REJ09B0009-0100Z

RENESAS

16

M16C/6N4 Group

Hardware Manual

RENESAS 16-BIT SINGLE-CHIP MICROCOMPUTER

M16C FAMILY / M16C/60 SERIES

Hardware Manual

Before using this material, please visit our website to confirm that this is the most current document available.

Rev. 1.00
Revision date: May 30, 2003

Renesas Technology
www.renesas.com

Keep safety first in your circuit designs!

- Renesas Technology Corporation 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

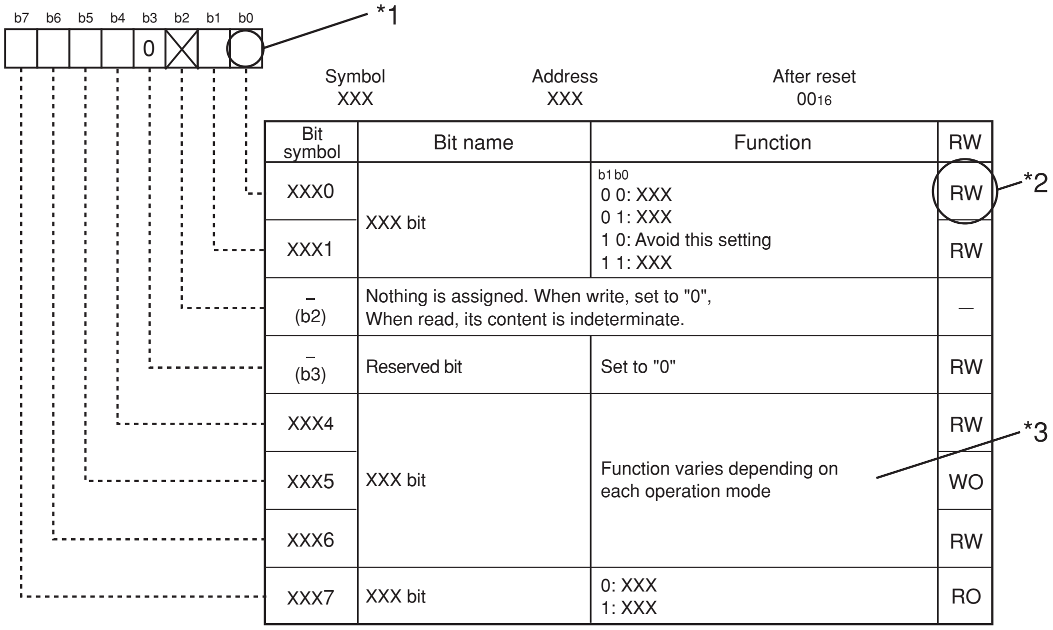
- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
- Renesas Technology Corporation 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.
- 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 Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
- 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 Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology Corporation 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 Corporation or an authorized Renesas Technology Corporation 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.
- The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- 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.
- Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.

How to Use This Manual

This hardware manual provides detailed information on features in the M16C/6N4 Group microcomputer. Users are expected to have basic knowledge of electric circuits, logical circuits and microcomputer.

Each register diagram contains bit functions with the following symbols and descriptions.

XXX register



*1

Blank: Set to “0” or “1” according to your intended use

0: Set to “0”

1: Set to “1”

X: Nothing is assigned

*2

RW: Read and write

RO: Read only

WO: Write only

–: Nothing is assigned

*3

Terms to use here are explained as follows.

- Nothing is assigned
  Nothing is assigned to the bit concerned. When write, set to “0” for new function in future plan.
- Reserved bit
  Reserved bit. Set the specified value.
- Avoid this setting
  The operation at having selected is not guaranteed.
- Function varies depending on each operation mode
  Bit function varies depending on peripheral function mode.
  Refer to register diagrams in each mode.

M16C Family Documents

The following document is prepared with the M16C family.

Document	Contents
Short Sheet	Hardware overview
Data Sheet	Hardware overview and electrical characteristics
Hardware Manual	Hardware specifications (pin assignments, memory maps, specifications of peripheral functions, electrical characteristics, timing charts)
Software Manual	Detailed description about instructions and microcomputer performance by each instruction
Application Note	• Application examples of peripheral functions • Sample programs • Introductory description about basic functions in M16C family • Programming method with the assembly and C languages

Table of Contents

M16C/6N4 Group Usage Note Reference Book

For the most current Usage Notes Reference Book, please visit our website.

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.

Quick Reference to Pages Classified by Address

Address	Register	Symbol	Page
0000_{16}			
0001_{16}			
0002_{16}			
0003_{16}			
0004_{16}	Processor mode register 0	PM0	31
0005_{16}	Processor mode register 1	PM1	32
0006_{16}	System clock control register 0	CM0	49
0007_{16}	System clock control register 1	CM1	50
0008_{16}	Chip select control register	CSR	37
0009_{16}	Address match interrupt enable register	AIER	89
$000A_{16}$	Protect register	PRCR	71
$000B_{16}$			
$000C_{16}$	Oscillation stop detection register	CM2	51
$000D_{16}$			
$000E_{16}$	Watchdog timer start register	WDTS	91
$000F_{16}$	Watchdog timer control register	WDC	91
0010_{16}			
0011_{16}	Address match interrupt register 0	RMAD0	89
0012_{16}			
0013_{16}			
0014_{16}			
0015_{16}	Address match interrupt register 1	RMAD1	89
0016_{16}			
0017_{16}			
0018_{16}			
0019_{16}			
$001A_{16}$			
$001B_{16}$	Chip select expansion control register	CSE	43
$001C_{16}$	PLL control register 0	PLC0	54
$001D_{16}$			
$001E_{16}$	Processor mode register 2	PM2	53
$001F_{16}$			
0020_{16}			
0021_{16}	DMA0 source pointer	SAR0	96
0022_{16}			
0023_{16}			
0024_{16}			
0025_{16}	DMA0 destination pointer	DAR0	96
0026_{16}			
0027_{16}			
0028_{16}	DMA0 transfer counter	TCR0	96
0029_{16}			
$002A_{16}$			
$002B_{16}$			
$002C_{16}$	DMA0 control register	DM0CON	95
$002D_{16}$			
$002E_{16}$			
$002F_{16}$			
0030_{16}			
0031_{16}	DMA1 source pointer	SAR1	96
0032_{16}			
0033_{16}			
0034_{16}			
0035_{16}	DMA1 destination pointer	DAR1	96
0036_{16}			
0037_{16}			
0038_{16}	DMA1 transfer counter	TCR1	96
0039_{16}			
$003A_{16}$			
$003B_{16}$			
$003C_{16}$	DMA1 control register	DM1CON	95
$003D_{16}$			
$003E_{16}$			
$003F_{16}$			

The blank areas are reserved.

Address	Register	Symbol	Page
0040_{16}			
0041_{16}	CAN0/1 wake up interrupt control register	C01WKIC	77
0042_{16}	CAN0 successful reception interrupt control register	C0RECIC	77
0043_{16}	CAN0 successful transmission interrupt control register	C0TRMIC	77
0044_{16}	INT3 interrupt control register	INT3IC	78
0045_{16}	Timer B5 interrupt control register	TB5IC	77
0046_{16}	Timer B4 interrupt control register	TB4IC	77
	UART1 bus collision detection interrupt control register	U1BCNIC	77
0047_{16}	Timer B3 interrupt control register	TB3IC	77
	UART0 bus collision detection interrupt control register	U0BCNIC	77
0048_{16}	CAN1 successful reception interrupt control register	C1RECIC	78
	INT5 interrupt control register	INT5IC	78
0049_{16}	CAN1 successful transmission interrupt control register	C1TRMIC	78
	SI/O3 interrupt control register	S3IC	78
	INT4 interrupt control register	INT4IC	78
$004A_{16}$	UART2 bus collision detection interrupt control register	U2BCNIC	77
$004B_{16}$	DMA0 interrupt control register	DM0IC	77
$004C_{16}$	DMA1 interrupt control register	DM1IC	77
$004D_{16}$	CAN0/1 error interrupt control register	C01ERRIC	77
$004E_{16}$	A-D conversion interrupt control register	ADIC	77
	Key input interrupt control register	KUPIC	77
$004F_{16}$	UART2 transmit interrupt control register	S2TIC	77
0050_{16}	UART2 receive interrupt control register	S2RIC	77
0051_{16}	UART0 transmit interrupt control register	S0TIC	77
0052_{16}	UART0 receive interrupt control register	S0RIC	77
0053_{16}	UART1 transmit interrupt control register	S1TIC	77
0054_{16}	UART1 receive interrupt control register	S1RIC	77
0055_{16}	Timer A0 interrupt control register	TA0IC	77
0056_{16}	Timer A1 interrupt control register	TA1IC	77
0057_{16}	Timer A2 interrupt control register	TA2IC	77
0058_{16}	Timer A3 interrupt control register	TA3IC	77
0059_{16}	Timer A4 interrupt control register	TA4IC	77
$005A_{16}$	Timer B0 interrupt control register	TB0IC	77
$005B_{16}$	Timer B1 interrupt control register	TB1IC	77
$005C_{16}$	Timer B2 interrupt control register	TB2IC	77
$005D_{16}$	INT0 interrupt control register	INT0IC	78
$005E_{16}$	INT1 interrupt control register	INT1IC	78
$005F_{16}$	INT2 interrupt control register	INT2IC	78
0060_{16}			
0061_{16}			
0062_{16}	CAN0 message box 0: Identifier / DLC		
0063_{16}			
0064_{16}			
0065_{16}			
0066_{16}			
0067_{16}			
0068_{16}			
0069_{16}	CAN0 message box 0: Data field		
$006A_{16}$			
$006B_{16}$			
$006C_{16}$			
$006D_{16}$			
$006E_{16}$	CAN0 message box 0: Time stamp		
$006F_{16}$			209
0070_{16}			210
0071_{16}			
0072_{16}	CAN0 message box 1: Identifier / DLC		
0073_{16}			
0074_{16}			
0075_{16}			
0076_{16}			
0077_{16}			
0078_{16}			
0079_{16}	CAN0 message box 1: data Field		
$007A_{16}$			
$007B_{16}$			
$007C_{16}$			
$007D_{16}$			
$007E_{16}$	CAN0 message box 1: Time stamp		
$007F_{16}$			

Address	Register	Symbol	Page
0080_{16}	CAN0 message box 2: Identifier / DLC		209 210
0081_{16}			
0082_{16}			
0083_{16}			
0084_{16}			
0085_{16}			
0086_{16}	CAN0 message box 2: Data field		
0087_{16}			
0088_{16}			
0089_{16}			
$008A_{16}$			
$008B_{16}$			
$008C_{16}$			
$008D_{16}$			
$008E_{16}$	CAN0 message box 2: Time stamp		
$008F_{16}$			
0090_{16}	CAN0 message box 3: Identifier / DLC		
0091_{16}			
0092_{16}			
0093_{16}			
0094_{16}			
0095_{16}			
0096_{16}	CAN0 message box 3: Data field		
0097_{16}			
0098_{16}			
0099_{16}			
$009A_{16}$			
$009B_{16}$			
$009C_{16}$			
$009D_{16}$			
$009E_{16}$	CAN0 message box 3: Time stamp		
$009F_{16}$			
$00A0_{16}$	CAN0 message box 4: Identifier / DLC		
$00A1_{16}$			
$00A2_{16}$			
$00A3_{16}$			
$00A4_{16}$			
$00A5_{16}$			
$00A6_{16}$	CAN0 message box 4: Data field		
$00A7_{16}$			
$00A8_{16}$			
$00A9_{16}$			
$00AA_{16}$			
$00AB_{16}$			
$00AC_{16}$			
$00AD_{16}$			
$00AE_{16}$	CAN0 message box 4: Time stamp		
$00AF_{16}$			
$00B0_{16}$	CAN0 message box 5: Identifier / DLC		
$00B1_{16}$			
$00B2_{16}$			
$00B3_{16}$			
$00B4_{16}$			
$00B5_{16}$			
$00B6_{16}$	CAN0 message box 5: Data field		
$00B7_{16}$			
$00B8_{16}$			
$00B9_{16}$			
$00BA_{16}$			
$00BB_{16}$			
$00BC_{16}$			
$00BD_{16}$			
$00BE_{16}$	CAN0 message box 5: Time stamp		
$00BF_{16}$			

Address	Register	Symbol	Page
$00C0_{16}$	CAN0 message box 6: Identifier / DLC		209 210
$00C1_{16}$			
$00C2_{16}$			
$00C3_{16}$			
$00C4_{16}$			
$00C5_{16}$			
$00C6_{16}$	CAN0 message box 6: Data field		
$00C7_{16}$			
$00C8_{16}$			
$00C9_{16}$			
$00CA_{16}$			
$00CB_{16}$			
$00CC_{16}$			
$00CD_{16}$			
$00CE_{16}$	CAN0 message box 6: Time stamp		
$00CF_{16}$			
$00D0_{16}$	CAN0 message box 7: Identifier / DLC		
$00D1_{16}$			
$00D2_{16}$			
$00D3_{16}$			
$00D4_{16}$			
$00D5_{16}$			
$00D6_{16}$	CAN0 message box 7: Data field		
$00D7_{16}$			
$00D8_{16}$			
$00D9_{16}$			
$00DA_{16}$			
$00DB_{16}$			
$00DC_{16}$			
$00DD_{16}$			
$00DE_{16}$	CAN0 message box 7: Time stamp		
$00DF_{16}$			
$00E0_{16}$	CAN0 message box 8: Identifier / DLC		
$00E1_{16}$			
$00E2_{16}$			
$00E3_{16}$			
$00E4_{16}$			
$00E5_{16}$			
$00E6_{16}$	CAN0 message box 8: Data field		
$00E7_{16}$			
$00E8_{16}$			
$00E9_{16}$			
$00EA_{16}$			
$00EB_{16}$			
$00EC_{16}$			
$00ED_{16}$			
$00EE_{16}$	CAN0 message box 8: Time stamp		
$00EF_{16}$			
$00F0_{16}$	CAN0 message box 9: Identifier / DLC		
$00F1_{16}$			
$00F2_{16}$			
$00F3_{16}$			
$00F4_{16}$			
$00F5_{16}$			
$00F6_{16}$	CAN0 message box 9: Data field		
$00F7_{16}$			
$00F8_{16}$			
$00F9_{16}$			
$00FA_{16}$			
$00FB_{16}$			
$00FC_{16}$			
$00FD_{16}$			
$00FE_{16}$	CAN0 message box 9: Time stamp		
$00FF_{16}$			

Address	Register	Symbol	Page
0100_{16}	CAN0 message box 10: Identifier / DLC		209 210
0101_{16}			
0102_{16}			
0103_{16}			
0104_{16}			
0105_{16}			
0106_{16}	CAN0 message box 10: Data field		
0107_{16}			
0108_{16}			
0109_{16}			
$010A_{16}$			
$010B_{16}$			
$010C_{16}$			
$010D_{16}$			
$010E_{16}$	CAN0 message box 10: Time stamp		
$010F_{16}$			
0110_{16}	CAN0 message box 11: Identifier / DLC		
0111_{16}			
0112_{16}			
0113_{16}			
0114_{16}			
0115_{16}			
0116_{16}	CAN0 message box 11: Data field		
0117_{16}			
0118_{16}			
0119_{16}			
$011A_{16}$			
$011B_{16}$			
$011C_{16}$			
$011D_{16}$			
$011E_{16}$	CAN0 message box 11: Time stamp		
$011F_{16}$			
0120_{16}	CAN0 message box 12: Identifier / DLC		
0121_{16}			
0122_{16}			
0123_{16}			
0124_{16}			
0125_{16}			
0126_{16}	CAN0 message box 12: Data field		
0127_{16}			
0128_{16}			
0129_{16}			
$012A_{16}$			
$012B_{16}$			
$012C_{16}$			
$012D_{16}$			
$012E_{16}$	CAN0 message box 12: Time stamp		
$012F_{16}$			
0130_{16}	CAN0 message box 13: Identifier / DLC		
0131_{16}			
0132_{16}			
0133_{16}			
0134_{16}			
0135_{16}			
0136_{16}	CAN0 message box 13: Data field		
0137_{16}			
0138_{16}			
0139_{16}			
$013A_{16}$			
$013B_{16}$			
$013C_{16}$			
$013D_{16}$			
$013E_{16}$	CAN0 message box 13: Time stamp		
$013F_{16}$			

Address	Register	Symbol	Page
0140_{16}	CAN0 message box 14: Identifier /DLC		209 210
0141_{16}			
0142_{16}			
0143_{16}			
0144_{16}			
0145_{16}			
0146_{16}	CAN0 message box 14: Data field		
0147_{16}			
0148_{16}			
0149_{16}			
$014A_{16}$			
$014B_{16}$			
$014C_{16}$			
$014D_{16}$			
$014E_{16}$	CAN0 message box 14: Time stamp		
$014F_{16}$			
0150_{16}	CAN0 message box 15: Identifier /DLC		
0151_{16}			
0152_{16}			
0153_{16}			
0154_{16}			
0155_{16}			
0156_{16}	CAN0 message box 15: Data field		
0157_{16}			
0158_{16}			
0159_{16}			
$015A_{16}$			
$015B_{16}$			
$015C_{16}$			
$015D_{16}$			
$015E_{16}$	CAN0 message box 15: Time stamp		
$015F_{16}$			
0160_{16}	CAN0 global mask register	C0GMR	211
0161_{16}			
0162_{16}			
0163_{16}			
0164_{16}			
0165_{16}			
0166_{16}	CAN0 local mask A register	C0LMAR	211
0167_{16}			
0168_{16}			
0169_{16}			
$016A_{16}$			
$016B_{16}$			
$016C_{16}$	CAN0 local mask B register	C0LMBR	211
$016D_{16}$			
$016E_{16}$			
$016F_{16}$			
0170_{16}			
0171_{16}			
0172_{16}			
0173_{16}			
0174_{16}			
0175_{16}			
0176_{16}			
0177_{16}			
0178_{16}			
0179_{16}			
$017A_{16}$			
$017B_{16}$			
$017C_{16}$			
$017D_{16}$			
$017E_{16}$			
$017F_{16}$			

The blank areas are reserved.

Address	Register	Symbol	Page
0180$_{16}$			
0181$_{16}$			
0182$_{16}$			
0183$_{16}$			
0184$_{16}$			
0185$_{16}$			
0186$_{16}$			
0187$_{16}$			
0188$_{16}$			
0189$_{16}$			
018A$_{16}$			
018B$_{16}$			
018C$_{16}$			
018D$_{16}$			
018E$_{16}$			
018F$_{16}$			
0190$_{16}$			
0191$_{16}$			
0192$_{16}$			
0193$_{16}$			
0194$_{16}$			
0195$_{16}$			
0196$_{16}$			
0197$_{16}$			
0198$_{16}$			
0199$_{16}$			
019A$_{16}$			
019B$_{16}$			
019C$_{16}$			
019D$_{16}$			
019E$_{16}$			
019F$_{16}$			
01A0$_{16}$			
01A1$_{16}$			
01A2$_{16}$			
01A3$_{16}$			
01A4$_{16}$			
01A5$_{16}$			
01A6$_{16}$			
01A7$_{16}$			
01A8$_{16}$			
01A9$_{16}$			
01AA$_{16}$			
01AB$_{16}$			
01AC$_{16}$			
01AD$_{16}$			
01AE$_{16}$			
01AF$_{16}$			
01B0$_{16}$			
01B1$_{16}$			
01B2$_{16}$			
01B3$_{16}$			
01B4$_{16}$			
01B5$_{16}$	Flash memory control register 1	FMR1	269
01B6$_{16}$			
01B7$_{16}$	Flash memory control register 0	FMR0	269
01B8$_{16}$	Address match interrupt register 2	RAMD2	89
01B9$_{16}$			
01BA$_{16}$			
01BB$_{16}$	Address match interrupt enable register 2	AIER2	89
01BC$_{16}$	Address match interrupt register 3	RAMD3	89
01BD$_{16}$			
01BE$_{16}$			
01BF$_{16}$			

Address	Register	Symbol	Page
01C0$_{16}$	Timer B3,4,5 count start flag	TBSR	121
01C1$_{16}$			
01C2$_{16}$	Timer A1-1 register	TA11	132
01C3$_{16}$			
01C4$_{16}$	Timer A2-1 register	TA21	132
01C5$_{16}$			
01C6$_{16}$	Timer A4-1 register	TA41	132
01C7$_{16}$			
01C8$_{16}$	Three-phase PWM control register 0	INVC0	129
01C9$_{16}$	Three-phase PWM control register 1	INVC1	130
01CA$_{16}$	Three-phase output buffer register 0	IDB0	131
01CB$_{16}$	Three-phase output buffer register 1	IDB1	131
01CC$_{16}$	Dead time timer	DTT	131
01CD$_{16}$	Timer B2 interrupt occurrence frequency set counter	ICTB2	133
01CE$_{16}$			
01CF$_{16}$			
01D0$_{16}$	Timer B3 register	TB3	120
01D1$_{16}$			
01D2$_{16}$	Timer B4 register	TB4	120
01D3$_{16}$			
01D4$_{16}$	Timer B5 register	TB5	120
01D5$_{16}$			
01D6$_{16}$			
01D7$_{16}$			
01D8$_{16}$			
01D9$_{16}$			
01DA$_{16}$			
01DB$_{16}$	Timer B3 mode register	TB3MR	120 122 123 125
01DC$_{16}$	Timer B4 mode register	TB4MR	
01DD$_{16}$	Timer B5 mode register	TB5MR	
01DE$_{16}$	Interrupt cause select register 0	IFSR0	86
01DF$_{16}$	Interrupt cause select register 1	IFSR1	86
01E0$_{16}$	SI/O3 transmit/receive register	S3TRR	183
01E1$_{16}$			
01E2$_{16}$	SI/O3 control register	S3C	183
01E3$_{16}$	SI/O3 bit rate generator	S3BRG	183
01E4$_{16}$			
01E5$_{16}$			
01E6$_{16}$			
01E7$_{16}$			
01E8$_{16}$			
01E9$_{16}$			
01EA$_{16}$			
01EB$_{16}$			
01EC$_{16}$	UART0 special mode register 4	U0SMR4	146
01ED$_{16}$	UART0 special mode register 3	U0SMR3	145
01EE$_{16}$	UART0 special mode register 2	U0SMR2	145
01EF$_{16}$	UART0 special mode register	U0SMR	144
01F0$_{16}$	UART1 special mode register 4	U1SMR4	146
01F1$_{16}$	UART1 special mode register 3	U1SMR3	145
01F2$_{16}$	UART1 special mode register 2	U1SMR2	145
01F3$_{16}$	UART1 special mode register	U1SMR	144
01F4$_{16}$	UART2 special mode register 4	U2SMR4	146
01F5$_{16}$	UART2 special mode register 3	U2SMR3	145
01F6$_{16}$	UART2 special mode register 2	U2SMR2	145
01F7$_{16}$	UART2 special mode register	U2SMR	144
01F8$_{16}$	UART2 transmit/receive mode register	U2MR	142
01F9$_{16}$	UART2 bit rate generator	U2BRG	141
01FA$_{16}$	UART2 transmit buffer register	U2TB	141
01FB$_{16}$			
01FC$_{16}$	UART2 transmit/receive mode register 0	U2C0	142
01FD$_{16}$	UART2 transmit/receive mode register 1	U2C1	143
01FE$_{16}$	UART2 receive buffer register	U2RB	141
01FF$_{16}$			

The blank areas are reserved.

Address	Register	Symbol	Page
0200_{16}	CAN0 message control register 0	C0MCTL0	212
0201_{16}	CAN0 message control register 1	C0MCTL1	
0202_{16}	CAN0 message control register 2	C0MCTL2	
0203_{16}	CAN0 message control register 3	C0MCTL3	
0204_{16}	CAN0 message control register 4	C0MCTL4	
0205_{16}	CAN0 message control register 5	C0MCTL5	
0206_{16}	CAN0 message control register 6	C0MCTL6	
0207_{16}	CAN0 message control register 7	C0MCTL7	
0208_{16}	CAN0 message control register 8	C0MCTL8	
0209_{16}	CAN0 message control register 9	C0MCTL9	
$020A_{16}$	CAN0 message control register 10	C0MCTL10	
$020B_{16}$	CAN0 message control register 11	C0MCTL11	
$020C_{16}$	CAN0 message control register 12	C0MCTL12	
$020D_{16}$	CAN0 message control register 13	C0MCTL13	
$020E_{16}$	CAN0 message control register 14	C0MCTL14	
$020F_{16}$	CAN0 message control register 15	C0MCTL15	
0210_{16}	CAN0 control register	C0CTLR	213
0211_{16}			
0212_{16}	CAN0 status register	C0STR	214
0213_{16}			
0214_{16}	CAN0 slot status register	C0SSTR	215
0215_{16}			
0216_{16}	CAN0 interrupt control register	C0ICR	216
0217_{16}			
0218_{16}	CAN0 extended register	C0IDR	216
0219_{16}			
$021A_{16}$	CAN0 configuration register	C0CONR	217
$021B_{16}$			
$021C_{16}$	CAN0 receive error count register	C0RECR	218
$021D_{16}$	CAN0 transmit error count register	C0TECR	218
$021E_{16}$	CAN0 time stamp register	C0TSR	219
$021F_{16}$			
0220_{16}	CAN1 message control register 0	C1MCTL0	212
0221_{16}	CAN1 message control register 1	C1MCTL1	
0222_{16}	CAN1 message control register 2	C1MCTL2	
0223_{16}	CAN1 message control register 3	C1MCTL3	
0224_{16}	CAN1 message control register 4	C1MCTL4	
0225_{16}	CAN1 message control register 5	C1MCTL5	
0226_{16}	CAN1 message control register 6	C1MCTL6	
0227_{16}	CAN1 message control register 7	C1MCTL7	
0228_{16}	CAN1 message control register 8	C1MCTL8	
0229_{16}	CAN1 message control register 9	C1MCTL9	
$022A_{16}$	CAN1 message control register 10	C1MCTL10	
$022B_{16}$	CAN1 message control register 11	C1MCTL11	
$022C_{16}$	CAN1 message control register 12	C1MCTL12	
$022D_{16}$	CAN1 message control register 13	C1MCTL13	
$022E_{16}$	CAN1 message control register 14	C1MCTL14	
$022F_{16}$	CAN1 message control register 15	C1MCTL15	
0230_{16}	CAN1 control register	C1CTLR	213
0231_{16}			
0232_{16}	CAN1 status register	C1STR	214
0233_{16}			
0234_{16}	CAN1 slot status register	C1SSTR	215
0235_{16}			
0236_{16}	CAN1 interrupt control register	C1ICR	216
0237_{16}			
0238_{16}	CAN1 extended register	C1IDR	216
0239_{16}			
$023A_{16}$	CAN1 configuration register	C1CONR	217
$023B_{16}$			
$023C_{16}$	CAN1 receive error count register	C1RECR	218
$023D_{16}$	CAN1 transmit error count register	C1TECR	218
$023E_{16}$	CAN1 time stamp register	C1TSR	219
$023F_{16}$			

Address	Register	Symbol	Page
0240_{16}			
0241_{16}			
0242_{16}	CAN0 acceptance filter support register	C0AFS	219
0243_{16}			
0244_{16}	CAN1 acceptance filter support register	C1AFS	219
0245_{16}			
0246_{16}			
0247_{16}			
0248_{16}			
0249_{16}			
$024A_{16}$			
$024B_{16}$			
$024C_{16}$			
$024D_{16}$			
$024E_{16}$			
$024F_{16}$			
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 function clock select register	PCLKR	52
$025F_{16}$	CAN0/1 clock select register	CCLKR	52
0260_{16}	CAN1 message box 0: Identifier / DLC		209 210
0261_{16}			
0262_{16}			
0263_{16}			
0264_{16}			
0265_{16}			
0266_{16}	CAN1 message box 0: Data field		
0267_{16}			
0268_{16}			
0269_{16}			
$026A_{16}$			
$026B_{16}$			
$026C_{16}$			
$026D_{16}$			
$026E_{16}$	CAN1 message box 0:Time stamp		
$026F_{16}$			
0270_{16}	CAN1 message box 1: Identifier / DLC		
0271_{16}			
0272_{16}			
0273_{16}			
0274_{16}			
0275_{16}			
0276_{16}	CAN1 message box 1: Data field		
0277_{16}			
0278_{16}			
0279_{16}			
$027A_{16}$			
$027B_{16}$			
$027C_{16}$			
$027D_{16}$			
$027E_{16}$	CAN1 message box 1:Time stamp		
$027F_{16}$			

The blank areas are reserved.

Address	Register	Symbol	Page
0280$_{16}$	CAN1 message box 2: Identifier / DLC		
0281$_{16}$			
0282$_{16}$			
0283$_{16}$			
0284$_{16}$			
0285$_{16}$			
0286$_{16}$	CAN1 message box 2: Data field		
0287$_{16}$			
0288$_{16}$			
0289$_{16}$			
028A$_{16}$			
028B$_{16}$			
028C$_{16}$			
028D$_{16}$			
028E$_{16}$	CAN1 message box 2: Time stamp		
028F$_{16}$			
0290$_{16}$	CAN1 message box 3: Identifier / DLC		
0291$_{16}$			
0292$_{16}$			
0293$_{16}$			
0294$_{16}$			
0295$_{16}$			
0296$_{16}$	CAN1 message box 3: Data field		
0297$_{16}$			
0298$_{16}$			
0299$_{16}$			
029A$_{16}$			
029B$_{16}$			
029C$_{16}$			
029D$_{16}$			
029E$_{16}$	CAN1 message box 3: Time stamp		
029F$_{16}$			209
02A0$_{16}$	CAN1 message box 4: Identifier / DLC		210
02A1$_{16}$			
02A2$_{16}$			
02A3$_{16}$			
02A4$_{16}$			
02A5$_{16}$			
02A6$_{16}$	CAN1 message box 4: Data field		
02A7$_{16}$			
02A8$_{16}$			
02A9$_{16}$			
02AA$_{16}$			
02AB$_{16}$			
02AC$_{16}$			
02AD$_{16}$			
02AE$_{16}$	CAN1 message box 4: Time stamp		
02AF$_{16}$			
02B0$_{16}$	CAN1 message box 5: Identifier / DLC		
02B1$_{16}$			
02B2$_{16}$			
02B3$_{16}$			
02B4$_{16}$			
02B5$_{16}$			
02B6$_{16}$	CAN1 message box 5: Data field		
02B7$_{16}$			
02B8$_{16}$			
02B9$_{16}$			
02BA$_{16}$			
02BB$_{16}$			
02BC$_{16}$			
02BD$_{16}$			
02BE$_{16}$	CAN1 message box 5: Time stamp		
02BF$_{16}$			

Address	Register	Symbol	Page
02C0$_{16}$	CAN1 message box 6: Identifier / DLC		
02C1$_{16}$			
02C2$_{16}$			
02C3$_{16}$			
02C4$_{16}$			
02C5$_{16}$			
02C6$_{16}$	CAN1 message box 6: Data field		
02C7$_{16}$			
02C8$_{16}$			
02C9$_{16}$			
02CA$_{16}$			
02CB$_{16}$			
02CC$_{16}$			
02CD$_{16}$			
02CE$_{16}$	CAN1 message box 6: Time stamp		
02CF$_{16}$			
02D0$_{16}$	CAN1 message box 7: Identifier / DLC		
02D1$_{16}$			
02D2$_{16}$			
02D3$_{16}$			
02D4$_{16}$			
02D5$_{16}$			
02D6$_{16}$	CAN1 message box 7: Data field		
02D7$_{16}$			
02D8$_{16}$			
02D9$_{16}$			
02DA$_{16}$			
02DB$_{16}$			
02DC$_{16}$			
02DD$_{16}$			
02DE$_{16}$	CAN1 message box 7: Time stamp		
02DF$_{16}$			209
02E0$_{16}$	CAN1 message box 8: Identifier / DLC		210
02E1$_{16}$			
02E2$_{16}$			
02E3$_{16}$			
02E4$_{16}$			
02E5$_{16}$			
02E6$_{16}$	CAN1 message box 8: Data field		
02E7$_{16}$			
02E8$_{16}$			
02E9$_{16}$			
02EA$_{16}$			
02EB$_{16}$			
02EC$_{16}$			
02ED$_{16}$			
02EE$_{16}$	CAN1 message box 8: Time stamp		
02EF$_{16}$			
02F0$_{16}$	CAN1 message box 9: Identifier / DLC		
02F1$_{16}$			
02F2$_{16}$			
02F3$_{16}$			
02F4$_{16}$			
02F5$_{16}$			
02F6$_{16}$	CAN1 message box 9: Data field		
02F7$_{16}$			
02F8$_{16}$			
02F9$_{16}$			
02FA$_{16}$			
02FB$_{16}$			
02FC$_{16}$			
02FD$_{16}$			
02FE$_{16}$	CAN1 message box 9: Time stamp		
02FF$_{16}$			

Address	Register	Symbol	Page
0300_{16}	CAN1 message box 10: Identifier / DLC		209 210
0301_{16}			
0302_{16}			
0303_{16}			
0304_{16}			
0305_{16}			
0306_{16}	CAN1 message box 10: Data field		
0307_{16}			
0308_{16}			
0309_{16}			
$030A_{16}$			
$030B_{16}$			
$030C_{16}$			
$030D_{16}$			
$030E_{16}$	CAN1 message box 10: Time stamp		
$030F_{16}$			
0310_{16}	CAN1 message box 11: Identifier / DLC		
0311_{16}			
0312_{16}			
0313_{16}			
0314_{16}			
0315_{16}			
0316_{16}	CAN1 message box 11: Data field		
0317_{16}			
0318_{16}			
0319_{16}			
$031A_{16}$			
$031B_{16}$			
$031C_{16}$			
$031D_{16}$			
$031E_{16}$	CAN1 message box 11: Time stamp		
$031F_{16}$			
0320_{16}	CAN1 message box 12: Identifier / DLC		
0321_{16}			
0322_{16}			
0323_{16}			
0324_{16}			
0325_{16}			
0326_{16}	CAN1 message box 12: Data field		
0327_{16}			
0328_{16}			
0329_{16}			
$032A_{16}$			
$032B_{16}$			
$032C_{16}$			
$032D_{16}$			
$032E_{16}$	CAN1 message box 12: Time stamp		
$032F_{16}$			
0330_{16}	CAN1 message box 13: Identifier / DLC		
0331_{16}			
0332_{16}			
0333_{16}			
0334_{16}			
0335_{16}			
0336_{16}	CAN1 message box 13: Data field		
0337_{16}			
0338_{16}			
0339_{16}			
$033A_{16}$			
$033B_{16}$			
$033C_{16}$			
$033D_{16}$			
$033E_{16}$	CAN1 message box 13: Time stamp		
$033F_{16}$			

Address	Register	Symbol	Page
0340_{16}	CAN1 message box 14: Identifier / DLC		209 210
0341_{16}			
0342_{16}			
0343_{16}			
0344_{16}			
0345_{16}			
0346_{16}	CAN1 message box 14: Data field		
0347_{16}			
0348_{16}			
0349_{16}			
$034A_{16}$			
$034B_{16}$			
$034C_{16}$			
$034D_{16}$			
$034E_{16}$	CAN1 message box 14: Time stamp		
$034F_{16}$			
0350_{16}	CAN1 message box 15: Identifier / DLC		
0351_{16}			
0352_{16}			
0353_{16}			
0354_{16}			
0355_{16}			
0356_{16}	CAN1 message box 15: Data field		
0357_{16}			
0358_{16}			
0359_{16}			
$035A_{16}$			
$035B_{16}$			
$035C_{16}$			
$035D_{16}$			
$035E_{16}$	CAN1 message box 15: Time stamp		
$035F_{16}$			
0360_{16}	CAN1 global mask register	C0GMR	211
0361_{16}			
0362_{16}			
0363_{16}			
0364_{16}			
0365_{16}			
0366_{16}	CAN1 local mask A register	C0LMAR	211
0367_{16}			
0368_{16}			
0369_{16}			
$036A_{16}$			
$036B_{16}$			
$036C_{16}$	CAN1 local mask B register	C0LMBR	211
$036D_{16}$			
$036E_{16}$			
$036F_{16}$			
0370_{16}			
0371_{16}			
0372_{16}			
0373_{16}			
0374_{16}			
0375_{16}			
0376_{16}			
0377_{16}			
0378_{16}			
0379_{16}			
$037A_{16}$			
$037B_{16}$			
$037C_{16}$			
$037D_{16}$			
$037E_{16}$			
$037F_{16}$			

The blank areas are reserved.

Address	Register	Symbol	Page
0380_{16}	Count start flag	TABSR	106,121,134
0381_{16}	Clock prescaler reset flag	CPSRF	107,121
0382_{16}	One-shot start flag	ONSF	107
0383_{16}	Trigger select register	TRGSR	107,134
0384_{16}	Up-down flag	UDF	106
0385_{16}			
0386_{16}	Timer A0 register	TA0	105
0387_{16}			
0388_{16}	Timer A1 register	TA1	105 132
0389_{16}			
$038A_{16}$	Timer A2 register	TA2	105 132
$038B_{16}$			
$038C_{16}$	Timer A3 register	TA3	105
$038D_{16}$			
$038E_{16}$	Timer A4 register	TA4	105 132
$038F_{16}$			
0390_{16}	Timer B0 register	TB0	120
0391_{16}			
0392_{16}	Timer B1 register	TB1	120
0393_{16}			
0394_{16}	Timer B2 register	TB2	120 132
0395_{16}			
0396_{16}	Timer A0 mode register	TA0MR	105
0397_{16}	Timer A1 mode register	TA1MR	108 135
0398_{16}	Timer A2 mode register	TA2MR	110 112,135
0399_{16}	Timer A3 mode register	TA3MR	115 112
$039A_{16}$	Timer A4 mode register	TA4MR	117 112,135
$039B_{16}$	Timer B0 mode register	TB0MR	120,122
$039C_{16}$	Timer B1 mode register	TB1MR	123,125
$039D_{16}$	Timer B2 mode register	TB2MR	135
$039E_{16}$	Timer B2 special mode register	TB2SC	133
$039F_{16}$			
$03A0_{16}$	UART0 transmit/receive mode register	U0MR	142
$03A1_{16}$	UART0 bit rate generator	U0BRG	141
$03A2_{16}$	UART0 transmit buffer register	U0TB	141
$03A3_{16}$			
$03A4_{16}$	UART0 transmit/receive control register 0	U0C0	142
$03A5_{16}$	UART0 transmit/receive control register 1	U0C1	143
$03A6_{16}$	UART0 receive buffer register	U0RB	141
$03A7_{16}$			
$03A8_{16}$	UART1 transmit/receive mode register	U1MR	142
$03A9_{16}$	UART1 bit rate generator	U1BRG	141
$03AA_{16}$	UART1 transmit buffer register	U1TB	141
$03AB_{16}$			
$03AC_{16}$	UART1 transmit/receive control register 0	U1C0	142
$03AD_{16}$	UART1 transmit/receive control register 1	U1C1	143
$03AE_{16}$	UART1 receive buffer register	U1RB	141
$03AF_{16}$			
$03B0_{16}$	UART transmit/receive control register 2	UCON	144
$03B1_{16}$			
$03B2_{16}$			
$03B3_{16}$			
$03B4_{16}$			
$03B5_{16}$			
$03B6_{16}$			
$03B7_{16}$			
$03B8_{16}$	DMA0 request cause select register	DM0SL	94
$03B9_{16}$			
$03BA_{16}$	DMA1 request cause select register	DM1SL	95
$03BB_{16}$			
$03BC_{16}$	CRC data register	CRCD	205
$03BD_{16}$			
$03BE_{16}$	CRC input register	CRCIN	205
$03BF_{16}$			

Address	Register	Symbol	Page
$03C0_{16}$	A-D register 0	AD0	190
$03C1_{16}$			
$03C2_{16}$	A-D register 1	AD1	
$03C3_{16}$			
$03C4_{16}$	A-D register 2	AD2	
$03C5_{16}$			
$03C6_{16}$	A-D register 3	AD3	
$03C7_{16}$			
$03C8_{16}$	A-D register 4	AD4	
$03C9_{16}$			
$03CA_{16}$	A-D register 5	AD5	
$03CB_{16}$			
$03CC_{16}$	A-D register 6	AD6	
$03CD_{16}$			
$03CE_{16}$	A-D register 7	AD7	
$03CF_{16}$			
$03D0_{16}$			
$03D1_{16}$			
$03D2_{16}$			
$03D3_{16}$			
$03D4_{16}$	A-D control register 2	ADCON2	190
$03D5_{16}$			
$03D6_{16}$	A-D control register 0	ADCON0	189,192,194
$03D7_{16}$	A-D control register 1	ADCON1	196,198,200
$03D8_{16}$	D-A register 0	DA0	204
$03D9_{16}$			
$03DA_{16}$	D-A register 1	DA1	204
$03DB_{16}$			
$03DC_{16}$	D-A control register	DACON	204
$03DD_{16}$			
$03DE_{16}$			
$03DF_{16}$			
$03E0_{16}$	Port P0 register	P0	239
$03E1_{16}$	Port P1 register	P1	239
$03E2_{16}$	Port P0 direction register	PD0	238
$03E3_{16}$	Port P1 direction register	PD1	238
$03E4_{16}$	Port P2 register	P2	239
$03E5_{16}$	Port P3 register	P3	239
$03E6_{16}$	Port P2 direction register	PD2	238
$03E7_{16}$	Port P3 direction register	PD3	238
$03E8_{16}$	Port P4 register	P4	239
$03E9_{16}$	Port P5 register	P5	239
$03EA_{16}$	Port P4 direction register	PD4	238
$03EB_{16}$	Port P5 direction register	PD5	238
$03EC_{16}$	Port P6 register	P6	239
$03ED_{16}$	Port P7 register	P7	239
$03EE_{16}$	Port P6 direction register	PD6	238
$03EF_{16}$	Port P7 direction register	PD7	238
$03F0_{16}$	Port P8 register	P8	239
$03F1_{16}$	Port P9 register	P9	239
$03F2_{16}$	Port P8 direction register	PD8	238
$03F3_{16}$	Port P9 direction register	PD9	238
$03F4_{16}$	Port P10 register	P10	239
$03F5_{16}$			
$03F6_{16}$	Port P10 direction register	PD10	238
$03F7_{16}$			
$03F8_{16}$			
$03F9_{16}$			
$03FA_{16}$			
$03FB_{16}$			
$03FC_{16}$	Pull-up control register 0	PUR0	240
$03FD_{16}$	Pull-up control register 1	PUR1	240
$03FE_{16}$	Pull-up control register 2	PUR2	240
$03FF_{16}$	Port control register	PCR	241

The blank areas are reserved.

Overview

The M16C/6N4 group of single-chip microcomputers are built using the high-performance silicon gate CMOS process using an M16C/60 Series CPU core and are packaged in a 100-pin plastic molded QFP. These single-chip microcomputers operate using sophisticated instructions featuring a high level of instruction efficiency. With 1 Mbyte of address space, they are capable of executing instructions at high speed. Being equipped with two CAN (Controller Area Network) modules in M16C/6N4 group, the microcomputer is suited to drive automotive and industrial control systems. The CAN modules comply with the 2.0B specification. In addition, this microcomputer contains a multiplier and DMAC which combined with fast instruction processing capability, makes it suitable for control of various OA, communication, and industrial equipment which requires high-speed arithmetic/logic operations.

Applications

Automotive, industrial control systems and other autmobile, other

Performance Outline

Table 1.1.1 lists a performance outline of M16C/6N4 group.

Table 1.1.1 Performance outline of M16C/6N4 Group

Item		Performance
Number of basic instructions		91 instructions
Shortest instruction execution time		50.0 ns (f(BCLK)=20MHz, 1/1 prescaler, without software wait)
Memory capacity	ROM	(Refer to the product list)
	RAM	(Refer to the product list)
I/O port	P0 to P10 (except $P8_5$)	8 bits × 10, 7 bits × 1
Input port	$P8_5$	1 bit × 1 ($\overline{NMI}$ pin level judgment)
Multifunction timer	TA0, TA1, TA2, TA3, TA4	Output: 16 bits × 5 channels
	TB0, TB1, TB2, TB3, TB4, TB5	Input: 16 bits × 6 channels
Serial I/O	UART0, UART1, UART2	3 channels: UART, clock synchronous, I^2C-bus (Note 1) (option) or IEBus (Note 2) (option)
	SI/O3	1 channel: Clock synchronous
A-D converter		10 bits × (8 × 3 + 2) channels
D-A converter		8 bits × 2 channels
DMAC		2 channels (trigger: 24 sources)
CRC calculation circuit		1 circuit: CRC-CCITT
CAN Module		2 channels with 2.0B specification
Watchdog timer		15 bits × 1 (with prescaler)
Interrupt		31 internal and 9 external sources, 4 software sources, 7 levels
Clock generation circuit		4 circuits · Main clock, · Sub clock: These circuit contain a built-in feedback resistor; and external ceramic/quartz oscillator · Ring oscillator · PLL frequency synthesizer Main clock oscillation stop and re-oscillation detection function
Power supply voltage		4.2 to 5.5V (f(BCLK) = 20MHz, 1/1 prescaler, without software wait)
Flash memory	Program/erase voltage	5.0 ± 0.5 V
	Number of program/erase	100 times
Power consumption		Mask ROM version: 18 mA (Vcc=5V, (f(BCLK)=20MHz, 1/1 prescaler, without software wait) Flash memory version: 20 mA (Vcc=5V, (f(BCLK)=20MHz, 1/1 prescaler, without software wait)
I/O characteristics	I/O withstand voltage	5.0 V
	Output current	5 mA
Operating ambient temperature		-40 to 85°C (T version) -40 to 125°C (V version) (option)
Memory expansion		Available (to 1 Mbyte)
Device configuration		CMOS high performance silicon gate
Package		100-pin plastic mold QFP

Note 1: I^2C-bus is a registered trademark of Koninklijke Philips Electronics N.V.
Note 2: IEBus is a registered trademark of NEC Electronics Corporation.

option: If you desire this option, please so specify.

Block Diagram

Figure 1.1.1 shows a block diagram of M16C/6N4 group.

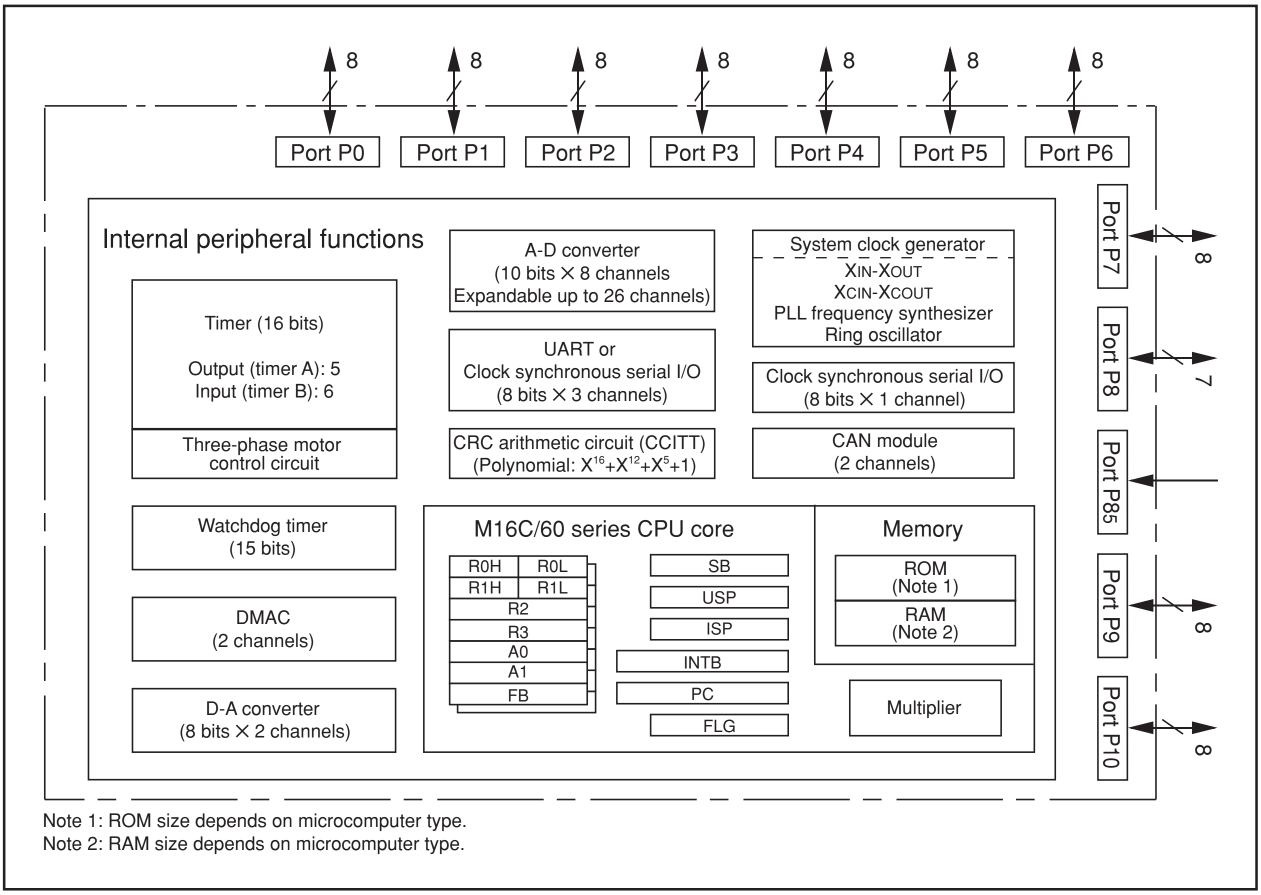


Figure 1.1.1 Block Diagram

Product List

Table 1.1.2 lists the M16C/6N4 group products and Figure 1.1.2 shows the type numbers, memory sizes and packages.

Table 1.1.2 Product List As of May 2003

Type No.	ROM capacity	RAM capacity	Package type	Remarks
M306N4MCT-XXXFP **	128 Kbytes	5 Kbytes	100P6S-A	Mask ROM version
M306N4MCV-XXXFP *				
M306N4FCTFP **				Flash memory version
M306N4FCVFP *				
M306N4MGT-XXXFP *	256 Kbytes	10 Kbytes		Mask ROM version
M306N4MGV-XXXFP *				
M306N4FGTFP **				Flash memory version
M306N4FGVFP *				

*: Under planning
**: Under development

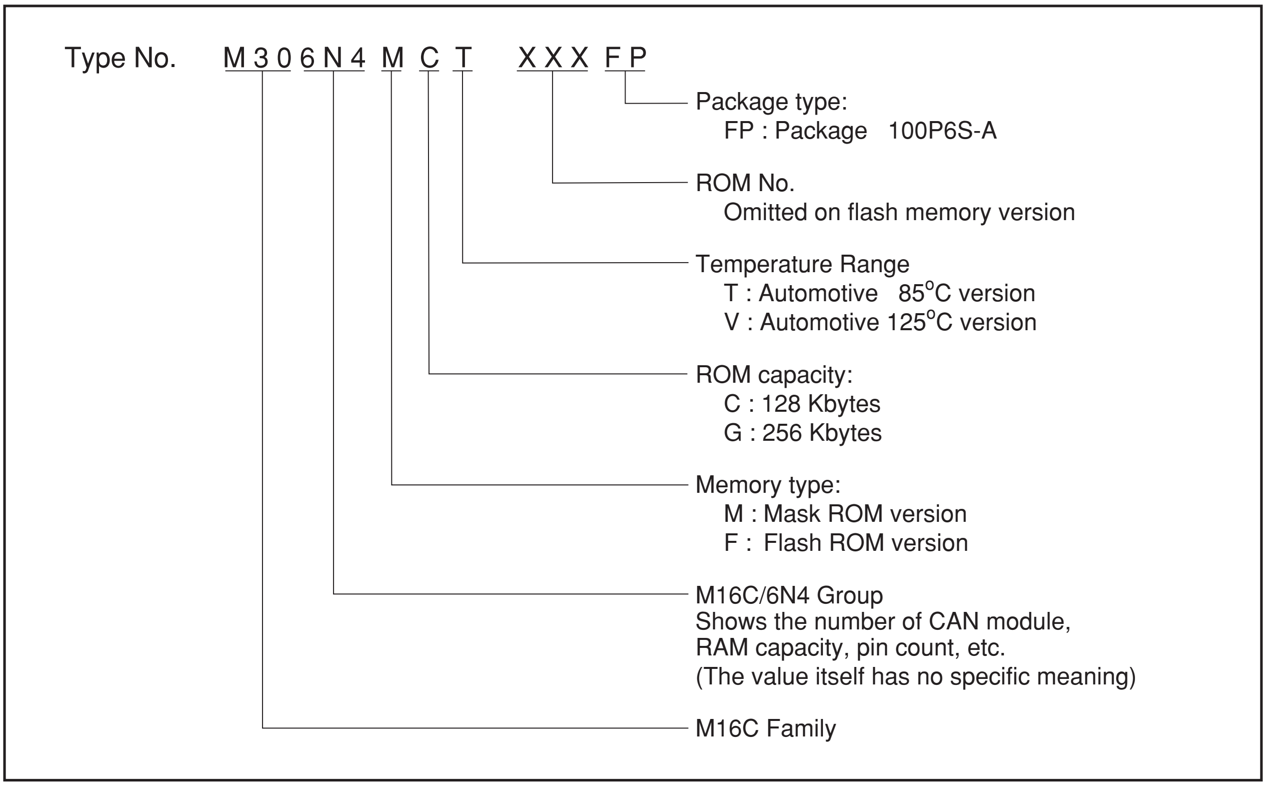
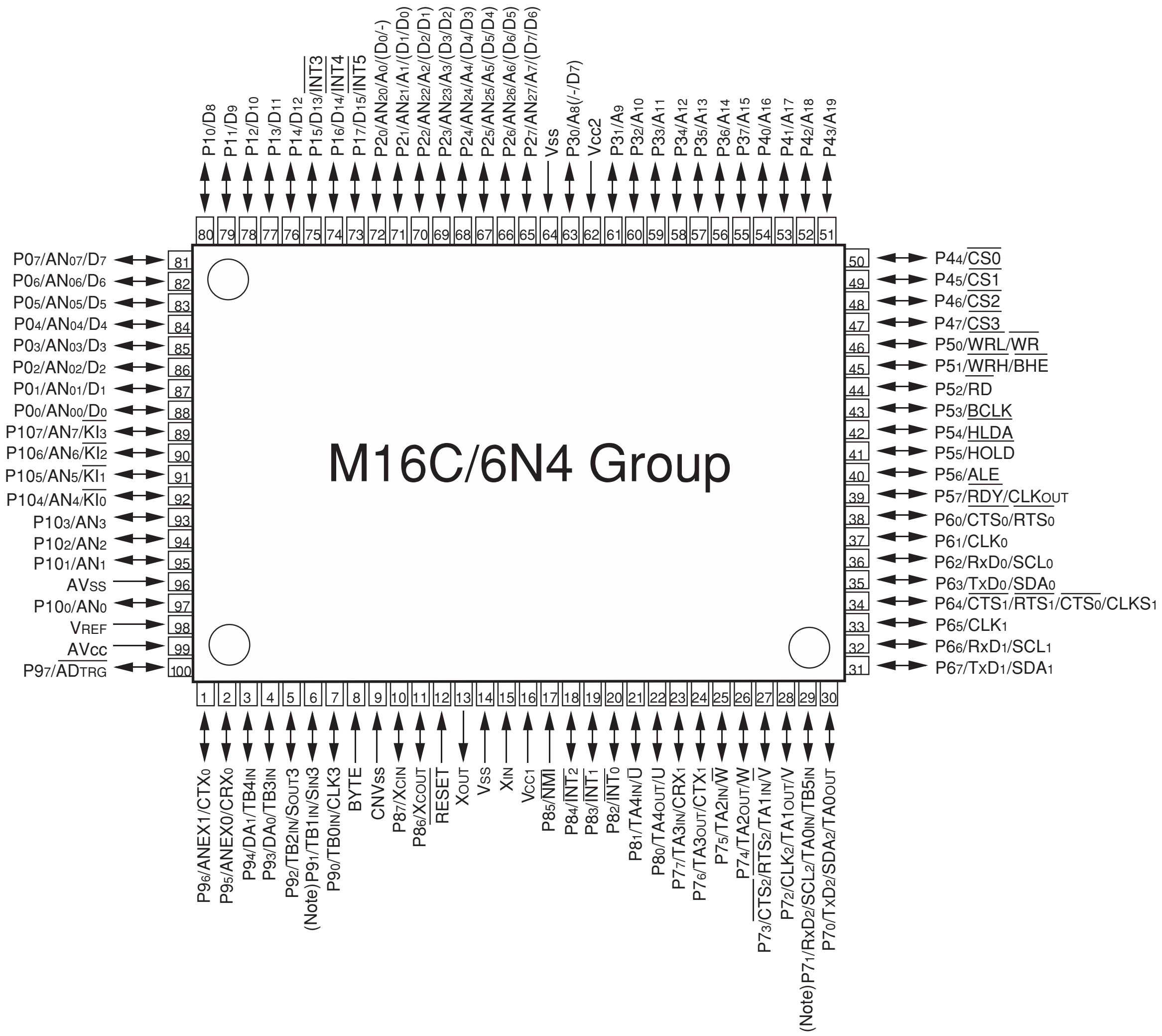


Figure 1.1.2 Type No., Memory Size, and Package

Pin Configuration

Figures 1.1.3 shows the pin configuration (top view).

PIN CONFIGURATION (top view)



Note: P7₁ and P9₁ are N channel open-drain output pins.

Package: 100P6S-A

Figure 1.1.3 Pin Configuration (Top View)

Pin Description

Tables 1.1.3 and 1.1.4 list the pin descriptions.

Table 1.1.3 Pin Description (1)

Pin name	Signal name	I/O type	Function
V_{CC1}, V_{CC2} V_{SS}	Power supply input		Apply 4.2 V to 5.5 V to the V_{CC1} and V_{CC2} pins and 0 V to the V_{SS} pin. The V_{CC} apply condition is that $V_{CC2} = V_{CC1}$.
CNV_{SS}	CNV_{SS}	Input	This pin switches between processor modes. Connect this pin to the V_{SS} pin when after a reset you want to start operation in single-chip mode (memory expansion mode) or the V_{CC1} pin when starting operation in microprocessor mode.
$\overline{RESET}$	Reset input	Input	"L" on this input resets the microcomputer.
X_{IN}	Clock input	Input	These pins are provided for the main clock generating circuit input/output. Connect a ceramic resonator or crystal between the X_{IN} and the X_{OUT} pins. To use an externally derived clock, input it to the X_{IN} pin and leave the X_{OUT} pin open.
X_{OUT}	Clock output	Output	
BYTE	External data bus width select input	Input	This pin selects the width of an external data bus. A 16-bit width is selected when this input is "L"; an 8-bit width is selected when this input is "H". This input must be fixed to either "H" or "L". Connect this pin to the V_{SS} pin when operating in single-chip mode.
AV_{CC}	Analog power supply input		This pin is a power supply input for the A-D converter. Connect this pin to V_{CC1}.
AV_{SS}	Analog power supply input		This pin is a power supply input for the A-D converter. Connect this pin to V_{SS}.
V_{REF}	Reference voltage input	Input	This pin is a reference voltage input for the A-D converter and D-A converter.
$P0_0$ to $P0_7$	I/O port P0	Input/output	This is an 8-bit CMOS I/O port. This port has an input/output select direction register, allowing each pin in that port to be directed for input or output individually. If any port is set for input, selection can be made for it in a program whether or not to have a pull-up resistor in 4-bit unit. This selection is unavailable in memory expansion and microprocessor modes. This port can function as input pins for the A-D converter when so selected in a program.
D_0 to D_7		Input/output	When set as a separate bus, these pins input and output data (D_0 to D_7).
$P1_0$ to $P1_7$	I/O port P1	Input/output	This is an 8-bit I/O port equivalent to P0. $P1_5$ to $P1_7$ also function as $\overline{INT}$ interrupt input pins as selected by a program.
D_8 to D_{15}		Input/output	When set as a separate bus, these pins input and output data (D_8 to D_{15}).
$P2_0$ to $P2_7$	I/O port P2	Input/output	This is an 8-bit I/O port equivalent to P0. This port can function as input pins for the A-D converter when so selected in a program.
A_0 to A_7		Output	These pins output 8 low-order address bits (A_0 to A_7).
A_0/D_0 to A_7/D_7		Input/output	If the external bus is set as an 8-bit width multiplexed bus, these pins input and output data (D_0 to D_7) and output 8 low-order address bits (A_0 to A_7) separated in time by multiplexing.
A_0, A_1/D_0 to A_7/D_6		Output Input/output	If the external bus is set as a 16-bit width multiplexed bus, these pins input and output data (D_0 to D_6) and output address (A_1 to A_7) separated in time by multiplexing. They also output address (A_0).
$P3_0$ to $P3_7$	I/O port P3	Input/output	This is an 8-bit I/O port equivalent to P0.
A_8 to A_{15}		Output	These pins output 8 middle-order address bits (A_8 to A_{15}).
A_8/D_7, A_9 to A_{15}		Input/output Output	If the external bus is set as a 16-bit width multiplexed bus, these pins input and output data (D_7) and output address (A_8) separated in time by multiplexing. They also output address (A_9 to A_{15}).
$P4_0$ to $P4_7$	I/O port P4	Input/output	This is an 8-bit I/O port equivalent to P0.
A_{16} to A_{19}, $\overline{CS0}$ to $\overline{CS3}$		Output Output	These pins output A_{16} to A_{19} and $\overline{CS0}$ to $\overline{CS3}$ signals. A_{16} to A_{19} are 4 high-order address bits. $\overline{CS0}$ to $\overline{CS3}$ are chip select signals used to specify an access space.

Table 1.1.4 Pin Description (2)

Pin name	Signal name	I/O type	Function
$P5_0$ to $P5_7$	I/O port P5	Input/output	This is an 8-bit I/O port equivalent to P0. In single-chip mode, $P5_7$ in this port outputs a divide-by-8 or divide-by-32 clock of X_{IN} or a clock of the same frequency as X_{CIN} as selected by program.
WRL / WR, WRH / BHE, RD, BCLK, HLDA, HOLD, ALE, RDY		Output Output Output Output Output Input Output Input	Output WRL/WR, WRH/BHE, RD, BCLK, HLDA, and ALE signals. WRL/WR and WRH/BHE are switchable in a program. Note that WRL and WRH are always used as a pair, so as WR and BHE. ■WRL, WRH, and RD selected If the external data bus is a 16-bit width, data are written to even addresses when the WRL signal is low, and written to odd addresses when the WRH signal is low. Data are read out when the RD signal is low. ■WR, BHE, and RD selected Data are written when the WR signal is low, or read out when the RD signal is low. Odd addresses are accessed when the BHE signal is low. Use this mode when the external data bus is an 8-bit width. The microcomputer goes to a hold state when input to the HOLD pin is held low. While in the hold state, HLDA outputs a low level. ALE is used to latch the address. While the input level of the RDY pin is low, the bus of the microcomputer goes to a wait state.
$P6_0$ to $P6_7$	I/O port P6	Input/output	This is an 8-bit I/O port equivalent to P0. Pins in this port also function as UART0 and UART1 I/O pins as selected by program.
$P7_0$ to $P7_7$	I/O port P7	Input/output	This is an 8-bit I/O port equivalent to P0 ($P7_1$ is an N channel open-drain output). This port can function as input/output pins for timers A0 to A3 when so selected in a program. Furthermore, $P7_0$ to $P7_3$, $P7_1$, $P7_2$ to $P7_5$ and $P7_6$, $P7_7$ can also function as input/output pins for UART2, an input pin for timer B5, output pins for the three-phase motor control timer, and input/output pin for the CAN1, respectively.
$P8_0$ to $P8_4$, $P8_6$, $P8_7$	I/O port P8	Input/output Input/output Input/output	$P8_0$ to $P8_4$, $P8_6$ and $P8_7$ are I/O ports with the same functions as P0. When so selected in a program, $P8_0$, $P8_1$, and $P8_2$ to $P8_4$ can function as input/output pins for timer A4 or output pins for the three-phase motor control timer and INT interrupt input pins, respectively. $P8_6$ and $P8_7$, when so selected in a program, both can function as input/output pins for the sub clock oscillator circuit. In that case, connect a crystal resonator between $P8_6$ (X_{COUT} pin) and $P8_7$ (X_{CIN} pin).
$P8_5$	Input port $P8_5$	Input	$P8_5$ is an input-only port shared with NMI. An NMI interrupt request is generated when input on this pin changes state from high to low. The NMI function cannot be disabled in a program. A pull-up cannot be set for this pin.
$P9_0$ to $P9_7$	I/O port P9	Input/output	This is an 8-bit I/O port equivalent to P0 ($P9_1$ is an N channel open-drain output). Pins in this port also function as input/output pins for SI/O3, input pins for times B0 to B4, output pins for D-A converter, and input pins for A-D converter or input/output pins for CAN0, or input pins for A-D trigger as selected by program.
$P10_0$ to $P10_7$	I/O port P10	Input/output	This is an 8-bit I/O port equivalent to P0. Pins in this port also function as input pins for A-D converter as selected by program. Furthermore, $P10_4$ to $P10_7$ also function as input pins for the key input interrupt function.

Memory

Figure 1.2.1 shows a memory map of the M16C/6N4 group. The address space extends the 1 Mbyte 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 128-Kbyte internal ROM is allocated to the addresses from $E0000_{16}$ to $FFFFF_{16}$.

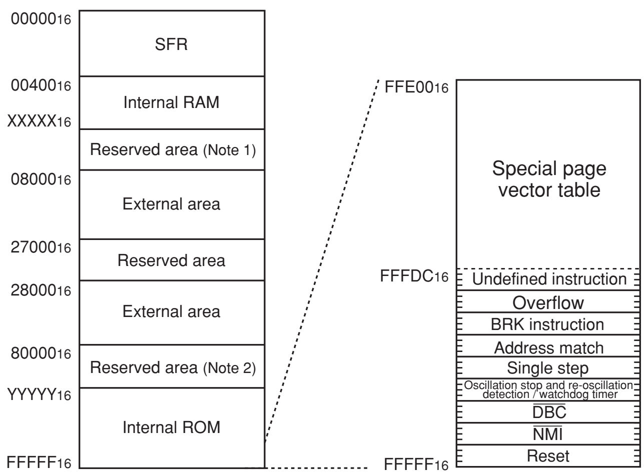
The fixed interrupt vector table is allocated to the addresses from $FFFDC_{16}$ to $FFFFF_{16}$. Therefore, store the start address of each interrupt routine here.

The internal RAM is allocated in an upper address direction beginning with address 00400_{16}. For example, a 5-Kbyte internal RAM is allocated to the addresses from 00400_{16} to $017FF_{16}$. In addition to storing data, the internal RAM also stores the stack used when calling subroutines and when interrupts are generated.

The SFR 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”.

In memory expansion and microprocessor modes, some areas are reserved for future use and cannot be used by users.



Internal RAM	
Size	Address $XXXXX_{16}$
5 Kbytes	$017FF_{16}$
10 Kbytes	$02BFF_{16}$

Internal ROM	
Size	Address $YYYYY_{16}$
128 Kbytes	$E0000_{16}$
256 Kbytes	$C0000_{16}$

Note 1: During memory expansion and microprocessor modes, can not be used.
Note 2: In memory expansion mode, can not be used.
Note 3: Shown here is a memory map for the case where the PM13 bit in the PM1 register is "0".
However, this shows the case where the PM13 bit is "1" on the devices whose internal ROM is 192 Kbytes or more.

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.

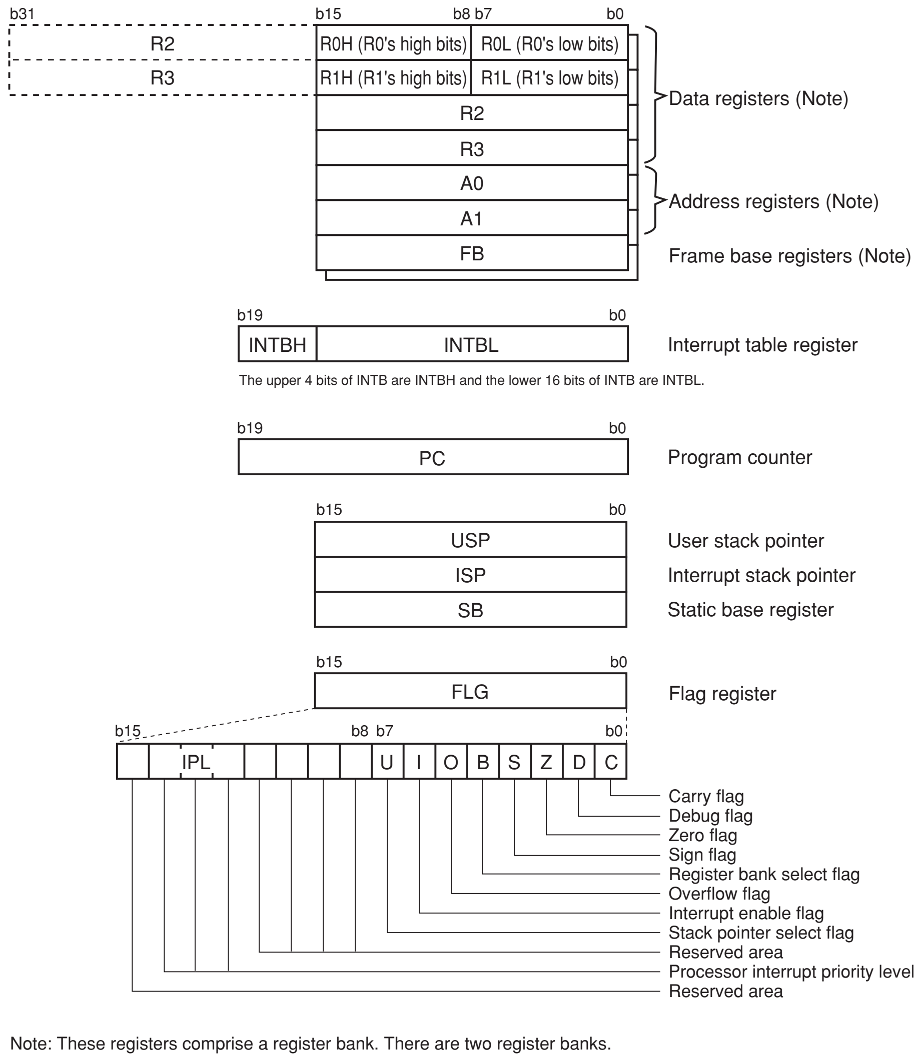


Figure 1.3.1 CPU Registers

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

In some instructions, 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), 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)**
  This 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 set 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 set 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 request is enabled.
- **Reserved Area**
  When white to this bit, write “0”. When read, its content is indeterminate.

SFR

Figures 1.4.1 to 1.4.16 show the location of peripheral function control registers and the value after reset.

Address	Register	Symbol	After reset
0000_{16}			
0001_{16}			
0002_{16}			
0003_{16}			
0004_{16}	Processor mode register 0 (Note 1)	PM0	00000000_2 (CNV$_{SS}$ pin is "L") 00000011_2 (CNV$_{SS}$ pin is "H")
0005_{16}	Processor mode register 1	PM1	$0XXX1000_2$
0006_{16}	System clock control register 0	CM0	01001000_2
0007_{16}	System clock control register 1	CM1	00100000_2
0008_{16}	Chip select control register	CSR	00000001_2
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 2)	CM2	$0X00X000_2$
$000D_{16}$			
$000E_{16}$	Watchdog timer start register	WDTS	XX_{16}
$000F_{16}$	Watchdog timer control register	WDC	$00XXXXXX_2$
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}			
$001A_{16}$			
$001B_{16}$	Chip select expansion control register	CSE	00_{16}
$001C_{16}$	PLL control register 0	PLC0	$0001X010_2$
$001D_{16}$			
$001E_{16}$	Processor mode register 2	PM2	$XXX00000_2$
$001F_{16}$			
0020_{16}	DMA0 source pointer	SAR0	XX_{16}
0021_{16}			XX_{16}
0022_{16}			XX_{16}
0023_{16}			
0024_{16}	DMA0 destination pointer	DAR0	XX_{16}
0025_{16}			XX_{16}
0026_{16}			XX_{16}
0027_{16}			
0028_{16}	DMA0 transfer counter	TCR0	XX_{16}
0029_{16}			XX_{16}
$002A_{16}$			
$002B_{16}$			
$002C_{16}$	DMA0 control register	DM0CON	$00000X00_2$
$002D_{16}$			
$002E_{16}$			
$002F_{16}$			
0030_{16}	DMA1 source pointer	SAR1	XX_{16}
0031_{16}			XX_{16}
0032_{16}			XX_{16}
0033_{16}			
0034_{16}	DMA1 destination pointer	DAR1	XX_{16}
0035_{16}			XX_{16}
0036_{16}			XX_{16}
0037_{16}			
0038_{16}	DMA1 transfer counter	TCR1	XX_{16}
0039_{16}			XX_{16}
$003A_{16}$			
$003B_{16}$			
$003C_{16}$	DMA1 control register	DM1CON	$00000X00_2$
$003D_{16}$			
$003E_{16}$			
$003F_{16}$			

X: Undefined

Note 1: The PM00 and PM01 bits do not change at software reset, watchdog timer reset and oscillation stop detection reset.
Note 2: The CM20, CM21, and CM27 bits do not change at oscillation stop detection reset.
Note 3: The blank areas are reserved and cannot be accessed by users.

Figure 1.4.1 Location of Peripheral Function Control Registers and Value at After Reset (1)

Address	Register	Symbol	After reset
0040$_{16}$			
0041$_{16}$	CAN0/1 wake up interrupt control register	C01WKIC	XXXXX000$_2$
0042$_{16}$	CAN0 successful reception interrupt control register	C0RECIC	XXXXX000$_2$
0043$_{16}$	CAN0 successful transmission interrupt control register	C0TRMIC	XXXXX000$_2$
0044$_{16}$	INT3 interrupt control register	INT3IC	XX00X000$_2$
0045$_{16}$	Timer B5 interrupt control register	TB5IC	XXXXX000$_2$
0046$_{16}$	Timer B4 interrupt control register	TB4IC	XXXXX000$_2$
	UART1 bus collision detection interrupt control register	U1BCNIC	
0047$_{16}$	Timer B3 interrupt control register	TB3IC	XXXXX000$_2$
	UART0 bus collision detection interrupt control register	U0BCNIC	
0048$_{16}$	CAN1 successful reception interrupt control register	C1RECIC	XX00X000$_2$
	INT5 interrupt control register	INT5IC	
0049$_{16}$	CAN1 successful transmission interrupt control register	C1TRMIC	XX00X000$_2$
	SI/O3 interrupt control register	S3IC	
	INT4 interrupt control register	INT4IC	
004A$_{16}$	UART2 bus collision detection interrupt control register	U2BCNIC	XXXXX000$_2$
004B$_{16}$	DMA0 interrupt control register	DM0IC	XXXXX000$_2$
004C$_{16}$	DMA1 interrupt control register	DM1IC	XXXXX000$_2$
004D$_{16}$	CAN0/1 error interrupt control register	C01ERRIC	XXXXX000$_2$
004E$_{16}$	A-D conversion interrupt control register	ADIC	XXXXX000$_2$
	Key input interrupt control register	KUPIC	
004F$_{16}$	UART2 transmit interrupt control register	S2TIC	XXXXX000$_2$
0050$_{16}$	UART2 receive interrupt control register	S2RIC	XXXXX000$_2$
0051$_{16}$	UART0 transmit interrupt control register	S0TIC	XXXXX000$_2$
0052$_{16}$	UART0 receive interrupt control register	S0RIC	XXXXX000$_2$
0053$_{16}$	UART1 transmit interrupt control register	S1TIC	XXXXX000$_2$
0054$_{16}$	UART1 receive interrupt control register	S1RIC	XXXXX000$_2$
0055$_{16}$	Timer A0 interrupt control register	TA0IC	XXXXX000$_2$
0056$_{16}$	Timer A1 interrupt control register	TA1IC	XXXXX000$_2$
0057$_{16}$	Timer A2 interrupt control register	TA2IC	XXXXX000$_2$
0058$_{16}$	Timer A3 interrupt control register	TA3IC	XXXXX000$_2$
0059$_{16}$	Timer A4 interrupt control register	TA4IC	XXXXX000$_2$
005A$_{16}$	Timer B0 interrupt control register	TB0IC	XXXXX000$_2$
005B$_{16}$	Timer B1 interrupt control register	TB1IC	XXXXX000$_2$
005C$_{16}$	Timer B2 interrupt control register	TB2IC	XXXXX000$_2$
005D$_{16}$	INT0 interrupt control register	INT0IC	XX00X000$_2$
005E$_{16}$	INT1 interrupt control register	INT1IC	XX00X000$_2$
005F$_{16}$	INT2 interrupt control register	INT2IC	XX00X000$_2$
0060$_{16}$	CAN0 message box 0: Identifier / DLC		XX$_{16}$
0061$_{16}$			XX$_{16}$
0062$_{16}$			XX$_{16}$
0063$_{16}$			XX$_{16}$
0064$_{16}$			XX$_{16}$
0065$_{16}$			XX$_{16}$
0066$_{16}$	CAN0 message box 0: Data field		XX$_{16}$
0067$_{16}$			XX$_{16}$
0068$_{16}$			XX$_{16}$
0069$_{16}$			XX$_{16}$
006A$_{16}$			XX$_{16}$
006B$_{16}$			XX$_{16}$
006C$_{16}$			XX$_{16}$
006D$_{16}$			XX$_{16}$
006E$_{16}$	CAN0 message box 0: Time stamp		XX$_{16}$
006F$_{16}$			XX$_{16}$
0070$_{16}$	CAN0 message box 1: Identifier / DLC		XX$_{16}$
0071$_{16}$			XX$_{16}$
0072$_{16}$			XX$_{16}$
0073$_{16}$			XX$_{16}$
0074$_{16}$			XX$_{16}$
0075$_{16}$			XX$_{16}$
0076$_{16}$	CAN0 message box 1: data Field		XX$_{16}$
0077$_{16}$			XX$_{16}$
0078$_{16}$			XX$_{16}$
0079$_{16}$			XX$_{16}$
007A$_{16}$			XX$_{16}$
007B$_{16}$			XX$_{16}$
007C$_{16}$			XX$_{16}$
007D$_{16}$			XX$_{16}$
007E$_{16}$	CAN0 message box 1: Time stamp		XX$_{16}$
007F$_{16}$			XX$_{16}$

X: Undefined

Note: The blank area is reserved and cannot be accessed by users.

Figure 1.4.2 Location of Peripheral Function Control Registers and Value at After Reset (2)

Address	Register	Symbol	After reset
0080_{16}	CAN0 message box 2: Identifier / DLC		XX_{16}
0081_{16}			XX_{16}
0082_{16}			XX_{16}
0083_{16}			XX_{16}
0084_{16}			XX_{16}
0085_{16}			XX_{16}
0086_{16}	CAN0 message box 2: Data field		XX_{16}
0087_{16}			XX_{16}
0088_{16}			XX_{16}
0089_{16}			XX_{16}
$008A_{16}$			XX_{16}
$008B_{16}$			XX_{16}
$008C_{16}$			XX_{16}
$008D_{16}$			XX_{16}
$008E_{16}$	CAN0 message box 2: Time stamp		XX_{16}
$008F_{16}$			XX_{16}
0090_{16}	CAN0 message box 3: Identifier / DLC		XX_{16}
0091_{16}			XX_{16}
0092_{16}			XX_{16}
0093_{16}			XX_{16}
0094_{16}			XX_{16}
0095_{16}			XX_{16}
0096_{16}	CAN0 message box 3: Data field		XX_{16}
0097_{16}			XX_{16}
0098_{16}			XX_{16}
0099_{16}			XX_{16}
$009A_{16}$			XX_{16}
$009B_{16}$			XX_{16}
$009C_{16}$			XX_{16}
$009D_{16}$			XX_{16}
$009E_{16}$	CAN0 message box 3: Time stamp		XX_{16}
$009F_{16}$			XX_{16}
$00A0_{16}$	CAN0 message box 4: Identifier / DLC		XX_{16}
$00A1_{16}$			XX_{16}
$00A2_{16}$			XX_{16}
$00A3_{16}$			XX_{16}
$00A4_{16}$			XX_{16}
$00A5_{16}$			XX_{16}
$00A6_{16}$	CAN0 message box 4: Data field		XX_{16}
$00A7_{16}$			XX_{16}
$00A8_{16}$			XX_{16}
$00A9_{16}$			XX_{16}
$00AA_{16}$			XX_{16}
$00AB_{16}$			XX_{16}
$00AC_{16}$			XX_{16}
$00AD_{16}$			XX_{16}
$00AE_{16}$	CAN0 message box 4: Time stamp		XX_{16}
$00AF_{16}$			XX_{16}
$00B0_{16}$	CAN0 message box 5: Identifier / DLC		XX_{16}
$00B1_{16}$			XX_{16}
$00B2_{16}$			XX_{16}
$00B3_{16}$			XX_{16}
$00B4_{16}$			XX_{16}
$00B5_{16}$			XX_{16}
$00B6_{16}$	CAN0 message box 5: Data field		XX_{16}
$00B7_{16}$			XX_{16}
$00B8_{16}$			XX_{16}
$00B9_{16}$			XX_{16}
$00BA_{16}$			XX_{16}
$00BB_{16}$			XX_{16}
$00BC_{16}$			XX_{16}
$00BD_{16}$			XX_{16}
$00BE_{16}$	CAN0 message box 5: Time stamp		XX_{16}
$00BF_{16}$			XX_{16}

X: Undefined

Figure 1.4.3 Location of Peripheral Function Control Registers and Value at After Reset (3)

Address	Register	Symbol	After reset
$00C0_{16}$	CAN0 message box 6: Identifier / DLC		XX_{16}
$00C1_{16}$			XX_{16}
$00C2_{16}$			XX_{16}
$00C3_{16}$			XX_{16}
$00C4_{16}$			XX_{16}
$00C5_{16}$			XX_{16}
$00C6_{16}$	CAN0 message box 6: Data field		XX_{16}
$00C7_{16}$			XX_{16}
$00C8_{16}$			XX_{16}
$00C9_{16}$			XX_{16}
$00CA_{16}$			XX_{16}
$00CB_{16}$			XX_{16}
$00CC_{16}$			XX_{16}
$00CD_{16}$			XX_{16}
$00CE_{16}$	CAN0 message box 6: Time stamp		XX_{16}
$00CF_{16}$			XX_{16}
$00D0_{16}$	CAN0 message box 7: Identifier / DLC		XX_{16}
$00D1_{16}$			XX_{16}
$00D2_{16}$			XX_{16}
$00D3_{16}$			XX_{16}
$00D4_{16}$			XX_{16}
$00D5_{16}$			XX_{16}
$00D6_{16}$	CAN0 message box 7: Data field		XX_{16}
$00D7_{16}$			XX_{16}
$00D8_{16}$			XX_{16}
$00D9_{16}$			XX_{16}
$00DA_{16}$			XX_{16}
$00DB_{16}$			XX_{16}
$00DC_{16}$			XX_{16}
$00DD_{16}$			XX_{16}
$00DE_{16}$	CAN0 message box 7: Time stamp		XX_{16}
$00DF_{16}$			XX_{16}
$00E0_{16}$	CAN0 message box 8: Identifier / DLC		XX_{16}
$00E1_{16}$			XX_{16}
$00E2_{16}$			XX_{16}
$00E3_{16}$			XX_{16}
$00E4_{16}$			XX_{16}
$00E5_{16}$			XX_{16}
$00E6_{16}$	CAN0 message box 8: Data field		XX_{16}
$00E7_{16}$			XX_{16}
$00E8_{16}$			XX_{16}
$00E9_{16}$			XX_{16}
$00EA_{16}$			XX_{16}
$00EB_{16}$			XX_{16}
$00EC_{16}$			XX_{16}
$00ED_{16}$			XX_{16}
$00EE_{16}$	CAN0 message box 8: Time stamp		XX_{16}
$00EF_{16}$			XX_{16}
$00F0_{16}$	CAN0 message box 9: Identifier / DLC		XX_{16}
$00F1_{16}$			XX_{16}
$00F2_{16}$			XX_{16}
$00F3_{16}$			XX_{16}
$00F4_{16}$			XX_{16}
$00F5_{16}$			XX_{16}
$00F6_{16}$	CAN0 message box 9: Data field		XX_{16}
$00F7_{16}$			XX_{16}
$00F8_{16}$			XX_{16}
$00F9_{16}$			XX_{16}
$00FA_{16}$			XX_{16}
$00FB_{16}$			XX_{16}
$00FC_{16}$			XX_{16}
$00FD_{16}$			XX_{16}
$00FE_{16}$	CAN0 message box 9: Time stamp		XX_{16}
$00FF_{16}$			XX_{16}

X: Undefined

Figure 1.4.4 Location of Peripheral Function Control Registers and Value at After Reset (4)

Address	Register	Symbol	After reset
0100$_{16}$	CAN0 message box 10: Identifier / DLC		XX$_{16}$
0101$_{16}$			XX$_{16}$
0102$_{16}$			XX$_{16}$
0103$_{16}$			XX$_{16}$
0104$_{16}$			XX$_{16}$
0105$_{16}$			XX$_{16}$
0106$_{16}$	CAN0 message box 10: Data field		XX$_{16}$
0107$_{16}$			XX$_{16}$
0108$_{16}$			XX$_{16}$
0109$_{16}$			XX$_{16}$
010A$_{16}$			XX$_{16}$
010B$_{16}$			XX$_{16}$
010C$_{16}$			XX$_{16}$
010D$_{16}$			XX$_{16}$
010E$_{16}$	CAN0 message box 10: Time stamp		XX$_{16}$
010F$_{16}$			XX$_{16}$
0110$_{16}$	CAN0 message box 11: Identifier / DLC		XX$_{16}$
0111$_{16}$			XX$_{16}$
0112$_{16}$			XX$_{16}$
0113$_{16}$			XX$_{16}$
0114$_{16}$			XX$_{16}$
0115$_{16}$			XX$_{16}$
0116$_{16}$	CAN0 message box 11: Data field		XX$_{16}$
0117$_{16}$			XX$_{16}$
0118$_{16}$			XX$_{16}$
0119$_{16}$			XX$_{16}$
011A$_{16}$			XX$_{16}$
011B$_{16}$			XX$_{16}$
011C$_{16}$			XX$_{16}$
011D$_{16}$			XX$_{16}$
011E$_{16}$	CAN0 message box 11: Time stamp		XX$_{16}$
011F$_{16}$			XX$_{16}$
0120$_{16}$	CAN0 message box 12: Identifier / DLC		XX$_{16}$
0121$_{16}$			XX$_{16}$
0122$_{16}$			XX$_{16}$
0123$_{16}$			XX$_{16}$
0124$_{16}$			XX$_{16}$
0125$_{16}$			XX$_{16}$
0126$_{16}$	CAN0 message box 12: Data field		XX$_{16}$
0127$_{16}$			XX$_{16}$
0128$_{16}$			XX$_{16}$
0129$_{16}$			XX$_{16}$
012A$_{16}$			XX$_{16}$
012B$_{16}$			XX$_{16}$
012C$_{16}$			XX$_{16}$
012D$_{16}$			XX$_{16}$
012E$_{16}$	CAN0 message box 12: Time stamp		XX$_{16}$
012F$_{16}$			XX$_{16}$
0130$_{16}$	CAN0 message box 13: Identifier / DLC		XX$_{16}$
0131$_{16}$			XX$_{16}$
0132$_{16}$			XX$_{16}$
0133$_{16}$			XX$_{16}$
0134$_{16}$			XX$_{16}$
0135$_{16}$			XX$_{16}$
0136$_{16}$	CAN0 message box 13: Data field		XX$_{16}$
0137$_{16}$			XX$_{16}$
0138$_{16}$			XX$_{16}$
0139$_{16}$			XX$_{16}$
013A$_{16}$			XX$_{16}$
013B$_{16}$			XX$_{16}$
013C$_{16}$			XX$_{16}$
013D$_{16}$			XX$_{16}$
013E$_{16}$	CAN0 message box 13: Time stamp		XX$_{16}$
013F$_{16}$			XX$_{16}$

X: Undefined

Figure 1.4.5 Location of Peripheral Function Control Registers and Value at After Reset (5)

Address	Register	Symbol	After reset
0140_{16}	CAN0 message box 14: Identifier /DLC		XX_{16}
0141_{16}			XX_{16}
0142_{16}			XX_{16}
0143_{16}			XX_{16}
0144_{16}			XX_{16}
0145_{16}			XX_{16}
0146_{16}	CAN0 message box 14: Data field		XX_{16}
0147_{16}			XX_{16}
0148_{16}			XX_{16}
0149_{16}			XX_{16}
$014A_{16}$			XX_{16}
$014B_{16}$			XX_{16}
$014C_{16}$			XX_{16}
$014D_{16}$			XX_{16}
$014E_{16}$	CAN0 message box 14: Time stamp		XX_{16}
$014F_{16}$			XX_{16}
0150_{16}	CAN0 message box 15: Identifier /DLC		XX_{16}
0151_{16}			XX_{16}
0152_{16}			XX_{16}
0153_{16}			XX_{16}
0154_{16}			XX_{16}
0155_{16}			XX_{16}
0156_{16}	CAN0 message box 15: Data field		XX_{16}
0157_{16}			XX_{16}
0158_{16}			XX_{16}
0159_{16}			XX_{16}
$015A_{16}$			XX_{16}
$015B_{16}$			XX_{16}
$015C_{16}$			XX_{16}
$015D_{16}$			XX_{16}
$015E_{16}$	CAN0 message box 15: Time stamp		XX_{16}
$015F_{16}$			XX_{16}
0160_{16}	CAN0 global mask register	C0GMR	XX_{16}
0161_{16}			XX_{16}
0162_{16}			XX_{16}
0163_{16}			XX_{16}
0164_{16}			XX_{16}
0165_{16}			XX_{16}
0166_{16}	CAN0 local mask A register	C0LMAR	XX_{16}
0167_{16}			XX_{16}
0168_{16}			XX_{16}
0169_{16}			XX_{16}
$016A_{16}$			XX_{16}
$016B_{16}$			XX_{16}
$016C_{16}$	CAN0 local mask B register	C0LMBR	XX_{16}
$016D_{16}$			XX_{16}
$016E_{16}$			XX_{16}
$016F_{16}$			XX_{16}
0170_{16}			XX_{16}
0171_{16}			XX_{16}
0172_{16}			
0173_{16}			
0174_{16}			
0175_{16}			
0176_{16}			
0177_{16}			
0178_{16}			
0179_{16}			
$017A_{16}$			
$017B_{16}$			
$017C_{16}$			
$017D_{16}$			
$017E_{16}$			
$017F_{16}$			

X: Undefined

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

Figure 1.4.6 Location of Peripheral Function Control Registers and Value at After Reset (6)

Address	Register	Symbol	After reset
0180_{16}			
0181_{16}			
0182_{16}			
0183_{16}			
0184_{16}			
0185_{16}			
0186_{16}			
0187_{16}			
0188_{16}			
0189_{16}			
$018A_{16}$			
$018B_{16}$			
$018C_{16}$			
$018D_{16}$			
$018E_{16}$			
$018F_{16}$			
0190_{16}			
0191_{16}			
0192_{16}			
0193_{16}			
0194_{16}			
0195_{16}			
0196_{16}			
0197_{16}			
0198_{16}			
0199_{16}			
$019A_{16}$			
$019B_{16}$			
$019C_{16}$			
$019D_{16}$			
$019E_{16}$			
$019F_{16}$			
$01A0_{16}$			
$01A1_{16}$			
$01A2_{16}$			
$01A3_{16}$			
$01A4_{16}$			
$01A5_{16}$			
$01A6_{16}$			
$01A7_{16}$			
$01A8_{16}$			
$01A9_{16}$			
$01AA_{16}$			
$01AB_{16}$			
$01AC_{16}$			
$01AD_{16}$			
$01AE_{16}$			
$01AF_{16}$			
$01B0_{16}$			
$01B1_{16}$			
$01B2_{16}$			
$01B3_{16}$			
$01B4_{16}$			
$01B5_{16}$	Flash memory control register 1 (Note 1)	FMR1	$0X00XX0X_2$
$01B6_{16}$			
$01B7_{16}$	Flash memory control register 0 (Note 1)	FMR0	$XX000001_2$
$01B8_{16}$	Address match interrupt register 2	RAMD2	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	RAMD3	00_{16}
$01BD_{16}$			00_{16}
$01BE_{16}$			$X0_{16}$
$01BF_{16}$			

X: Undefined

Note 1: This register is included in flash memory version.
Note 2: The blank areas are reserved and cannot be accessed by users.

Figure 1.4.7 Location of Peripheral Function Control Registers and Value at After Reset (7)

RENESAS

Address	Register	Symbol	After reset
$01C0_{16}$	Timer B3,4,5 count start flag	TBSR	$000XXXXX_2$
$01C1_{16}$			
$01C2_{16}$	Timer A1-1 register	TA11	XX_{16}
$01C3_{16}$			XX_{16}
$01C4_{16}$	Timer A2-1 register	TA21	XX_{16}
$01C5_{16}$			XX_{16}
$01C6_{16}$	Timer A4-1 register	TA41	XX_{16}
$01C7_{16}$			XX_{16}
$01C8_{16}$	Three-phase PWM control register 0	INVC0	00_{16}
$01C9_{16}$	Three-phase PWM control register 1	INVC1	00_{16}
$01CA_{16}$	Three-phase output buffer register 0	IDB0	00_{16}
$01CB_{16}$	Three-phase output buffer register 1	IDB1	00_{16}
$01CC_{16}$	Dead time timer	DTT	XX_{16}
$01CD_{16}$	Timer B2 interrupt occurrence frequency set counter	ICTB2	XX_{16}
$01CE_{16}$			
$01CF_{16}$			
$01D0_{16}$	Timer B3 register	TB3	XX_{16}
$01D1_{16}$			XX_{16}
$01D2_{16}$	Timer B4 register	TB4	XX_{16}
$01D3_{16}$			XX_{16}
$01D4_{16}$	Timer B5 register	TB5	XX_{16}
$01D5_{16}$			XX_{16}
$01D6_{16}$			
$01D7_{16}$			
$01D8_{16}$			
$01D9_{16}$			
$01DA_{16}$			
$01DB_{16}$	Timer B3 mode register	TB3MR	$00XX0000_2$
$01DC_{16}$	Timer B4 mode register	TB4MR	$00XX0000_2$
$01DD_{16}$	Timer B5 mode register	TB5MR	$00XX0000_2$
$01DE_{16}$	Interrupt cause select register 0	IFSR0	$00XXX000_2$
$01DF_{16}$	Interrupt cause select register 1	IFSR1	00_{16}
$01E0_{16}$	SI/O3 transmit/receive register	S3TRR	XX_{16}
$01E1_{16}$			
$01E2_{16}$	SI/O3 control register	S3C	01000000_2
$01E3_{16}$	SI/O3 bit rate generator	S3BRG	XX_{16}
$01E4_{16}$			
$01E5_{16}$			
$01E6_{16}$			
$01E7_{16}$			
$01E8_{16}$			
$01E9_{16}$			
$01EA_{16}$			
$01EB_{16}$			
$01EC_{16}$	UART0 special mode register 4	U0SMR4	00_{16}
$01ED_{16}$	UART0 special mode register 3	U0SMR3	$000X0X0X_2$
$01EE_{16}$	UART0 special mode register 2	U0SMR2	$X0000000_2$
$01EF_{16}$	UART0 special mode register	U0SMR	$X0000000_2$
$01F0_{16}$	UART1 special mode register 4	U1SMR4	00_{16}
$01F1_{16}$	UART1 special mode register 3	U1SMR3	$000X0X0X_2$
$01F2_{16}$	UART1 special mode register 2	U1SMR2	$X0000000_2$
$01F3_{16}$	UART1 special mode register	U1SMR	$X0000000_2$
$01F4_{16}$	UART2 special mode register 4	U2SMR4	00_{16}
$01F5_{16}$	UART2 special mode register 3	U2SMR3	$000X0X0X_2$
$01F6_{16}$	UART2 special mode register 2	U2SMR2	$X0000000_2$
$01F7_{16}$	UART2 special mode register	U2SMR	$X0000000_2$
$01F8_{16}$	UART2 transmit/receive mode register	U2MR	00_{16}
$01F9_{16}$	UART2 bit rate generator	U2BRG	XX_{16}
$01FA_{16}$	UART2 transmit buffer register	U2TB	XX_{16}
$01FB_{16}$			XX_{16}
$01FC_{16}$	UART2 transmit/receive mode register 0	U2C0	00001000_2
$01FD_{16}$	UART2 transmit/receive mode register 1	U2C1	00000010_2
$01FE_{16}$	UART2 receive buffer register	U2RB	XX_{16}
$01FF_{16}$			XX_{16}

X: Undefined

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

Figure 1.4.8 Location of Peripheral Function Control Registers and Value at After Reset (8)

Address	Register	Symbol	After reset
0200_{16}	CAN0 message control register 0	C0MCTL0	00_{16}
0201_{16}	CAN0 message control register 1	C0MCTL1	00_{16}
0202_{16}	CAN0 message control register 2	C0MCTL2	00_{16}
0203_{16}	CAN0 message control register 3	C0MCTL3	00_{16}
0204_{16}	CAN0 message control register 4	C0MCTL4	00_{16}
0205_{16}	CAN0 message control register 5	C0MCTL5	00_{16}
0206_{16}	CAN0 message control register 6	C0MCTL6	00_{16}
0207_{16}	CAN0 message control register 7	C0MCTL7	00_{16}
0208_{16}	CAN0 message control register 8	C0MCTL8	00_{16}
0209_{16}	CAN0 message control register 9	C0MCTL9	00_{16}
$020A_{16}$	CAN0 message control register 10	C0MCTL10	00_{16}
$020B_{16}$	CAN0 message control register 11	C0MCTL11	00_{16}
$020C_{16}$	CAN0 message control register 12	C0MCTL12	00_{16}
$020D_{16}$	CAN0 message control register 13	C0MCTL13	00_{16}
$020E_{16}$	CAN0 message control register 14	C0MCTL14	00_{16}
$020F_{16}$	CAN0 message control register 15	C0MCTL15	00_{16}
0210_{16}	CAN0 control register	C0CTLR	$X0000001_2$
0211_{16}			$XX0X0000_2$
0212_{16}	CAN0 status register	C0STR	00_{16}
0213_{16}			$X0000001_2$
0214_{16}	CAN0 slot status register	C0SSTR	00_{16}
0215_{16}			00_{16}
0216_{16}	CAN0 interrupt control register	C0ICR	00_{16}
0217_{16}			00_{16}
0218_{16}	CAN0 extended register	C0IDR	00_{16}
0219_{16}			00_{16}
$021A_{16}$	CAN0 configuration register	C0CONR	XX_{16}
$021B_{16}$			XX_{16}
$021C_{16}$	CAN0 receive error count register	C0RECR	00_{16}
$021D_{16}$	CAN0 transmit error count register	C0TECR	00_{16}
$021E_{16}$	CAN0 time stamp register	C0TSR	00_{16}
$021F_{16}$			00_{16}
0220_{16}	CAN1 message control register 0	C1MCTL0	00_{16}
0221_{16}	CAN1 message control register 1	C1MCTL1	00_{16}
0222_{16}	CAN1 message control register 2	C1MCTL2	00_{16}
0223_{16}	CAN1 message control register 3	C1MCTL3	00_{16}
0224_{16}	CAN1 message control register 4	C1MCTL4	00_{16}
0225_{16}	CAN1 message control register 5	C1MCTL5	00_{16}
0226_{16}	CAN1 message control register 6	C1MCTL6	00_{16}
0227_{16}	CAN1 message control register 7	C1MCTL7	00_{16}
0228_{16}	CAN1 message control register 8	C1MCTL8	00_{16}
0229_{16}	CAN1 message control register 9	C1MCTL9	00_{16}
$022A_{16}$	CAN1 message control register 10	C1MCTL10	00_{16}
$022B_{16}$	CAN1 message control register 11	C1MCTL11	00_{16}
$022C_{16}$	CAN1 message control register 12	C1MCTL12	00_{16}
$022D_{16}$	CAN1 message control register 13	C1MCTL13	00_{16}
$022E_{16}$	CAN1 message control register 14	C1MCTL14	00_{16}
$022F_{16}$	CAN1 message control register 15	C1MCTL15	00_{16}
0230_{16}	CAN1 control register	C1CTLR	$X0000001_2$
0231_{16}			$XX0X0000_2$
0232_{16}	CAN1 status register	C1STR	00_{16}
0233_{16}			$X0000001_2$
0234_{16}	CAN1 slot status register	C1SSTR	00_{16}
0235_{16}			00_{16}
0236_{16}	CAN1 interrupt control register	C1ICR	00_{16}
0237_{16}			00_{16}
0238_{16}	CAN1 extended register	C1IDR	00_{16}
0239_{16}			00_{16}
$023A_{16}$	CAN1 configuration register	C1CONR	XX_{16}
$023B_{16}$			XX_{16}
$023C_{16}$	CAN1 receive error count register	C1RECR	00_{16}
$023D_{16}$	CAN1 transmit error count register	C1TECR	00_{16}
$023E_{16}$	CAN1 time stamp register	C1TSR	00_{16}
$023F_{16}$			00_{16}

X: Undefined

Figure 1.4.9 Location of Peripheral Function Control Registers and Value at After Reset (9)

Address	Register	Symbol	After reset
0240₁₆			
0241₁₆			
0242₁₆	CAN0 acceptance filter support register	C0AFS	XX₁₆
0243₁₆			XX₁₆
0244₁₆	CAN1 acceptance filter support register	C1AFS	XX₁₆
0245₁₆			XX₁₆
0246₁₆			
0247₁₆			
0248₁₆			
0249₁₆			
024A₁₆			
024B₁₆			
024C₁₆			
024D₁₆			
024E₁₆			
024F₁₆			
0250₁₆			
0251₁₆			
0252₁₆			
0253₁₆			
0254₁₆			
0255₁₆			
0256₁₆			
0257₁₆			
0258₁₆			
0259₁₆			
025A₁₆			
025B₁₆			
025C₁₆			
025D₁₆			
025E₁₆	Peripheral function clock select register	PCLKR	00₁₆
025F₁₆	CAN0/1 clock select register	CCLKR	00₁₆
0260₁₆	CAN1 message box 0: Identifier / DLC		XX₁₆
0261₁₆			XX₁₆
0262₁₆			XX₁₆
0263₁₆			XX₁₆
0264₁₆			XX₁₆
0265₁₆			XX₁₆
0266₁₆	CAN1 message box 0: Data field		XX₁₆
0267₁₆			XX₁₆
0268₁₆			XX₁₆
0269₁₆			XX₁₆
026A₁₆			XX₁₆
026B₁₆			XX₁₆
026C₁₆			XX₁₆
026D₁₆			XX₁₆
026E₁₆	CAN1 message box 0:Time stamp		XX₁₆
026F₁₆			XX₁₆
0270₁₆	CAN1 message box 1: Identifier / DLC		XX₁₆
0271₁₆			XX₁₆
0272₁₆			XX₁₆
0273₁₆			XX₁₆
0274₁₆			XX₁₆
0275₁₆			XX₁₆
0276₁₆	CAN1 message box 1: Data field		XX₁₆
0277₁₆			XX₁₆
0278₁₆			XX₁₆
0279₁₆			XX₁₆
027A₁₆			XX₁₆
027B₁₆			XX₁₆
027C₁₆			XX₁₆
027D₁₆			XX₁₆
027E₁₆	CAN1 message box 1:Time stamp		XX₁₆
027F₁₆			XX₁₆

X: Undefined

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

Figure 1.4.10 Location of Peripheral Function Control Registers and Value at After Reset (10)

Address	Register	Symbol	After reset
0280_{16}	CAN1 message box 2: Identifier / DLC		XX_{16}
0281_{16}			XX_{16}
0282_{16}			XX_{16}
0283_{16}			XX_{16}
0284_{16}			XX_{16}
0285_{16}			XX_{16}
0286_{16}	CAN1 message box 2: Data field		XX_{16}
0287_{16}			XX_{16}
0288_{16}			XX_{16}
0289_{16}			XX_{16}
$028A_{16}$			XX_{16}
$028B_{16}$			XX_{16}
$028C_{16}$			XX_{16}
$028D_{16}$			XX_{16}
$028E_{16}$	CAN1 message box 2: Time stamp		XX_{16}
$028F_{16}$			XX_{16}
0290_{16}	CAN1 message box 3: Identifier / DLC		XX_{16}
0291_{16}			XX_{16}
0292_{16}			XX_{16}
0293_{16}			XX_{16}
0294_{16}			XX_{16}
0295_{16}			XX_{16}
0296_{16}	CAN1 message box 3: Data field		XX_{16}
0297_{16}			XX_{16}
0298_{16}			XX_{16}
0299_{16}			XX_{16}
$029A_{16}$			XX_{16}
$029B_{16}$			XX_{16}
$029C_{16}$			XX_{16}
$029D_{16}$			XX_{16}
$029E_{16}$	CAN1 message box 3: Time stamp		XX_{16}
$029F_{16}$			XX_{16}
$02A0_{16}$	CAN1 message box 4: Identifier / DLC		XX_{16}
$02A1_{16}$			XX_{16}
$02A2_{16}$			XX_{16}
$02A3_{16}$			XX_{16}
$02A4_{16}$			XX_{16}
$02A5_{16}$			XX_{16}
$02A6_{16}$	CAN1 message box 4: Data field		XX_{16}
$02A7_{16}$			XX_{16}
$02A8_{16}$			XX_{16}
$02A9_{16}$			XX_{16}
$02AA_{16}$			XX_{16}
$02AB_{16}$			XX_{16}
$02AC_{16}$			XX_{16}
$02AD_{16}$			XX_{16}
$02AE_{16}$	CAN1 message box 4: Time stamp		XX_{16}
$02AF_{16}$			XX_{16}
$02B0_{16}$	CAN1 message box 5: Identifier / DLC		XX_{16}
$02B1_{16}$			XX_{16}
$02B2_{16}$			XX_{16}
$02B3_{16}$			XX_{16}
$02B4_{16}$			XX_{16}
$02B5_{16}$			XX_{16}
$02B6_{16}$	CAN1 message box 5: Data field		XX_{16}
$02B7_{16}$			XX_{16}
$02B8_{16}$			XX_{16}
$02B9_{16}$			XX_{16}
$02BA_{16}$			XX_{16}
$02BB_{16}$			XX_{16}
$02BC_{16}$			XX_{16}
$02BD_{16}$			XX_{16}
$02BE_{16}$	CAN1 message box 5: Time stamp		XX_{16}
$02BF_{16}$			XX_{16}

X: Undefined

Figure 1.4.11 Location of Peripheral Function Control Registers and Value at After Reset (11)

Address	Register	Symbol	After reset
02C0$_{16}$	CAN1 message box 6: Identifier / DLC		XX$_{16}$
02C1$_{16}$			XX$_{16}$
02C2$_{16}$			XX$_{16}$
02C3$_{16}$			XX$_{16}$
02C4$_{16}$			XX$_{16}$
02C5$_{16}$			XX$_{16}$
02C6$_{16}$	CAN1 message box 6: Data field		XX$_{16}$
02C7$_{16}$			XX$_{16}$
02C8$_{16}$			XX$_{16}$
02C9$_{16}$			XX$_{16}$
02CA$_{16}$			XX$_{16}$
02CB$_{16}$			XX$_{16}$
02CC$_{16}$			XX$_{16}$
02CD$_{16}$			XX$_{16}$
02CE$_{16}$	CAN1 message box 6: Time stamp		XX$_{16}$
02CF$_{16}$			XX$_{16}$
02D0$_{16}$	CAN1 message box 7: Identifier / DLC		XX$_{16}$
02D1$_{16}$			XX$_{16}$
02D2$_{16}$			XX$_{16}$
02D3$_{16}$			XX$_{16}$
02D4$_{16}$			XX$_{16}$
02D5$_{16}$			XX$_{16}$
02D6$_{16}$	CAN1 message box 7: Data field		XX$_{16}$
02D7$_{16}$			XX$_{16}$
02D8$_{16}$			XX$_{16}$
02D9$_{16}$			XX$_{16}$
02DA$_{16}$			XX$_{16}$
02DB$_{16}$			XX$_{16}$
02DC$_{16}$			XX$_{16}$
02DD$_{16}$			XX$_{16}$
02DE$_{16}$	CAN1 message box 7: Time stamp		XX$_{16}$
02DF$_{16}$			XX$_{16}$
02E0$_{16}$	CAN1 message box 8: Identifier / DLC		XX$_{16}$
02E1$_{16}$			XX$_{16}$
02E2$_{16}$			XX$_{16}$
02E3$_{16}$			XX$_{16}$
02E4$_{16}$			XX$_{16}$
02E5$_{16}$			XX$_{16}$
02E6$_{16}$	CAN1 message box 8: Data field		XX$_{16}$
02E7$_{16}$			XX$_{16}$
02E8$_{16}$			XX$_{16}$
02E9$_{16}$			XX$_{16}$
02EA$_{16}$			XX$_{16}$
02EB$_{16}$			XX$_{16}$
02EC$_{16}$			XX$_{16}$
02ED$_{16}$			XX$_{16}$
02EE$_{16}$	CAN1 message box 8: Time stamp		XX$_{16}$
02EF$_{16}$			XX$_{16}$
02F0$_{16}$	CAN1 message box 9: Identifier / DLC		XX$_{16}$
02F1$_{16}$			XX$_{16}$
02F2$_{16}$			XX$_{16}$
02F3$_{16}$			XX$_{16}$
02F4$_{16}$			XX$_{16}$
02F5$_{16}$			XX$_{16}$
02F6$_{16}$	CAN1 message box 9: Data field		XX$_{16}$
02F7$_{16}$			XX$_{16}$
02F8$_{16}$			XX$_{16}$
02F9$_{16}$			XX$_{16}$
02FA$_{16}$			XX$_{16}$
02FB$_{16}$			XX$_{16}$
02FC$_{16}$			XX$_{16}$
02FD$_{16}$			XX$_{16}$
02FE$_{16}$	CAN1 message box 9: Time stamp		XX$_{16}$
02FF$_{16}$			XX$_{16}$

X: Undefined

Figure 1.4.12 Location of Peripheral Function Control Registers and Value at After Reset (12)

Address	Register	Symbol	After reset
0300_{16}	CAN1 message box 10: Identifier / DLC		XX_{16}
0301_{16}			XX_{16}
0302_{16}			XX_{16}
0303_{16}			XX_{16}
0304_{16}			XX_{16}
0305_{16}			XX_{16}
0306_{16}	CAN1 message box 10: Data field		XX_{16}
0307_{16}			XX_{16}
0308_{16}			XX_{16}
0309_{16}			XX_{16}
$030A_{16}$			XX_{16}
$030B_{16}$			XX_{16}
$030C_{16}$			XX_{16}
$030D_{16}$			XX_{16}
$030E_{16}$	CAN1 message box 10: Time stamp		XX_{16}
$030F_{16}$			XX_{16}
0310_{16}	CAN1 message box 11: Identifier / DLC		XX_{16}
0311_{16}			XX_{16}
0312_{16}			XX_{16}
0313_{16}			XX_{16}
0314_{16}			XX_{16}
0315_{16}			XX_{16}
0316_{16}	CAN1 message box 11: Data field		XX_{16}
0317_{16}			XX_{16}
0318_{16}			XX_{16}
0319_{16}			XX_{16}
$031A_{16}$			XX_{16}
$031B_{16}$			XX_{16}
$031C_{16}$			XX_{16}
$031D_{16}$			XX_{16}
$031E_{16}$	CAN1 message box 11: Time stamp		XX_{16}
$031F_{16}$			XX_{16}
0320_{16}	CAN1 message box 12: Identifier / DLC		XX_{16}
0321_{16}			XX_{16}
0322_{16}			XX_{16}
0323_{16}			XX_{16}
0324_{16}			XX_{16}
0325_{16}			XX_{16}
0326_{16}	CAN1 message box 12: Data field		XX_{16}
0327_{16}			XX_{16}
0328_{16}			XX_{16}
0329_{16}			XX_{16}
$032A_{16}$			XX_{16}
$032B_{16}$			XX_{16}
$032C_{16}$			XX_{16}
$032D_{16}$			XX_{16}
$032E_{16}$	CAN1 message box 12: Time stamp		XX_{16}
$032F_{16}$			XX_{16}
0330_{16}	CAN1 message box 13: Identifier / DLC		XX_{16}
0331_{16}			XX_{16}
0332_{16}			XX_{16}
0333_{16}			XX_{16}
0334_{16}			XX_{16}
0335_{16}			XX_{16}
0336_{16}	CAN1 message box 13: Data field		XX_{16}
0337_{16}			XX_{16}
0338_{16}			XX_{16}
0339_{16}			XX_{16}
$033A_{16}$			XX_{16}
$033B_{16}$			XX_{16}
$033C_{16}$			XX_{16}
$033D_{16}$			XX_{16}
$033E_{16}$	CAN1 message box 13: Time stamp		XX_{16}
$033F_{16}$			XX_{16}

X: Undefined

Figure 1.4.13 Location of Peripheral Function Control Registers and Value at After Reset (13)

Address	Register	Symbol	After reset
0340_{16}	CAN1 message box 14: Identifier / DLC		XX_{16}
0341_{16}			XX_{16}
0342_{16}			XX_{16}
0343_{16}			XX_{16}
0344_{16}			XX_{16}
0345_{16}			XX_{16}
0346_{16}	CAN1 message box 14: Data field		XX_{16}
0347_{16}			XX_{16}
0348_{16}			XX_{16}
0349_{16}			XX_{16}
$034A_{16}$			XX_{16}
$034B_{16}$			XX_{16}
$034C_{16}$			XX_{16}
$034D_{16}$			XX_{16}
$034E_{16}$	CAN1 message box 14: Time stamp		XX_{16}
$034F_{16}$			XX_{16}
0350_{16}	CAN1 message box 15: Identifier / DLC		XX_{16}
0351_{16}			XX_{16}
0352_{16}			XX_{16}
0353_{16}			XX_{16}
0354_{16}			XX_{16}
0355_{16}			XX_{16}
0356_{16}	CAN1 message box 15: Data field		XX_{16}
0357_{16}			XX_{16}
0358_{16}			XX_{16}
0359_{16}			XX_{16}
$035A_{16}$			XX_{16}
$035B_{16}$			XX_{16}
$035C_{16}$			XX_{16}
$035D_{16}$			XX_{16}
$035E_{16}$	CAN1 message box 15: Time stamp		XX_{16}
$035F_{16}$			XX_{16}
0360_{16}	CAN1 global mask register	C1GMR	XX_{16}
0361_{16}			XX_{16}
0362_{16}			XX_{16}
0363_{16}			XX_{16}
0364_{16}			XX_{16}
0365_{16}			XX_{16}
0366_{16}	CAN1 local mask A register	C1LMAR	XX_{16}
0367_{16}			XX_{16}
0368_{16}			XX_{16}
0369_{16}			XX_{16}
$036A_{16}$			XX_{16}
$036B_{16}$			XX_{16}
$036C_{16}$	CAN1 local mask B register	C1LMBR	XX_{16}
$036D_{16}$			XX_{16}
$036E_{16}$			XX_{16}
$036F_{16}$			XX_{16}
0370_{16}			XX_{16}
0371_{16}			XX_{16}
0372_{16}			
0373_{16}			
0374_{16}			
0375_{16}			
0376_{16}			
0377_{16}			
0378_{16}			
0379_{16}			
$037A_{16}$			
$037B_{16}$			
$037C_{16}$			
$037D_{16}$			
$037E_{16}$			
$037F_{16}$			

X: Undefined

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

Figure 1.4.14 Location of Peripheral Function Control Registers and Value at After Reset (14)

Address	Register	Symbol	After reset
0380_{16}	Count start flag	TABSR	00_{16}
0381_{16}	Clock prescaler reset flag	CPSRF	$0XXXXXXX_2$
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}	Timer A0 register	TA0	XX_{16}
0387_{16}			XX_{16}
0388_{16}	Timer A1 register	TA1	XX_{16}
0389_{16}			XX_{16}
$038A_{16}$	Timer A2 register	TA2	XX_{16}
$038B_{16}$			XX_{16}
$038C_{16}$	Timer A3 register	TA3	XX_{16}
$038D_{16}$			XX_{16}
$038E_{16}$	Timer A4 register	TA4	XX_{16}
$038F_{16}$			XX_{16}
0390_{16}	Timer B0 register	TB0	XX_{16}
0391_{16}			XX_{16}
0392_{16}	Timer B1 register	TB1	XX_{16}
0393_{16}			XX_{16}
0394_{16}	Timer B2 register	TB2	XX_{16}
0395_{16}			XX_{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}
$039B_{16}$	Timer B0 mode register	TB0MR	$00XX0000_2$
$039C_{16}$	Timer B1 mode register	TB1MR	$00XX0000_2$
$039D_{16}$	Timer B2 mode register	TB2MR	$00XX0000_2$
$039E_{16}$	Timer B2 special mode register	TB2SC	$XXXXXX00_2$
$039F_{16}$			
$03A0_{16}$	UART0 transmit/receive mode register	U0MR	00_{16}
$03A1_{16}$	UART0 bit rate generator	U0BRG	XX_{16}
$03A2_{16}$	UART0 transmit buffer register	U0TB	XX_{16}
$03A3_{16}$			XX_{16}
$03A4_{16}$	UART0 transmit/receive control register 0	U0C0	00001000_2
$03A5_{16}$	UART0 transmit/receive control register 1	U0C1	00000010_2
$03A6_{16}$	UART0 receive buffer register	U0RB	XX_{16}
$03A7_{16}$			XX_{16}
$03A8_{16}$	UART1 transmit/receive mode register	U1MR	00_{16}
$03A9_{16}$	UART1 bit rate generator	U1BRG	XX_{16}
$03AA_{16}$	UART1 transmit buffer register	U1TB	XX_{16}
$03AB_{16}$			XX_{16}
$03AC_{16}$	UART1 transmit/receive control register 0	U1C0	00001000_2
$03AD_{16}$	UART1 transmit/receive control register 1	U1C1	00000010_2
$03AE_{16}$	UART1 receive buffer register	U1RB	XX_{16}
$03AF_{16}$			XX_{16}
$03B0_{16}$	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}
$03BB_{16}$			
$03BC_{16}$	CRC data register	CRCD	XX_{16}
$03BD_{16}$			XX_{16}
$03BE_{16}$	CRC input register	CRCIN	XX_{16}
$03BF_{16}$			

X: Undefined

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

Figure 1.4.15 Location of Peripheral Function Control Registers and Value at After Reset (15)

Address	Register	Symbol	After reset
$03C0_{16}$	A-D register 0	AD0	XX_{16}
$03C1_{16}$			XX_{16}
$03C2_{16}$	A-D register 1	AD1	XX_{16}
$03C3_{16}$			XX_{16}
$03C4_{16}$	A-D register 2	AD2	XX_{16}
$03C5_{16}$			XX_{16}
$03C6_{16}$	A-D register 3	AD3	XX_{16}
$03C7_{16}$			XX_{16}
$03C8_{16}$	A-D register 4	AD4	XX_{16}
$03C9_{16}$			XX_{16}
$03CA_{16}$	A-D register 5	AD5	XX_{16}
$03CB_{16}$			XX_{16}
$03CC_{16}$	A-D register 6	AD6	XX_{16}
$03CD_{16}$			XX_{16}
$03CE_{16}$	A-D register 7	AD7	XX_{16}
$03CF_{16}$			XX_{16}
$03D0_{16}$			
$03D1_{16}$			
$03D2_{16}$			
$03D3_{16}$			
$03D4_{16}$	A-D control register 2	ADCON2	00_{16}
$03D5_{16}$			
$03D6_{16}$	A-D control register 0	ADCON0	$00000XXX_2$
$03D7_{16}$	A-D control register 1	ADCON1	00_{16}
$03D8_{16}$	D-A register 0	DA0	XX_{16}
$03D9_{16}$			
$03DA_{16}$	D-A register 1	DA1	XX_{16}
$03DB_{16}$			
$03DC_{16}$	D-A control register	DACON	00_{16}
$03DD_{16}$			
$03DE_{16}$			
$03DF_{16}$			
$03E0_{16}$	Port P0 register	P0	XX_{16}
$03E1_{16}$	Port P1 register	P1	XX_{16}
$03E2_{16}$	Port P0 direction register	PD0	00_{16}
$03E3_{16}$	Port P1 direction register	PD1	00_{16}
$03E4_{16}$	Port P2 register	P2	XX_{16}
$03E5_{16}$	Port P3 register	P3	XX_{16}
$03E6_{16}$	Port P2 direction register	PD2	00_{16}
$03E7_{16}$	Port P3 direction register	PD3	00_{16}
$03E8_{16}$	Port P4 register	P4	XX_{16}
$03E9_{16}$	Port P5 register	P5	XX_{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	XX_{16}
$03ED_{16}$	Port P7 register	P7	XX_{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	XX_{16}
$03F1_{16}$	Port P9 register	P9	XX_{16}
$03F2_{16}$	Port P8 direction register	PD8	$00X00000_2$
$03F3_{16}$	Port P9 direction register	PD9	00_{16}
$03F4_{16}$	Port P10 register	P10	XX_{16}
$03F5_{16}$			
$03F6_{16}$	Port P10 direction register	PD10	00_{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 (Note 1) 00000010_2
$03FE_{16}$	Pull-up control register 2	PUR2	00_{16}
$03FF_{16}$	Port control register	PCR	00_{16}

X: Undefined

Note 1: At hardware reset, the register is as follows:
- "00000000_2" where "L" is input to the CNV_{SS} pin
- "00000010_2" where "H" is input to the CNV_{SS} pin

At software reset, watchdog timer reset and oscillation stop detection reset, the register is as follows:
- "00000000_2" where the PM01 to PM00 bits in the PM0 register are "00_2" (single-chip mode)
- "00000010_2" where the PM01 to PM00 bits in the PM0 register are "01_2" (memory expansion mode) or "11_2" (microprocessor mode)

Note 2: The blank areas are reserved and cannot be accessed by users.

Figure 1.4.16 Location of Peripheral Function Control Registers and Value at After Reset (16)

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 (refer to “Table 1.5.1 Pin Status When $\overline{\text{RESET}}$ Pin Level is “L” ”). 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 for $t_{d(P-R)}$ 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.

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, re-oscillation detection), the microcomputer initializes its pins, CPU and SFR, coming to a halt if it detects main clock oscillation circuit stop. Refer to “Oscillation Stop and Re-oscillation Detection Function”.

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

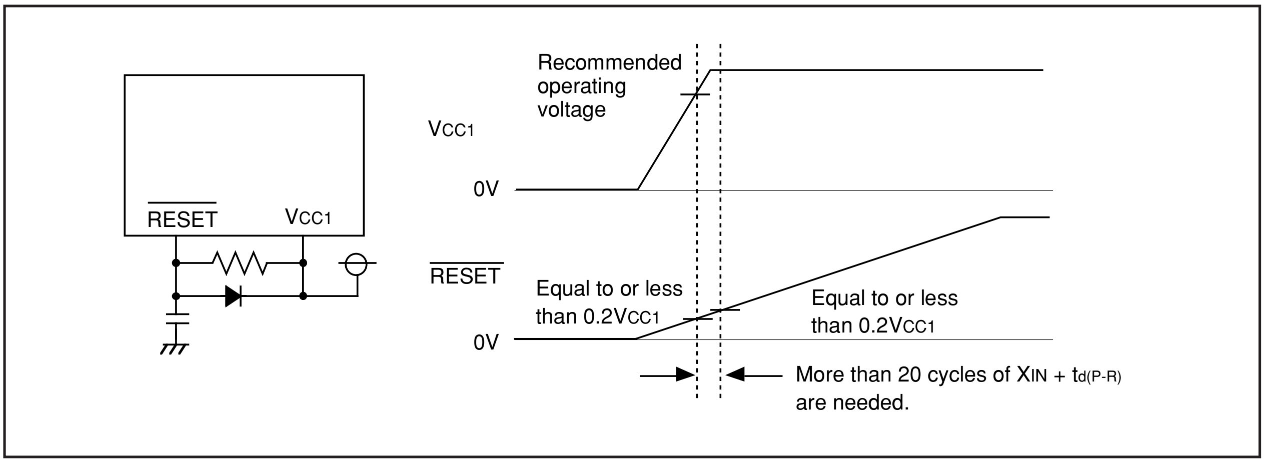


Figure 1.5.1 Example Reset Circuit

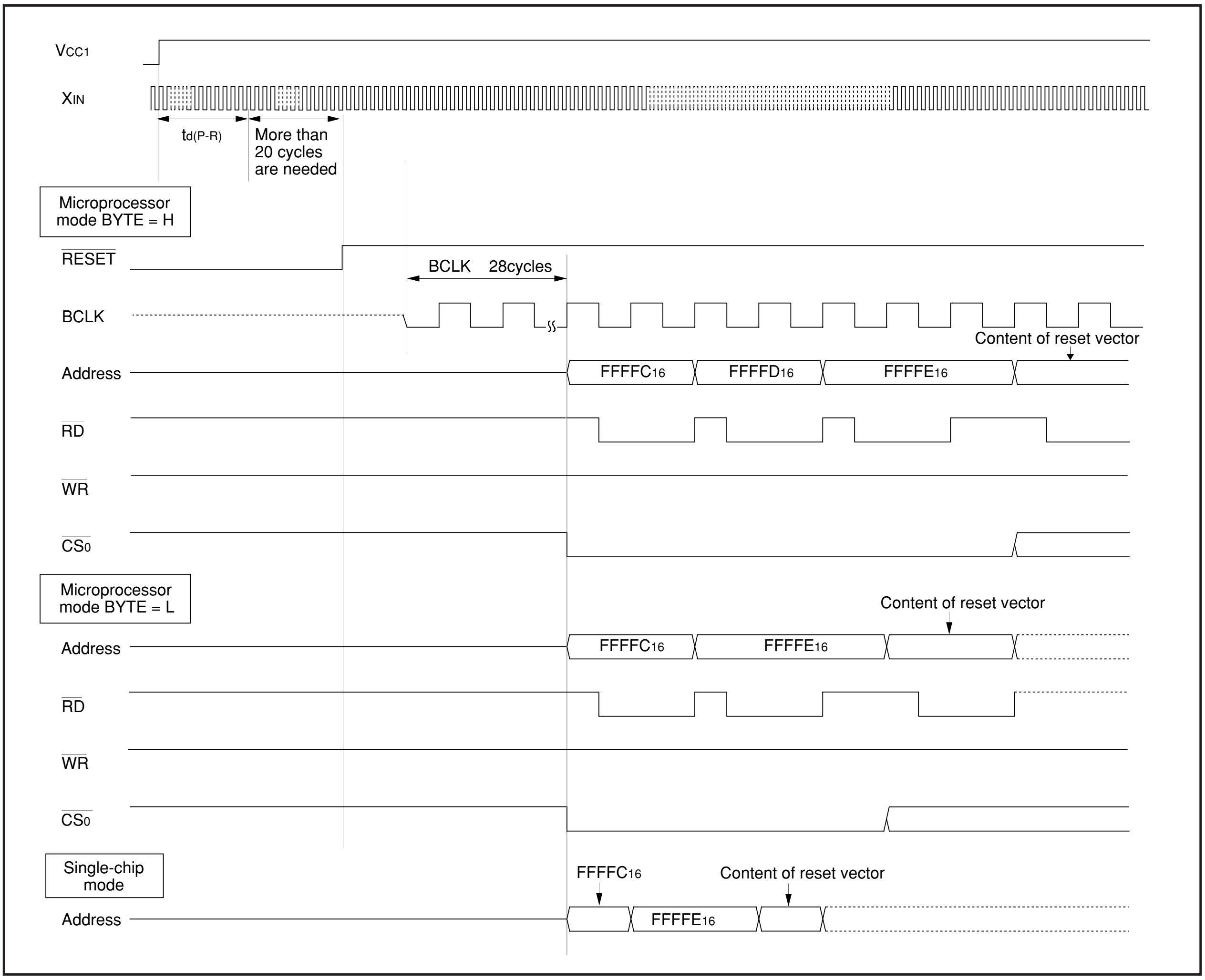


Figure 1.5.2 Reset Sequence

Table 1.5.1 Pin Status When $\overline{\text{RESET}}$ Pin Level is "L"

Pin name	Status		
	$CNV_{SS} = V_{SS}$	$CNV_{SS} = V_{CC1}$ (Note)	
		$BYTE = V_{SS}$	$BYTE = V_{CC1}$
P0	Input port	Data input	Data input
P1	Input port	Data input	Input port
P2, P3, $P4_0$ to $P4_3$	Input port	Address output (undefined)	Address output (undefined)
$P4_4$	Input port	$\overline{CS_0}$ output ("H" is output)	$\overline{CS_0}$ output ("H" is output)
$P4_5$ to $P4_7$	Input port	Input port (Pulled high)	Input port (Pulled high)
$P5_0$	Input port	$\overline{\text{WR}}$ output ("H" is output)	$\overline{\text{WR}}$ output ("H" is output)
$P5_1$	Input port	$\overline{\text{BHE}}$ output (undefined)	$\overline{\text{BHE}}$ output (undefined)
$P5_2$	Input port	$\overline{\text{RD}}$ output ("H" is output)	$\overline{\text{RD}}$ output ("H" is output)
$P5_3$	Input port	BCLK output	BCLK output
$P5_4$	Input port	$\overline{\text{HLDA}}$ output (The output value depends on the input to the $\overline{\text{HOLD}}$ pin)	$\overline{\text{HLDA}}$ output (The output value depends on the input to the $\overline{\text{HOLD}}$ pin)
$P5_5$	Input port	$\overline{\text{HOLD}}$ input	$\overline{\text{HOLD}}$ input
$P5_6$	Input port	ALE output ("L" is output)	ALE output ("L" is output)
$P5_7$	Input port	$\overline{\text{RDY}}$ input	$\overline{\text{RDY}}$ input
P6, P7, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, P9, P10	Input port	Input port	Input port

Note: Shown here is the valid pin state when the internal power supply voltage has stabilized after power-on. When $CNV_{SS} = V_{CC1}$, the pin state is indeterminate until the internal power supply voltage stabilizes.

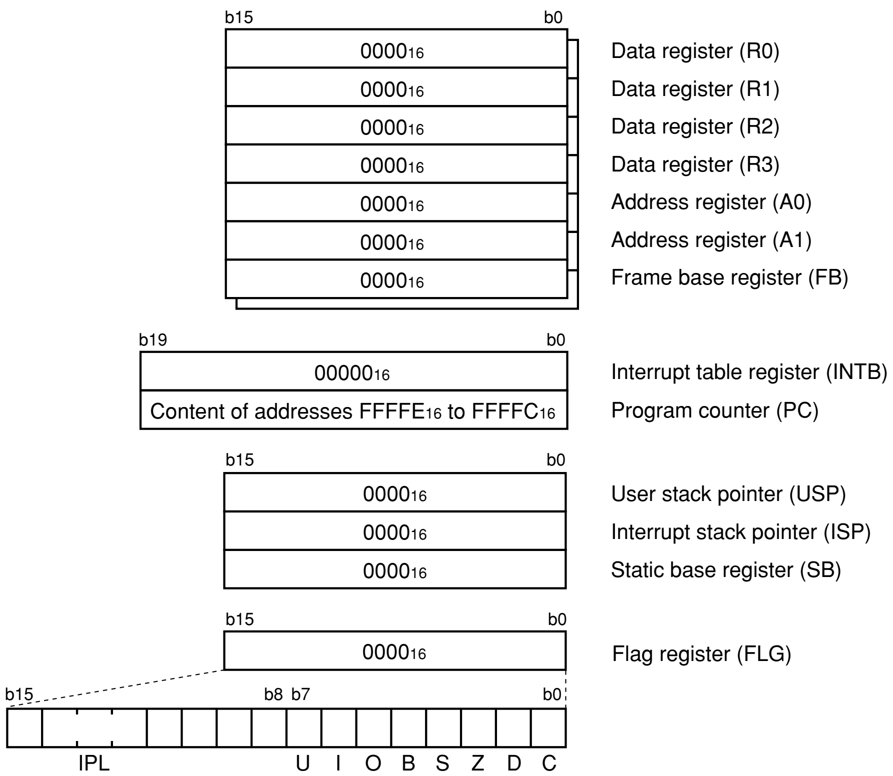


Figure 1.5.3 CPU Register Status After Reset

Processor Mode

(1) Types of Processor Mode

Three processor modes are available to choose from: single-chip mode, memory expansion mode, and microprocessor mode. Table 1.6.1 shows the features of these processor modes.

Table 1.6.1 Features of Processor Modes

Processor mode	Access space	Pins which are assigned I/O ports
Single-chip mode	SFR, internal RAM, internal ROM	All pins are I/O ports or peripheral function I/O pins
Memory expansion mode	SFR, internal RAM, internal ROM, external area (Note)	Some pins serve as bus control pins (Note)
Microprocessor mode	SFR, internal RAM, external area (Note)	Some pins serve as bus control pins (Note)

Note: Refer to “Bus”.

(2) Setting Processor Modes

Processor mode is set by using the CNV_{SS} 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 bits set values and processor modes.

Table 1.6.2 Processor Mode After Hardware Reset

CNV_{SS} pin input level	Processor mode
V_{SS}	Single-chip mode
V_{CC1} (Notes 1, 2)	Microprocessor mode

Note 1: If the microcomputer is reset in hardware by applying V_{CC1} to the CNV_{SS} pin, the internal ROM cannot be accessed regardless of PM01 to PM00 bits.
Note 2: The multiplexed bus cannot be assigned to the entire $\overline{CS}$ space.

Table 1.6.3 PM01 to PM00 Bits Set Values and Processor Modes

PM01 to PM 00 bits	Processor mode
00_2	Single-chip mode
01_2	Memory expansion mode
10_2	Must not be set
11_2	Microprocessor mode

Rewriting the PM01 to PM00 bits places the microcomputer in the corresponding processor mode regardless of whether the input level on the CNV_{SS} pin is “H” or “L”. Note, however, that the PM01 to PM00 bits cannot be rewritten to “01_2” (memory expansion mode) or “11_2” (microprocessor mode) at the same time the PM07 to PM02 bits are rewritten. Note also that these bits cannot be rewritten to enter microprocessor mode in the internal ROM, nor can they be rewritten to exit microprocessor mode in areas overlapping the internal ROM.
If the microcomputer is reset in hardware by applying V_{CC1} to the CNV_{SS} pin (hardware reset), the internal ROM cannot be accessed regardless of PM01 to PM00 bits.
Figures 1.6.1 and 1.6.2 show the processor mode related registers. Figure 1.6.3 shows the memory map in single-chip mode. Figures 1.6.4 to 1.6.7 show the memory map and $\overline{CS}$ area in memory expansion mode and microprocessor mode.

Processor mode register 0 (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

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 : Memory expansion mode 1 0 : Must not be set 1 1 : Microprocessor mode	RW
PM01			RW
PM02	R/W mode select bit (Note 3)	0 : $\overline{RD}$, $\overline{BHE}$, $\overline{WR}$ 1 : $\overline{RD}$, $\overline{WRH}$, $\overline{WRL}$	RW
PM03	Software reset bit	Setting this bit to "1" resets the microcomputer. When read, its content is "0"	RW
PM04	Multiplexed bus space select bit (Note 3)	b5 b4 0 0 : Multiplexed bus is unused (Separate bus in the entire $\overline{CS}$ space) 0 1 : Allocated to $\overline{CS2}$ space 1 0 : Allocated to $\overline{CS1}$ space 1 1 : Allocated to the entire $\overline{CS}$ space (Note 4)	RW
PM05			RW
PM06	Port $P4_0$ to $P4_3$ function select bit (Note 3)	0 : Address output 1 : Port function (Address is not output)	RW
PM07	BCLK output disable bit (Note 3)	0 : BCLK is output 1 : BCLK is not output (Pin is left high-impedance)	RW

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.
Note 3: Effective when the PM01 to PM00 bits are set to "01_2" (memory expansion mode) or "11_2" (microprocessor mode).
Note 4: To set the PM01 to PM00 bits are "01_2" and the PM05 to PM04 bits are "11_2" (multiplexed bus assigned to the entire $\overline{CS}$ space), apply an "H" signal to the BYTE pin (external data bus is 8-bit width).
While the CNV_{SS} pin is held "H" (V_{CC1}), do not rewrite the PM05 to PM04 bits to "11_2" after reset.
If the PM05 to PM04 bits are set to "11_2" during memory expansion mode, $P3_1$ to $P3_7$ and $P4_0$ to $P4_3$ become I/O ports, in which case the accessible area for each $\overline{CS}$ is 256 bytes.

Figure 1.6.1 PM0 Register

Processor mode register 1 (Note 1)

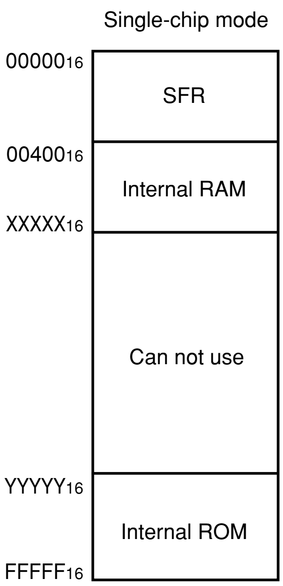
b7	b6	b5	b4	b3	b2	b1	b0
	0	0	0				

Symbol	Address	After reset
PM1	0005_{16}	$0XXX1000_2$

Bit symbol	Bit name	Function	RW
PM10	$\overline{CS2}$ area switch bit (data block enable bit) (Note 2)	0 : 08000_{16} to $26FFF_{16}$ (block A disable) 1 : 10000_{16} to $26FFF_{16}$ (block A enable)	RW
PM11	Port $P3_7$ to $P3_4$ function select bit (Note 3)	0 : Address output 1 : Port function	RW
PM12	Watchdog timer function select bit	0 : Watchdog timer interrupt 1 : Watchdog timer reset (Note 4)	RW
PM13	Internal reserved area expansion bit (Note 5)	Internal ROM area is: 0 : 192 Kbytes or smaller 1 : Expanded over 192 Kbytes	RW
– (b6-b4)	Reserved bit	Set to "0".	RW
PM17	Wait bit (Note 6)	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: For the mask ROM version, this bit must be set to "0".
For the flash memory version, the PM10 bit also controls block A by enabling or disabling it.
However, the PM10 bit is automatically set to "1" when the FMR01 bit in the FMR0 register is "1" (CPU rewrite mode).
Note 3: Effective when the PM01 to PM00 bits are set to "01_2" (memory expansion mode) or "11_2" (microprocessor mode).
Note 4: The PM12 bit is set to "1" by writing a "1" in a program. (Writing a "0" has no effect.)
Note 5: Be sure to set this bit to "0" except for products with internal ROM area over 192 Kbytes.
The PM13 bit is automatically set to "1" when the FMR01 bit in the FMR0 register is "1" (CPU rewrite mode).
Note 6: When the 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.
If the CSiW bit (i = 0 to 3) in the CSR register is "0" (with wait state), the $\overline{CS_i}$ area is always accessed with one or more wait states regardless of whether the PM17 bit is set or not.
Where the $\overline{RDY}$ signal is used or multiplexed bus is used, set the CSiW bit to "0" (with wait state).

Figure 1.6.2 PM1 Register



PM13 = 0

Internal RAM		Internal ROM	
Capacity	Address XXXXX$_{16}$	Capacity	Address YYYYY$_{16}$
5 Kbytes	017FF$_{16}$	128 Kbytes	E0000$_{16}$
10 Kbytes	02BFF$_{16}$	256 Kbytes	D0000$_{16}$ (Note 1)

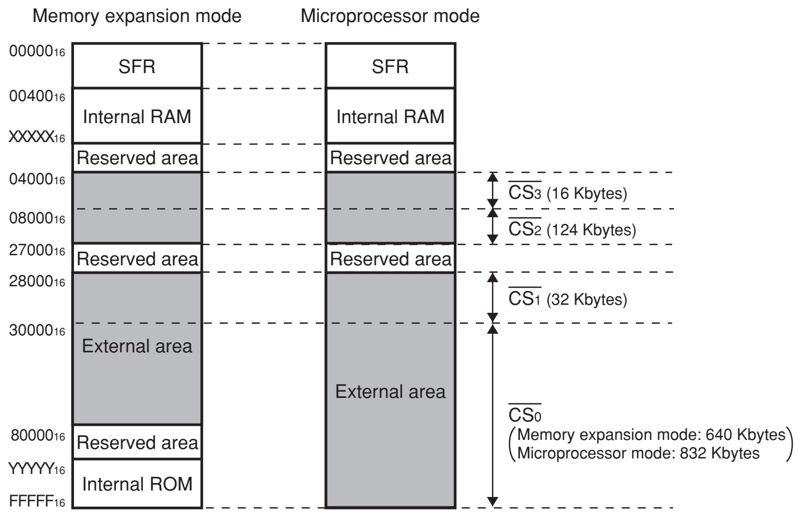
PM13 = 1

Internal RAM		Internal ROM	
Capacity	Address XXXXX$_{16}$	Capacity	Address YYYYY$_{16}$
5 Kbytes	017FF$_{16}$	128 Kbytes	E0000$_{16}$
10 Kbytes	02BFF$_{16}$	256 Kbytes	C0000$_{16}$

Note 1: When the PM13 bit in the PM1 register = 0, 192 Kbytes or smaller of the internal ROM can be used.
Note 2: For the mask ROM version, set the PM10 bit in the PM1 register to "0" (08000$_{16}$ to 26FFF$_{16}$ for $\overline{CS2}$ area).

Figure 1.6.3 Memory Map in Single-chip Mode

■ When PM13 = 0 and PM10 = 0 (A memory space of 1MB)

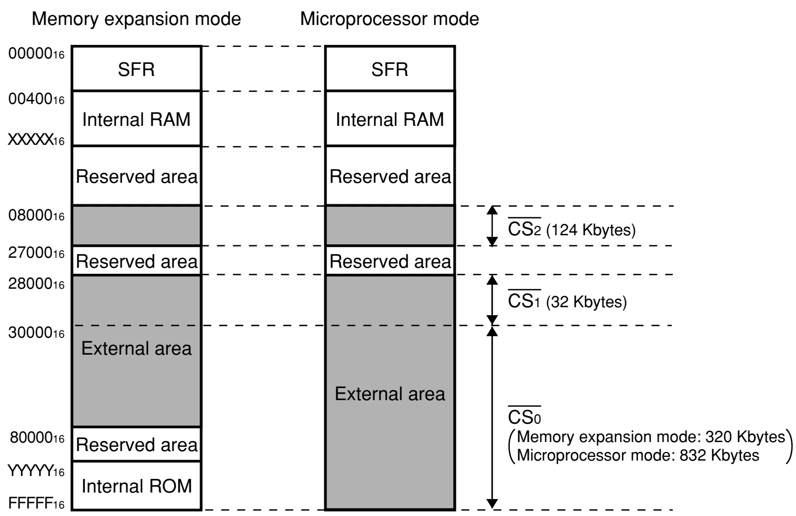


Internal RAM		Internal ROM	
Capacity	Address $XXXXX_{16}$	Capacity	Address $YYYYY_{16}$
5 Kbytes	$017FF_{16}$	128 Kbytes	$E0000_{16}$
10 Kbytes	$02BFF_{16}$	256 Kbytes	$D0000_{16}$ (Note)

Note: When the PM13 bit in the PM1 register = 0, 192 Kbytes or smaller of the internal ROM can be used.

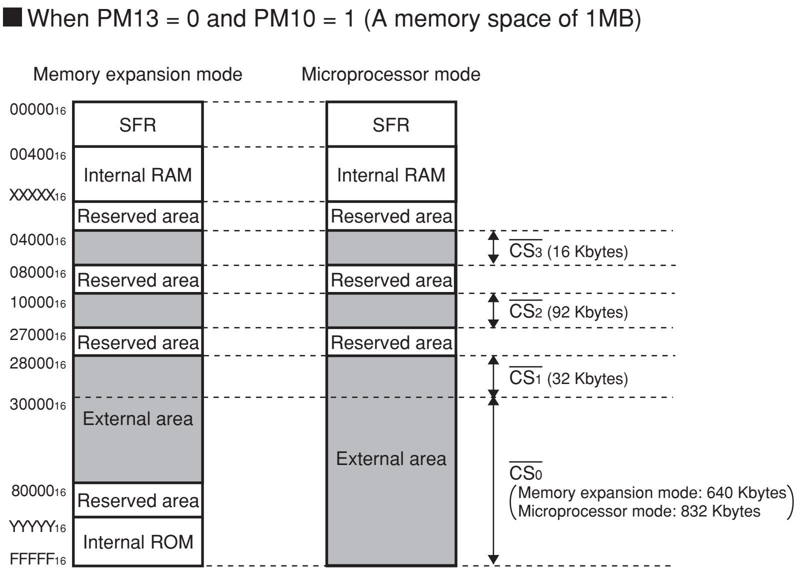
Figure 1.6.4 Memory Map and $\overline{CS}$ Area in Memory Expansion Mode and Microprocessor Mode (1)

■ When PM13 = 1 and PM10 = 0 (A memory space of 1MB)



Internal RAM		Internal ROM	
Capacity	Address $XXXXX_{16}$	Capacity	Address $YYYYY_{16}$
5 Kbytes	$017FF_{16}$	128 Kbytes	$E0000_{16}$
10 Kbytes	$02BFF_{16}$	256 Kbytes	$C0000_{16}$

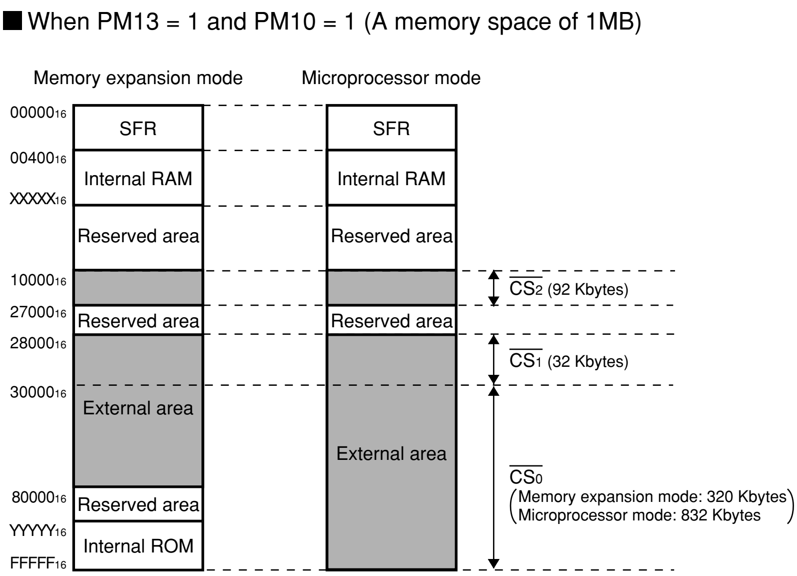
Figure 1.6.5 Memory Map and $\overline{CS}$ Area in Memory Expansion Mode and Microprocessor Mode (2)



Internal RAM		Internal ROM	
Capacity	Address XXXXX16	Capacity	Address YYYYY16
5 Kbytes	017FF16	128 Kbytes	E000016
10 Kbytes	02BFF16	256 Kbytes	D000016 (Note)

Note: When the PM13 bit in the PM1 register = 0, 192 Kbytes or smaller of the internal ROM can be used.

Figure 1.6.6 Memory Map and $\overline{CS}$ Area in Memory Expansion Mode and Microprocessor Mode (3)



Internal RAM		Internal ROM	
Capacity	Address XXXXX16	Capacity	Address YYYYY16
5 Kbytes	017FF16	128 Kbytes	E000016
10 Kbytes	02BFF16	256 Kbytes	C000016

Figure 1.6.7 Memory Map and $\overline{CS}$ Area in Memory Expansion Mode and Microprocessor Mode (4)

Bus

During memory expansion or microprocessor mode, some pins serve as the bus control pins to perform data input/output to and from external devices. These bus control pins include A_0 to A_{19}, D_0 to D_{15}, $\overline{CS_0}$ to $\overline{CS_3}$, $\overline{RD}$, $\overline{WRL}/\overline{WR}$, $\overline{WRH}/\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$, $\overline{HLDA}$ and BCLK.

Bus Mode

The bus mode, either multiplexed or separate, can be selected using the PM05 to PM04 bits in the PM0 register.

Separate Bus

In this bus mode, data and address are separate.

Multiplexed Bus

In this bus mode, data and address are multiplexed.

- When the input level on BYTE pin is high (8-bit data bus)
  D_0 to D_7 and A_0 to A_7 are multiplexed.
- When the input level on BYTE pin is low (16-bit data bus)
  D_0 to D_7 and A_1 to A_8 are multiplexed. D_8 to D_{15} are not multiplexed. Do not use D_8 to D_{15}.
  External buses connecting to a multiplexed bus are allocated to only the even addresses of the microcomputer. Odd addresses cannot be accessed.

Bus Control

The following describes the signals needed for accessing external devices and the functionality of software wait.

(1) Address Bus

The address bus consists of 20 lines, A_0 to A_{19}. The address bus width can be chosen to be 12, 16 or 20 bits by using the PM06 bit in the PM0 register and the PM11 bit in the PM1 register. Table 1.7.1 shows the PM06 and PM11 bits set values and address bus widths.
When processor mode is changed from single-chip mode to memory expansion mode, the address bus is indeterminate until any external area is accessed.

Table 1.7.1 PM06 and PM11 Bits Set Value and Address Bus Width

Set value (Note)	Pin function	Address bus width
PM11 = 1	$P3_4$ to $P3_7$	12 bits
PM06 = 1	$P4_0$ to $P4_3$	
PM11 = 0	A_{12} to A_{15}	16 bits
PM06 = 1	$P4_0$ to $P4_3$	
PM11 = 0	A_{12} to A_{15}	20 bits
PM06 = 0	A_{16} to A_{19}	

Note: No values other than those shown above can be set.

(2) Data Bus

When input on the BYTE pin is high (data bus is an 8-bit width), 8 lines D_0 to D_7 comprise the data bus; when input on the BYTE pin is low (data bus is a 16-bit width), 16 lines D_0 to D_{15} comprise the data bus. Do not change the input level on the BYTE pin while in operation.

(3) Chip Select Signal

The chip select (hereafter referred to as the $\overline{CS_i}$) signals are output from the $\overline{CS_i}$ (i = 0 to 3) pins. These pins can be chosen to function as I/O ports or as $\overline{CS}$ by using the CSi bit in the CSR register.
Figure 1.7.1 shows the CSR register.
During 1 Mbyte mode, the external area can be separated into up to 4 by the $\overline{CS_i}$ signal which is output from the $\overline{CSi}$ pin.
Figure 1.7.2 shows the example of address bus and $\overline{CS_i}$ signal output in 1 Mbyte mode.

Chip select control register

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
CSR	0008_{16}	01_{16}

Bit symbol	Bit name	Function	RW
CS0	$\overline{CS_0}$ output enable bit	0 : Chip select output disabled (functions as I/O port) 1 : Chip select output enabled	RW
CS1	$\overline{CS_1}$ output enable bit		RW
CS2	$\overline{CS_2}$ output enable bit		RW
CS3	$\overline{CS_3}$ output enable bit		RW
CS0W	$\overline{CS_0}$ wait bit	0 : With wait state 1 : Without wait state (Notes 1, 2, 3)	RW
CS1W	$\overline{CS_1}$ wait bit		RW
CS2W	$\overline{CS_2}$ wait bit		RW
CS3W	$\overline{CS_3}$ wait bit		RW

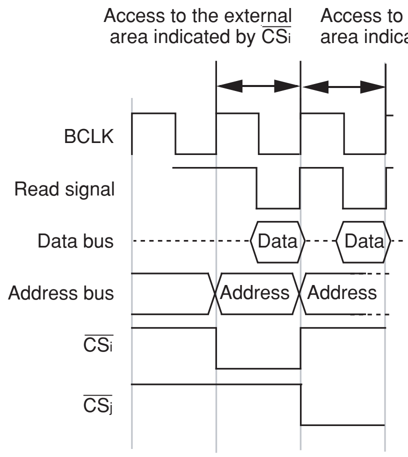
Note 1: Where the $\overline{RDY}$ signal is used in the area indicated by $\overline{CS_i}$ (i = 0 to 3) or the multiplexed bus is used, set the CSiW bit to "0" (Wait state).
Note 2: If the PM17 bit in the PM1 register is set to "1" (with wait state), the external area indicated by $\overline{CS_0}$ to $\overline{CS_3}$ is always accessed with one wait state even when the CSiW bit is "1" (without wait state).
Note 3: When the CSiW bit is "0" (with wait state), the number of wait states (in terms of clock cycles) can be selected using the CSEi1W to CSEi0W bits in the CSE register.

Figure 1.7.1 CSR Register

Example 1

To access the external area indicated by $\overline{CS_j}$ in the next cycle after accessing the external area indicated by $\overline{CS_i}$.

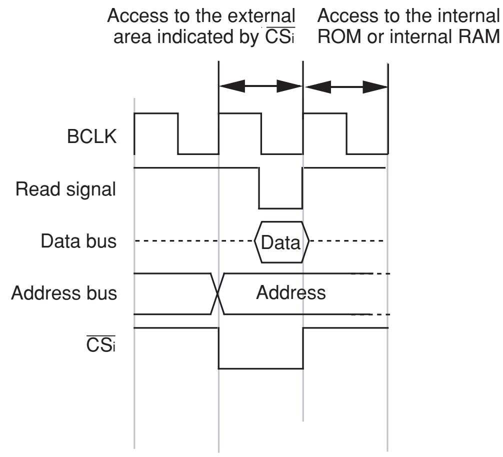
The address bus and the chip select signal both change state between these two cycles.



Example 2

To access the internal ROM or internal RAM in the next cycle after accessing the external area indicated by $\overline{CS_i}$.

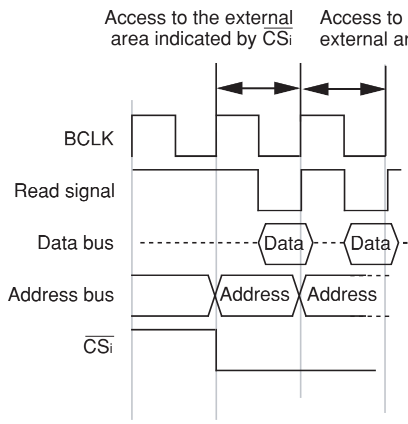
The chip select signal changes state but the address bus does not change state.



Example 3

To access the external area indicated by $\overline{CS_i}$ in the next cycle after accessing the external area indicated by the same $\overline{CS_i}$.

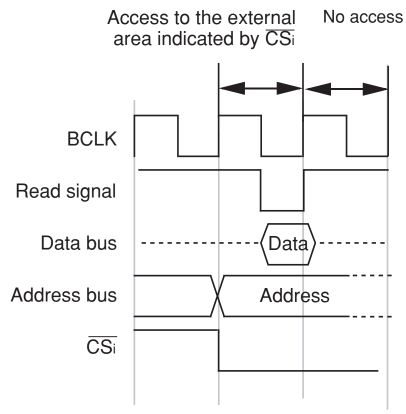
The address bus changes state but the chip select signal does not change state.



Example 4

Not to access any area (nor instruction prefetch generated) in the next cycle after accessing the external area indicated by $\overline{CS_i}$.

Neither the address bus nor the chip select signal changes state between these two cycles.



Note : These examples show the address bus and chip select signal when accessing areas in two successive cycles. The chip select bus cycle may be extended more than two cycles depending on a combination of these examples.

Shown above is the case where separate bus is selected and the area is accessed for read without wait states. i = 0 to 3, j = 0 to 3 (not including i, however)

Figure 1.7.2 Example of Address Bus and $\overline{CS_i}$ Signal Output in 1 Mbyte Mode

(4) Read and Write Signals

When the data bus is 16-bit width, the read and write signals can be chosen to be a combination of $\overline{RD}$, $\overline{WR}$ and $\overline{BHE}$ or a combination of $\overline{RD}$, $\overline{WRL}$ and $\overline{WRH}$ by using the PM02 bit in the PM0 register. When the data bus is 8-bit width, use a combination of $\overline{RD}$, $\overline{WR}$ and $\overline{BHE}$.

Table 1.7.2 shows the operation of $\overline{RD}$, $\overline{WRL}$, and $\overline{WRH}$ signals. Table 1.7.3 shows the operation of $\overline{RD}$, $\overline{WR}$, and $\overline{BHE}$ signals.

Table 1.7.2 Operation of $\overline{RD}$, $\overline{WRL}$ and $\overline{WRH}$ Signals

Data bus width	$\overline{RD}$	$\overline{WRL}$	$\overline{WRH}$	Status of external data bus
16 bits (BYTE pin input = L)	L	H	H	Read data
	H	L	H	Write 1 byte of data to an even address
	H	H	L	Write 1 byte of data to an odd address
	H	L	L	Write data to both even and odd addresses

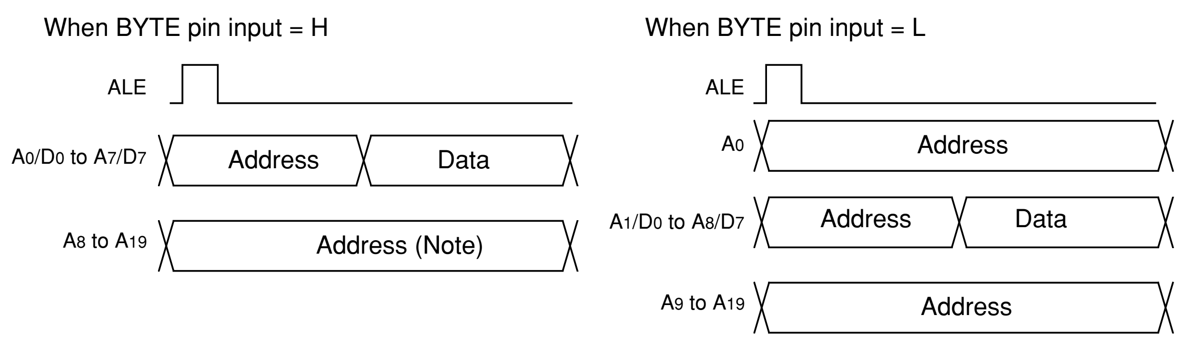
Table 1.7.3 Operation of $\overline{RD}$, $\overline{WR}$ and $\overline{BHE}$ Signals

Data bus width	$\overline{RD}$	$\overline{WR}$	$\overline{BHE}$	A_0	Status of external data bus
16 bits (BYTE pin input = L)	H	L	L	H	Write 1 byte of data to an odd address
	L	H	L	H	Read 1 byte of data from an odd address
	H	L	H	L	Write 1 byte of data to an even address
	L	H	H	L	Read 1 byte of data from an even address
	H	L	L	L	Write data to both even and odd addresses
	L	H	L	L	Read data from both even and odd addresses
8 bits (BYTE pin input = H)	H	L	− (Note)	H to L	Write 1 byte of data
	L	H	− (Note)	H to L	Read 1 byte of data

Note: Do not use.

(5) ALE Signal

The ALE signal latches the address when accessing the multiplexed bus space. Latch the address when the ALE signal falls. Figure 1.7.3 shows the ALE signal, address bus and data bus.



Note: If the entire $\overline{CS}$ space is assigned a multiplexed bus, these pins function as I/O ports.

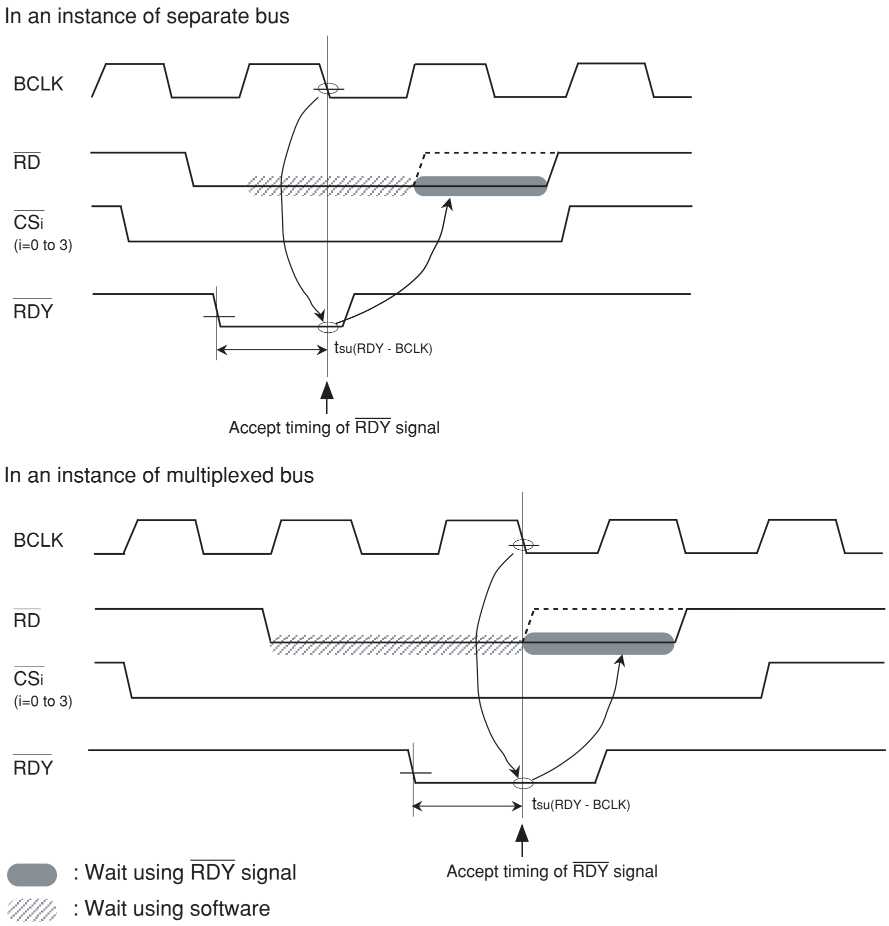
Figure 1.7.3 ALE Signal, Address Bus, Data Bus

(6) The $\overline{\text{RDY}}$ Signal

This signal is provided for accessing external devices which need to be accessed at low speed. If input on the $\overline{\text{RDY}}$ pin is asserted low at the last falling edge of BCLK of the bus cycle, one wait state is inserted in the bus cycle. While in a wait state, the following signals retain the state in which they were when the $\overline{\text{RDY}}$ signal was acknowledged.

A_0 to A_{19}, D_0 to D_{15}, $\overline{\text{CS}_0}$ to $\overline{\text{CS}_3}$, $\overline{\text{RD}}$, $\overline{\text{WRL}}$, $\overline{\text{WRH}}$, $\overline{\text{WR}}$, $\overline{\text{BHE}}$, ALE, $\overline{\text{HLDA}}$

Then, when the input on the $\overline{\text{RDY}}$ pin is detected high at the falling edge of BCLK, the remaining bus cycle is executed. Figure 1.7.4 shows example in which the wait state was inserted into the read cycle by the $\overline{\text{RDY}}$ signal. To use the $\overline{\text{RDY}}$ signal, set the corresponding bit (CS3W to CS0W bits) in the CSR register to "0" (with wait state). When not using the $\overline{\text{RDY}}$ signal, process the $\overline{\text{RDY}}$ pin as an unused pin.



$t_{SU(RDY-BCLK)}$: $\overline{\text{RDY}}$ input setup time

Shown above is the case where CSEi1W to CSEi0W (i = 0 to 3) bits in the CSE register are "00_2" (one wait state).

Figure 1.7.4 Example in which Wait State was Inserted into Read Cycle by $\overline{\text{RDY}}$ Signal

(7) $\overline{\text{HOLD}}$ Signal

This signal is used to transfer control of the bus from CPU or DMAC to an external circuit. When the input on $\overline{\text{HOLD}}$ pin is pulled low, the microcomputer is placed in a hold state after the bus access then in process finishes. The microcomputer remains in a hold state while the $\overline{\text{HOLD}}$ pin is held low, during which time the $\overline{\text{HLDA}}$ pin outputs a low-level signal.
Table 1.7.4 shows the microcomputer status in the hold state.
Bus-using priorities are given to $\overline{\text{HOLD}}$, DMAC, and CPU in order of decreasing precedence (refer to "Figure 1.7.5 Bus-using Priorities"). However, if the CPU is accessing an odd address in word units, the DMAC cannot gain control of the bus during two separate accesses.

$\overline{\text{HOLD}}$ > DMAC > CPU

Figure 1.7.5 Bus-using Priorities

Table 1.7.4 Microcomputer Status in Hold State

Item		Status
BCLK		Output
A_0 to A_{19}, D_0 to D_{15}, $\overline{CS_0}$ to $\overline{CS_3}$, $\overline{RD}$, $\overline{WRL}$, $\overline{WRH}$, $\overline{WR}$, $\overline{BHE}$		High-impedance
I/O ports	P0, P1, P3, P4 (Note 1)	High-impedance
	P6 to P10	Maintains status when hold signal is received
$\overline{\text{HLDA}}$		Output "L"
Internal peripheral circuits		ON (but watchdog timer stops (Note 2))
ALE signal		Undefined

Note 1: When I/O port function is selected.
Note 2: The watchdog timer does not stop when the PM22 bit in the PM2 register is set to "1" (the count source for the watchdog timer is the ring oscillator clock).

(8) BCLK Output

If the PM07 bit in the PM0 register is set to "0" (output enable), a clock with the same frequency as that of the CPU clock is output as BCLK from the BCLK pin. Refer to "CPU Clock and Peripheral Function Clock".

Table 1.7.5 shows the pin functions for each processor mode.

Table 1.7.5 Pin Functions for Each Processor Mode

Processor mode		Memory expansion mode or microprocessor mode				Memory expansion mode
PM05 to PM04 bits		00_2 (separate bus)		01_2 ($\overline{CS_2}$ is for multiplexed bus and others are for separate bus) 10_2 ($\overline{CS_1}$ is for multiplexed bus and others are for separate bus)		11_2 (multiplexed bus for the entire space) (Note 1)
Data bus width		8 bits	16 bits	8 bits	16 bits	8 bits
BYTE pin		"H"	"L"	"H"	"L"	"H"
$P0_0$ to $P0_7$		D_0 to D_7		D_0 to D_7 (Note 4)		I/O ports
$P1_0$ to $P1_7$		I/O ports	D_8 to D_{15}	I/O ports	D_8 to D_{15} (Note 4)	I/O ports
$P2_0$		A_0		A_0/D_0 (Note 2)	A_0	A_0/D_0
$P2_1$ to $P2_7$		A_1 to A_7		A_1 to A_7/D_1 to D_7 (Note 2)	A_1 to A_7/D_0 to D_6 (Note 2)	A_1 to A_7/D_1 to D_7
$P3_0$		A_8			A_8/D_7 (Note 2)	A_8
$P3_1$ to $P3_3$		A_9 to A_{11}				I/O ports
$P3_4$ to $P3_7$	PM11 = 0	A_{12} to A_{15}				I/O ports
	PM11 = 1	I/O ports				
$P4_0$ to $P4_3$	PM06 = 0	A_{16} to A_{19}				I/O ports
	PM06 = 1	I/O ports				
$P4_4$	CS0 = 0	I/O ports				
	CS0 = 1	$\overline{CS_0}$				
$P4_5$	CS1 = 0	I/O ports				
	CS1 = 1	$\overline{CS_1}$				
$P4_6$	CS2 = 0	I/O ports				
	CS2 = 1	$\overline{CS_2}$				
$P4_7$	CS3 = 0	I/O ports				
	CS3 = 1	$\overline{CS_3}$				
$P5_0$	PM02 = 0	$\overline{WR}$				
	PM02 = 1	− (Note 3)	$\overline{WRL}$	− (Note 3)	$\overline{WRL}$	− (Note 3)
$P5_1$	PM02 = 0	$\overline{BHE}$				
	PM02 = 1	− (Note 3)	$\overline{WRH}$	− (Note 3)	$\overline{WRH}$	− (Note 3)
$P5_2$		$\overline{RD}$				
$P5_3$		BCLK				
$P5_4$		$\overline{HLDA}$				
$P5_5$		$\overline{HOLD}$				
$P5_6$		ALE				
$P5_7$		$\overline{RDY}$				

I/O ports: Function as I/O ports or peripheral function I/O pins.

Note 1: For setting the PM01 to PM00 bits to "01_2" (memory expansion mode) and the PM05 to PM04 bits to "11_2" (multiplexed bus assigned to the entire $\overline{CS}$ space), apply "H" to the BYTE pin (external data bus is an 8-bit width). While the CNV_{SS} pin is held "H" (V_{CC1}), do not rewrite the PM05 to PM04 bits to "11_2" after reset. If the PM05 to PM04 bits are set to "11_2" during memory expansion mode, $P3_1$ to $P3_7$ and $P4_0$ to $P4_3$ become I/O ports, in which case the accessible area for each $\overline{CS}$ is 256 bytes.

Note 2: In separate bus mode, these pins serve as the address bus.

Note 3: If the data bus is 8-bit width, make sure the PM02 bit is set to "0" ($\overline{RD}$, $\overline{BHE}$, $\overline{WR}$).

Note 4: When accessing the area that uses a multiplexed bus, these pins output an indeterminate value during a write.

(9) External Bus Status When Internal Area Accessed

Table 1.7.6 shows the external bus status when the internal area is accessed.

Table 1.7.6 External Bus Status When Internal Area Accessed

Item		SFR accessed	Internal ROM, internal RAM accessed
A_0 to A_{19}		Address output	Maintain status before accessed address of external area or SFR
D_0 to D_{15}	When read	High-impedance	High-impedance
	When write	Output data	Undefined
$\overline{RD}$, $\overline{WR}$, $\overline{WRL}$, $\overline{WRH}$		$\overline{RD}$, $\overline{WR}$, $\overline{WRL}$, $\overline{WRH}$ output	Output "H"
$\overline{BHE}$		$\overline{BHE}$ output	Maintain status before accessed status of external area or SFR
$\overline{CS_0}$ to $\overline{CS_3}$		Output "H"	Output "H"
ALE		Output "L"	Output "L"

(10) Software Wait

Software wait states can be inserted by using the PM17 bit in the PM1 register, the CS0W to CS3W bits in the CSR register, and the CSE register. The SFR area is unaffected by these control bits. This area is always accessed in 2 BCLK or 3 BCLK cycles as determined by the PM20 bit in the PM2 register. Refer to "Table 1.7.7 Bit and Bus Cycle Related to Software Wait " for details.

To use the $\overline{RDY}$ signal, set the corresponding CS3W to CS0W bit to "0" (with wait state). Figure 1.7.6 shows the CSE register. Table 1.7.7 shows the software wait related bits and bus cycles. Figures 1.7.7 and 1.7.8 show the typical bus timings using software wait.

Chip select expansion control register

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
CSE	$001B_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
CSE00W	$\overline{CS_0}$ wait expansion bit (Note)	b1 b0 0 0 : 1 wait 0 1 : 2 waits 1 0 : 3 waits 1 1 : Must not be set	RW
CSE01W			RW
CSE10W	$\overline{CS_1}$ wait expansion bit (Note)	b3 b2 0 0 : 1 wait 0 1 : 2 waits 1 0 : 3 waits 1 1 : Must not be set	RW
CSE11W			RW
CSE20W	$\overline{CS_2}$ wait expansion bit (Note)	b5 b4 0 0 : 1 wait 0 1 : 2 waits 1 0 : 3 waits 1 1 : Must not be set	RW
CSE21W			RW
CSE30W	$\overline{CS_3}$ wait expansion bit (Note)	b7 b6 0 0 : 1 wait 0 1 : 2 waits 1 0 : 3 waits 1 1 : Must not be set	RW
CSE31W			RW

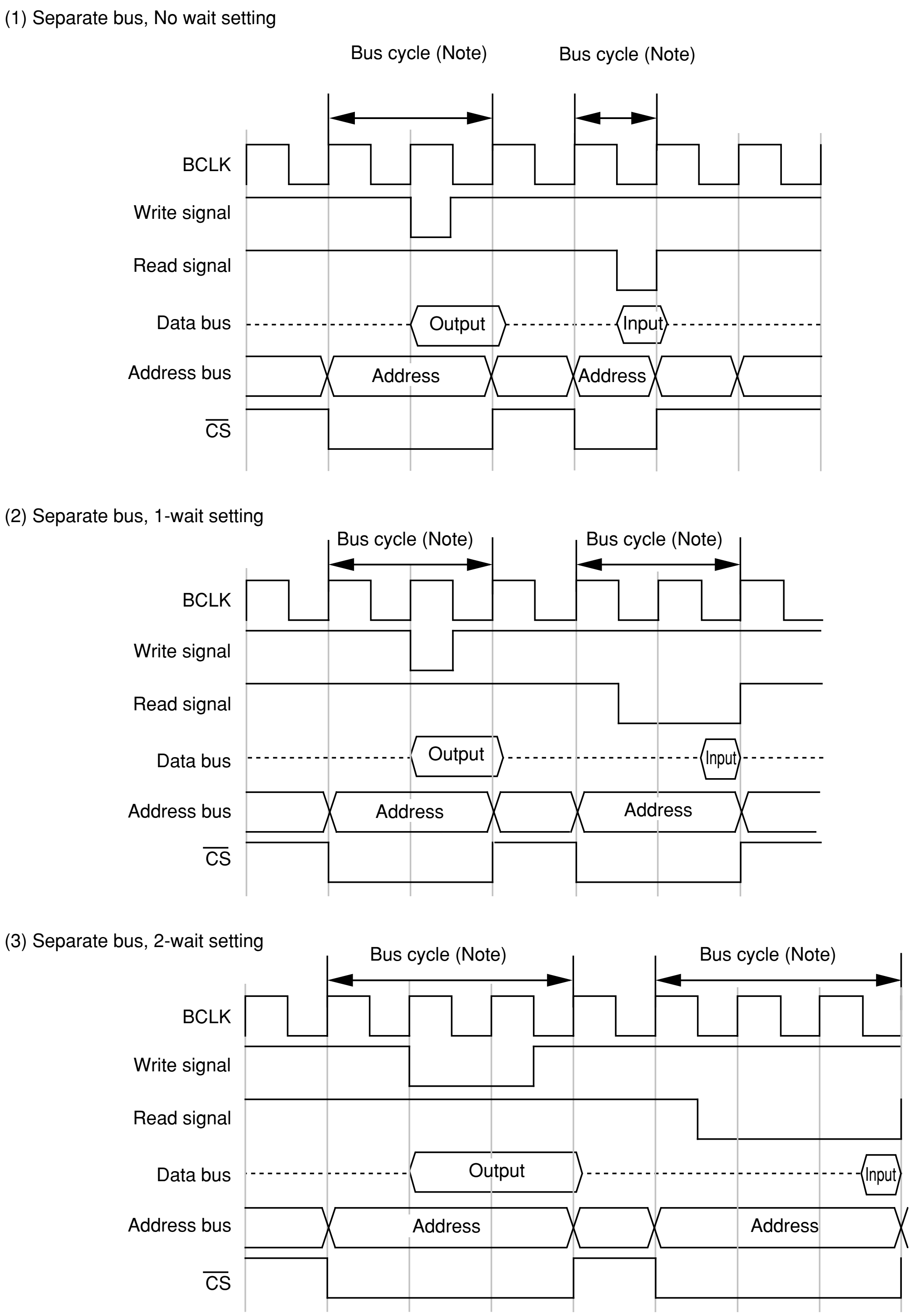
Note: Set the CSiW bit (i = 0 to 3) in the CSR register to "0" (with wait state) before writing to the CSEi1W to CSEi0W bits. If the CSiW bit needs to be set to "1" (without wait state), set the CSEi1W to CSEi0W bits to "00_2" before setting it.

Figure 1.7.6 CSE Register

Table 1.7.7 Software Wait Related Bits and Bus Cycles

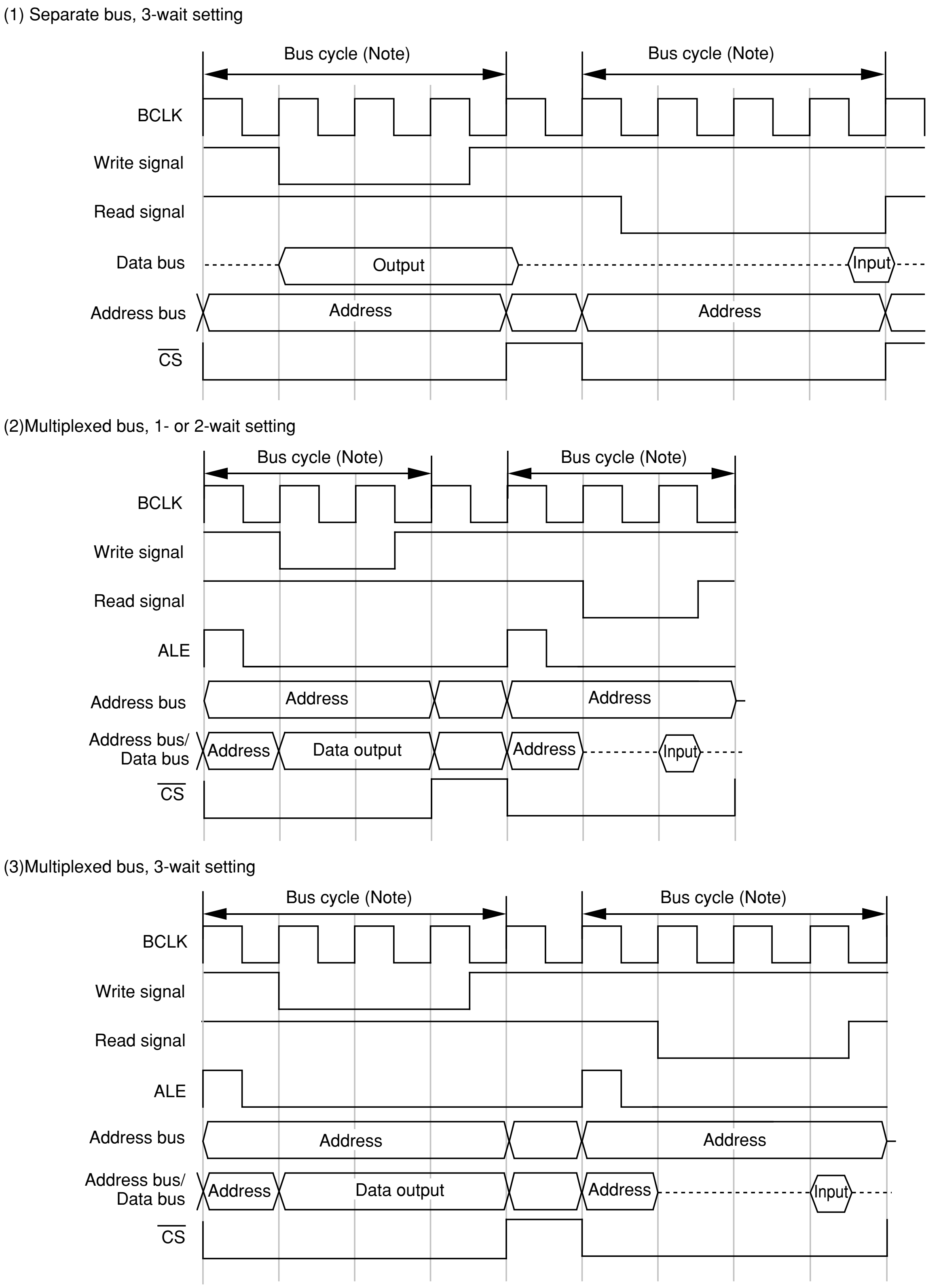
Area	Bus mode	PM2 Register PM20 bit	PM1 Register PM17 bit	CSR register CS3W bit (Note 1) CS2W bit (Note 1) CS1W bit (Note 1) CS0W bit (Note 1)	CSE register CS31W to CS30W bits CS21W to CS20W bits CS11W to CS10W bits CS01W to CS00W bits	Software wait	Bus cycle
SFR	–	0	–	–	–	–	2 BCLK cycles (Note 4)
	–	1	–	–	–	–	3 BCLK cycles (Note 4)
Internal ROM, RAM	–	–	0	–	–	No wait	1 BCLK cycle (Note 3)
	–	–	1	–	–	1 wait	2 BCLK cycles
External area	Separate bus	–	0	1	00_2	No wait	1 BCLK cycle (read)
							2 BCLK cycles (write)
		–	–	0	00_2	1 wait	2 BCLK cycles (Note 3)
		–	–	0	01_2	2 waits	3 BCLK cycles
		–	–	0	10_2	3 waits	4 BCLK cycles
		–	1	1	00_2	1 wait	2 BCLK cycles
	Multiplexed bus (Note 2)	–	–	0	00_2	1 wait	3 BCLK cycles
		–	–	0	01_2	2 waits	3 BCLK cycles
		–	–	0	10_2	3 waits	4 BCLK cycles
		–	1	0	00_2	1 wait	3 BCLK cycles

Note 1: To use the $\overline{RDY}$ signal, set this bit to “0 ”.
Note 2: To access in multiplexed bus mode, set the corresponding bit of CS0W to CS3W to “0” (with wait state).
Note 3: After reset, the PM17 bit is set to “0” (without wait state), all of the CS0W to CS3W bits are set to “0” (with wait state), and the CSE register is set to “00_{16}” (one wait state for $\overline{CS_0}$ to $\overline{CS_3}$). Therefore, the internal RAM and internal ROM are accessed with no wait state, and all external areas are accessed with one wait state.
Note 4: When the selected CPU clock source is the PLL clock, the number of wait cycles can be altered by the PM20 bit in the PM2 register. When using a 16 MHz or higher PLL clock, be sure to set the PM20 bit to “0” (2 wait cycles).



Note: These example timing charts indicate bus cycle length. After this bus cycle sometimes come read and write cycles in succession.

Figure 1.7.7 Typical Bus Timings Using Software Wait (1)



Note: These example timing charts indicate bus cycle length. After this bus cycle sometimes come read and write cycles in succession.

Figure 1.7.8 Typical Bus Timings Using Software Wait (2)

Clock Generation Circuit

The clock generation circuit contains four oscillator circuits as follows:

(1) Main clock oscillation circuit
(2) Sub clock oscillation circuit
(3) Ring oscillator
(4) PLL frequency synthesizer

Table 1.8.1 lists the clock generation circuit specifications. Figure 1.8.1 shows the clock generation circuit. Figures 1.8.2 to 1.8.8 show the clock-related registers.

Table 1.8.1 Clock Generation Circuit Specifications

Item	Main clock oscillation circuit	Sub clock oscillation circuit	Ring oscillator	PLL frequency synthesizer
Use of clock	• CPU clock source • Peripheral function clock source	• CPU clock source • Timer A, B's clock source	• CPU clock source • Peripheral function clock source • CPU and peripheral function clock sources when the main clock stops oscillating	• CPU clock source • Peripheral function clock source
Clock frequency	0 to 16 MHz	32.768 kHz	About 1 MHz	20 MHz
Usable oscillator	•Ceramic oscillator •Crystal oscillator	•Crystal oscillator	-	-
Pins to connect oscillator	X_{IN}, X_{OUT}	X_{CIN}, X_{COUT}	-	-
Oscillation stop and re-oscillation detection function	Present	Present	Present	Present
Oscillation status after reset	Oscillating	Stopped	Stopped	Stopped
Other	Externally derived clock can be input		-	-

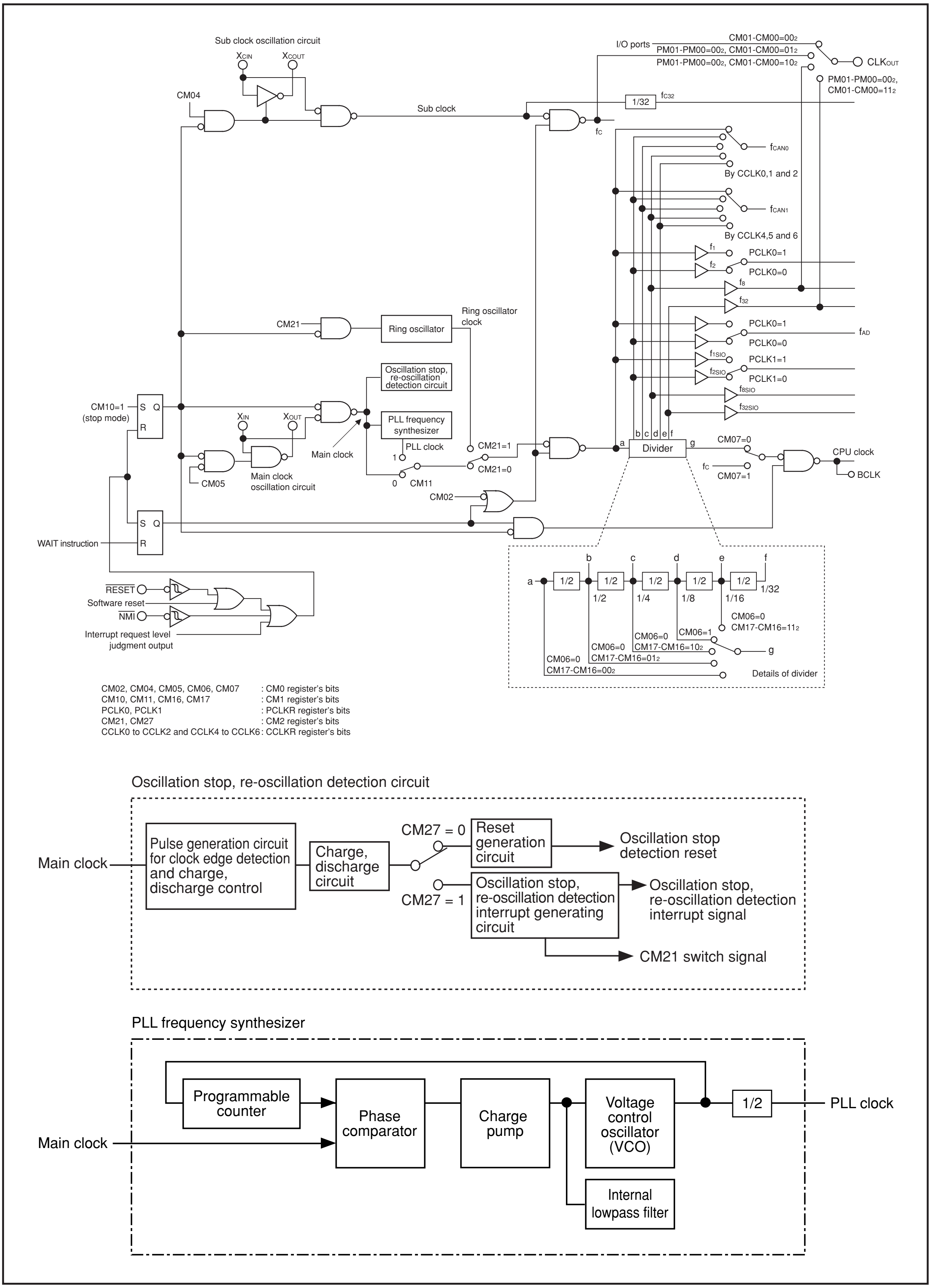


Figure 1.8.1 Clock Generation Circuit

System clock control register 0 (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

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

Bit symbol	Bit name	Function	RW
CM00	Clock output function select bit (Valid only in single-chip mode)	b1 b0 0 0 : I/O port $P5_7$ 0 1 : f_C output 1 0 : f_8 output 1 1 : f_{32} output	RW
CM01			RW
CM02	WAIT peripheral function clock stop bit	0 : Do not stop peripheral function clock in wait mode 1 : Stop peripheral function clock in wait mode (Note 2)	RW
CM03	X_{CIN}-X_{COUT} drive capacity select bit (Note 3)	0 : LOW 1 : HIGH	RW
CM04	Port X_C select bit (Note 3)	0 : I/O port $P8_6$, $P8_7$ 1 : X_{CIN}-X_{COUT} generation function (Note 4)	RW
CM05	Main clock stop bit (Notes 5, 6, 7)	0 : On 1 : Off (Notes 8, 9)	RW
CM06	Main clock division select bit 0 (Notes 7, 10, 12)	0 : CM16 and CM17 valid 1 : Division by 8 mode	RW
CM07	System clock select bit (Notes 6, 11)	0 : Main clock, PLL clock, or ring oscillator clock 1 : Sub clock	RW

Note 1: Write to this register after setting the PRC0 bit of PRCR register to "1" (write enable).
Note 2: The f_{C32} clock does not stop. During low speed or low power dissipation mode, do not set this bit to "1" (peripheral clock turned off when in wait mode).
Note 3: The CM03 bit is set to "1" (high) when the CM04 bit is set to "0" (I/O port) or the microcomputer goes to stop mode.
Note 4: To use a sub clock, set this bit to "1". Also make sure ports $P8_6$ and $P8_7$ are directed for input, with no pull-ups.
Note 5: This bit is provided to stop the main clock when the low power dissipation mode or ring oscillator low power dissipation mode is selected. This bit cannot be used for detection as to whether the main clock stopped or not. To stop the main clock, the following setting is required:
(1) Set the CM07 bit to "1" (sub clock select) or the CM21 bit of CM2 register to "1" (ring oscillator select) with the sub clock stably oscillating.
(2) Set the CM20 bit of CM2 register to "0" (oscillation stop, re-oscillation detection function disabled).
(3) Set the CM05 bit to "1" (stop).
Note 6: To use the main clock as the clock source for the CPU clock, follow the procedure below.
(1) Set the CM05 bit to "0" (oscillate)
(2) Wait until $t_{d(M-L)}$ elapses or the main clock oscillation stabilizes, whichever is longer.
(3) Set the CM11, CM21 and CM07 bits all to "0".
Note 7: When the CM21 bit = 0 (ring oscillator turned off) and the CM05 bit = 1 (main clock turned off), the CM06 bit is fixed to "1" (divide-by-8 mode) and the CM15 bit is fixed to "1" (drive capability High).
Note 8: During external clock input, only the clock oscillation buffer is turned off and clock input is accepted if the sub clock is not selected as a CPU clock.
Note 9: When CM05 bit is set to "1", the X_{OUT} pin goes "H". Furthermore, because the internal feedback resistor remains connected, the X_{IN} pin is pulled "H" to the same level as X_{OUT} via the feedback resistor.
Note 10: 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 11: After setting the CM04 bit to "1" (X_{CIN}-X_{COUT} oscillator function), wait until the sub clock oscillates stably before switching the CM07 bit from "0" to "1" (sub clock).
Note 12: To return from ring oscillator mode to high-speed or middle-speed mode, set the CM06 and CM15 bits both to "1".

Figure 1.8.2 CM0 Register

System clock control register 1 (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
			0	0	0		

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

Bit symbol	Bit name	Function	RW
CM10	All clock stop control bit (Notes 2, 3)	0 : Clock on 1 : All clocks off (stop mode)	RW
CM11	System clock select bit 1 (Notes 3, 4)	0 : Main clock 1 : PLL clock (Note 5)	RW
- (b4-b2)	Reserved bit	Set to "0"	RW
CM15	X_{IN}-X_{OUT} drive capacity select bit (Note 6)	0 : LOW 1 : HIGH	RW
CM16	Main clock division select bit 1 (Note 7)	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: If the CM10 bit is "1" (stop mode), X_{OUT} goes "H" and the internal feedback resistor is disconnected. The X_{CIN} and X_{COUT} pins are placed in the high-impedance state. When the CM11 bit is set to "1" (PLL clock), or 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 3: 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.
Note 4: Effective when CM07 bit is "0" and CM21 bit is "0".
Note 5: After setting the PL07 bit in PLC0 register to "1" (PLL operation), wait until t_{su}(PLL) elapses before setting the CM11 bit to "1" (PLL clock).
Note 6: When entering stop mode from high- or middle-speed mode, or when the CM05 bit is set to "1" (main clock turned off) in low-speed mode, the CM15 bit is set to "1" (drive capability high).
Note 7: Effective when the CM06 bit is "0" (CM16 and CM17 bits enabled).

Figure 1.8.3 CM1 Register

Oscillation stop detection register (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
	X	0	0				

Symbol	Address	After reset
CM2	$000C_{16}$	$0X00X000_2$ (Note 2)

Bit symbol	Bit name	Function	RW
CM20	Oscillation stop, re-oscillation detection enable bit (Notes 2, 3, 4)	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, 5, 6, 7, 8, 11)	0 : Main clock or PLL clock (Ring oscillator turned off) 1 : Ring oscillator clock (Ring oscillator oscillating)	RW
CM22	Oscillation stop, re-oscillation detection flag (Note 9)	0 : Main clock stop, re-oscillation not detected 1 : Main clock stop, re-oscillation detected	RW
CM23	X_{IN} monitor flag (Note 10)	0 : Main clock oscillating 1 : Main clock turned off	RO
– (b5-b4)	Reserved bit	Set to "0"	RW
– (b6)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		–
CM27	Operation select bit (behavior if oscillation stop, re-oscillation is detected) (Note 2)	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: The CM20, CM21 and CM27 bits do not change at oscillation stop detection reset.
Note 3: Set the CM20 bit to "0" (disable) before entering stop mode. After exiting stop mode, set the CM20 bit back to "1" (enable).
Note 4: Set the CM20 bit to "0" (disable) before setting the CM05 bit of CM0 register
Note 5: 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 6: 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 7: Effective when the CM07 bit of CM0 register is "0".
Note 8: Where 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 CM11 bit is "1" (the CPU clock source is PLL clock), the CM21 bit remains unchanged even when main clock stop is detected. If the CM22 bit is "0" under these conditions, oscillation stop, re-oscillation detection interrupt generate at main clock stop detection; it is, therefore, necessary to set the CM21 bit to "1" (ring oscillator clock) inside the interrupt routine.
Note 9: This bit is set to "1" when the main clock is detected to have stopped and when the main clock is detected to have restarted oscillating. When this bit changes state from "0" to "1", an oscillation stop, re-oscillation detection interrupt request is generated. Use this bit in an interrupt routine to discriminate the causes of interrupts between the oscillation stop, re-oscillation detection interrupt and the watchdog timer interrupt. This bit is set to "0" by writing "0" in a program. (Writing "1" has no effect. Nor is it set to "0" by an oscillation stop, re-oscillation detection interrupt request acknowledged.)
If an oscillation stop or a re-oscillation is detected when the CM22 bit = 1, no oscillation stop and re-oscillation detection interrupt requests are generated.
Note 10: Read the CM23 bit in an oscillation stop, re-oscillation detection interrupt handling routine to determine the main clock status.
Note 11: When the CM21 bit = 0 (ring oscillator turned off) and the CM05 bit = 1 (main clock turned off), the CM06 bit is fixed to "1" (divide-by-8 mode) and the CM15 bit is fixed to "1" (drive capability High).

Figure 1.8.4 CM2 Register

Peripheral clock select register (Note)

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0		

Symbol	Address	After reset
PCLKR	$025E_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
PCLK0	Timers A, B, and A-D clock select bit (Clock source for the timers A, B, the dead time timer and A-D)	0 : Divide-by-2 of f_{AD2}, f_2 1 : f_{AD}, f_1	RW
PCLK1	SI/O clock select bit (Clock source for UART0 to UART2, SI/O3)	0 : f_{2SIO} 1 : f_{1SIO}	RW
– (b7-b2)	Reserved bit	Set to "0"	RW

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

Figure 1.8.5 PCLKR Register

CAN0/1 clock select register (Notes 1, 2)

b7	b6	b5	b4	b3	b2	b1	b0

Symbol	Address	After reset
CCLKR	$025F_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
CCLK0	CAN0 clock select bits	b2 b1 b0 0 0 0 No division 0 0 1 : Divide-by-2 0 1 0 : Divide-by-4 0 1 1 : Divide-by-8 1 0 0: Divide-by-16 1 0 1 : Inhibited 1 1 0 : Inhibited 1 1 1 : Inhibited	RW
CCLK1			RW
CCLK2			RW
CCLK3	CAN0 CPU interface sleep bit	0: CAN0 CPU interface operating 1: CAN0 CPU interface in sleep	RW
CCLK4	CAN1 clock select bits	b6 b5 b4 0 0 0 No division 0 0 1 : Divide-by-2 0 1 0 : Divide-by-4 0 1 1 : Divide-by-8 1 0 0: Divide-by-16 1 0 1 : Inhibited 1 1 0 : Inhibited 1 1 1 : Inhibited	RW
CCLK5			RW
CCLK6			RW
CCLK7	CAN1 CPU interface sleep bit	0: CAN1 CPU interface operating 1: CAN1 CPU interface in sleep	RW

Note 1: Write to this register after setting the PRC0 bit of PRCR register to "1" (write enable).
Note 2: Configuration of this register can be done only when the Reset bit of C0CTLR, C1CTLR registers = 1 (Reset/Initialization mode).

Figure 1.8.6 CCLKR Register

Processor mode register 2 (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
X	X	X	0	0		0	

Symbol: PM2 Address: 001E16 After reset: XXX000002

Bit symbol	Bit name	Function	RW
PM20	Specifying wait when accessing SFR at PLL operation (Note 2)	0 : 2 waits 1 : 1 wait	RW
- (b1)	Reserved bit	Set to "0"	RW
PM22	WDT count source protective bit (Notes 3, 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	Set to "0"	RW
- (b7-b5)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		–

Note 1: Write to this register after setting the PRC1 bit of PRCR register to "1" (write enable).
Note 2: This bit can only be rewritten while the PLC07 bit is "0" (PLL turned off). Also, to select a 16 MHz or higher PLL clock, set this bit to "0" (2 waits). Note that if the clock source for the CPU clock is to be changed from the PLL clock to another, the PLC07 bit must be set to "0" before setting the PM20 bit.
Note 3: Once this bit is set to "1", it cannot be set to "0" in a program.
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 1.8.7 PM2 Register

PLL control register 0 (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
	0	0	1	X			

Symbol	Address	After reset
PLC0	$001C_{16}$	$0001X010_2$

Bit symbol	Bit name	Function	RW
PLC00	PLL multiplying factor select bit (Note 2)	b2 b1 b0 0 0 0 : Must not be set 0 0 1 : Multiply by 2 0 1 0 : Multiply by 4 0 1 1 : Multiply by 6 1 0 0 : Multiply by 8 1 0 1 : 1 1 0 : } Must not be set 1 1 1 :	RW
PLC01			RW
PLC02			RW
– (b3)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		–
– (b4)	Reserved bit	Set to "1"	RW
– (b6-b5)	Reserved bit	Set to "0"	RW
PLC07	Operation enable bit (Note 3)	0 : PLL Off 1 : PLL On	RW

Note 1: Write to this register after setting the PRC0 bit of PRCR register to "1" (write enable).
Note 2: These three bits can only be modified when the PLC07 bit = 0 (PLL turned off). The value once written to this bit cannot be modified.
Note 3: Before setting this bit to "1", set the CM07 bit to "0" (main clock), set the CM17 to CM16 bits to "00_2" (main clock undivided mode), and set the CM06 bit to "0" (CM16 and CM17 bits enable).

Figure 1.8.8 PLC0 Register

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

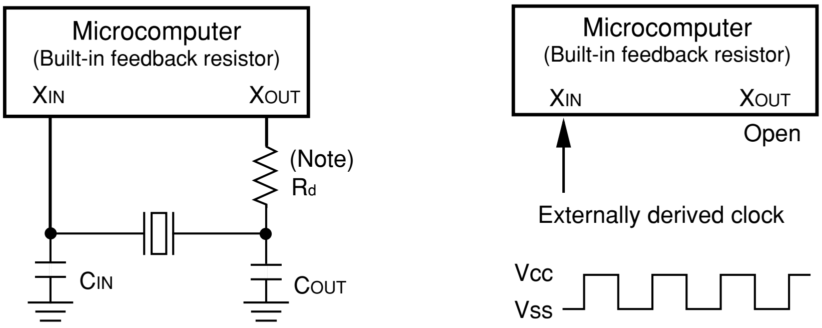
(1) Main Clock

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 X_{IN} and X_{OUT} pins. The main clock oscillator circuit contains a feedback resistor, which is disconnected from the oscillator circuit during stop mode in order to reduce the amount of power consumed in the chip. The main clock oscillator circuit may also be configured by feeding an externally generated clock to the X_{IN} pin. Figure 1.8.9 shows the examples of main clock connection circuit.

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

The power consumption in the chip can be reduced by setting the CM05 bit of CM0 register to "1" (main clock oscillator circuit turned off) after switching the clock source for the CPU clock to a sub clock or ring oscillator clock. In this case, X_{OUT} goes "H". Furthermore, because the internal feedback resistor remains on, X_{IN} is pulled "H" to X_{OUT} via the feedback resistor. Note, that if an externally generated clock is fed into the X_{IN} pin, the main clock cannot be turned off by setting the CM05 bit to "1" unless the sub clock is selected as a CPU clock. If necessary, use an external circuit to turn off the clock.

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



Note: 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.
When the oscillation drive capacity is set to low, check that oscillation is stable.
Also, if the oscillator manufacturer's data sheet specifies that a feedback resistor be added external to the chip, insert a feedback resistor between X_{IN} and X_{OUT} following the instruction.

Figure 1.8.9 Examples of Main Clock Connection Circuit

(2) Sub Clock

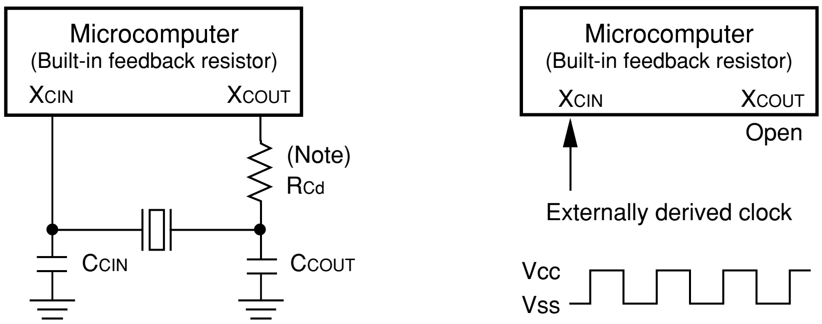
The sub clock is generated by the sub clock oscillation circuit. This clock is used as the clock source for the CPU clock, as well as the timer A and timer B count sources. In addition, an f_C clock with the same frequency as that of the sub clock can be output from the CLK_{OUT} pin.

The sub clock oscillator circuit is configured by connecting a crystal resonator between the X_{CIN} and X_{COUT} pins. The sub clock oscillator circuit contains a feedback resistor, which is disconnected from the oscillator circuit during stop mode in order to reduce the amount of power consumed in the chip. The sub clock oscillator circuit may also be configured by feeding an externally generated clock to the X_{CIN} pin. Figure 1.8.10 shows the examples of sub clock connection circuit.

After reset, the sub clock is turned off. At this time, the feedback resistor is disconnected from the oscillator circuit.

To use the sub clock for the CPU clock, set the CM07 bit of CM0 register to "1 " (sub clock) after the sub clock becomes oscillating stably.

During stop mode, all clocks including the sub clock are turned off. Refer to "power control".



Note: 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.
When the oscillation drive capacity is set to low, check that oscillation is stable.
Also, if the oscillator manufacturer's data sheet specifies that a feedback resistor be added external to the chip, insert a feedback resistor between X_{CIN} and X_{COUT} following the instruction.

Figure 1.8.10 Examples of Sub 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 (refer to “Watchdog Timer • Count source protective mode”).

After reset, the ring oscillator 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. If the main clock stops oscillating when the CM20 bit of CM2 register is “1” (oscillation stop, re-oscillation detection function enabled) and the CM27 bit is “1” (oscillation stop, re-oscillation detection interrupt), the ring oscillator automatically starts operating, supplying the necessary clock for the microcomputer.

(4) PLL Clock

The PLL clock is generated by a PLL frequency synthesizer. This clock is used as the clock source for the CPU and peripheral function clocks. After reset, the PLL clock is turned off. The PLL frequency synthesizer is activated by setting the PLC07 bit to “1” (PLL operation). When the PLL clock is used as the clock source for the CPU clock, wait a fixed period of t_{su}(PLL) for the PLL clock to be stable, and then set the CM11 bit in the CM1 register to “1”.

Before entering wait mode or stop mode, be sure to set the CM11 bit to “0” (CPU clock source is the main clock). Furthermore, before entering stop mode, be sure to set the PLC07 bit in the PLC0 register to “0” (PLL stops). Figure 1.8.11 shows the procedure for using the PLL clock as the clock source for the CPU.
The PLL clock frequency is determined by the equation below.

Figure 1.8.11 shows the procedure for using the PLL clock as the clock source for the CPU.
The PLL clock frequency is determined by the equation below.

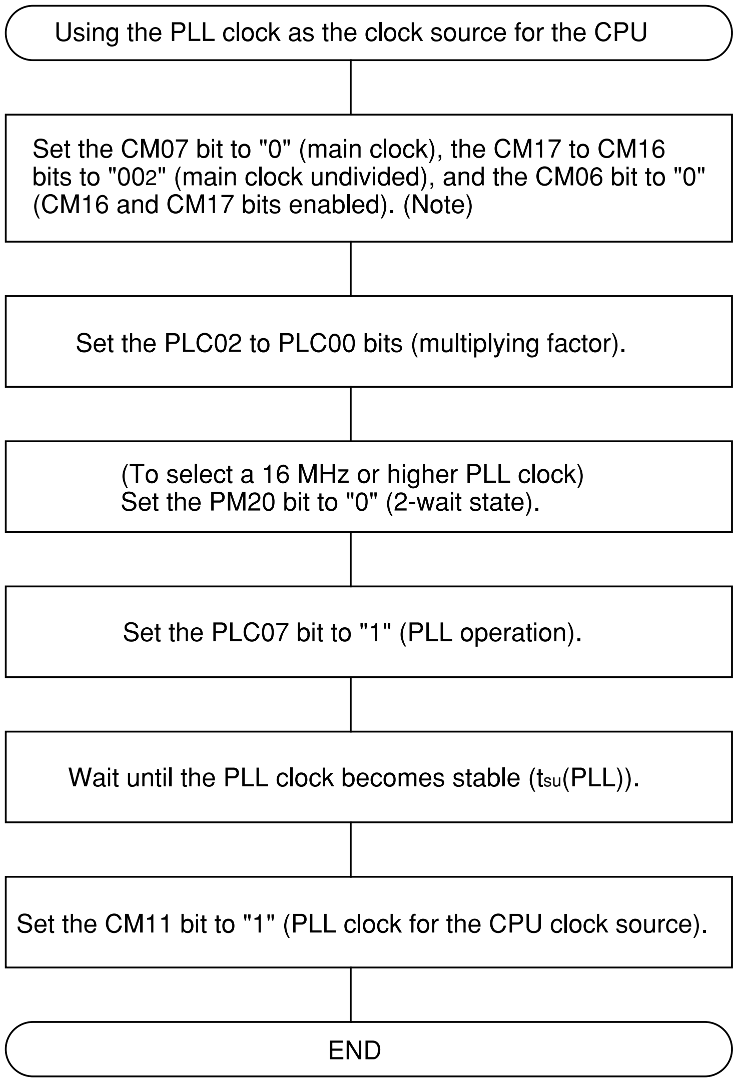
PLL clock frequency = $f(X_{IN}) \times$ (multiplying factor set by the PLC02 to PLC00 bits of the PLC0 register)
(However, PLL clock frequency = 20 MHz)

The PLC02 to PLC00 bits can be set only once after reset. Table 1.8.2 shows the example for setting PLL clock frequencies.

Table 1.8.2 Example for Setting PLL Clock Frequencies

X_{IN} (MHz)	PLC02	PLC01	PLC00	Multiply factor	PLL clock (MHz) (Note)
10	0	0	1	2	20
5	0	1	0	4	

Note: PLL clock frequency = 20 MHz



Note: PLL operation mode can be entered from high-speed mode.

Figure 1.8.11 Procedure to Use PLL Clock as CPU Clock Source

CPU Clock and Peripheral Function Clock

There are existing two type clocks: The CPU clock to operate the CPU and the 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, sub clock, ring oscillator clock or the PLL 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 of CM0 register and the CM17 to CM16 bits of CM1 register to select the divide-by-n value.
When the PLL clock is selected as the clock source for the CPU clock, the CM06 bit should be set to "0" and the CM17 to CM16 bits to "00_2" (undivided).
After reset, the main clock divided by 8 provides the CPU clock.
During memory expansion or microprocessor mode, a BCLK signal with the same frequency as the CPU clock can be output from the BCLK pin by setting the PM07 bit of PM0 register to "0" (output enabled).
Note that when entering stop mode from high- or middle-speed mode, ring oscillator mode or ring oscillator low power dissipation mode, or when the CM05 bit of CM0 register is set to "1" (main clock turned off) in low-speed 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}, f_{AD}, f_{CAN0}, f_{CAN1}, f_{C32})

These are operating clocks for the peripheral functions.
Two of these, f_i (i = 1, 2, 8, 32) and f_{iSIO} are derived from the main clock, PLL clock or ring oscillator clock by dividing them by i. The clock f_i is used for timers A and B, and f_{iSIO} is used for serial I/O. The f_8 and f_{32} clocks can be output from the CLK_{OUT} pin.
The f_{AD} clock is produced from the main clock, PLL clock or ring oscillator clock, and is used for the A-D converter.
The f_{CANi} (i = 0, 1) clock is derived from the main clock, PLL clock or ring oscillator clock by dividing them by 1 (undivided), 2, 4, 8 or 16, and is used for the CAN module.

When the WAIT instruction is executed after setting the CM02 bit of CM0 register to "1" (peripheral function clock turned off during wait mode), or when the microcomputer is in low power dissipation mode, the f_i, f_{iSIO}, f_{AD}, f_{CAN0} and f_{CAN1} clocks are turned off (Note).
The f_{C32} clock is derived from the sub clock, and is used for timers A and B. This clock can be used when the sub clock is activated.

Note: f_{CAN0} and f_{CAN1} clocks stop at "H" in CAN0, 1 sleep mode.

Clock Output Function

During single-chip mode, the f_8, f_{32} or f_C clock can be output from the CLK_{OUT} pin. Use the CM01 to CM00 bits of CM0 register to select.

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 seven sub 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, sub clock or PLL clock, allow a sufficient wait time in a program until it becomes oscillating stably.

Note that operation modes cannot be changed directly from low speed or low power dissipation mode to ring oscillator or ring oscillator low power dissipation mode. Nor can operation modes be changed directly from ring oscillator or ring oscillator low power dissipation mode to low speed or low power dissipation mode. Where the CPU clock source is changed from the ring oscillator to the main clock, change the operation mode to the medium-speed mode (divide-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. If the sub clock is activated, f_{C32} can be used as the count source for timers A and B.

- **PLL Operation Mode**

  The main clock multiplied by 2, 4, 6 or 8 provides the PLL clock, and this PLL clock serves as the CPU clock. If the sub clock is activated, f_{C32} can be used as the count source for timers A and B. PLL operation mode can be entered from high speed mode. If PLL operation mode is to be changed to wait or stop mode, first go to high speed mode before changing.

- **Medium-speed Mode**

  The main clock divided by 2, 4, 8 or 16 provides the CPU clock. If the sub clock is activated, f_{C32} can be used as the count source for timers A and B.

- **Low-speed Mode**

  The sub clock provides the CPU clock. The main clock is used as the clock source for the peripheral function clock when the CM21 bit is set to “0” (ring oscillator turned off), and the ring oscillator clock is used when the CM21 bit is set to “1” (ring oscillator oscillating).

  The f_{C32} clock can be used as the count source for timers A and B.

- **Low Power Dissipation Mode**

  In this mode, the main clock is turned off after being placed in low speed mode. The sub clock provides the CPU clock. The f_{C32} clock can be used as the count source for timers A and B.

  Simultaneously when this mode is selected, the CM06 bit of CM0 register becomes “1” (divide-by-8 mode). In the low power dissipation mode, do not change the CM06 bit. Consequently, the medium speed (divide-by-8) mode is to be selected when the main clock is operated next.

- **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. If the sub clock is activated, f_{C32} can be used as the count source for timers A and B.

- **Ring Oscillator Low Power Dissipation Mode**

  The main clock is turned off after being placed in ring oscillator mode. The CPU clock can be selected like in the ring oscillator mode. The ring oscillator clock is the clock source for the peripheral function clocks. If the sub clock is activated, f_{C32} can be used as the count source for timers A and B. When the operation mode is returned to the high- and medium-speed modes, set the CM06 bit to “1” (divide-by-8 mode).

  Table 1.8.3 lists the setting clock related bit and modes

Table 1.8.3 Setting Clock Related Bit and Modes

Modes		CM2 register	CM1 register		CM0 register			
		CM21	CM11	CM17, CM16	CM07	CM06	CM05	CM04
PLL operation mode		0	1	00_2	0	0	0	-
High-speed mode		0	0	00_2	0	0	0	-
Medium-speed mode	divided by 2	0	0	01_2	0	0	0	-
	divided by 4	0	0	10_2	0	0	0	-
	divided by 8	0	0	-	0	1	0	-
	divided by 16	0	0	11_2	0	0	0	-
Low-speed mode		-	-	-	1	-	0	1
Low power dissipation mode		-	-	-	1	1 (Note 1)	1 (Note 1)	1
Ring oscillator mode	divided by 1	1	-	00_2	0	0	0	-
	divided by 2	1	-	01_2	0	0	0	-
	divided by 4	1	-	10_2	0	0	0	-
	divided by 8	1	-	-	0	1	0	-
	divided by 16	1	-	11_2	0	0	0	-
Ring oscillator low power dissipation mode		1	-	(Note 2)	0	(Note 2)	1	-

Note 1: When the CM05 bit is set to “1” (main clock turned off) in low-speed mode, the mode goes to low power dissipation mode and CM06 bit is set to “1” (divide-by-8 mode) simultaneously.

Note 2: 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, sub clock, ring oscillator clock and PLL clock all 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}, f_{32SIO}, f_{AD}, f_{CAN0} and f_{CAN1} clocks are turned off when in wait mode, with the power consumption reduced that much. However, f_{C32} remains on.

- **Entering Wait Mode**

  The microcomputer is placed into wait mode by executing the WAIT instruction.
  When the CM11 bit = 1 (CPU clock source is the PLL clock), be sure to set the CM11 bit to “0” (CPU clock source is the main clock) before going to wait mode. The power consumption of the chip can be reduced by setting the PLC07 bit to “0” (PLL stops).

- **Pin Status During Wait Mode**

  Table 1.8.4 lists the pin status during wait mode.

Table 1.8.4 Pin Status During Wait Mode

Pin		Memory expansion mode Microprocessor mode	Single-chip mode
A_0 to A_{19}, D_0 to D_{15}, $\overline{CS_0}$ to $\overline{CS_3}$, $\overline{BHE}$		Retains status before wait mode	-
$\overline{RD}$, $\overline{WR}$, $\overline{WRL}$, $\overline{WRH}$		“H”	-
$\overline{HLDA}$, BCLK		“H”	-
ALE		“H”	-
I/O ports		Retains status before wait mode	Retains status before wait mode
CLK_{OUT}	When f_C selected	-	Does not stop
	When f_8, f_{32} selected	-	•CM02 bit = 0: Does not stop •CM02 bit = 1: Retains status before wait mode

- **Exiting Wait Mode**

  The microcomputer is moved out of wait mode by a hardware reset, $\overline{NMI}$ interrupt or peripheral function interrupt.
  If the microcomputer is to be moved out of exit wait mode by a hardware reset or $\overline{NMI}$ interrupt, 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 the 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 the 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 1.8.5 lists the interrupts to exit wait mode.

Table 1.8.5 Interrupts to Exit Wait Mode

Interrupt	CM02 = 0	CM02 = 1
$\overline{\text{NMI}}$ interrupt	Can be used	Can be used
Serial I/O interrupt	Can be used when operating with internal or external clock	Can be used when operating with external clock
Key input interrupt	Can be used	Can be used
A-D conversion interrupt	Can be used in one-shot mode or single sweep mode	- (Do not use)
Timer A interrupt Timer B interrupt	Can be used in all modes	Can be used in event counter mode or when the count source is f_{c32}
$\overline{\text{INT}}$ interrupt	Can be used	Can be used
CAN0/1 Wake-up interrupt	Can be used	Can be used

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

1. In the ILVL2 to ILVL0 bits of interrupt control register, set the interrupt priority level of the peripheral function interrupt to be used to exit wait mode.
   Also, for all of the peripheral function interrupts not used to exit wait mode, set the ILVL2 to ILVL0 bits to “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 V_{CC} 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.

- $\overline{NMI}$ interrupt
- Key interrupt
- $\overline{INT}$ interrupt
- Timer A, Timer B interrupt (when counting external pulses in event counter mode)
- Serial I/O interrupt (when external clock is selected)
- CAN0/1 Wake-up interrupt

- **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) and the CM15 bit of CM1 register is set to "1" (main clock oscillator circuit drive capability high).

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

  Also, if the CM11 bit is "1" (PLL clock for the CPU clock source), set the CM11 bit to "0" (main clock for the CPU clock source) and the PLC07 bit to "0" (PLL turned off) before entering stop mode.

- **Pin Status During Stop Mode**

  Table 1.8.6 lists the pin status during stop mode.

Table 1.8.6 Pin Status During Stop Mode

Pin		Memory expansion mode Microprocessor mode	Single-chip mode
A_0 to A_{19}, D_0 to D_{15}, $\overline{CS_0}$ to $\overline{CS_3}$, $\overline{BHE}$		Retains status before stop mode	-
$\overline{RD}$, $\overline{WR}$, $\overline{WRL}$, $\overline{WRH}$		"H"	-
$\overline{HLDA}$, BCLK		"H"	-
ALE		"H"	-
I/O ports		Retains status before stop mode	Retains status before stop mode
CLK_{OUT}	When f_C selected	-	"H"
	When f_8, f_{32} selected	-	Retains status before stop mode

• Exiting Stop Mode

The microcomputer is moved out of stop mode by a hardware reset, $\overline{NMI}$ interrupt or peripheral function interrupt.

If the microcomputer is to be moved out of stop mode by a hardware reset or $\overline{NMI}$ interrupt, 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".

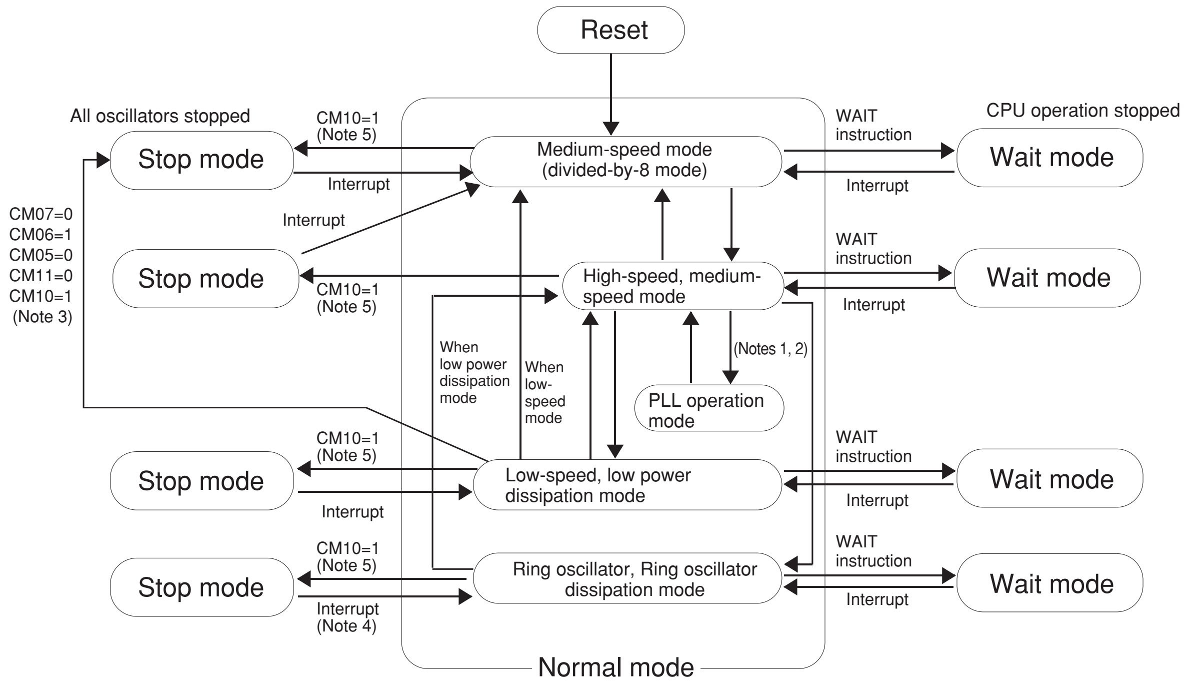
1. In the ILVL2 to ILVL0 bits of interrupt control register, set the interrupt priority level of the peripheral function interrupt to be used to exit stop mode.
   Also, for all of the peripheral function interrupts not used to exit stop mode, set the ILVL2 to ILVL0 bits to "000_2".
2. Set the I flag 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 or $\overline{NMI}$ interrupt 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 sub clock: sub clock
- 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

Figure 1.8.12 shows the state transition from normal operation mode to stop mode and wait mode. Figure 1.8.13 shows the state transition in normal operation mode.

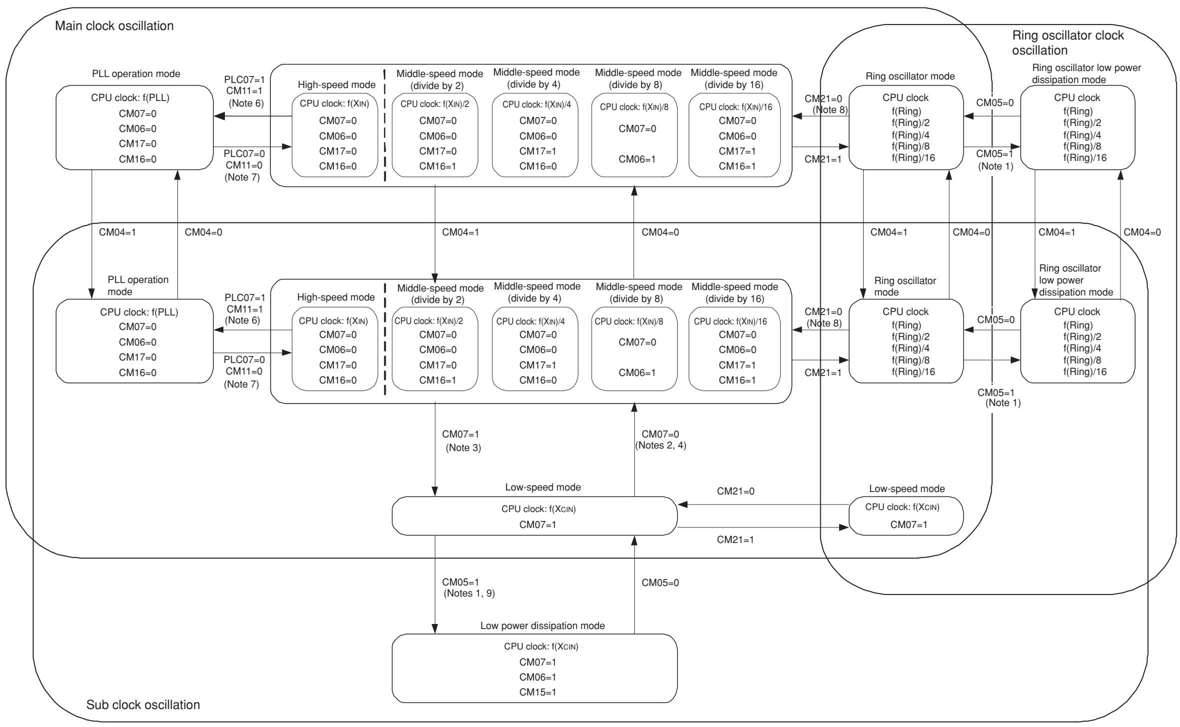
Table 1.8.7 shows a state transition matrix describing allowed transition and setting. The vertical line shows current state and horizontal line show state after transition.



CM05, CM06, CM07 : CM0 register's bits
CM10, CM11 : CM1 register's bits

Note 1: Do not go directly from PLL operation mode to wait or stop mode.
Note 2: PLL operation mode can be entered from high speed mode. Similarly, PLL operation mode can be changed back to high speed mode.
Note 3: Write to the CM0 register and CM1 register simultaneously by accessing in word unit while CM21 = 0 (ring oscillator turned off).
Note 4: The ring oscillator clock divided by 8 provides the CPU clock.
Note 5: Before entering stop mode, be sure to set the CM20 bit in the CM2 register to "0" (oscillation stop, re-oscillation detection function disabled).

Figure 1.8.12 State Transition to Stop Mode and Wait Mode



CM04, CM05, CM06, CM07 : CM0 register's bits
CM11, CM15, CM16, CM17 : CM1 register's bits
CM20, CM21 : CM2 register's bits
PLC07 : PLC0 register's bit

Note 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.
Note 2: Wait for $t_{d(M-L)}$ or the main clock oscillation stabilization time whichever is longer before switching over.
Note 3: Switch clock after oscillation of sub clock is sufficiently stable.
Note 4: Change CM17 and CM16 before changing CM06.
Note 5: Transit in accordance with arrow.
Note 6: PLL operation mode can only be entered from high speed mode. Also, wait until the PLL clock is sufficiently stable before changing operation modes. To select a 16 MHz or higher PLL clock, set the PM20 bit to "0" (SFR accessed with two wait states) before setting PLC07 to "1" (PLL operation).
Note 7: PLL operation mode can only be changed to high speed mode. If the PM20 bit = 0 (SFR accessed with two wait states), set PLC07 to "0" (PLL turned off) before setting the PM20 bit to "1" (SFR accessed with one wait state).
Note 8: 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.
Note 9: When the CM21 bit = 0 (ring oscillator turned off) and the CM05 bit = 1 (main clock turned off), the CM06 bit is fixed to "1" (divide-by-8 mode) and the CM15 bit is fixed to "1" (drive capability High).

Figure 1.8.13 State Transition in Normal Operation Mode

Table 1.8.7 Allowed Transition and Setting

Current state	State after transition: High-speed mode, middle-speed mode	Low-speed mode (Note 2)	Low power dissipation mode	PLL operation mode (Note 2)	Ring oscillator mode	Ring oscillator low power dissipation mode	Stop mode	Wait mode
High-speed mode, Middle-speed mode	(Note 8)	(9) (Note 7)	–	(13) (Note 3)	(15)	–	(16) (Note 1)	(17)
Low-speed mode (Note 2)	(8)		(11) (Notes 1, 6)	–	–	–	(16) (Note 1)	(17)
Low power dissipation mode	–	(10)		–	–	–	(16) (Note 1)	(17)
PLL operation mode (Note 2)	(12) (Note 3)	–	–		–	–	–	–
Ring oscillator mode	(14) (Note 4)	–	–	–	(Note 8)	(11) (Note 1)	(16) (Note 1)	(17)
Ring oscillator low power dissipation mode	–	–	–	–	(10)	(Note 8)	(16) (Note 1)	(17)
Stop mode	(18) (Note 5)	(18)	(18)	–	(18) (Note 5)	(18) (Note 5)		–
Wait mode	(18)	(18)	(18)	–	(18)	(18)	–	

-: Cannot transit

Note 1: Avoid making a transition when the CM20 bit = 1 (oscillation stop, re-oscillation detection function enabled). Set the CM20 bit to "0" (oscillation stop, re-oscillation detection function disabled) before transiting.

Note 2: Ring oscillator clock oscillates and stops in low-speed mode. In this mode, the ring oscillator can be used as peripheral function clock. Sub clock oscillates and stops in PLL operation mode. In this mode, sub clock can be used as peripheral function clock.

Note 3: PLL operation mode can only be entered from and changed to high-speed mode.

Note 4: Set the CM06 bit to "1" (division by 8 mode) before transiting from ring oscillator mode to high- or middle-speed mode.

Note 5: When exiting stop mode, the CM06 bit is set to "1" (division by 8 mode).

Note 6: If the CM05 bit is set to "1" (main clock stop), then the CM06 bit is set to "1" (division by 8 mode).

Note 7: A transition can be made only when sub clock is oscillating.

Note 8: State transitions within the same mode (divide-by-n values changed or sub clock oscillation turned on or off) are shown in the table below.

		Sub clock oscillating: No division	Divided by 2	Divided by 4	Divided by 8	Divided by 16	Sub clock turned off: No division	Divided by 2	Divided by 4	Divided by 8	Divided by 16
Sub clock oscillating	No division		(4)	(5)	(7)	(6)	(1)	–	–	–	–
	Divided by 2	(3)		(5)	(7)	(6)	–	(1)	–	–	–
	Divided by 4	(3)	(4)		(7)	(6)	–	–	(1)	–	–
	Divided by 8	(3)	(4)	(5)		(6)	–	–	–	(1)	–
	Divided by 16	(3)	(4)	(5)	(7)		–	–	–	–	(1)
Sub clock turned off	No division	(2)	–	–	–	–		(4)	(5)	(7)	(6)
	Divided by 2	–	(2)	–	–	–	(3)		(5)	(7)	(6)
	Divided by 4	–	–	(2)	–	–	(3)	(4)		(7)	(6)
	Divided by 8	–	–	–	(2)	–	(3)	(4)	(5)		(6)
	Divided by 16	–	–	–	–	(2)	(3)	(4)	(5)	(7)	

Note 9: ( ):setting method. Refer to right table.

	Setting	Operation
(1)	CM04=0	Sub clock turned off
(2)	CM04=1	Sub clock oscillating
(3)	CM06=0 CM17=0 CM16=0	CPU clock no division mode
(4)	CM06=0 CM17=0 CM16=1	CPU clock division by 2 mode
(5)	CM06=0 CM17=1 CM16=0	CPU clock division by 4 mode
(6)	CM06=0 CM17=1 CM16=1	CPU clock division by 16 mode
(7)	CM06=1	CPU clock division by 8 mode
(8)	CM07=0	Main clock, PLL clock or ring oscillator clock selected
(9)	CM07=1	Sub clock selected
(10)	CM05=0	Main clock oscillating
(11)	CM05=1	Main clock turned off
(12)	PLC07=0 CM11=0	Main clock selected
(13)	PLC07=1 CM11=1	PLL clock selected
(14)	CM21=0	Main clock or PLL clock selected
(15)	CM21=1	Ring oscillator clock selected
(16)	CM10=1	Transition to stop mode
(17)	WAIT instruction	Transition to wait mode
(18)	Hardware interrupt	Exit stop mode or wait mode

CM04, CM05, CM06, CM07 :CM0 register's bits
CM10, CM11, CM16, CM17 :CM1 register's bits
CM20, CM21 :CM2 register's bits
PLC07 :PLC0 register's bit

Oscillation Stop and Re-oscillation Detection Function

The oscillation stop and re-oscillation detection 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 one is to be generated can be selected using the CM27 bit of CM2 register.

The oscillation stop and re-oscillation detection function can be enabled or disabled using the CM20 bit of CM2 register.

Table 1.8.8 lists a specification overview of the oscillation stop and re-oscillation detection function.

Table 1.8.8 Specification Overview of Oscillation Stop and Re-oscillation Detection Function

Item	Specification
Oscillation stop detectable clock and frequency bandwidth	$f(X_{IN}) \geq 2$ MHz
Enabling condition for oscillation stop and re-oscillation detection function	Set CM20 bit to “1” (enable)
Operation at oscillation stop, re-oscillation detection	•Reset occurs (when CM27 bit = 0) •Oscillation stop, re-oscillation detection interrupt occurs (when the CM27 bit =1)

(1) Operation When CM27 Bit = 0 (Oscillation Stop Detection Reset)

Where main clock stop is detected when the CM20 bit is “1” (oscillation stop, re-oscillation detection function enabled), the microcomputer is initialized, coming to a halt (oscillation stop reset; refer to “SFR”, “Reset”).

This status is reset with hardware reset. Also, even when re-oscillation is detected, the microcomputer can be initialized and stopped; it is, however, necessary to avoid such usage. (During main clock stop, do not set the CM20 bit to “1” and the CM27 bit to “0”.)

(2) Operation When CM27 Bit = 1 (Oscillation Stop, Re-oscillation Detection Interrupt)

Where the main clock corresponds to the CPU clock source and the CM20 bit is “1” (oscillation stop, re-oscillation detection function enabled), the system is placed in the following state if the main clock comes to a halt:

- Oscillation stop, re-oscillation detection interrupt request occurs.
- The ring oscillator starts oscillation, and the ring oscillator clock becomes the clock source for CPU clock and peripheral functions in place of the main clock.
- CM21 bit = 1 (ring oscillator clock is the clock source for CPU clock)
- CM22 bit = 1 (main clock stop detected)
- CM23 bit = 1 (main clock stopped)

Where the PLL clock corresponds to the CPU clock source and the CM20 bit is “1”, the system is placed in the following state if the main clock comes to a halt: Since the CM21 bit remains unchanged, set it to “1” (ring oscillator clock) inside the interrupt routine.

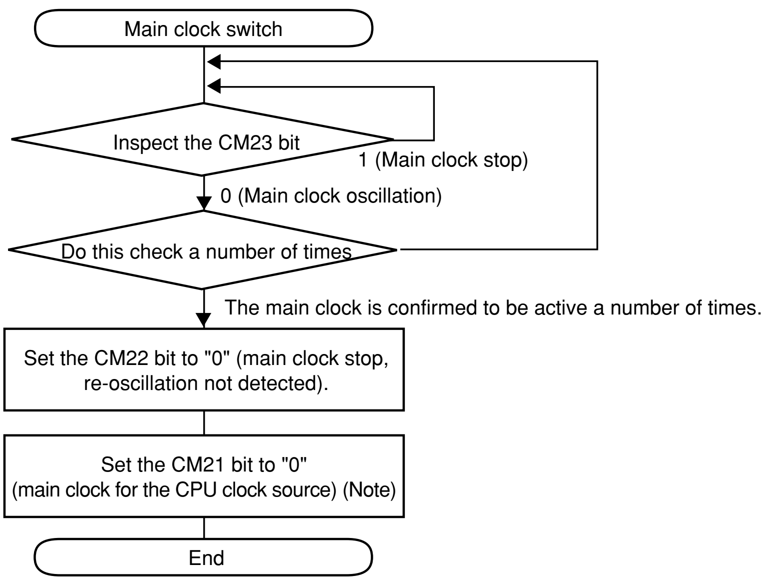
- Oscillation stop, re-oscillation detection interrupt request occurs.
- CM22 bit = 1 (main clock stop detected)
- CM23 bit = 1 (main clock stopped)
- CM21 bit remains unchanged

Where the CM20 bit is “1”, the system is placed in the following state if the main clock re-oscillates from the stop condition:

- Oscillation stop, re-oscillation detection interrupt request occurs.
- CM22 bit = 1 (main clock re-oscillation detected)
- CM23 bit = 0 (main clock oscillation)
- CM21 bit remains unchanged

How to Use Oscillation Stop and Re-oscillation Detection Function

- The oscillation stop, re-oscillation detection interrupt shares the vector with the watchdog timer interrupt. If the oscillation stop, re-oscillation detection and watchdog timer interrupts both are used, read the CM22 bit in an interrupt routine to determine which interrupt source is requesting the interrupt.
- Where the main clock re-oscillated after oscillation stop, the clock source for CPU clock and peripheral function must be switched to the main clock in the program. Figure 1.8.14 shows the procedure to switch the clock source from the ring oscillator to the main clock.
- Simultaneously with oscillation stop, re-oscillation detection interrupt request occurrence, the CM22 bit becomes “1”. When the CM22 bit is set at “1”, oscillation stop, re-oscillation detection interrupt are disabled. By setting the CM22 bit to “0” in the program, oscillation stop, re-oscillation detection interrupt are enabled.
- If the main clock stops during low speed mode where the CM20 bit is “1”, an oscillation stop, re-oscillation detection interrupt request is generated. At the same time, the ring oscillator starts oscillating. In this case, although the CPU clock is derived from the sub clock as it was before the interrupt occurred, the peripheral function clocks now are derived from the ring oscillator clock.
- To enter wait mode while using the oscillation stop and re-oscillation detection function, set the CM02 bit to “0” (peripheral function clocks not turned off during wait mode).
- Since the oscillation stop and re-oscillation detection function is provided in preparation for main clock stop due to external factors, set the CM20 bit to “0” (Oscillation stop, re-oscillation detection function disabled) where the main clock is stopped or oscillated in the program, that is where the stop mode is selected or the CM05 bit is altered.
- This function cannot be used if the main clock frequency is 2 MHz or less. In that case, set the CM20 bit to “0”.



CM21, CM22, CM 23 bits are the CM2 register bits

Note: If the clock source for CPU clock is to be changed to PLL clock, set to PLL operation mode after set to high-speed mode.

Figure 1.8.14 Procedure to Switch Clock Source from Ring Oscillator to Main Clock

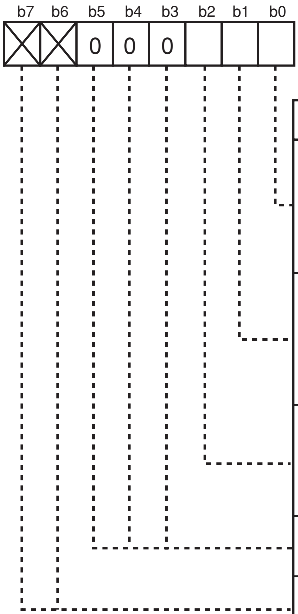
Protection

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 1.9.1 shows the PRCR register. The following lists the registers protected by the PRCR register.

- Registers protected by the PRC0 bit: CM0, CM1, CM2, PLC0, PCLKR and CCLKR registers
- Registers protected by the PRC1 bit: PM0, PM1, PM2, TB2SC, INVC0 and INVC1 registers
- Registers protected by the PRC2 bit: PD7, PD9 and S3C registers

Set the PRC2 bit to “1” (write enabled) and then write to any address, and the PRC2 bit will be set 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 set to “0” by writing to any address. They can only be set to “0” 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, PCLKR, CCLKR registers 0 : Write protected 1 : Write enabled	RW
PRC1	Protect bit 1	Enable write to PM0, PM1, PM2, TB2SC, INVC0, INVC1 registers 0 : Write protected 1 : Write enabled	RW
PRC2	Protect bit 2	Enable write to PD7, PD9, S3C registers 0 : Write protected 1 : Write enabled (Note)	RW
– (b5-b3)	Reserved bit	Set to "0"	RW
– (b7-b6)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		–

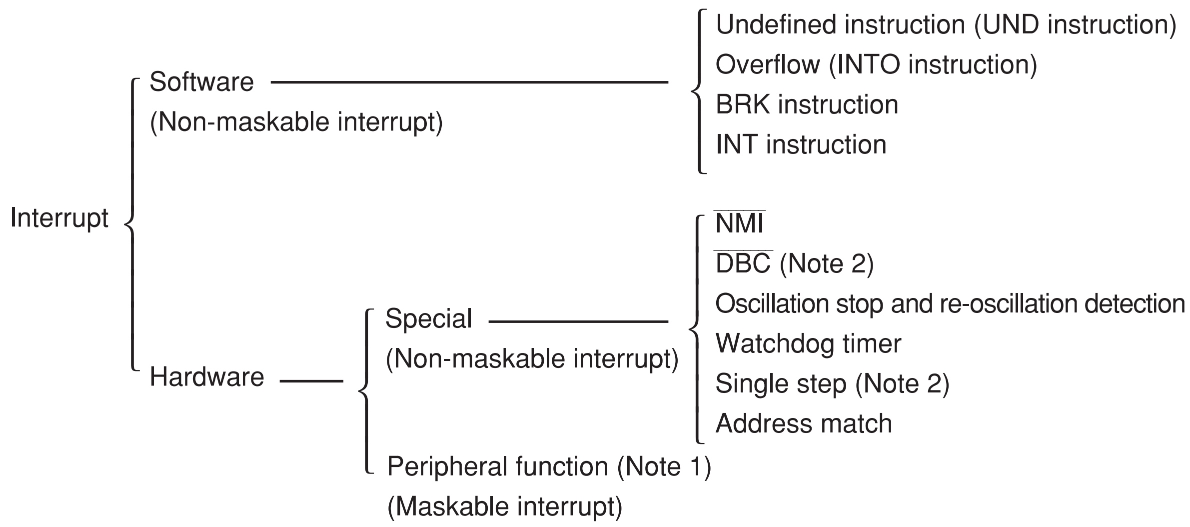
Note: The PRC2 bit is set to "0" by writing to any address after setting it to "1". Other bits are not set to "0" by writing to any address, and must therefore be set in a program.

Figure 1.9.1 PRCR Register

Interrupts

Type of Interrupts

Figure 1.10.1 shows the 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 1.10.1 Interrupts

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

Software Interrupts

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

- **Undefined Instruction Interrupt**
  An undefined instruction interrupt occurs when executing the UND instruction.
- **Overflow Interrupt**
  An overflow interrupt occurs when executing the INTO instruction with the O flag set to “1” (the operation resulted in an overflow). The following are instructions whose O flag changes by arithmetic:
  ABS, ADC, ADCF, ADD, CMP, DIV, DIVU, DIVX, NEG, RMPA, SBB, SHA, SUB
- **BRK Interrupt**
  A BRK interrupt occurs when executing the BRK instruction.
- **INT Instruction Interrupt**
  An INT instruction interrupt occurs when executing the INT instruction. Software interrupt Nos. 0 to 63 can be specified for the INT instruction. Because software interrupt Nos. 1 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 set 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.

- **$\overline{\text{NMI}}$ Interrupt**

  An $\overline{\text{NMI}}$ interrupt is generated when input on the $\overline{\text{NMI}}$ pin changes state from high to low. For details, refer to "$\overline{\text{NMI}}$ Interrupt".

- **$\overline{\text{DBC}}$ 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 "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 and re-oscillation detection function, refer to "Clock Generation 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 registers 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, refer to "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 1.10.2 Relocatable Vector Tables”.

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 1.10.2 shows the interrupt vector.

	MSB	LSB
Vector address (L)	Low address	
	Medium address	
	0 0 0 0	High address
Vector address (H)	0 0 0 0	0 0 0 0

Figure 1.10.2 Interrupt Vector

- **Fixed Vector Tables**

The fixed vector tables are allocated to the addresses from FFFDC$_{16}$ to FFFFF$_{16}$. Table 1.10.1 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 "Functions to Prevent Flash Memory from Rewriting".

Table 1.10.1 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 series software manual
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$_{16}$		
Oscillation stop and re-oscillation detection, Watchdog timer	FFFF0$_{16}$ to FFFF3$_{16}$		Clock generation circuit Watchdog timer
$\overline{\text{DBC}}$ (Note)	FFFF4$_{16}$ to FFFF7$_{16}$		
$\overline{\text{NMI}}$	FFFF8$_{16}$ to FFFFB$_{16}$		$\overline{\text{NMI}}$ interrupt
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 relocatable vector table area. Table 1.10.2 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 1.10.2 Relocatable Vector Tables

Interrupt source	Vector address (Note 1) Address (L) to address (H)	Software interrupt number	Reference
BRK instruction (Note 2)	+0 to +3(0000_{16} to 0003_{16})	0	M16C/60, M16C/20 series software manual
CAN0/1 wake-up	+4 to +7 (0004_{16} to 0007_{16})	1	CAN module
CAN0 successful reception	+8 to +11 (0008_{16} to $000B_{16}$)	2	
CAN0 successful transmission	+12 to +15 ($000C_{16}$ to $000F_{16}$)	3	
$\overline{INT3}$	+16 to +19 (0010_{16} to 0013_{16})	4	$\overline{INT}$ interrupt
Timer B5	+20 to +23 (0014_{16} to 0017_{16})	5	Timer
Timer B4, UART1 bus collision detection (Note 3. 9)	+24 to +27 (0018_{16} to $001B_{16}$)	6	Timer, Serial I/O
Timer B3, UART0 bus collision detection (Note 4, 9)	+28 to +31 ($001C_{16}$ to $001F_{16}$)	7	
CAN1 successful reception, $\overline{INT5}$ (Note 5)	+32 to +35 (0020_{16} to 0023_{16})	8	CAN module, $\overline{INT}$ interrupt
SIO3, CAN1 successful transmission, $\overline{INT4}$ (Note 6)	+36 to +39 (0024_{16} to 0027_{16})	9	Serial I/O, CAN module, $\overline{INT}$ interrupt
UART2 bus collision detection (Note 9)	+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	
CAN0/1 error	+52 to +55 (0034_{16} to 0037_{16})	13	CAN module
A-D, Key input (Note 7)	+56 to +59 (0038_{16} to $003B_{16}$)	14	A-D convertor, Key input interrupt
UART2 transmission, NACK2 (Note 8)	+60 to +63 ($003C_{16}$ to $003F_{16}$)	15	Serial I/O
UART2 reception, ACK2 (Note 8)	+64 to +67 (0040_{16} to 0043_{16})	16	
UART0 transmission, NACK0 (Note 8)	+68 to +71 (0044_{16} to 0047_{16})	17	
UART0 reception, ACK0 (Note 8)	+72 to +75 (0048_{16} to $004B_{16}$)	18	
UART1 transmission, NACK1 (Note 8)	+76 to +79 ($004C_{16}$ to $004F_{16}$)	19	
UART1 reception, ACK1 (Note 8)	+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	+100to +103 (0064_{16} to 0067_{16})	25	
Timer B0	+104to +107 (0068_{16} to $006B_{16}$)	26	
Timer B1	+108to +111 ($006C_{16}$ to $006F_{16}$)	27	
Timer B2	+112to +115 (0070_{16} to 0073_{16})	28	
$\overline{INT0}$	+116to +119 (0074_{16} to 0077_{16})	29	$\overline{INT}$ interrupt
$\overline{INT1}$	+120to +123 (0078_{16} to $007B_{16}$)	30	
$\overline{INT2}$	+124to +127 ($007C_{16}$ to $007F_{16}$)	31	
Software interrupt (Note 2)	+128to +131 (0080_{16} to 0083_{16}) ⋮ +252to +255 ($00FC_{16}$ to $00FF_{16}$)	32 ⋮ 63	M16C/60, M16C/20 series software manual

Note 1: Address relative to address in INTB.
Note 2: These interrupts cannot be disabled using the I flag.
Note 3: Use the IFSR0 register's IFSR07 bit to select.
Note 4: Use the IFSR0 register's IFSR06 bit to select.
Note 5: Use the IFSR1 register's IFSR17 bit to select.
Note 6: Use the IFSR1 register's IFSR16 bit to select.
Furthermore, use the IFSR0 register's IFSR00 bit to select, when selecting SI/O3 or CAN1 successful transmission.
Note 7: Use the IFSR0 register's IFSR01 bit to select.
Note 8: During I^2C mode, NACK and ACK interrupts comprise the interrupt source.
Note 9: Bus collision detection: During IE mode, this bus collision detection constitutes the cause of an interrupt.
During I^2C mode, a start condition or a stop condition detection constitutes the cause of an interrupt.

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 non-maskable 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.

Figures 1.10.3 and 1.10.4 show the interrupt control registers.

Interrupt control register (Note 1)

Symbol	Address	After reset
C01WKIC	0041_{16}	$XXXXX000_2$
C0RECIC	0042_{16}	$XXXXX000_2$
C0TRMIC	0043_{16}	$XXXXX000_2$
TB5IC	0045_{16}	$XXXXX000_2$
TB4IC/U1BCNIC (Note 2)	0046_{16}	$XXXXX000_2$
TB3IC/U0BCNIC (Note 3)	0047_{16}	$XXXXX000_2$
U2BCNIC	$004A_{16}$	$XXXXX000_2$
DM0IC, DM1IC	$004B_{16}$, $004C_{16}$	$XXXXX000_2$
C01ERRIC	$004D_{16}$	$XXXXX000_2$
ADIC/KUPIC	$004E_{16}$	$XXXXX000_2$
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}	$XXXXX000_2$
TB0IC to TB2IC	$005A_{16}$ to $005C_{16}$	$XXXXX000_2$

b7 b6 b5 b4 b3 b2 b1 b0

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 4)
– (b7-b4)	Noting is assigned. When write, set to "0". When read, their contents are indeterminate.		–

Note 1: To rewrite the interrupt control registers, do so at a point that does not generate the interrupt request for that register. For details, refer to "Precautions for Interrupts" of the Usage Notes Reference Book.
Note 2: Use the IFSR07 bit of IFSR0 register to select.
Note 3: Use the IFSR06 bit of IFSR0 register to select.
Note 4: This bit can only be reset by writing "0" (Do not write "1").

Figure 1.10.3 Interrupt Control Registers (1)

Interrupt control register (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0
X X 0

Symbol	Address	After reset
INT3IC (Note 2)	0044_{16}	$XX00X000_2$
C1RECIC/INT5IC	0048_{16}	$XX00X000_2$
C1TRMIC/S3IC/INT4IC	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 3)
POL	Polarity select bit	0 : Selects falling edge (Notes 4, 5) 1 : Selects rising edge	RW
– (b5)	Reserved bit	Set to "0"	RW
– (b7-b6)	Nothing is assigned. When write, set to "0". When read, their contents are indeterminate.		–

Note 1: To rewrite the interrupt control registers, do so at a point that does not generate the interrupt request for that register. For details, refer to "Precautions for Interrupts" of the Usage Notes Reference Book.
Note 2: When the BYTE pin is low and the processor mode is memory expansion or microprocessor mode, set the ILVL2 to ILVL0 bits in the INT5IC to INT3IC registers to "000_2" (interrupt disabled).
Note 3: This bit can only be reset by writing "0" (Do not write "1").
Note 4: If the IFSR1 register's IFSR1i bit (i = 0 to 5) is "1" (both edges), set the INTiIC register's POL bit to "0" (falling edge).
Note 5: Set the S3IC register's POL bit to "0" (falling edge) when the IFSR0 register's IFSR00 bit = 1 and the IFSR1 register's IFSR16 bit = 0 (SI/O3 selected).

Figure 1.10.4 Interrupt Control Registers (2)

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 set to "0" (interrupt not requested).
The IR bit can be set 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 1.10.3 shows the settings of interrupt priority levels and Table 1.10.4 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 1.10.3 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 1.10.4 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 5 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

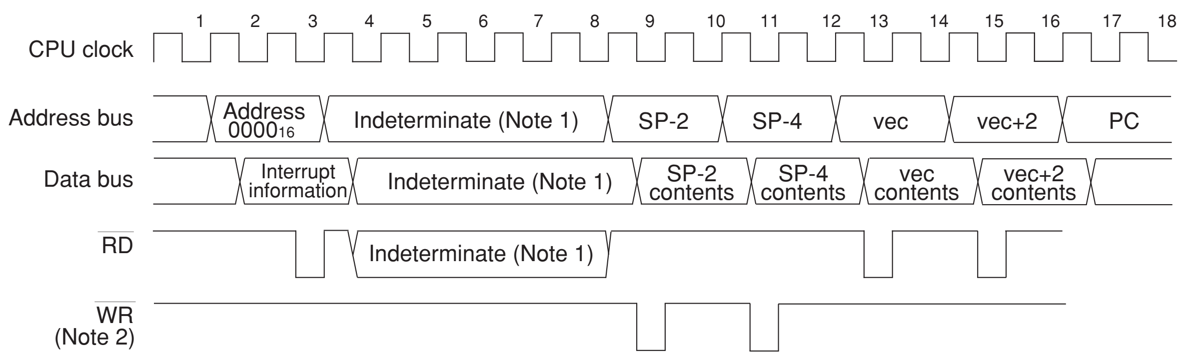
An interrupt sequence — what are performed over a period from the instant an interrupt is accepted to the instant the interrupt routine is executed — is described here.
If an interrupt occurs during execution of an instruction, the processor determines its priority when the execution of the instruction 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 1.10.5 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 set 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).
(3) The I, D and U flags in the FLG register become as follows:
- The I flag = 0 (interrupts disabled).
- The D flag = 0 (single-step interrupt disabled).
- The U flag = 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) 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.

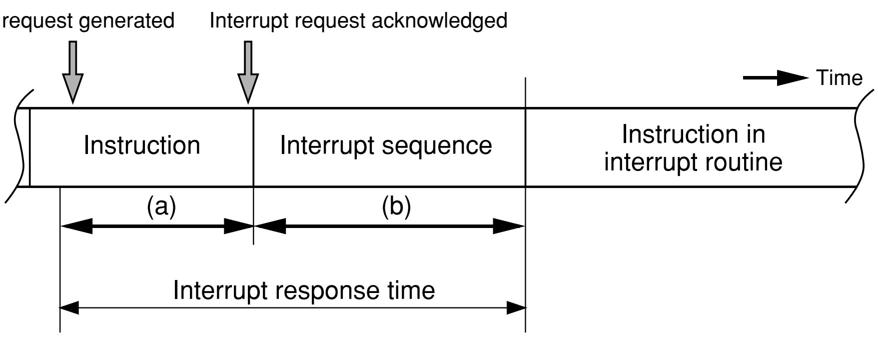


Note 1: The indeterminate state depends on the instruction queue buffer.
A read cycle occurs when the instruction queue buffer is ready to accept instructions.
Note 2: The WR signal timing shown here is for the case where the stack is located in the internal RAM.

Figure 1.10.5 Time Required for Executing Interrupt Sequence

Interrupt Response Time

Figure 1.10.6 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 1.10.6) and a time during which the interrupt sequence is executed ((b) in Figure 1.10.6).



(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
	Odd	19 cycles	
Odd	Even	19 cycles	
	Odd	20 cycles	

Figure 1.10.6 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 1.10.5 is set in the IPL. Table 1.10.5 shows the IPL values of software and special interrupts when they are accepted.

Table 1.10.5 IPL Level that is Set to IPL When A Software or Special Interrupt is Accepted

Interrupt sources	Value set in the IPL
Oscillation stop and re-oscillation detection, Watchdog timer, $\overline{\text{NMI}}$	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 1.10.7 shows the stack status before and after an interrupt request is accepted.

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

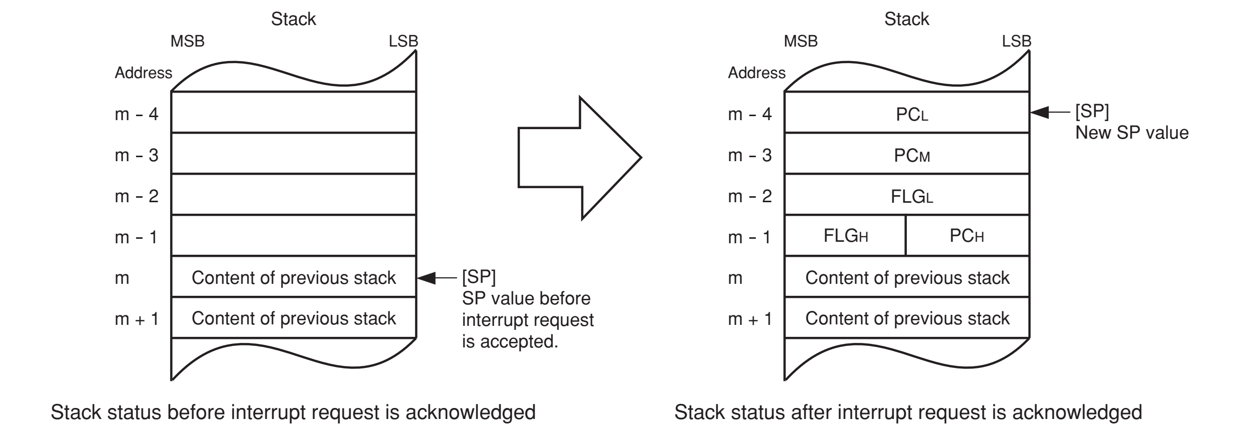


Figure 1.10.7 Stack Status Before and After Acceptance of Interrupt Request

The operation of saving registers carried out in the interrupt sequence is dependent on whether the SP (Note), at the time of acceptance of an interrupt request, is even or odd. If the SP (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 1.10.8 shows the operation of the saving registers.

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

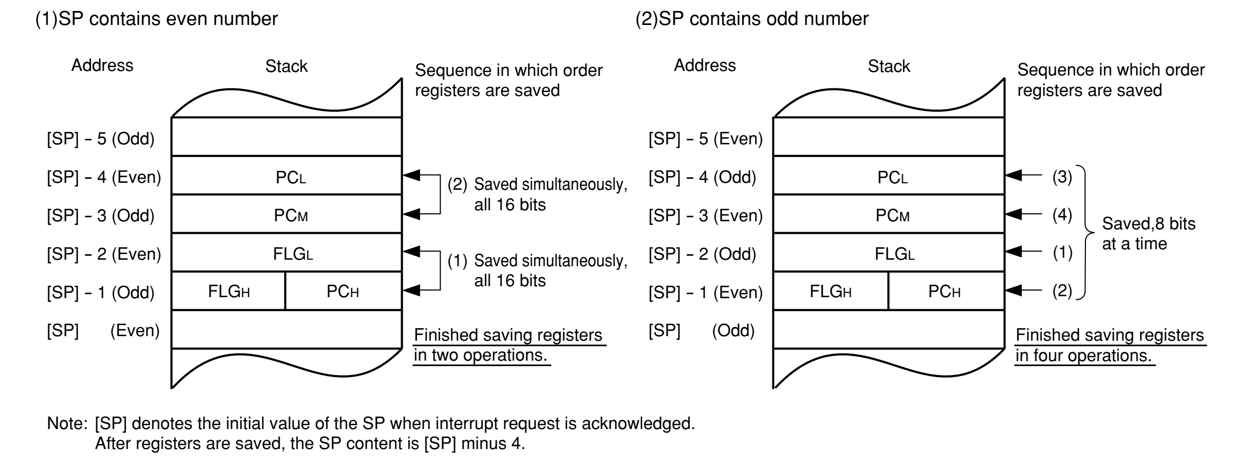


Figure 1.10.8 Operation of Saving Registers

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 1.10.9 shows the priorities of hardware interrupts.

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

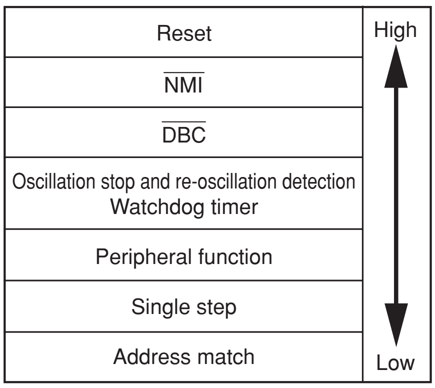


Figure 1.10.9 Hardware Interrupt Priority

Interrupt Priority Resolution Circuit

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

Figure 1.10.10 shows the circuit that judges the interrupt priority level.

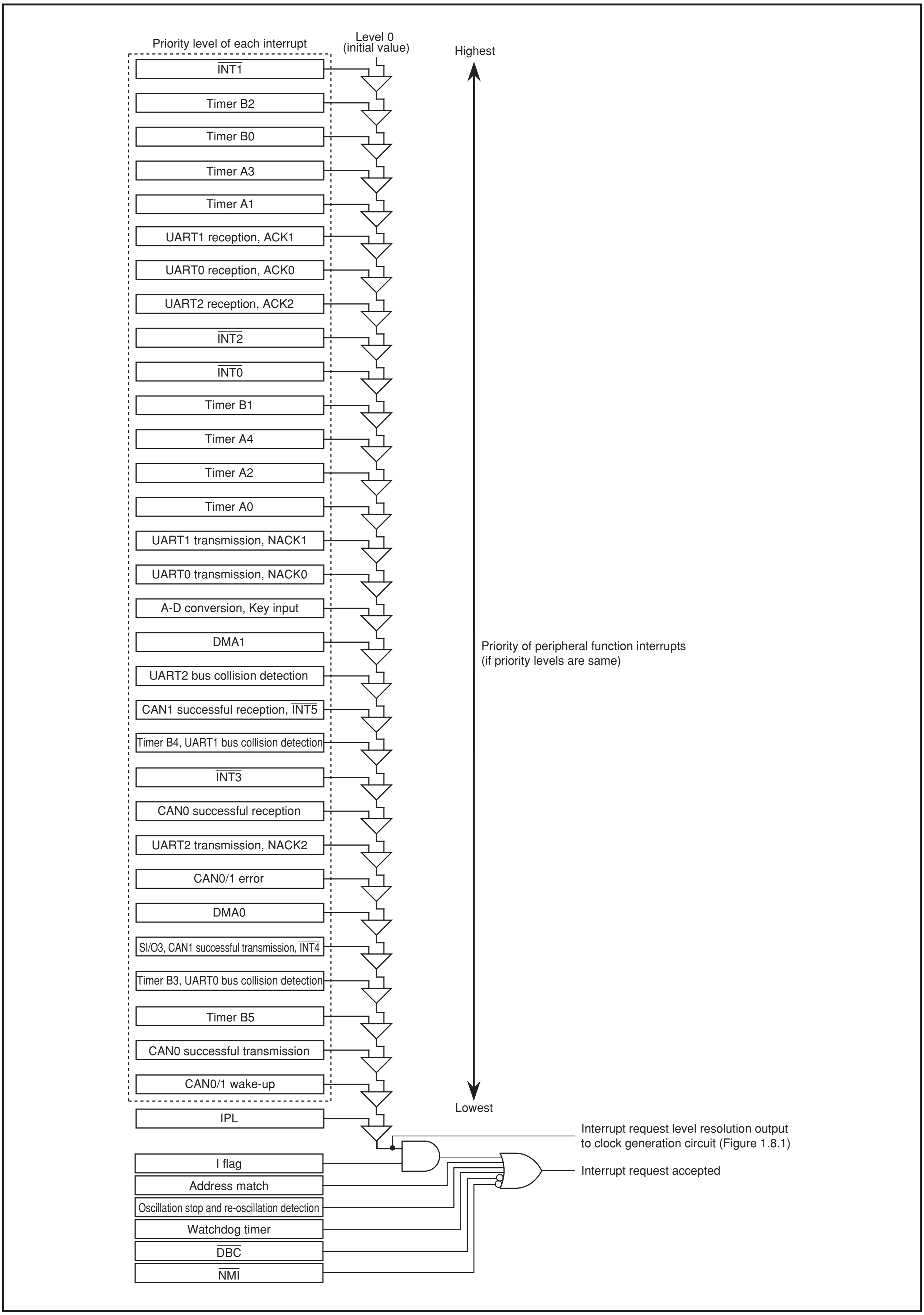


Figure 1.10.10 Interrupts Priority Select Circuit

INT Interrupt

INTi interrupt (i = 0 to 5) is triggered by the edges of external inputs. The edge polarity is selected using the IFSR1 register's IFSR1i bit.

INT4 share the interrupt vector and interrupt control register with SI/O3 and CAN1 successful transmission. INT5 share the interrupt vector and interrupt control register with CAN1 successful reception. To use the INT4 interrupt, set the IFSR1 register's IFSR16 bit to "1" (INT4). To use the INT5 interrupt, set the IFSR1 register's IFSR17 bit to "1" (INT5).

After modifying the IFSR16 or IFSR17 bit, set the corresponding IR bit to "0" (interrupt not requested) before enabling the interrupt.

Figure 1.10.11 shows the IFSR0 register and IFSR1 register.

Interrupt request cause select register 0

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
IFSR0	$01DE_{16}$	$00XXX000_2$

Bit symbol	Bit name	Function	RW
IFSR00	Interrupt request cause select bit	0 : CAN1 successful transmission 1 : SI/O3	RW
IFSR01	Interrupt request cause select bit	0 : A-D conversion 1 : Key input	RW
IFSR02	Interrupt request cause select bit	0 : CAN0/1 wake-up error 1 : CAN0 wake-up error/ CAN1 wake-up error	RW
- (b5-b3)	Nothing is assigned. When write, set to "0". When read, their contents are indeterminate.		–
IFSR06	Interrupt request cause select bit (Note 1)	0 : Timer B3 1 : UART0 bus collision detection	RW
IFSR07	Interrupt request cause select bit (Note 2)	0 : Timer B4 1 : UART1 bus collision detection	RW

Note 1: Timer B3 and UART0 bus collision detection share the vector and interrupt control register.
When using the timer B3 interrupt, set the IFSR06 bit in the IFSR0 register to "0" (timer B3).
When using UART0 bus collision detection, set the IFSR06 bit to "1" (UART0 bus collision detection).
Note 2: Timer B4 and UART1 bus collision detection share the vector and interrupt control register.
When using the timer B4 interrupt, set the IFSR07 bit in the IFSR0 register to "0" (timer B4).
When using UART1 bus collision detection, set the IFSR07 bit to "1" (UART1 bus collision detection).

Interrupt request cause select register 1

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
IFSR1	$01DF_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
IFSR10	INT0 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
IFSR11	INT1 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
IFSR12	INT2 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
IFSR13	INT3 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
IFSR14	INT4 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
IFSR15	INT5 interrupt polarity switching bit	0 : One edge 1 : Both edges (Note 1)	RW
IFSR16	Interrupt request cause select bit (Note 2)	0 : SI/O3/CAN1 successful transmission (Note 3) 1 : $\overline{INT4}$	RW
IFSR17	Interrupt request cause select bit	0 : CAN1 successful reception 1 : $\overline{INT5}$	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: During memory expansion and microprocessor modes, set this bit to "0" (SI/O3, CAN1 successful transmission).
Note 3: When setting this bit to "0" (SI/O3, CAN1 successful transmission), make sure the IFSR0 register's IFSR00 bit is set to "0" (CAN1 successful transmission) or "1" (SI/O3).
And, make sure the C1TRMIC register's POL bit is set to "0" (falling edge).

Figure 1.10.11 IFSR0 Register and IFSR1 Register

$\overline{\text{NMI}}$ Interrupt

An $\overline{\text{NMI}}$ interrupt request is generated when input on the $\overline{\text{NMI}}$ pin changes state from high to low. The $\overline{\text{NMI}}$ interrupt is a non-maskable interrupt.
The input level of this $\overline{\text{NMI}}$ interrupt input pin can be read by accessing the P8 register's P8_5 bit.
This pin cannot be used as an input port.

Key Input Interrupt

Of $P10_4$ to $P10_7$, a key input interrupt is generated when input on any of the $P10_4$ to $P10_7$ pins which has had the PD10 register's PD10_4 to PD10_7 bits set to "0" (input) goes low. Key input interrupts can be used as a key-on wakeup function, the function which gets the microcomputer out of wait or stop mode. However, if you intend to use the key input interrupt, do not use $P10_4$ to $P10_7$ as analog input ports. Figure 1.10.12 shows the block diagram of the key input interrupt. Note, however, that while input on any pin which has had the PD10_4 to PD10_7 bits set to "0" (input mode) is pulled low, inputs on all other pins of the port are not detected as interrupts.

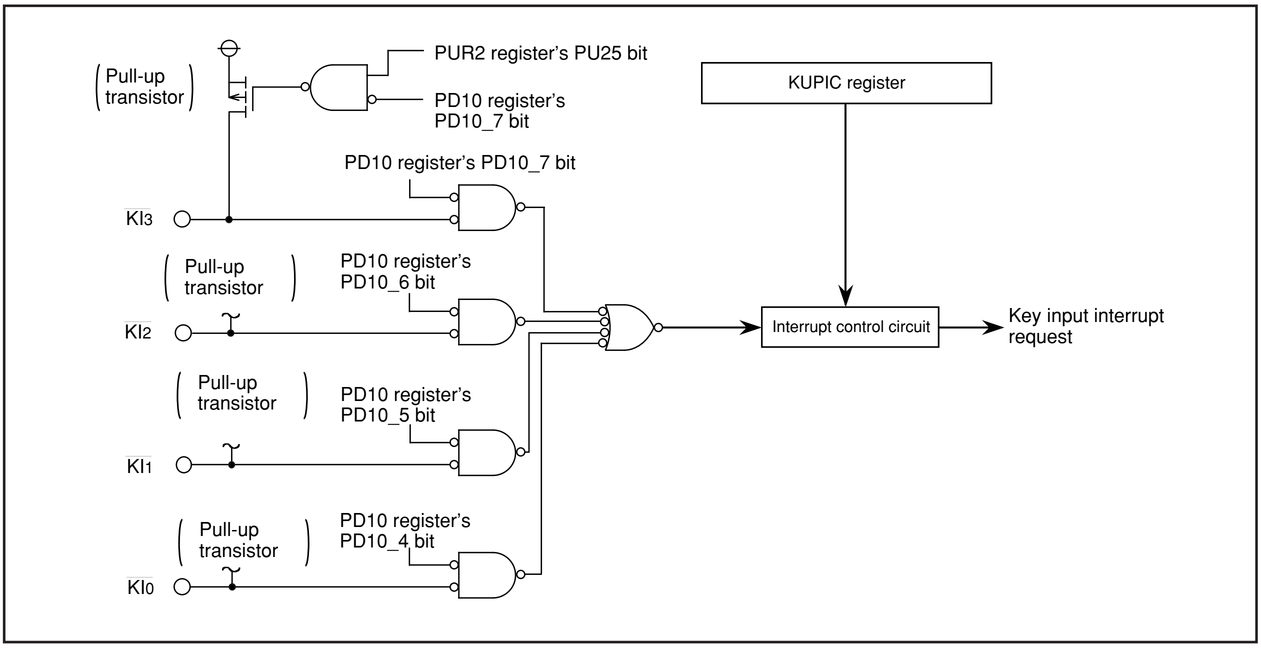


Figure 1.10.12 Key Input Interrupt Block Diagram

CAN0/1 Wake-up Interrupt

CAN0/1 wake-up interrupt is occurs when a falling edge is input to CRx_0 or CRx_1. Use the interrupt in stop/wait mode or CAN sleep mode. The CAN0/1 wake-up interrupt is enabled only when the port is defined as the CAN port. One interrupt is allocated to CAN0/1. Figure 1.10.13 shows the block diagram of the CAN0/1 wake-up interrupt. Please note that the wake-up message will be lost.

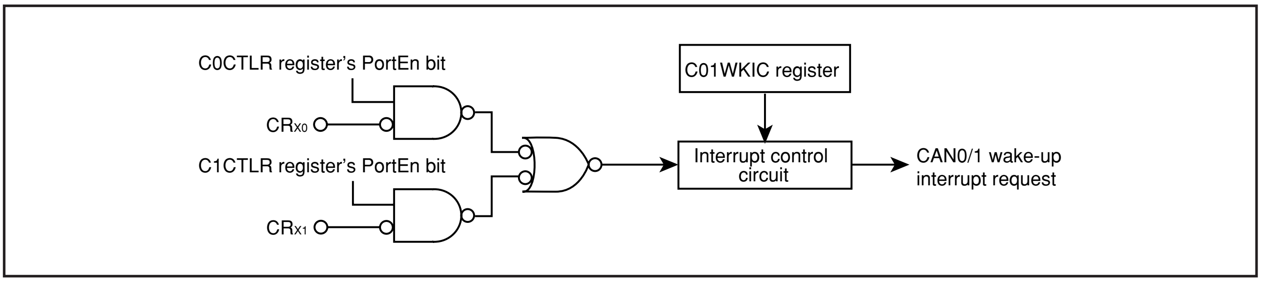


Figure 1.10.13 CAN0/1 Wake-up Interrupt Block Diagram

Address Match Interrupt

An address match interrupt request is generated immediately before executing the instruction at the address indicated by the RMADi register (i = 0 to 3). Set the start address of any instruction in the RMADi register. Use the AIER register's AIER0 and AIER1 bits and the AIER2 register's AIER20 and AIER21 bits to enable or disable the interrupt. Note that the address match interrupt is unaffected by the I flag and IPL. For address match interrupts, the value of the PC that is saved to the stack area varies depending on the instruction being executed (refer to "Saving Registers"). (The value of the PC that is saved to the stack area is not the correct return address.) Therefore, follow one of the methods described below to return from the address match interrupt.

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

Table 1.10.6 shows the value of the PC that is saved to the stack area when an address match interrupt request is accepted.
Note that when using the external bus in 8-bit width, no address match interrupts can be used for external areas. Table 1.10.7 shows the relationship between address match interrupt sources and associated registers. Figure 1.10.14 shows the AIER, AIER2, and RMAD0 to RMAD3 registers.

Table 1.10.6 Value of PC That is Saved to Stack Area When Address Match Interrupt Request is Accepted

Instruction at address indicated by RMADi register	Value at PC that is saved to stack area
• 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)	• Address indicated by RMADi register + 2
• Instructions other than the above	• Address indicated by RMADi register + 1

Value of PC that is saved to stack area: Refer to "Saving Registers".

Table 1.10.7 Relationship Between Address Match Interrupt Sources and Associated Registers

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

Address match interrupt enable register

b7 b6 b5 b4 b3 b2 b1 b0

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

b7 b6 b5 b4 b3 b2 b1 b0

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)

(b23) b7 (b19) b3 (b16)(b15) b0 b7 (b8) b0 b7 b0

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 $01BC_{16}$	$X00000_{16}$

Bit symbol	Function	Setting range	RW
– (b19-b0)	Address setting register for address match interrupt	00000_{16} to $FFFFF_{16}$	RW
– (b23-b20)	Nothing is assigned. When write, set to "0". When read, their contents are indeterminate.		–

Figure 1.10.14 AIER Register, AIER2 Register and RMAD0 to RMAD3 Registers

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" (watchdog timer reset). Once this bit is set to "1", it cannot be set to "0" (watchdog timer interrupt) in a program. Refer to "Watchdog Timer Reset" for details about watchdog timer reset.

When the main clock is selected for CPU clock, ring oscillator clock, PLL clock, the divide-by-n value for the prescaler can be selected 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 selected for CPU clock, ring oscillator clock, PLL clock

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

With sub clock selected for CPU clock

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

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 1.11.1 shows the block diagram of the watchdog timer. Figure 1.11.2 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 runaway.

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

(1) Set the PRC1 bit of the PRCR register to "1" (enable writes to the PM1 and PM2 registers).
(2) Set the PM12 bit of the PM1 register to "1" (reset when the watchdog timer underflows).
(3) Set the PM22 bit of the PM2 register to "1" (ring oscillator clock used for the watchdog timer count source).
(4) Set the PRC1 bit of the PRCR register to "0" (disable writes to the 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.

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

- The CM10 bit of the 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.

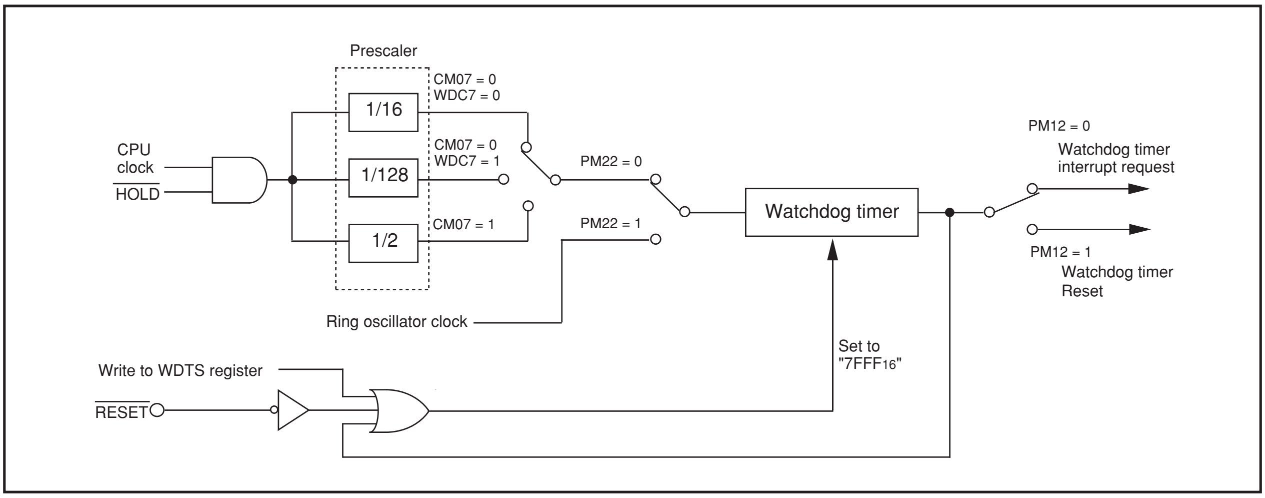


Figure 1.11.1 Watchdog Timer Block Diagram

Watchdog timer control register

b7 b6 b5 b4 b3 b2 b1 b0 — (b6, b5 = 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	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 1.11.2 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 1.12.1 shows the block diagram of the DMAC. Table 1.12.1 shows the DMAC specifications. Figures 1.12.2 to 1.12.4 show the DMAC related-registers.

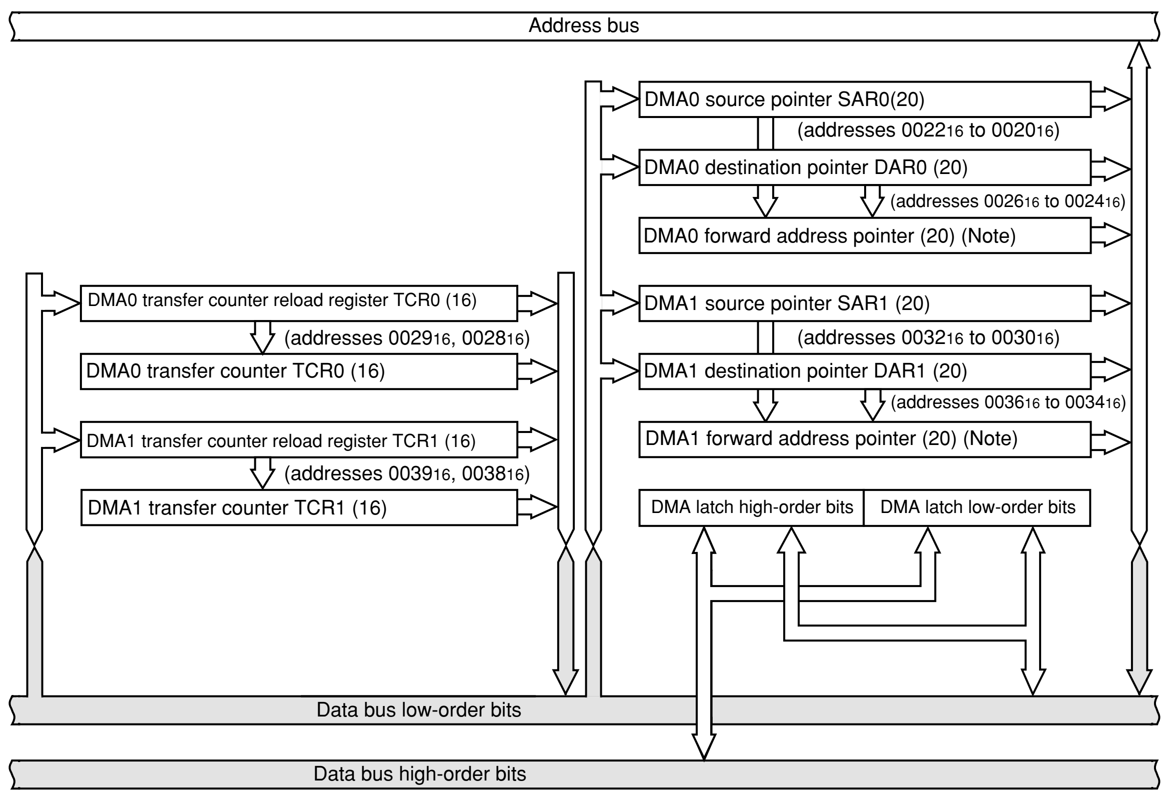


Figure 1.12.1 DMAC Block Diagram

A DMA request is generated by a write to the DSR bit of the DMiSL register (i = 0, 1), as well as by an interrupt request which is generated by any function specified by the DMS and DSEL3 to DSEL0 bits of the DMiSL register. 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 IR bit of the interrupt control register does not change state due to a DMA transfer.
A data transfer is initiated each time a DMA request is generated when the DMAE bit = 1 (DMA enabled) of the DMiCON register. 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 1.12.1 DMAC Specifications

Item		Specification
No. of channels		2 (cycle steal method)
Transfer memory space		• From any address in the 1 Mbyte space to a fixed address • From a fixed address to any address in the 1 Mbyte space • From a fixed address to a fixed address
Maximum No. of bytes transferred		128 Kbytes (with 16-bit transfer) or 64 Kbytes (with 8-bit transfer)
DMA request factors (Notes 1, 2)		Falling edge of $\overline{INT0}$ or $\overline{INT1}$ Both edge of $\overline{INT0}$ or $\overline{INT1}$ Timer A0 to timer A4 interrupt requests Timer B0 to timer B5 interrupt requests UART0 transfer, UART0 reception interrupt requests UART1 transfer, UART1 reception interrupt requests UART2 transfer, UART2 reception interrupt requests SI/O3 interrupt request A-D conversion interrupt requests Software triggers
Channel priority		DMA0 > DMA1 (DMA0 takes precedence)
Transfer unit		8 bits or 16 bits
Transfer address direction		forward or fixed (The source and destination addresses cannot both be in the forward direction.)
Transfer mode	Single transfer	Transfer is completed when the DMAi transfer counter 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 start-up		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.

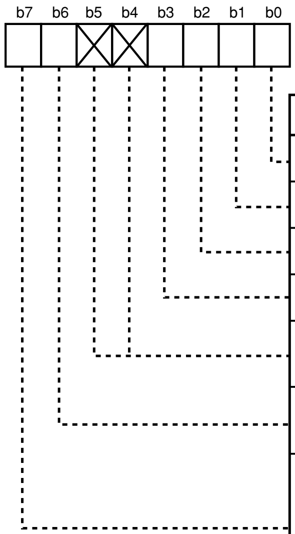
i = 0, 1

Note 1: DMA transfer is not effective to any interrupt. DMA transfer is affected neither by the I flag nor by the interrupt control register.

Note 2: The selectable causes of DMA requests differ with each channel.

Note 3: Make sure that no DMAC-related registers (addresses 0020_{16} to $003F_{16}$) 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 $\overline{INT0}$ 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 $\overline{INT0}$ pin
0 1 1 1$_2$	Timer B0	Timer B3
1 0 0 0$_2$	Timer B1	Timer B4
1 0 0 1$_2$	Timer B2	Timer B5
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$_2$	A-D conversion	—
1 1 1 1$_2$	UART1 transmit	—

Figure 1.12.2 DM0SL Register

DMA1 request cause select register

b7 b6 b5 b4 b3 b2 b1 b0

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)
0000_2	Falling edge of $\overline{INT1}$ pin	—
0001_2	Software trigger	—
0010_2	Timer A0	—
0011_2	Timer A1	—
0100_2	Timer A2	—
0101_2	Timer A3	SI/O3
0110_2	Timer A4	—
0111_2	Timer B0	Two edges of $\overline{INT1}$ pin
1000_2	Timer B1	—
1001_2	Timer B2	—
1010_2	UART0 transmit	—
1011_2	UART0 receive/ACK0	—
1100_2	UART2 transmit	—
1101_2	UART2 receive/ACK2	—
1110_2	A-D conversion	—
1111_2	UART1 transmit/ACK1	—

DMAi control register (i = 0, 1)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
DM0CON	$002C_{16}$	$00000X00_2$
DM1CON	$003C_{16}$	$00000X00_2$

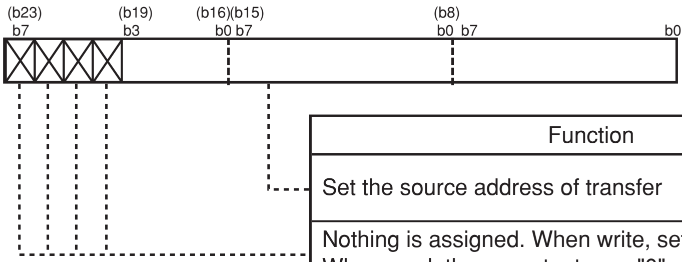
Bit symbol	Bit name	Function	RW
DMBIT	Transfer unit bit select bit	0 : 16 bits 1 : 8 bits	RW
DMASL	Repeat transfer mode select bit	0 : Single transfer 1 : Repeat transfer	RW
DMAS	DMA request bit	0 : DMA not requested 1 : DMA requested	RW (Note 1)
DMAE	DMA enable bit	0 : Disabled 1 : Enabled	RW
DSD	Source 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 1.12.3 DM1SL Register, DM0CON Register and DM1CON Register

RENESAS

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

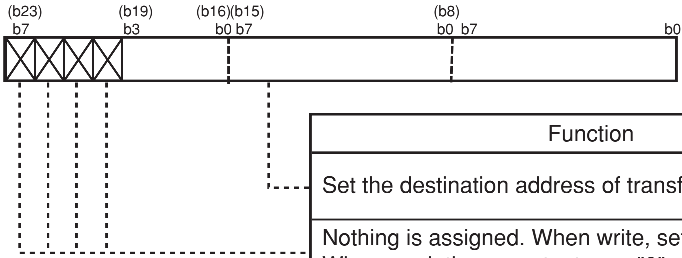


Symbol	Address	After reset
SAR0	0022_{16} to 0020_{16}	Indeterminate
SAR1	0032_{16} to 0030_{16}	Indeterminate

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

Note: If the DSD bit of the DMiCON register is "0" (fixed), this register can only be written to when the DMAE bit of the 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)

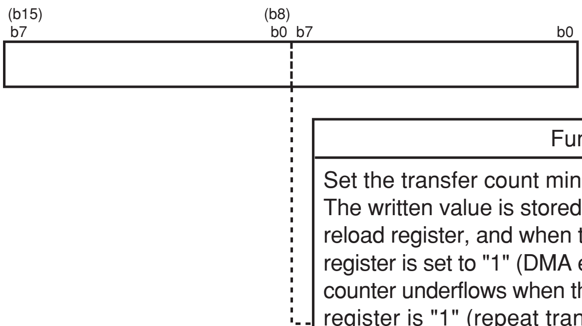


Symbol	Address	After reset
DAR0	0026_{16} to 0024_{16}	Indeterminate
DAR1	0036_{16} to 0034_{16}	Indeterminate

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

Note: If the DAD bit of the DMiCON register is "0" (fixed), this register can only be written to when the DMAE bit of the 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 the DMiCON register is set to "1" (DMA enabled) or the DMAi transfer counter underflows when the DMASL bit of the 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 1.12.4 SAR0, SAR1, DAR0, DAR1, TCR0 and TCR1 Registers

1. Transfer Cycle

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. During memory expansion and microprocessor modes, it is also affected by the BYTE pin level. 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 BYTE Pin Level

During memory expansion and microprocessor modes, if 16 bits of data are to be transferred on an 8-bit data bus (input on the BYTE pin = high), the operation is accomplished by transferring 8 bits of data twice. Therefore, this operation requires two bus cycles to read data and two bus cycles to write data. Furthermore, if the DMAC is to access the internal area (internal ROM, internal RAM, or SFR), unlike in the case of the CPU, the DMAC does it through the data bus width selected by the BYTE pin.

(c) 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.

(d) Effect of $\overline{\text{RDY}}$ Signal

During memory expansion and microprocessor modes, DMA transfers to and from an external area are affected by the $\overline{\text{RDY}}$ signal. Refer to “$\overline{\text{RDY}}$ Signal”.

Figure 1.12.5 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 unit using an 8-bit bus ((2) in Figure 1.12.5), two source read bus cycles and two destination write bus cycles are required.

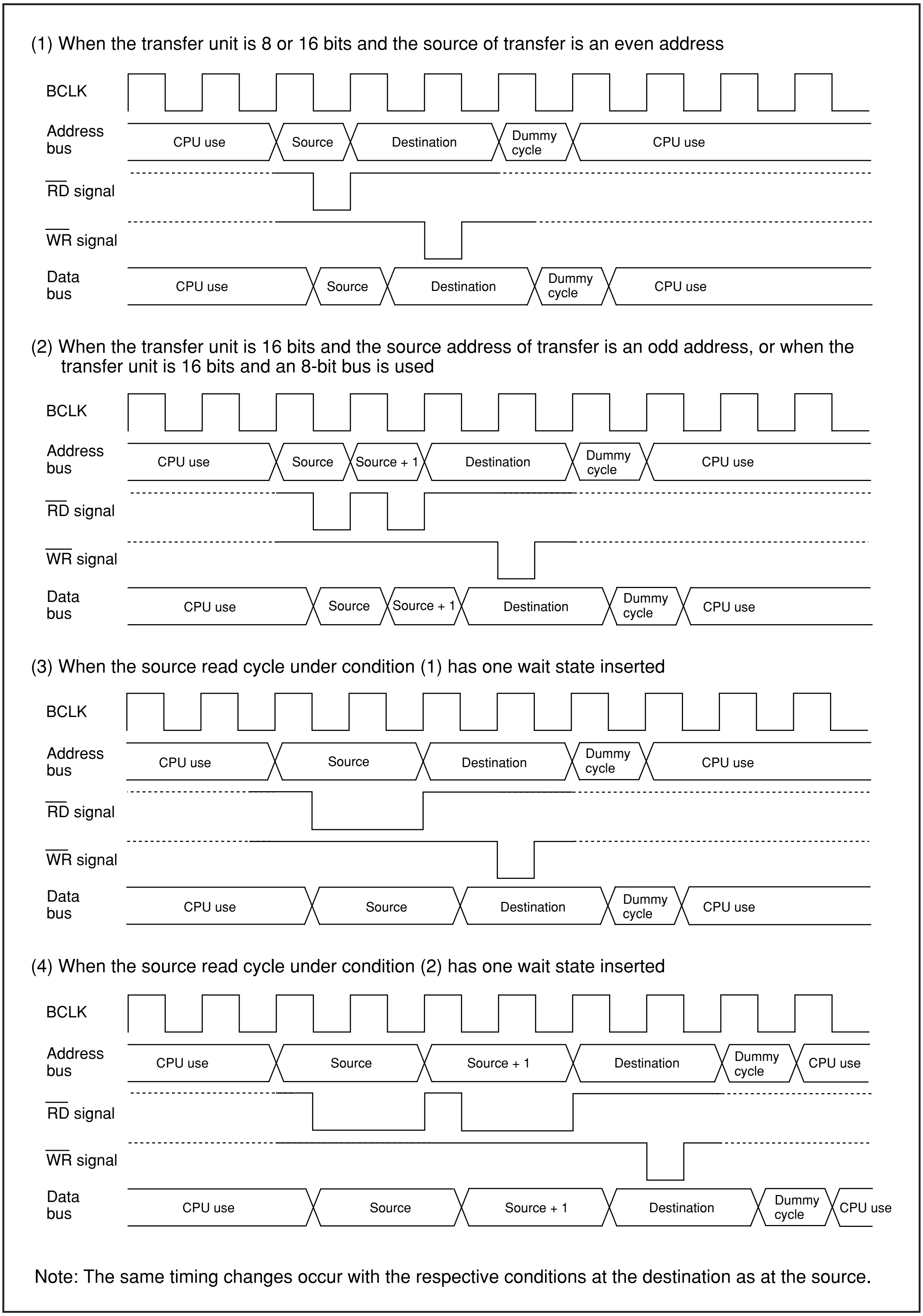


Figure 1.12.5 Transfer Cycles for Source Read

2. DMA Transfer Cycles

Any combination of even or odd transfer read and write addresses is possible.

Table 1.12.2 shows the number of DMA transfer cycles. Table 1.12.3 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 $\times$ j + No. of write cycles $\times$ k

Table 1.12.2 DMA Transfer Cycles

Transfer unit	Bus width	Access address	Single-chip mode		Memory expansion mode Microprocessor mode	
			No. of read cycles	No. of write cycles	No. of read cycles	No. of write cycles
8-bit transfer (DMBIT =1)	16 bits (BYTE = L)	Even	1	1	1	1
		Odd	1	1	1	1
	8 bits (BYTE= H)	Even	–	–	1	1
		Odd	–	-	1	1
16-bit transfer (DMBIT = 0)	16 bits (BYTE =L)	Even	1	1	1	1
		Odd	2	2	2	2
	8 bits (BYTE = H)	Even	–	–	2	2
		Odd	–	–	2	2

Table 1.12.3 Coefficient j, k

	Internal area				External area						
	Internal ROM, RAM		SFR		Separate bus				Multiplexed bus		
	No wait	With wait	1 wait (Note 1)	2 waits (Note 1)	No wait	With wait (Note 2)			With wait (Note 2)		
						1 wait	2 waits	3 waits	1 wait	2 waits	3 waits
j	1	2	2	3	1	2	3	4	3	3	4
k	1	2	2	3	2	2	3	4	3	3	4

Note 1: Depends on the set value of the PM20 bit of the PM2 register.

Note 2: Depends on the set value of the CSE register.

3. DMA Enable

When a data transfer starts after setting the DMAE bit of the 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 of the DMiCON register is “1” (forward) or the DARi register value when the DAD bit of the 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 of the 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 the DMiSL register (i = 0, 1) on either channel. Table 1.12.4 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 1.12.4 Timing at Which DMAS bit Changes State

DMA factor	DMAS bit of 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 the DMiSL 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 the DMiSL register has its IR bit set to “1”.	

i = 0, 1

5. 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 1.12.6 shows an example of DMA transfer effected by external factors.
In Figure 1.12.6, DMA0 request having priority is received first to start a transfer when a DMA0 request and DMA1 request are generated simultaneously. After one DMA0 transfer is completed, a bus arbitration is returned to the CPU. When the CPU has completed one bus access, a DMA1 transfer starts. After one DMA1 transfer is completed, the bus arbitration is again returned to the CPU.
In addition, DMA requests cannot be counted up since each channel has one DMAS bit. Therefore, when DMA requests, as DMA1 in Figure 1.12.6, occurs more than one time, the DMAS bit is set to “0” as soon as getting the bus arbitration. The bus arbitration is returned to the CPU when one transfer is completed. Refer to “(7) $\overline{\text{HOLD}}$ Signal in Bus Control” for details about bus arbitration between the CPU and DMA.

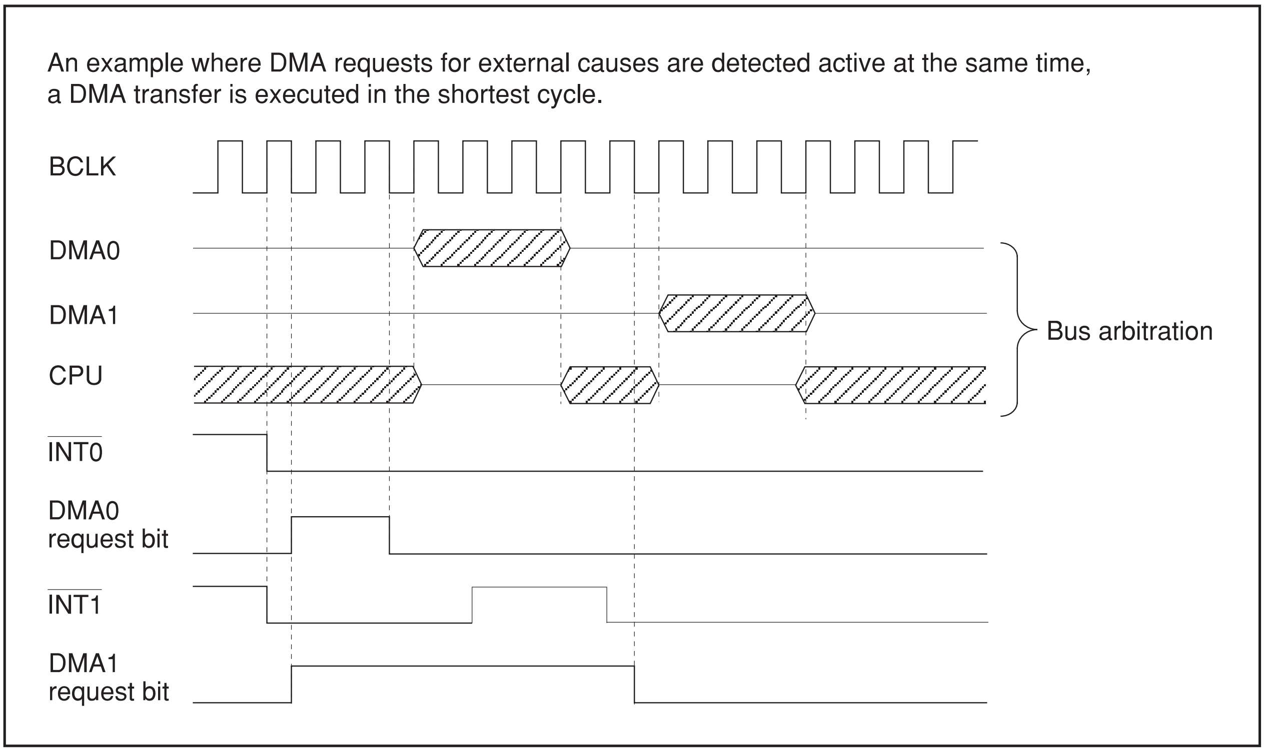
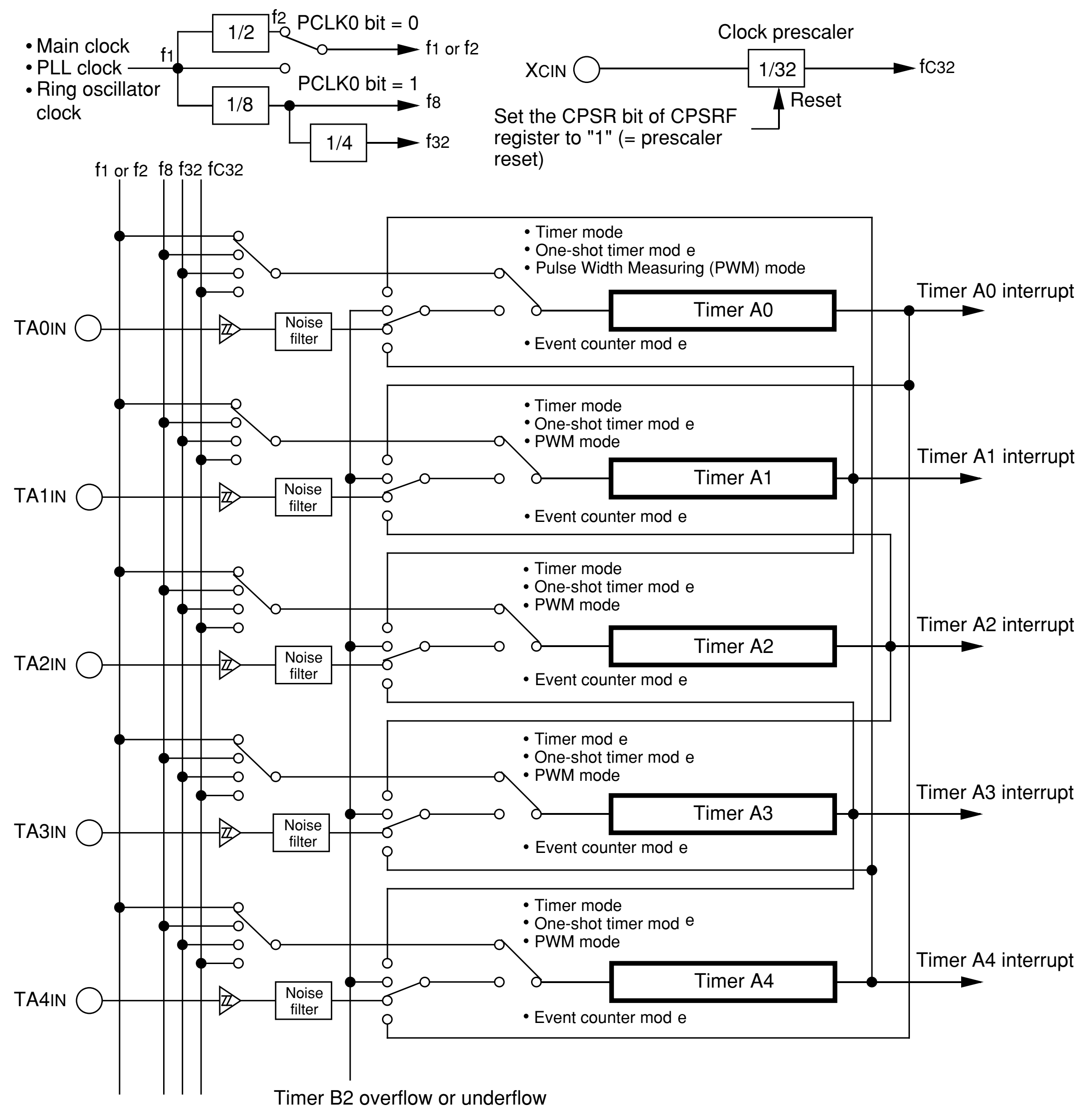


Figure 1.12.6 DMA Transfer by External Factors

Timers

Eleven 16-bit timers, each capable of operating independently of the others, can be classified by function as either timer A (five) and timer B (six). The count source for each timer acts as a clock, to control such timer operations as counting, reloading, etc.

Figures 1.13.1 and 1.13.2 show block diagrams of timer A and timer B configuration, respectively.



Note: Be aware that TA0IN shares the pin with RxD2, SCL2 and TB5IN.

Figure 1.13.1 Timer A Configuration

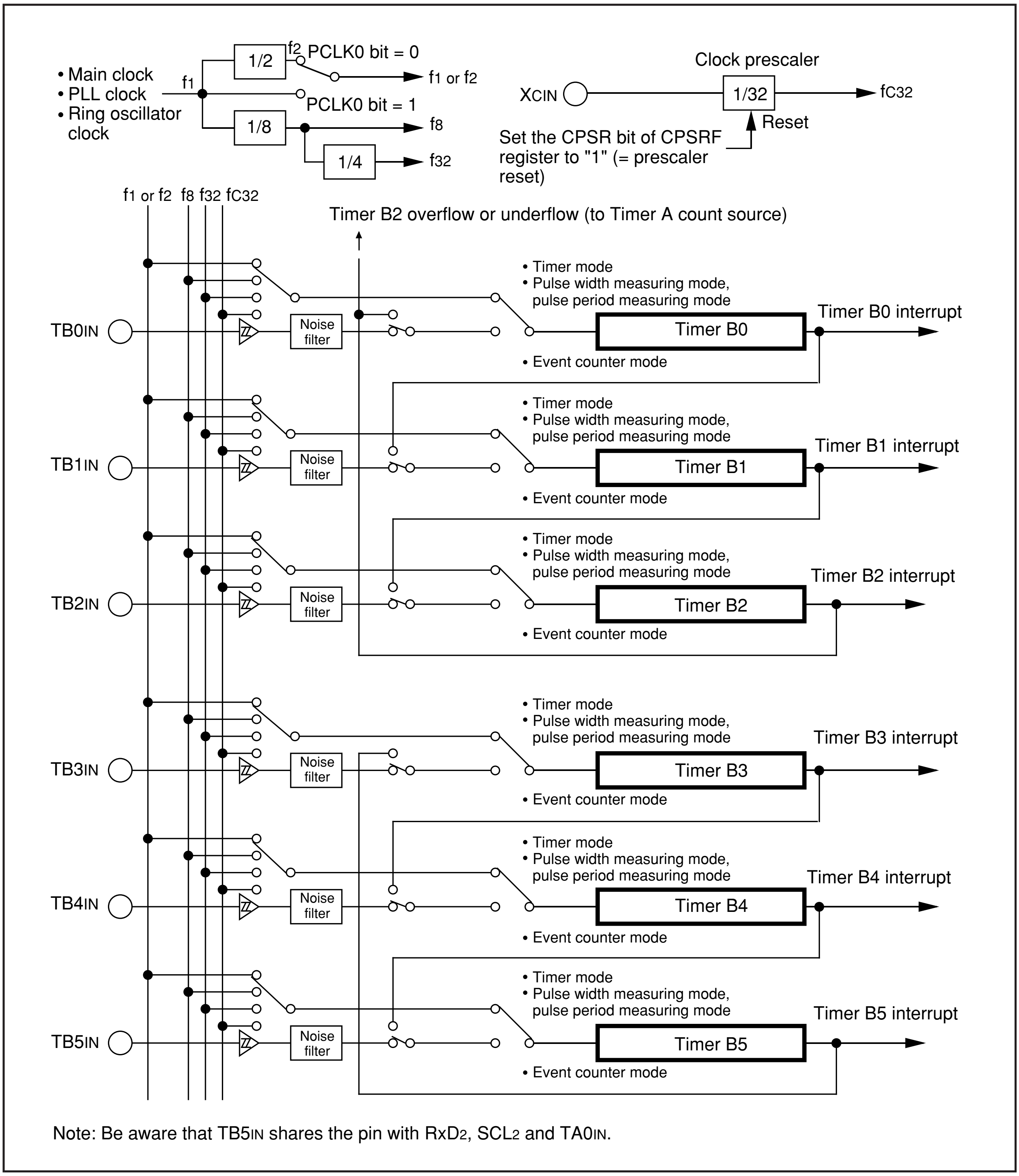


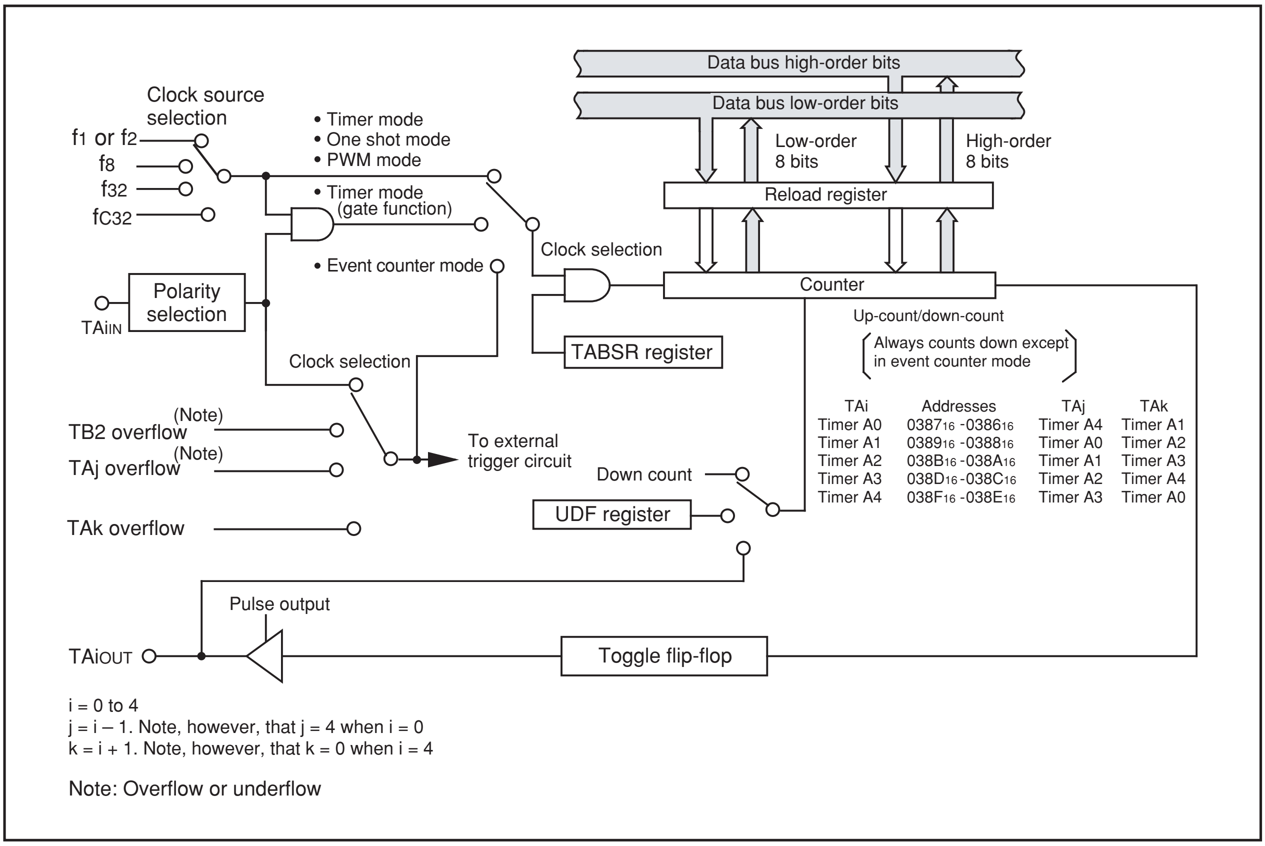
Figure 1.13.2 Timer B Configuration

Timer A

Figure 1.13.3 shows a block diagram of the timer A. Figures 1.13.4 to 1.13.6 show the timer A-related registers.

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 1.13.3 Timer A Block Diagram

Timer Ai mode register (i = 0 to 4)

b7 b6 b5 b4 b3 b2 b1 b0

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

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 2)	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 sto p	0000_{16} to $FFFF_{16}$ (Notes 3, 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 4, 5)	WO
Pulse width modulation mode (8-bit PWM)	Modify the pulse width as follows: PWM period: $(2^8 - 1) \times (m + 1)$/ 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 4, 5)	WO

Note 1: The register must be accessed in 16-bit unit.
Note 2: The timer counts pulses from an external device or overflows or underflows in other timers.
Note 3: 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 4: Use the MOV instruction to write to the TAi register.
Note 5: 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 "00_{16}" while operating as an 8-bit pulse width modulator.

Figure 1.13.4 TA0MR to TA4MR Registers and TA0 to TA4 Registers

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
TB0S	Timer B0 count start flag		RW
TB1S	Timer B1 count start flag		RW
TB2S	Timer B2 count start flag		RW

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 $TA4_{IN}$ 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 timer A2 to timer A4 to "0".

Figure 1.13.5 TABSR Register and UFD Register

One-shot start flag

b7 b6 b5 b4 b3 b2 b1 b0

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) is "10_2" (one-shot timer mode) and the MR2 bit of TAiMR register is "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
TAZIE	Z-phase input enable bit	0 : Z-phase input disabled 1 : Z-phase input enabled	RW
TA0TGL	Timer A0 event/trigger select bit	b7 b6 0 0 : Input on $TA0_{IN}$ is selected (Note) 0 1 : TB2 is selected 1 0 : TA4 is selected 1 1 : TA1 is selected	RW
TA0TGH			RW

Note: Make sure the PD7_1 bit of PD7 register is set to "0" (input mode).

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) 0 1 : TB2 is selected 1 0 : TA0 is selected 1 1 : TA2 is selected	RW
TA1TGH			RW
TA2TGL	Timer A2 event/trigger select bit	b3 b2 0 0 : Input on $TA2_{IN}$ is selected (Note) 0 1 : TB2 is selected 1 0 : TA1 is selected 1 1 : TA3 is selected	RW
TA2TGH			RW
TA3TGL	Timer A3 event/trigger select bit	b5 b4 0 0 : Input on $TA3_{IN}$ is selected (Note) 0 1 : TB2 is selected 1 0 : TA2 is selected 1 1 : TA4 is selected	RW
TA3TGH			RW
TA4TGL	Timer A4 event/trigger select bit	b7 b6 0 0 : Input on $TA4_{IN}$ is selected (Note) 0 1 : TB2 is selected 1 0 : TA3 is selected 1 1 : TA0 is selected	RW
TA4TGH			RW

Note: Make sure the port direction bits for the $TA1_{IN}$ to $TA4_{IN}$ pins are set to "0" (input mode).

Clock prescaler reset flag

b7 b6 b5 b4 b3 b2 b1 b0

Symbol CPSRF — Address 0381_{16} — After reset $0XXXXXXX_2$

Bit symbol	Bit name	Function	RW
— (b6-b0)	Nothing is assigned. When write, set to "0". When read, their contents are indeterminate.		—
CPSR	Clock prescaler reset flag	Setting this bit to "1" initializes the prescaler for the timekeeping clock. (When read, its content is "0".)	RW

Figure 1.13.6 ONSF Register, TRGSR Register and CPSRF Register

1. Timer Mode

In timer mode, the timer counts a count source generated internally. Table 1.13.1 lists specifications in timer mode. Figure 1.13.7 shows TAiMR register in timer mode.

Table 1.13.1. Specifications in Timer Mode

Item	Specification
Count source	f_1, f_2, f_8, f_{32}, f_{C32}
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 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
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.

i = 0 to 4

Timer Ai mode register (i = 0 to 4)

b7 b6 b5 b4 b3 b2 b1 b0: [ ] [ ] [0] [ ] [ ] [ ] [0] [0]

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 (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) 1 1 : Counts while input on the TAi_{IN} pin is high (Note)	RW
MR2			RW
MR3	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 : f_{C32}	RW
TCK1			RW

Note: The port direction bit for the TAi_{IN} pin must be set to "0" (input mode).

Figure 1.13.7 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. Timers A2, A3 and A4 can count two-phase external signals. Table 1.13.2 lists specifications in event counter mode (when not processing two-phase pulse signal). Figure 1.13.8 shows TAiMR register in event counter mode (when not processing two-phase pulse signal). Table 1.13.3 lists specifications in event counter mode (when processing two-phase pulse signal with the timers A2, A3 and A4). Figure 1.13.9 shows TA2MR to TA4MR registers in event counter mode (when processing two-phase pulse signal with the timers A2, A3 and A4).

Table 1.13.2 Specifications in Event Counter Mode (when not processing two-phase pulse signal)

Item	Specification
Count source	• External signals input to TAi_{IN} pin (effective edge can be selected in program) • Timer B2 overflows or underflows, 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
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.

i = 0 to 4

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

b7	b6	b5	b4	b3	b2	b1	b0
		0				0	1

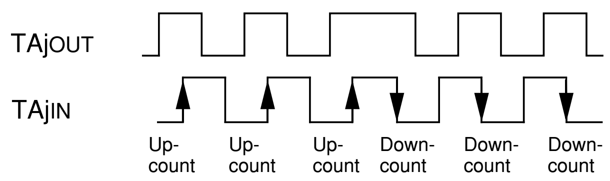
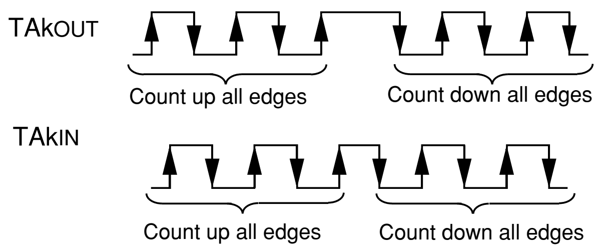
Symbol	Address	After reset
TA0MR to TA4MR	0396_{16} to $039A_{16}$	00_{16}

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 (TAiOUT pin functions as pulse output pin)	RW
MR1	Count polarity select bit (Note 2)	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 3)	RW
MR3	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: Effective when the TAiTGH and TAiTGL bits of ONSF or TRGSR register are"00_2" (TAiIN pin input).
Note 3: 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 1.13.8 TAiMR Register in Event Counter Mode (when not using two-phase pulse signal processing)

Table 1.13.3 Specifications in Event Counter Mode (when processing two-phase pulse signal with timers A2, A3 and A4)

Item	Specification
Count source	• Two-phase pulse signals input to TAi_{IN} or TAi_{OUT} pins
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
TAi_{IN} pin function	Two-phase pulse input
TAi_{OUT} pin function	Two-phase pulse input
Read from timer	Count value can be read by reading timer A2, A3 or 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 (Note)	• Normal processing operation (timer A2 and timer A3) The timer counts up rising edges or counts down falling edges on TAj_{IN} pin when input signals on TAj_{OUT} pin is “H”.  • Multiply-by-4 processing operation (timer A3 and timer A4) If the phase relationship is such that TAk_{IN} pin goes “H” when the input signal on TAk_{OUT} pin is “H”, the timer counts up rising and falling edges on TAk_{OUT} and TAk_{IN} pins. If the phase relationship is such that TAk_{IN} pin goes “L” when the input signal on TAk_{OUT} pin is “H”, the timer counts down rising and falling edges on TAk_{OUT} and TAk_{IN} pins.  • Counter initialization by Z-phase input (timer A3) The timer count value is initialized to “0” by Z-phase input.

i = 2 to 4
j = 2, 3
k = 3, 4
Note : Only timer A3 is selectable. Timer A2 is fixed to normal processing operation, and timer A4 is fixed to multiply-by-4 processing operation.

Timer Ai mode register (i = 2 to 4)
(When using two-phase pulse signal processing)

b6	b5	b4	b3	b2	b1	b0
	0	1	0	0	0	1

Symbol	Address	After reset
TA2MR to TA4MR	0398_{16} to $039A_{16}$	00_{16}

Bit symbol	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			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 (Notes 1, 2)	0 : Normal processing operation 1 : Multiply-by-4 processing operation	RW

Note 1: TCK1 bit is valid for timer A3 mode register. No matter how this bit is set, timers A2 and A4 always operate in normal processing mode and x4 processing mode, respectively.

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 TAiTGH and TAiTGL bits to "00_2" (TAi_{IN} pin input).
- Set the port direction bits for TAi_{IN} and TAi_{OUT} to "0" (input mode).

Figure 1.13.9 TA2MR to TA4MR Registers in Event Counter Mode (when using two-phase pulse signal processing with timer A2, A3 or A4)

• **Counter Initialization by Two-Phase Pulse Signal Processing**

This function initializes the timer count value to “0” by Z-phase (counter initialization) input during two-phase pulse signal processing.

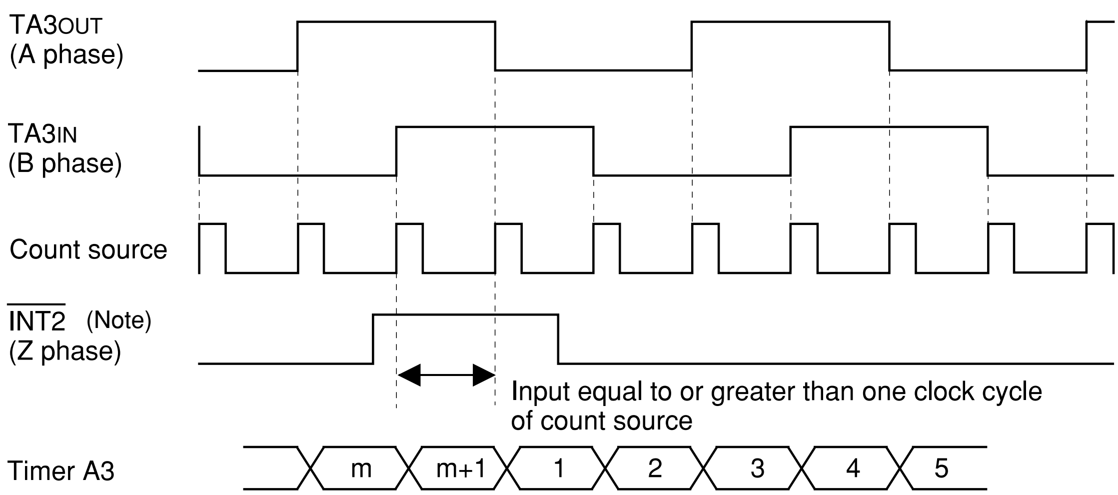
This function can only be used in timer A3 event counter mode during two-phase pulse signal processing, free-running type, x4 processing, with Z-phase entered from the $\overline{INT2}$ pin.

Counter initialization by Z-phase input is enabled by writing “0000_{16}” to the TA3 register and setting the TAZIE bit in ONSF register to “1” (Z-phase input enabled).

Counter initialization is accomplished by detecting Z-phase input edge. The active edge can be selected to be the rising or falling edge by using the POL bit of INT2IC register. The Z-phase pulse width applied to the $\overline{INT2}$ pin must be equal to or greater than one clock cycle of the timer A3 count source.

The counter is initialized at the next count timing after recognizing Z-phase input. Figure 1.13.10 shows the relationship between the two-phase pulse (A phase and B phase) and the Z phase.

If timer A3 overflow or underflow coincides with the counter initialization by Z-phase input, a timer A3 interrupt request is generated twice in succession. Do not use the timer A3 interrupt when using this function.



Note: This timing diagram is for the case where the POL bit of INT2IC register = 1 (rising edge).

Figure 1.13.10 Two-phase Pulse (A phase and B phase) and Z Phase

3. One-shot Timer Mode

In one-shot timer mode, the timer is activated only once by one trigger. When the trigger occurs, the timer starts up and continues operating for a given period. Table 1.13.4 lists specifications in one-shot timer mode. Figure 1.13.11 shows the TAiMR register in one-shot timer mode.

Table 1.13.4 Specifications in One-shot Timer Mode

Item	Specification
Count source	f_1, f_2, f_8, f_{32}, f_{C32}
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 B2 overflow or underflow, 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.

i = 0 to 4

Timer Ai mode register (i = 0 to 4)

b7	b6	b5	b4	b3	b2	b1	b0
		0				1	0

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 0 : One-shot timer mode	RW
TMOD1			RW
MR0	Pulse output function select bit	0 : Pulse is not output (TAi_{OUT} pin functions as I/O port) 1 : Pulse is output (TAi_{OUT} pin functions as a pulse output pin)	RW
MR1	External trigger select bit (Note 1)	0 : Falling edge of input signal to TAi_{IN} pin (Note 2) 1 : Rising edge of input signal to TAi_{IN} pin (Note 2)	RW
MR2	Trigger select bit	0 : TAiOS bit is enabled 1 : Selected by TAiTGH to TAiTGL bits	RW
MR3	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 : fC_{32}	RW
TCK1			RW

Note 1: Effective when the TAiTGH and TAiTGL bits of ONSF or TRGSR register are "00_2" (TAi_{IN} pin input).
Note 2: The port direction bit for the TAi_{IN} pin must be set to "0" (input mode).

Figure 1.13.11 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. The counter functions as either 16-bit pulse width modulator or 8-bit pulse width modulator.
Table 1.13.5 lists specifications in PWM mode. Figure 1.13.12 shows TAiMR register in PWM mode. Figures 1.13.13 and 1.13.14 show examples of how a 16-bit pulse width modulator operates and how an 8-bit pulse width modulator operates, respectively.

Table 1.13.5 Specifications in PWM Mode

Item	Specification
Count source	f_1, f_2, f_8, f_{32}, f_{C32}
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 • Cycle time $(2^{16}-1)$ / fj fixed fj: count source frequency (f_1, f_2, f_8, f_{32}, f_{C32})
8-bit PWM	• High level width n × (m+1) / fj n : set value of TAiMR register high-order address • Cycle time $(2^{8}-1)$ × (m+1) / fj m : set value of TAiMR register low-order address
Count start condition	• TAiS bit of TABSR register is set to “1” (start counting) • TAiS bit = 1 and external trigger input from the TAi_{IN} pin • TAiS bit = 1 and one of the following external triggers occurs Timer B2 overflow or underflow, 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
Count stop condition	TAiS bit is set to “0” (stop counting)
Interrupt request generation timing	PWM pulse goes “L”
TAi_{IN} pin function	I/O port or trigger input
TAi_{OUT} 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)

i = 0 to 4

Timer Ai mode register (i = 0 to 4)

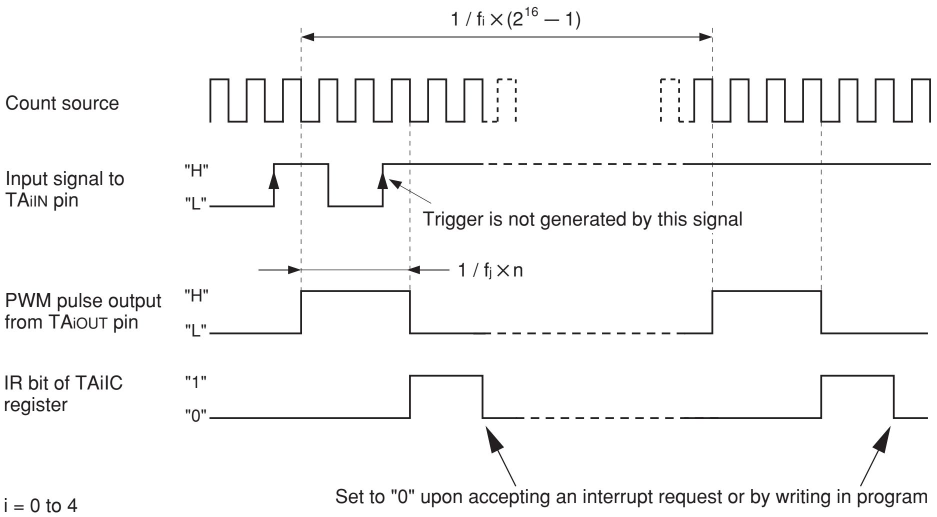
b7	b6	b5	b4	b3	b2	b1	b0
					1	1	1

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	RW
TMOD1			RW
MR0	Set to "1" in PWM mode		RW
MR1	External trigger select bit (Note 1)	0: Falling edge of input signal to TAi_{IN} pin (Note 2) 1: Rising edge of input signal to TAi_{IN} pin (Note 2)	RW
MR2	Trigger select bit	0 : Write "1" to TAiS bit in the TABSR register 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 : fC_{32}	RW
TCK1			RW

Note 1: Effective when the TAiTGH and TAiTGL bits of ONSF or TRGSR register are "00_2" (TAi_{IN} pin input).
Note 2: The port direction bit for the TAi_{IN} pin must be set to "0" (input mode).

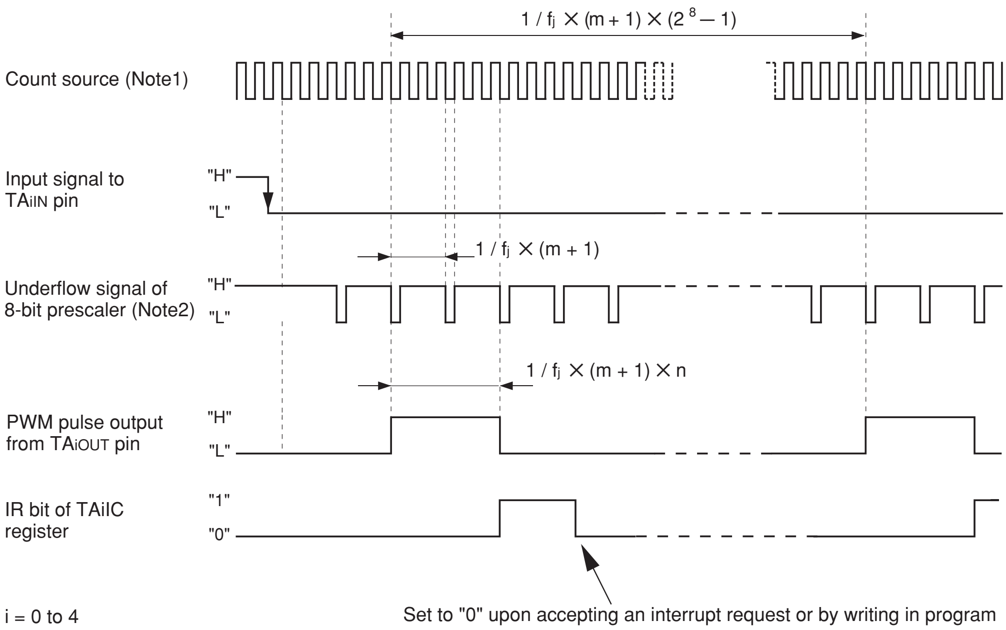
Figure 1.13.12 TAiMR Register in PWM Mode



i = 0 to 4
f_j: Frequency of count source (f_1, f_2, f_8, f_{32}, f_{C32})

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

Figure 1.13.13 Example of 16-bit Pulse Width Modulator Operation



i = 0 to 4
f_j: Frequency of count source (f_1, f_2, f_8, f_{32}, f_{C32})

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 the following case.
- TAi register = 0202_{16}
- The TAiTGH and TAiTGL bits of ONSF or TRGSR register = 00_2 (TAiIN pin input)
- The MR1 bit of TAiMR register = 0 (falling edge)
- The MR2 bit of TAiMR register = 1 (trigger selected by TAiTGH and TAiTGL bits)

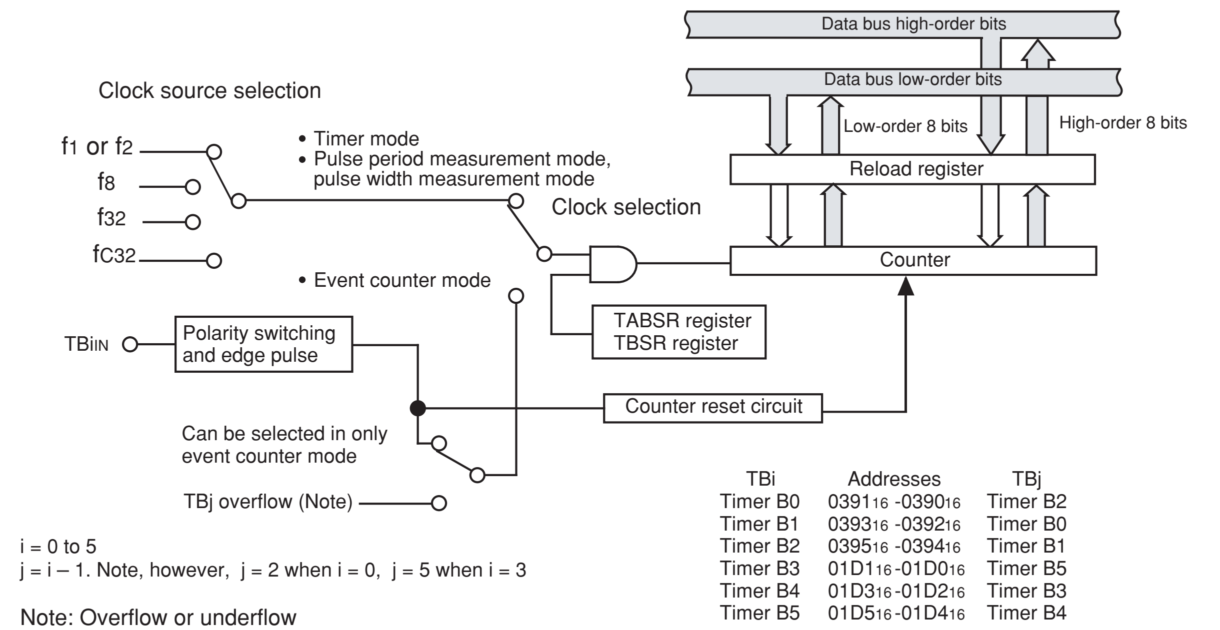
Figure 1.13.14 Example of 8-bit Pulse Width Modulator Operation

Timer B

Figure 1.13.15 shows a block diagram of the timer B. Figures 1.13.16 and 1.13.17 show the timer B-related registers.

Timer B supports the following three modes. Use the TMOD1 and TMOD0 bits of TBiMR register (i = 0 to 5) 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 or underflows of other timers.
- Pulse period/pulse width measuring mode: The timer measures an external signal's pulse period or pulse width.



TBi	Addresses	TBj
Timer B0	0391_{16} -0390_{16}	Timer B2
Timer B1	0393_{16} -0392_{16}	Timer B0
Timer B2	0395_{16} -0394_{16}	Timer B1
Timer B3	$01D1_{16}$-$01D0_{16}$	Timer B5
Timer B4	$01D3_{16}$-$01D2_{16}$	Timer B3
Timer B5	$01D5_{16}$-$01D4_{16}$	Timer B4

Figure 1.13.15 Timer B Block Diagram

Timer Bi mode register (i = 0 to 5)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
TB0MR to TB2MR	$039B_{16}$ to $039D_{16}$	$00XX0000_2$
TB3MR to TB5MR	$01DB_{16}$ to $01DD_{16}$	$00XX0000_2$

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 0 0 : Timer mode 0 1 : Event counter mode 1 0 : Pulse period measurement mode, pulse width measurement mode 1 1 : Must not be set	RW
TMOD1			RW
MR0	———	Function varies with each operation mode	RW
MR1			RW
MR2			RW (Note 1) — (Note 2)
MR3			RO
TCK0	Count source select bit	Function varies with each operation mode	RW
TCK1			RW

Note 1: Timer B0, timer B3.
Note 2: Timer B1, timer B2, timer B4, timer B5.

Timer Bi register (i = 0 to 5) (Note 1)

(b15) b7 (b8) b0 b7 b0

Symbol	Address	After reset
TB0	0391_{16}, 0390_{16}	Indeterminate
TB1	0393_{16}, 0392_{16}	Indeterminate
TB2	0395_{16}, 0394_{16}	Indeterminate
TB3	$01D1_{16}$, $01D0_{16}$	Indeterminate
TB4	$01D3_{16}$, $01D2_{16}$	Indeterminate
TB5	$01D5_{16}$, $01D4_{16}$	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 n + 1 where n = set value (Note 2)	0000_{16} to $FFFF_{16}$	RW
Pulse period modulation mode, Pulse width modulation mode	Measures a pulse period or width	———	RO

Note 1: The register must be accessed in 16-bit unit.
Note 2: The timer counts pulses from an external device or overflows or underflows of other timers.

Figure 1.13.16 TB0MR to TB5MR Registers and TB0 to TB5 Registers

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
TB0S	Timer B0 count start flag		RW
TB1S	Timer B1 count start flag		RW
TB2S	Timer B2 count start flag		RW

Timer B3, B4, B5 count start flag

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
TBSR	$01C0_{16}$	$000XXXXX_2$

Bit symbol	Bit name	Function	RW
— (b4-b0)	Nothing is assigned. When write, set to "0". When read, their contents are indeterminate.		—
TB3S	Timer B3 count start flag	0 : Stops counting 1 : Starts counting	RW
TB4S	Timer B4 count start flag		RW
TB5S	Timer B5 count start flag		RW

Clock prescaler reset flag

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
CPSRF	0381_{16}	$0XXXXXXX_2$

Bit symbol	Bit name	Function	RW
— (b6-b0)	Nothing is assigned. When write, set to "0". When read, their contents are indeterminate.		—
CPSR	Clock prescaler reset flag	Setting this bit to "1" initializes the prescaler for the timekeeping clock. (When read, the value of this bit is "0".)	RW

Figure 1.13.17 TABSR Register, TBSR Register and CPSRF Register

1. Timer Mode

In timer mode, the timer counts a count source generated internally.

Table 1.13.6 lists specifications in timer mode. Figure 1.13.18 shows TBiMR register in timer mode.

Table 1.13.6 Specifications in Timer Mode

Item	Specification
Count source	f_1, f_2, f_8, f_{32}, f_{C32}
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 TBiMR register 0000_{16} to $FFFF_{16}$
Count start condition	Set TBiS bit (Note) to “1” (start counting)
Count stop condition	Set TBiS bit to “0” (stop counting)
Interrupt request generation timing	Timer underflow
TBi_{IN} pin function	I/O port
Read from timer	Count value can be read by reading TBi register
Write to timer	• When not counting and until the 1st count source is input after counting start Value written to TBi register is written to both reload register and counter • When counting (after 1st count source input) Value written to TBi register is written to only reload register (Transferred to counter when reloaded next)

i = 0 to 5

Note : The TB0S to TB2S bits are assigned to the TABSR register bit 5 to bit 7, and the TB3S to TB5S bits are assigned to the TBSR register bit 5 to bit 7.

Timer Bi mode register (i = 0 to 5)

b7 b6 b5 b4 b3 b2 b1 b0: b1 = 0, b0 = 0

Symbol	Address	After reset
TB0MR to TB2MR	$039B_{16}$ to $039D_{16}$	$00XX0000_2$
TB3MR to TB5MR	$01DB_{16}$ to $01DD_{16}$	$00XX0000_2$

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 0 0 : Timer mode	RW
TMOD1			RW
MR0	Has no effect in timer mode Can be set to "0" or "1"		RW
MR1			RW
MR2	TB0MR, TB3MR registers Set to "0" in timer mode		RW
	TB1MR, TB2MR, TB4MR, TB5MR registers Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—
MR3	When write in timer mode, set to "0". When read in timer mode, its content is indeterminate.		RO
TCK0	Count source select bit	b7 b6 0 0 : f_1 or f_2 0 1 : f_8 1 0 : f_{32} 1 1 : f_{C32}	RW
TCK1			RW

Figure 1.13.18 TBiMR 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. Table 1.13.7 lists specifications in event counter mode. Figure 1.13.19 shows TBiMR register in event counter mode.

Table 1.13.7 Specifications in Event Counter Mode

Item	Specification
Count source	• External signals input to TBi_{IN} pin (effective edge can be selected in program) • Timer Bj overflow or underflow (j = i - 1, except j = 2 if i = 0, j = 5 if i = 3)
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 TBi register 0000_{16} to $FFFF_{16}$
Count start condition	Set TBiS bit (Note) to “1” (start counting)
Count stop condition	Set TBiS bit to “0” (stop counting)
Interrupt request generation timing	Timer underflow
TBi_{IN} pin function	Count source input
Read from timer	Count value can be read by reading TBi register
Write to timer	• When not counting and until the 1st count source is input after counting start Value written to TBi register is written to both reload register and counter • When counting (after 1st count source input) Value written to TBi register is written to only reload register (Transferred to counter when reloaded next)

i = 0 to 5
Note: The TB0S to TB2S bits are assigned to the TABSR register bit 5 to bit 7, and the TB3S to TB5S bits are assigned to the TBSR register bit 5 to bit 7.

Timer Bi mode register (i= 0 to 5)

b7 b6 b5 b4 b3 b2 b1 b0: [ ] [ ] [ ] [ ] [ ] [ ] [0] [1]

Symbol	Address	After reset
TB0MR to TB2MR	$039B_{16}$ to $039D_{16}$	$00XX0000_2$
TB3MR to TB5MR	$01DB_{16}$ to $01DD_{16}$	$00XX0000_2$

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 0 1 : Event counter mode	RW
TMOD1			RW
MR0	Count polarity select bit (Note 1)	b3 b2 0 0 : Counts external signal’s falling edges 0 1 : Counts external signal’s rising edges 1 0 : Counts external signal’s falling and rising edges 1 1 : Must not be set	RW
MR1			RW
MR2	TB0MR, TB3MR registers Set to "0" in event counter mode		RW
	TB1MR, TB2MR, TB4MR, TB5MR registers Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—
MR3	When write in event counter mode, set to "0". When read in event counter mode, its content is indeterminate.		RO
TCK0	Has no effect in event counter mode. Can be set to "0" or "1".		RW
TCK1	Event clock select bit	0 : Input from TBi_{IN} pin (Note 2) 1 : TBj overflow or underflow (j = i – 1, except j = 2 if i = 0, j = 5 if i = 3)	RW

Note 1: Effective when the TCK1 bit = 0 (input from TBiIN pin). If the TCK1 bit = 1 (TBj overflow or underflow), these bits can be set to "0" or "1" .
Note 2: The port direction bit for the TBi_{IN} pin must be set to "0" (input mode).

Figure 1.13.19 TBiMR Register in Event Counter Mode

3. Pulse Period and Pulse Width Measurement Mode

In pulse period and pulse width measurement mode, the timer measures pulse period or pulse width of an external signal. Table 1.13.8 lists specifications in pulse period and pulse width measurement mode. Figure 1.13.20 shows TBiMR register in pulse period and pulse width measurement mode. Figure 1.13.21 shows the operation timing when measuring a pulse period. Figure 1.13.22 shows the operation timing when measuring a pulse width.

Table 1.13.8 Specifications in Pulse Period and Pulse Width Measurement Mode

Item	Specification
Count source	f_1, f_2, f_8, f_{32}, f_{C32}
Count operation	• Up-count • Counter value is transferred to reload register at an effective edge of measurement pulse. The counter value is set to "0000_{16}" to continue counting.
Count start condition	Set TBiS bit (Note 1) to "1" (start counting)
Count stop condition	Set TBiS bit to "0" (stop counting)
Interrupt request generation timing	• When an effective edge of measurement pulse is input (Note 2) • Timer overflow. When an overflow occurs, the MR3 bit of TBiMR register is set to "1" (overflow) simultaneously. The MR3 bit is set to "0" (no overflow) by writing to TBiMR register at the next count timing or later after the MR3 bit was set to "1". At this time, make sure TBiS bit is set to "1" (start counting).
TBi_{IN} pin function	Measurement pulse input
Read from timer	Contents of the reload register (measurement result) can be read by reading TBi register (Note 3)
Write to timer	Value written to TBi register is written to neither reload register nor counter

i = 0 to 5

Note 1: The TB0S to TB2S bits are assigned to the TABSR register bit 5 to bit 7, and the TB3S to TB5S bits are assigned to the TBSR register bit 5 to bit 7.

Note 2: Interrupt request is not generated when the first effective edge is input after the timer started counting.

Note 3: Value read from TBi register is indeterminate until the second valid edge is input after the timer starts counting.

Timer Bi mode register (i = 0 to 5)

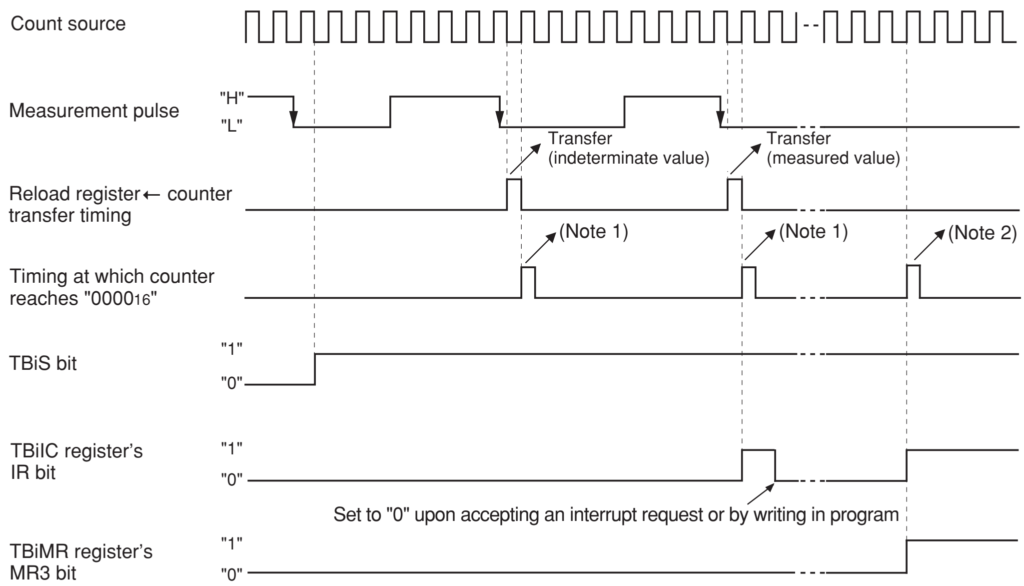
b7	b6	b5	b4	b3	b2	b1	b0
						1	0

Symbol	Address	After reset
TB0MR to TB2MR	$039B_{16}$ to $039D_{16}$	$00XX0000_2$
TB3MR to TB5MR	$01DB_{16}$ to $01DD_{16}$	$00XX0000_2$

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	b1 b0 1 0 : Pulse period / pulse width measurement mode	RW
TMOD1			RW
MR0	Measurement mode select bit	b3 b2 0 0 : Pulse period measurement (Measurement between a falling edge and the next falling edge of measured pulse) 0 1 : Pulse period measurement (Measurement between a rising edge and the next rising edge of measured pulse) 1 0 : Pulse width measurement (Measurement between a falling edge and the next rising edge of measured pulse and between a rising edge and the next falling edge) 1 1 : Must not be set.	RW
MR1			RW
MR2	TB0MR and TB3MR registers Set to "0" in pulse period and pulse width measurement mode		RW
	TB1MR, TB2MR, TB4MR, TB5MR registers Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—
MR3	Timer Bi overflow flag (Note)	0 : Timer did not overflow 1 : Timer has overflown	RO
TCK0	Count source select bit	b7 b6 0 0 : f_1 or f_2 0 1 : f_8 1 0 : f_{32} 1 1 : fC_{32}	RW
TCK1			RW

Note: This flag is indeterminate after reset. When the TBiS bit = 1 (start counting), the MR3 bit is set to "0" (no overflow) by writing to the TBiMR register at the next count timing or later after the MR3 bit was set to "1" (overflow). The MR3 bit cannot be set to "1" in a program. The TB0S to TB2S bits are assigned to the TABSR register's bit 5 to bit 7, and the TB3S to TB5S bits are assigned to the TBSR register's bit 5 to bit 7.

Figure 1.13.20 TBiMR Register in Pulse Period and Pulse Width Measurement Mode

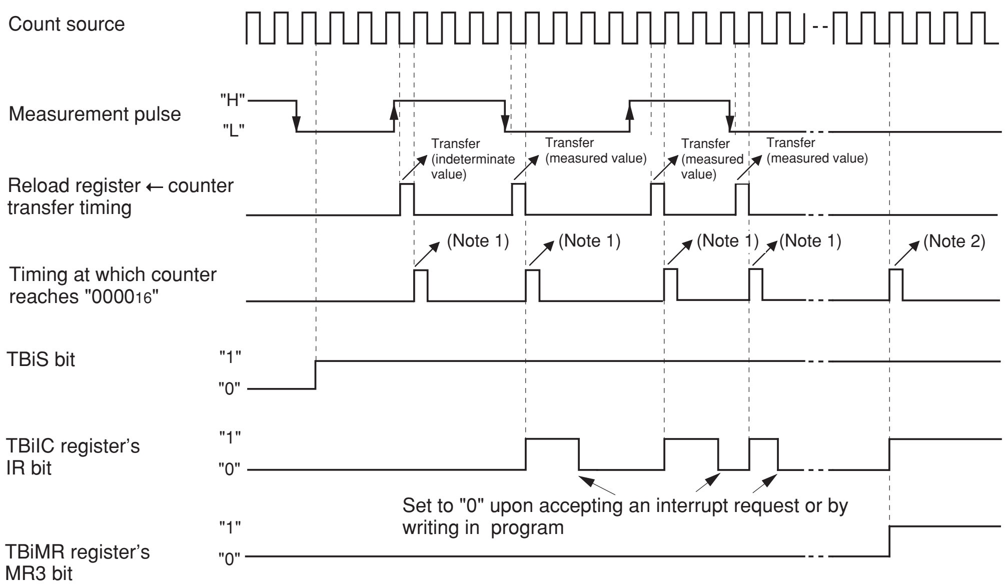


The TB0S to TB2S bits are assigned to the TABSR register's bit 5 to bit 7, and the TB3S to TB5S bits are assigned to the TBSR register's bit 5 to bit 7.

i = 0 to 5

Note 1: Counter is initialized at completion of measurement.
Note 2: Timer has overflown.
Note 3: This timing diagram is for the case where the TBiMR register's MR1 to MR0 bits are "00_2" (measure the interval from falling edge to falling edge of the measurement pulse).

Figure 1.13.21 Operation Timing When Measuring Pulse Period



The TB0S to TB2S bits are assigned to the TABSR register's bit 5 to bit 7, and the TB3S to TB5S bits are assigned to the TBSR register's bit 5 to bit 7.

i = 0 to 5

Note 1: Counter is initialized at completion of measurement.
Note 2: Timer has overflown.
Note 3: This timing diagram is for the case where the TBiMR register's MR1 to MR0 bits are "10_2" (measure the interval from a falling edge to the next rising edge and the interval from a rising edge to the next falling edge of the measurement pulse).

Figure 1.13.22 Operation Timing When Measuring Pulse Width

Three-phase Motor Control Timer Function

Timers A1, A2, A4 and B2 can be used to output three-phase motor drive waveforms. Table 1.14.1 lists the specifications of the three-phase motor control timer function. Figure 1.14.1 shows the block diagram for three-phase motor control timer function. Also, the related registers are shown on Figures 1.14.2 to 1.14.8.

Table 1.14.1 Three-phase Motor Control Timer Function Specifications

Item	Specification
Three-phase waveform output pin	Six pins (U, $\overline{U}$, V, $\overline{V}$, W, $\overline{W}$)
Forced cutoff input (Note)	Input “L” to $\overline{NMI}$ pin
Used Timers	Timer A4, A1, A2 (used in the one-shot timer mode) • Timer A4: U- and $\overline{U}$-phase waveform control • Timer A1: V- and $\overline{V}$-phase waveform control • Timer A2: W- and $\overline{W}$-phase waveform control Timer B2 (used in the timer mode) • Carrier wave cycle control Dead time timer (3 eight-bit timer and shared reload register) • Dead time control
Output waveform	Triangular wave modulation, Sawtooth wave modification • Enable to output “H” or “L” for one cycle • Enable to set positive-phase level and negative-phase level respectively
Carrier wave cycle	Triangular wave modulation: count source × (m+1) × 2 Sawtooth wave modulation: count source × (m+1) m: Setting value of TB2 register, 0 to 65535 Count source: f_1, f_2, f_8, f_{32}, f_{C32}
Three-phase PWM output width	Triangular wave modulation: count source × n × 2 Sawtooth wave modulation: count source × n n: Setting value of TA4, TA1 and TA2 registers (of TA4, TA41, TA1, TA11, TA2 and TA21 registers when setting the INV11 bit to “1”), 1 to 65535 Count source: f_1, f_2, f_8, f_{32}, f_{C32}
Dead time	Count source × p, or no dead time p: Setting value of DTT register, 1 to 255 Count source: f_1, f_2, f_1 divided by 2, f_2 divided by 2
Active level	Enable to select “H” or “L”
Positive and negative-phase concurrent active disable function	Positive and negative-phases concurrent active disable function Positive and negative-phases concurrent active detect function
Interrupt frequency	For Timer B2 interrupt, select a carrier wave cycle-to-cycle basis through 15 times carrier wave cycle-to-cycle basis

Note: Forced cutoff with $\overline{NMI}$ input is effective when the IVPCR1 bit of TB2SC register is set to “1” (three-phase output forcible cutoff by $\overline{NMI}$ input enabled). If an “L” signal is applied to the $\overline{NMI}$ pin when the IVPCR1 bit is “1”, the related pins go to a high-impedance state regardless of which functions of those pins are being used.

Related pins:
- $P7_2$/CLK_2/$TA1_{OUT}$/V
- $P7_3$/$\overline{CTS_2}$/$\overline{RTS_2}$/$TA1_{IN}$/$\overline{V}$
- $P7_4$/$TA2_{OUT}$/W
- $P7_5$/$TA2_{IN}$/$\overline{W}$
- $P8_0$/$TA4_{OUT}$/U
- $P8_1$/$TA4_{IN}$/$\overline{U}$

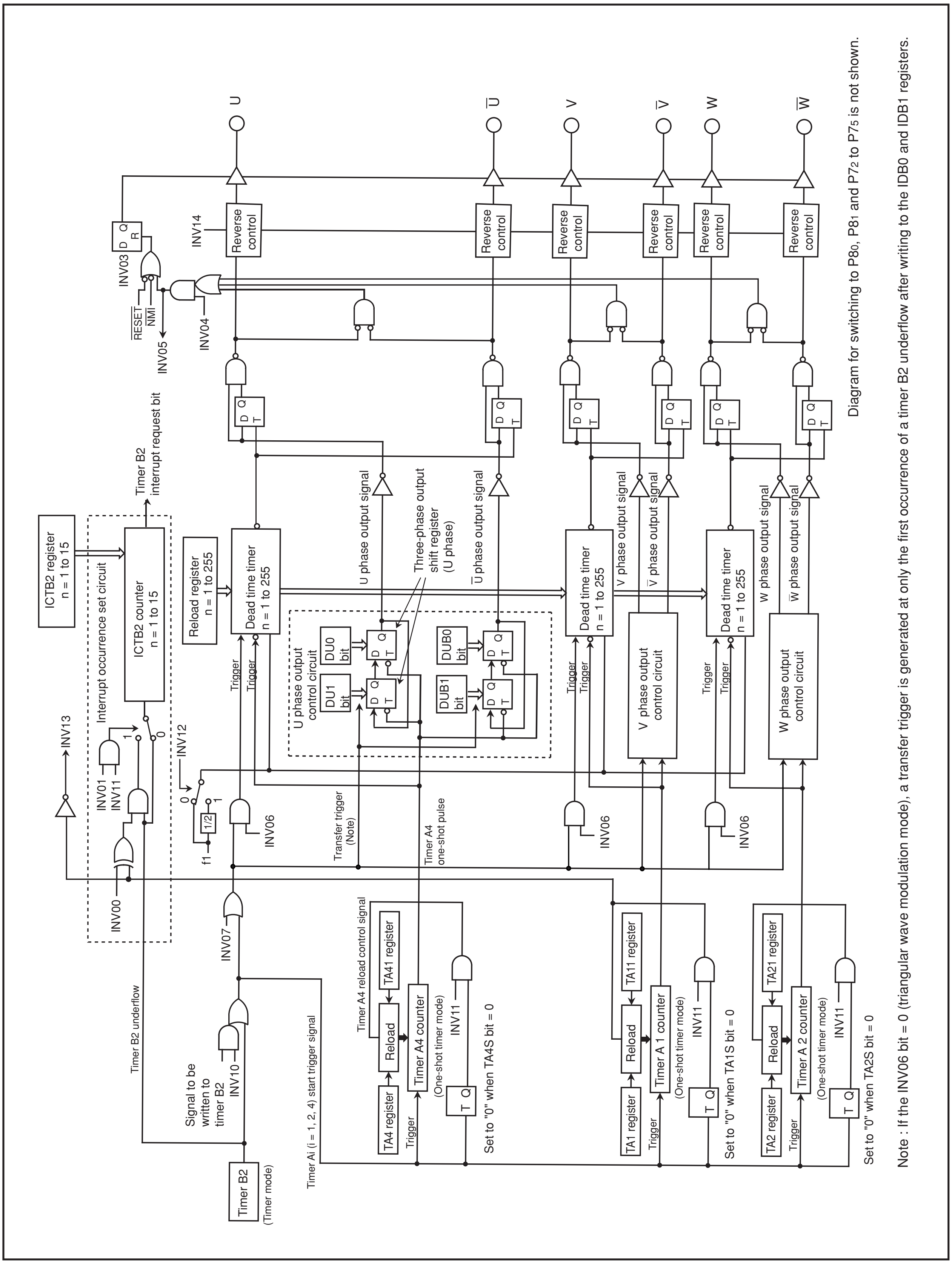


Figure 1.14.1 Three-phase Motor Control Timer Function Block Diagram

Three-phase PWM control register 0 (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
INVC0	$01C8_{16}$	00_{16}

Bit symbol	Bit name	Description	RW
INV00	Effective interrupt output polarity select bit (Note 2)	0: ICTB2 counter incremented by 1 at odd-numbered occurrences of a timer B2 underflow 1: ICTB2 counter incremented by 1 at even-numbered occurrences of a timer B2 underflow	RW
INV01	Effective interrupt output specification bit (Notes 2, 3)	0: ICTB2 counter incremented by 1 at a timer B2 underflow 1: Selected by INV00 bit	RW
INV02	Mode select bit (Note 4)	0: Three-phase motor control timer function unused (Note 5) 1: Three-phase motor control timer function	RW
INV03	Output control bit (Note 6)	0: Three-phase motor control timer output disabled (Note 5) 1: Three-phase motor control timer output enabled	RW
INV04	Positive and negative phases concurrent output disable bit	0: Simultaneous active output enabled 1: Simultaneous active output disabled	RW
INV05	Positive and negative phases concurrent output detect flag	0: Not detected yet 1: Already detected (Note 7)	RW
INV06	Modulation mode select bit (Note 8)	0: Triangular wave modulation mode 1: Sawtooth wave modulation mode (Note 9)	RW
INV07	Software trigger select bit	Setting this bit to "1" generates a transfer trigger. If the INV06 bit is "1", a trigger for the dead time timer is also generated. The value of this bit when read is "0".	RW

Note 1: Write to this register after setting the PRC1 bit of PRCR register to "1" (write enable). Note also that INV00 to INV02, INV04 and INV06 bits can only be rewritten when timers A1, A2, A4 and B2 are idle.
Note 2: Effective when the INV11 bit is "1" (three-phase mode 1). If INV11 is "0" (three-phase mode 0), the ICTB2 counter is incremented by "1" each time the timer B2 underflows, regardless of whether the INV00 and INV01 bits are set.
Note 3: If this bit needs to be set to "1", set any value in the ICTB2 register before writing to it.
Note 4: Setting the INV02 bit to "1" activates the dead time timer, U/V/W-phase output control circuits and ICTB2 counter.
Note 5: All of the U, $\overline{U}$, V, $\overline{V}$, W and $\overline{W}$ pins are placed in the high-impedance state by setting the INV02 bit to 1 (three-phase motor control timer function) and setting the INV03 bit to "0" (three-phase motor control timer output disable).
Note 6: The INV03 bit is set to "0" in the following cases:
- When reset
- When positive and negative go active simultaneously while INV04 bit is "1"
- When set to "0" in a program
- When input on the $\overline{NMI}$ pin changes state from "H" to "L" (The INV03 bit cannot be set to "1" when $\overline{NMI}$ input is "L".)

Note 7: Can only be set by writing "0" in a program, and cannot be set to "1".
Note 8: The effects of the INV06 bit are described in the table below.

Item	INV06 = 0	INV06 = 1
Mode	Triangular wave modulation mode	Sawtooth wave modulation mode
Timing at which transferred from IDB0 to IDB1 registers to three-phase output shift register	Transferred only once synchronously with the transfer trigger after writing to the IDB0 to IDB1 registers	Transferred every transfer trigger
Timing at which dead time timer trigger is generated when INV16 bit is "0"	Synchronous with the falling edge of timer A1, A2, or A4 one-shot pulse	Synchronous with the transfer trigger and the falling edge of timer A1, A2, or A4 one-shot pulse
INV13 bit	Effective when INV11 is "1" and INV06 is "0"	Has no effect

Transfer trigger: Timer B2 underflow, write to the INV07 bit or write to the TB2 register when INV10 is "1"
Note 9: If the INV06 bit is "1", set the INV11 bit to "0" (three-phase mode 0) and set the PWCON bit to "0" (timer B2 reloaded by a timer B2 underflow).

Figure 1.14.2 INVC0 Register

Three-phase PWM control register 1 (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
0							

Symbol	Address	After reset
INVC1	$01C9_{16}$	00_{16}

Bit symbol	Bit name	Description	RW
INV10	Timer A1, A2, A4 start trigger signal select bit	0: Timer B2 underflow 1: Timer B2 underflow and write to the TB2 register	RW
INV11	Timer A1-1, A2-1, A4-1 control bit (Note 2)	0: Three-phase mode 0 1: Three-phase mode 1 (Note 3)	RW
INV12	Dead time timer count source select bit	0 : f_1 or f_2 1 : f_1 divided by 2 or f_2 divided by 2	RW
INV13	Carrier wave detect flag (Note 4)	0: Timer A output at even-numbered occurrences (TA11, TA21, TA41 register value counted) 1: Timer A output at odd-numbered occurrences (TA1, TA2, TA4 register value counted)	RO
INV14	Output polarity control bit	0 : Output waveform "L" active 1 : Output waveform "H" active	RW
INV15	Dead time invalid bit	0: Dead time timer enabled 1: Dead time timer disabled	RW
INV16	Dead time timer trigger select bit	0: Falling edge of timer A4, A1 or A2 one-shot pulse (Note 5) 1: Rising edge of three-phase output shift register (U, V or W phase) output	RW
– (b7)	Reserved bit	Set to "0"	RW

Note 1: Write to this register after setting the PRC1 bit of PRCR register to "1" (write enable). Note also that this register can only be rewritten when timers A1, A2, A4 and B2 are idle.

Note 2: The effects of the INV11 bit are described in the table below.

Item	INV11 = 0	INV11 = 1
Mode	Three-phase mode 0	Three-phase mode 1
TA11, TA21, TA41 registers	Not used	Used
INV00 bit, INV01 bit	Has no effect. ICTB2 counted every time timer B2 underflows regardless of whether the INV00 to INV01 bits are set.	Effect
INV13 bit	Has no effect	Effective when INV11 bit is "1" and INV06 bit is "0"

Note 3: If the INV06 bit is "1" (sawtooth wave modulation mode), set this bit to "0" (three-phase mode 0). Also, if the INV11 bit is "0", set the PWCON bit to "0" (timer B2 reloaded by a timer B2 underflow).

Note 4: The INV13 bit is effective only when the INV06 bit is "0" (triangular wave modulation mode) and the INV11 bit is "1" (three-phase mode 1).

Note 5: If all of the following conditions hold true, set the INV16 bit to "1" (dead time timer triggered by the rising edge of three-phase output shift register output).
- The INV15 bit is "0" (dead time timer enabled)
- When the INV03 bit is set to "1" (three-phase motor control timer output enabled), the Dij bit and DiBj bit (i: U, V, or W, j: 0, 1) have always different values (the positive-phase and negative-phase always output different levels during the period other than dead time).

Conversely, if either one of the above conditions holds false, set the INV16 bit to "0" (dead time timer triggered by the falling edge of one-shot timer pulse).

Figure 1.14.3 INVC1 Register

Three-phase output buffer register i (i = 0, 1) (Note)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
IDB0	$01CA_{16}$	00_{16}
IDB1	$01CB_{16}$	00_{16}

Bit Symbol	Bit name	Function	RW
DUi	U phase output buffer i	Write the output level 0: Active level 1: Inactive level When read, these bits show the three-phase output shift register value.	RW
DUBi	$\overline{U}$ phase output buffer i		RW
DVi	V phase output buffer i		RW
DVBi	$\overline{V}$ phase output buffer i		RW
DWi	W phase output buffer i		RW
DWBi	$\overline{W}$ phase output buffer i		RW
—— (b7-b6)	Nothing is assigned. When write, set to "0". When read, its content is "0".		——

Note: The IDB0 and IDB1 register values are transferred to the three-phase output shift register by a transfer trigger. The value written to the IDB0 register after a transfer trigger generates the output signal of each phase, and the next value written to the IDB1 register at the falling edge of the timer A1, A2 or A4 one-shot pulse represents the output signal of each phase.

Dead time timer (Notes 1, 2)

b7 b0

Symbol	Address	After reset
DTT	$01CC_{16}$	Indeterminate

Function	Setting range	RW
Assuming the set value = n, upon a start trigger the timer starts counting the count source selected by the INV12 bit and stops after counting it n times. The positive or negative phase whichever is going from an inactive to an active level changes at the same time the dead time timer stops.	1 to 255	WO

Note 1: Use MOV instruction to write to this register.
Note 2: Effective when the INV15 bit is "0" (dead time timer enabled). If the INV15 bit is "1", the dead time timer is disabled and has no effect.

Figure 1.14.4 IDB0 Register, IDB1 Register and DTT Register

Timer Ai, Ai-1 register (i = 1, 2, 4) (Notes 1 to 6)

(b15) b7 ... (b8) b0 | b7 ... b0

Symbol	Address	After reset
TA1	0389_{16}-0388_{16}	Indeterminate
TA2	$038B_{16}$-$038A_{16}$	Indeterminate
TA4	$038F_{16}$-$038E_{16}$	Indeterminate
TA11 (Note 7)	$01C3_{16}$-$01C2_{16}$	Indeterminate
TA21 (Note 7)	$01C5_{16}$-$01C4_{16}$	Indeterminate
TA41 (Note 7)	$01C7_{16}$-$01C6_{16}$	Indeterminate

Function	Setting range	RW
Assuming the set value = n, upon a start trigger the timer starts counting the count source and stops after counting it n times. The positive and negative phases change at the same time timer A1, A2 or A4 stops.	0000_{16} to $FFFF_{16}$	WO

Note 1: The register must be accessed in 16-bit unit.
Note 2: When these registers are set to "0000_{16}", the counter does not operate and a timer Ai interrupt does not occur.
Note 3: Use MOV instruction to write to these registers.
Note 4: If the INV15 bit is "0" (dead time timer enabled), the positive or negative phase whichever is going from an inactive to an active level changes at the same time the dead time timer stops.
Note 5: If the INV11 bit is "0" (three-phase mode 0), the TAi register value is transferred to the reload register by a timer Ai (i = 1, 2 or 4) start trigger.
If the INV11 bit is "1" (three-phase mode 1), the TAi1 register value is transferred to the reload register by a timer Ai start trigger first and then the TAi register value is transferred to the reload register by the next timer Ai start trigger. Thereafter, the TAi1 register and TAi register values are transferred to the reload register alternately.
Note 6: Do not write to these registers synchronously with a timer B2 underflow.
Note 7: Write to TAi1 register as follows:
(1) Write a value to the TAi1 register.
(2) Wait for one cycle of timer Ai count source.
(3) Write the same value to the TAi1 register again.

Timer B2 register (Note)

(b15) b7 ... (b8) b0 | b7 ... b0

Symbol	Address	After reset
TB2	0395_{16}-0394_{16}	Indeterminate

Function	Setting range	RW
Divide the count source by n + 1 where n = set value. Timer A1, A2 and A4 are started at every occurrence of underflow.	0000_{16} to $FFFF_{16}$	RW

Note: The register must be accessed in 16-bit unit.

Figure 1.14.5 TA1, TA2, TA4, TA11, TA21 and TA41 Registers, and TB2 Register

Timer B2 interrupt occurrences frequency set counter

b7 b0

Symbol	Address	After reset
ICTB2	$01CD_{16}$	Indeterminate

Function	Setting range	RW
If the INV01 bit is "0" (ICTB2 counter counted every time timer B2 underflows), assuming the set value = n, a timer B2 interrupt is generated at every n'th occurrence of a timer B2 underflow. If the INV01 bit is "1" (ICTB2 counter count timing selected by the INV00 bit), assuming the set value = n, a timer B2 interrupt is generated at every n'th occurrence of a timer B2 underflow that meets the condition selected by the INV00 bit. (Note)	1 to 15	WO
Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—

Note: Use MOV instruction to write to this register.
If the INV01 bit = 1, make sure the TB2S bit also = 0 (timer B2 count stopped) when writing to this register.
If the INV01 bit = 0, although this register can be written even when the TB2S bit = 1 (timer B2 count start), do not write synchronously with a timer B2 underflow

Timer B2 special mode register (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
TB2SC	$039E_{16}$	$XXXXXX00_2$

Bit symbol	Bit name	Function	RW
PWCOM	Timer B2 reload timing switching bit	0 : Timer B2 underflow 1 : Timer A output at odd-numbered occurrences (Note 2)	RW
IVPCR1	Three-phase output port $\overline{NMI}$ control bit 1 (Note 3)	0 : Three-phase output forcible cutoff by $\overline{NMI}$ input (high-impedance) disabled 1 : Three-phase output forcible cutoff by $\overline{NMI}$ input (high-impedance) enabled	RW
— (b7-b2)	Nothing is assigned. When write, set to "0". When read, its content is "0".		—

Note 1: Write to this register after setting the PRC1 bit of PRCR register to "1" (write enabled).
Note 2: If the INV11 bit is "0" (three-phase mode 0) or the INV06 bit is "1" (sawtooh wave modulation mode), set this bit to "0" (timer B2 underflow).
Note 3: Related pins are U($P8_0/TA4_{OUT}$), $\overline{U}$($P8_1/TA4_{IN}$), V($P7_2/CLK_2/TA1_{OUT}$), $\overline{V}$($P7_3/\overline{CTS_2}/\overline{RTS_2}/TA1_{IN}$), W($P7_4/TA2_{OUT}$), $\overline{W}$($P7_5/TA2_{IN}$). If a low-level signal is applied to the $\overline{NMI}$ pin when the IVPCR1 bit = 1, the target pins go to a high-impedance state regardless of which functions of those pins are being used. After forced interrupt (cutoff), input "H" to the $\overline{NMI}$ pin and set IVPCR1 bit to "0": this forced cutoff will be reset.

Figure 1.14.6 ICTB2 Register and TB2SC Register

Trigger select register

b7 b6 b5 b4 b3 b2 b1 b0

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

Bit	Bit symbol	Bit name	Function	RW
b0	TA1TGL	Timer A1 event/trigger select bit	To use the V-phase output control circuit, set these bits to "01_2" (TB2 underflow).	RW
b1	TA1TGH			RW
b2	TA2TGL	Timer A2 event/trigger select bit	To use the W-phase output control circuit, set these bits to "01_2" (TB2 underflow).	RW
b3	TA2TGH			RW
b4	TA3TGL	Timer A3 event/trigger select bit	b5 b4 0 0 : Input on $TA3_{IN}$ is selected (Note) 0 1 : TB2 is selected 1 0 : TA2 is selected 1 1 : TA4 is selected	RW
b5	TA3TGH			RW
b6	TA4TGL	Timer A4 event/trigger select bit	To use the U-phase output control circuit, set these bits to "01_2" (TB2 underflow).	RW
b7	TA4TGH			RW

Note: Set the corresponding port direction bit to "0" (input mode).

Count start flag

b7 b6 b5 b4 b3 b2 b1 b0

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

Bit	Bit symbol	Bit name	Function	RW
b0	TA0S	Timer A0 count start flag	0 : Stops counting 1 : Starts counting	RW
b1	TA1S	Timer A1 count start flag		RW
b2	TA2S	Timer A2 count start flag		RW
b3	TA3S	Timer A3 count start flag		RW
b4	TA4S	Timer A4 count start flag		RW
b5	TB0S	Timer B0 count start flag		RW
b6	TB1S	Timer B1 count start flag		RW
b7	TB2S	Timer B2 count start flag		RW

Figure 1.14.7 TRGSR Register and TRBSR Register

Timer Ai mode register (i = 1, 2, 4)

b7	b6	b5	b4	b3	b2	b1	b0
		0	1		0	1	0

Symbol	Address	After reset
TA1MR	0397_{16}	00_{16}
TA2MR	0398_{16}	00_{16}
TA4MR	$039A_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	Set to "10_2" (one-shot timer mode) for the three-phase motor control timer function	RW
TMOD1			RW
MR0	Pulse output function select bit	Set to "0" for the three-phase motor control timer function	RW
MR1	External trigger select bit	Has no effect for the three-phase motor control timer function	RW
MR2	Trigger select bit	Set to "1" (selected by TRGSR register) for the three-phase motor control timer function	RW
MR3	Set to "0" for the three-phase motor control timer function		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 : fC_{32}	RW
TCK1			RW

Timer B2 mode register

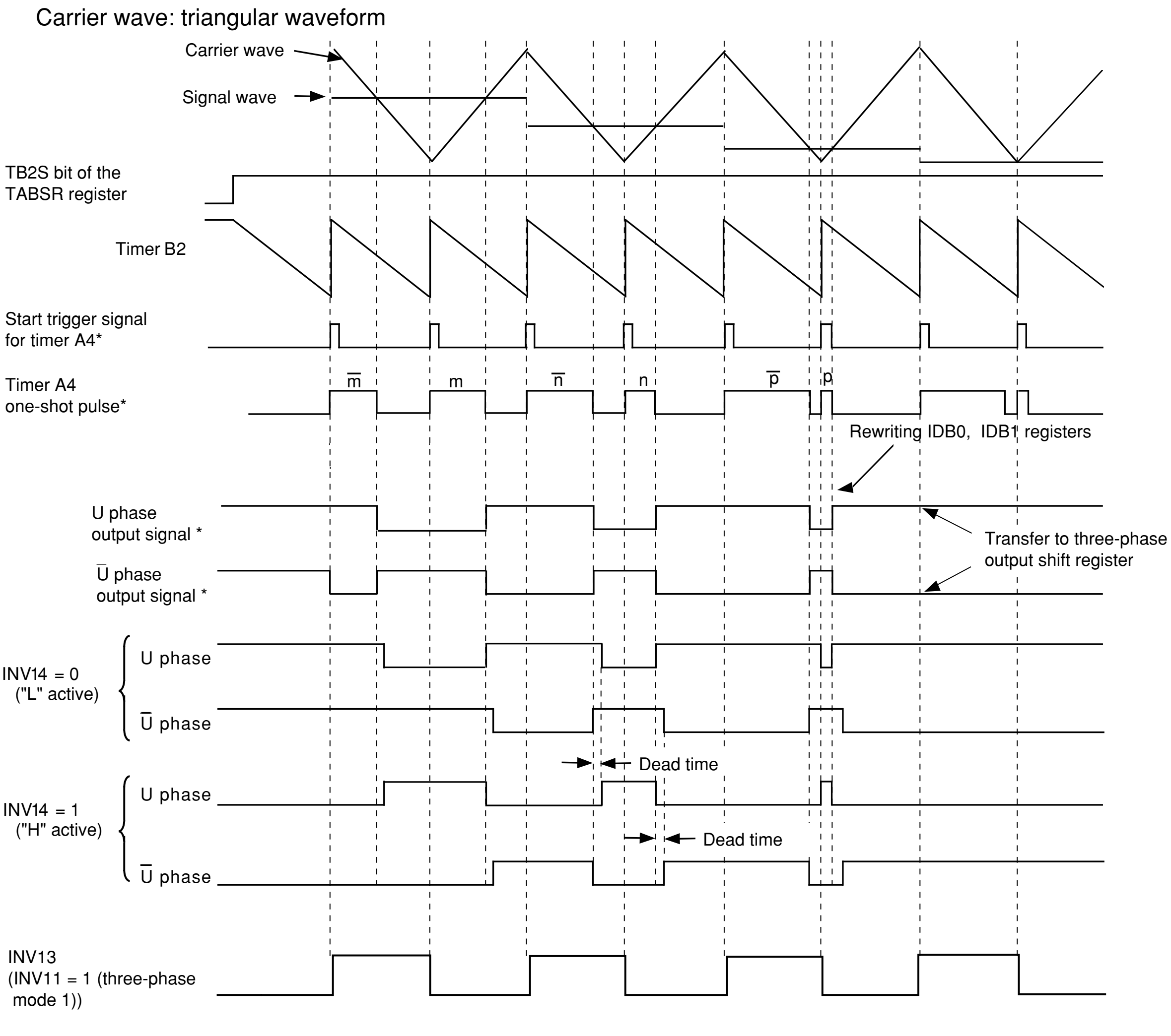
b7	b6	b5	b4	b3	b2	b1	b0
			0			0	0

Symbol	Address	After reset
TB2MR	$039D_{16}$	$00XX0000_2$

Bit symbol	Bit name	Function	RW
TMOD0	Operation mode select bit	Set to "00_2" (timer mode) for the three-phase motor control timer function	RW
TMOD1			RW
MR0	Has no effect for the three-phase motor control timer function . When write, set to "0". When read, its content is indeterminate .		RW
MR1			RW
MR2	Set to "0" for the three-phase motor control timer function		RW
MR3	When write in the three-phase motor control timer function, write "0". When read, its content is indeterminate.		RO
TCK0	Count source select bit	b7 b6 0 0 : f_1 or f_2 0 1 : f_8 1 0 : f_{32} 1 1 : fC_{32}	RW
TCK1			RW

Figure 1.14.8 TA1MR, TA2MR and TA4MR Registers, and TB2MR Register

The three-phase motor control timer function is enabled by setting the INV02 bit of INVC0 register to “1”. When this function is selected, timer B2 is used to control the carrier wave, and timers A4, A1 and A2 are used to control three-phase PWM outputs (U, $\overline{U}$, V, $\overline{V}$, W and $\overline{W}$). The dead time is controlled by a dedicated dead-time timer. Figure 1.14.9 shows the example of triangular modulation waveform and Figure 1.14.10 shows the example of sawtooth modulation waveform.



* Internal signals. See the block diagram of the three-phase motor control timer function.

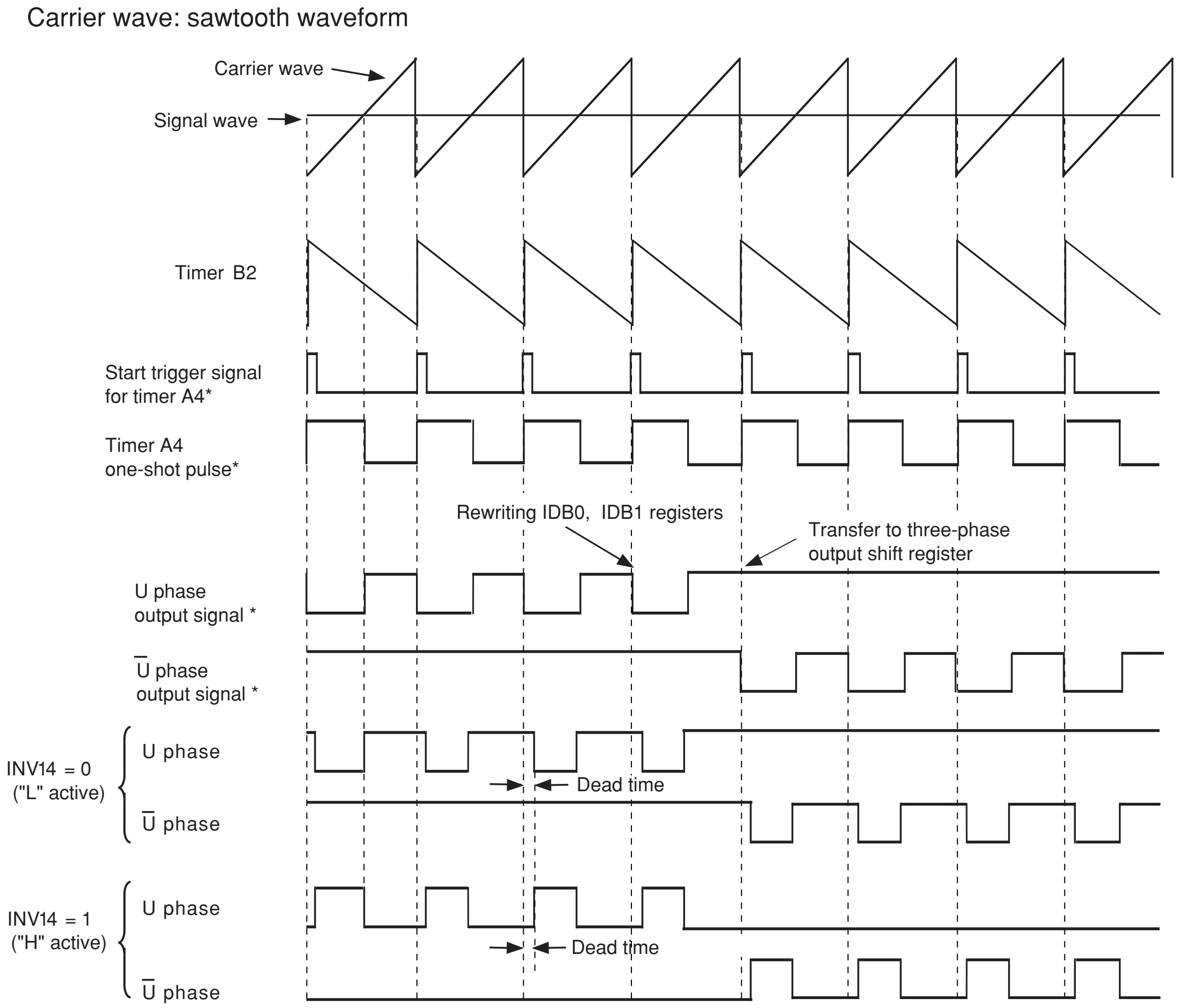
Shown here is a typical waveform for the case where INVC0 = 00XX11XX2 (X = set as suitable for the system) and INVC1 = 010XXXX02.
An example for changing PWM outputs is shown below.

(1)When INV11 = 1 (three-phase mode 1)
- INV01 = 0, ICTB2 = 216 (timer B2 interrupt is generated at every 2’th occurrence of a timer B2 underflow), or INV01 = 1, INV00 = 1, ICTB2=116(timer B2 interrupt is generated at even-numbered occurrences of a timer B2 underflow).
- Initial timer value: TA41 = $\overline{m}$, TA4 = m. The TA4 and TA41 registers are modified every time a timer B2 interrupt occurs. First time, TA41 = $\overline{n}$, TA4 = n. Second time, TA41 = $\overline{p}$, TA4 = p.
- Initial values of IDB0 and IDB1 registers: DU0 = 1, DUB0 = 0, DU1 = 0, DUB1 = 1.The register values are changed to DU0 = 1, DUB0 = 0, DU1= 1 and DUB1 = 0 the third time a timer B2 interrupt occurs.

(2)When INV11 = 0 (three-phase mode 0)
- INV01 = 0, ICTB2 = 116 (timer B2 interrupt is generated at every occurrence of a timer B2 underflow)
- Initial timer value: TA4 = $\overline{m}$. The TA4 register is modified each time a timer B2 interrupt occurs. First time, TA4 = m. Second time, TA4 = $\overline{n}$. Third time, TA4 = n. Fourth time, TA4 = $\overline{p}$. Fifth time, TA4 = p.
- Initial values of IDB0 and IDB1 registers: DU0 = 1, DUB0 = 0, DU1 = 0, DUB1 = 1. The register values are changed to DU0 = 1, DUB0 = 0, DU1= 1 and DUB1 = 0 the sixth time a timer B2 interrupt occurs.

The value written to the TA4 register and TA41 register are inverted at odd-numbered timer A outputs.

Figure 1.14.9 Triangular Wave Modulation Operation



* Internal signals. See the block diagram of the three-phase motor control timer function.

Shown here is a typical waveform for the case where INVC0 = 01XX110X2 (X = set as suitable for the system) and INVC1 = 010XXX002.
An example for changing PWM outputs is shown below.
• Initial values of IDB0 and IDB1 registers: DU0 = 0, DUB0 = 1, DU1 = 1, DUB1 = 1.
The register values are changed to DU0 = 1, DUB0 = 0, DU1 = 1, DUB1 = 1 a timer B2 interrupt occurs.

Figure 1.14.10 Sawtooth Wave Modulation Operation

Serial I/O

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

UARTi (i = 0 to 2)

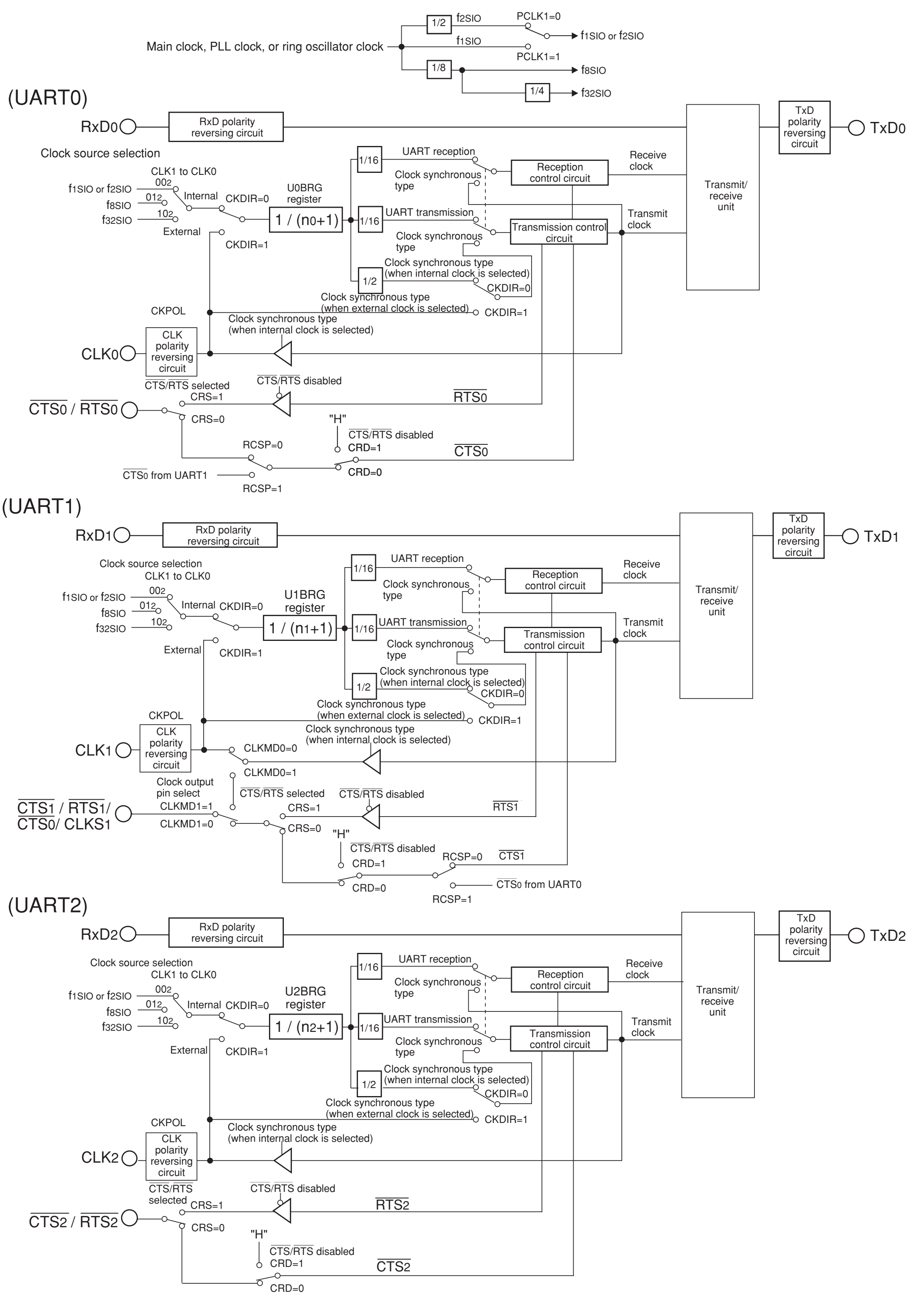
Each UARTi has an exclusive timer to generate a transfer clock, so they operate independently of each other.

Figure 1.15.1 shows the block diagram of UARTi. Figures 1.15.2 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
- Special mode 3 (Bus collision detection function, IE mode) : UART0, UART1
- Special mode 4 (SIM mode) : UART2

Figures 1.15.3 to 1.15.8 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

Figure 1.15.1 UARTi Block Diagram

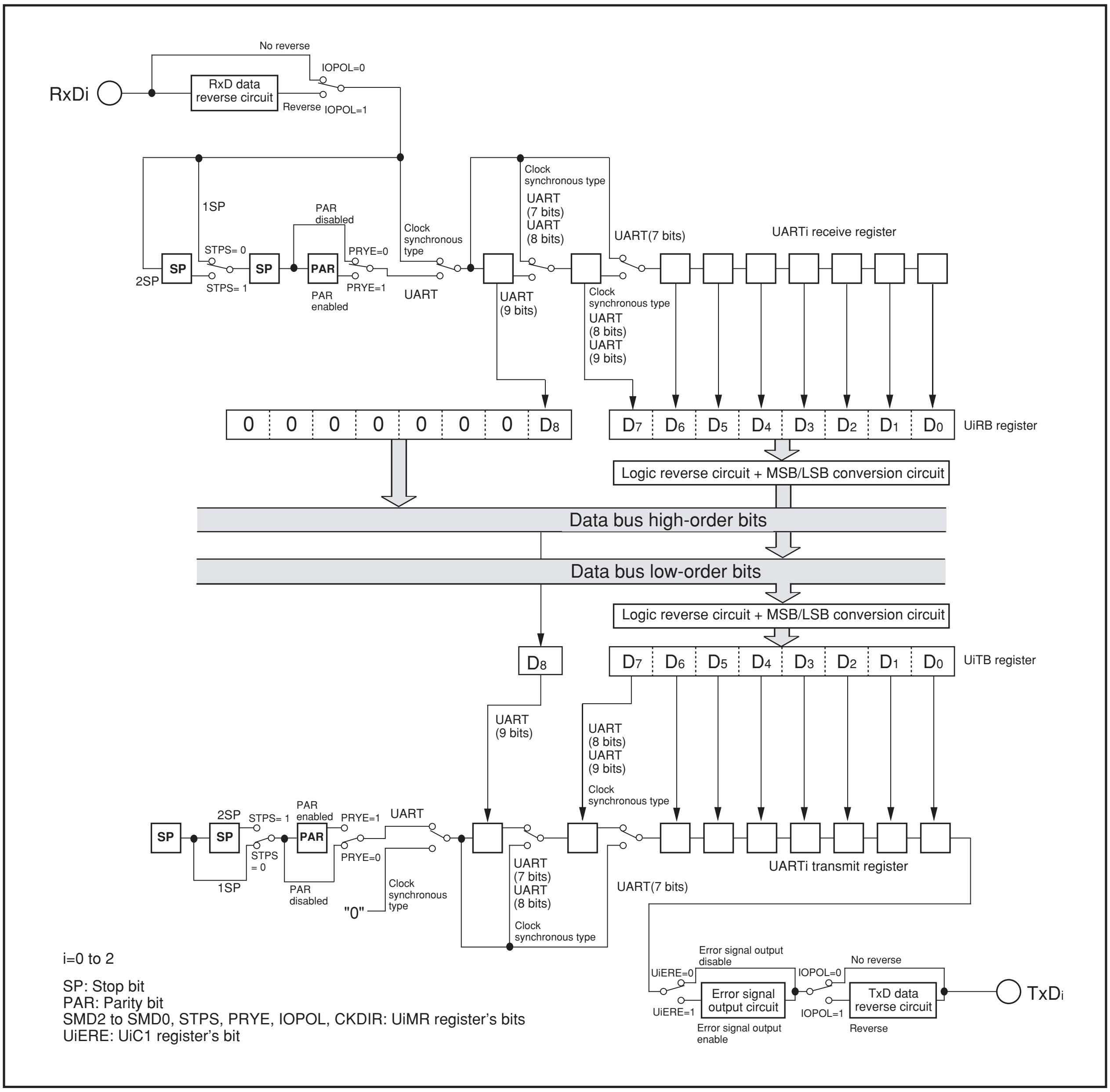


Figure 1.15.2 UARTi Transmit/Receive Unit

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

(b15) b7 (b8) b0 b7 b0

Symbol	Address	After reset
U0TB	$03A3_{16}$-$03A2_{16}$	Indeterminate
U1TB	$03AB_{16}$-$03AA_{16}$	Indeterminate
U2TB	$01FB_{16}$-$01FA_{16}$	Indeterminate

Function	RW
Transmit data	WO
Nothing is assigned· When write, set to "0". When read, their contents are indeterminate.	—

Note: Use MOV instruction to write to this register.

UARTi receive buffer register (i = 0 to 2)

(b15) b7 (b8) b0 b7 b0

Symbol	Address	After reset
U0RB	$03A7_{16}$-$03A6_{16}$	Indeterminate
U1RB	$03AF_{16}$-$03AE_{16}$	Indeterminate
U2RB	$01FF_{16}$-$01FE_{16}$	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 When write, set to "0". When read, their contents are indeterminate.		—
ABT	Arbitration lost detecting flag (Note 1)	0 : Not detected 1 : Detected	RW
OER	Overrun error flag (Note 2)	0 : No overrun error 1 : Overrun error found	RO
FER	Framing error flag (Note 2)	0 : No framing error 1 : Framing error found	RO
PER	Parity error flag (Note 2)	0 : No parity error 1 : Parity error found	RO
SUM	Error sum flag (Note 2)	0 : No error 1 : Error found	RO

Note 1: The ABT bit is set to "0" by writing "0" in a program. (Writing "1" has no effect.)
Note 2: 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 are = 0 (no error). Also, the PER and FER bits are set to "0" by reading the lower byte of the UiRB register.

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

b7 b0

Symbol	Address	After reset
U0BRG	$03A1_{16}$	Indeterminate
U1BRG	$03A9_{16}$	Indeterminate
U2BRG	$01F9_{16}$	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 1.15.3 U0TB to U2TB Registers, U0RB to U2RB Registers, and U0BRG to U2BRG Registers

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

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
U0MR to U2MR	03A0$_{16}$, 03A8$_{16}$, 01F8$_{16}$	00$_{16}$

Bit symbol	Bit name	Function	RW
SMD0	Serial I/O mode select bit (Note 1)	b2 b1 b0 0 0 0 : Serial I/O disabled 0 0 1 : Clock synchronous serial I/O mode 0 1 0 : I^{2}C mode (Note 2) 1 0 0 : UART mode transfer data 7-bit long 1 0 1 : UART mode transfer data 8-bit long 1 1 0 : UART mode transfer data 9-bit 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 3)	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: To receive data, set the corresponding port direction bit for each RxDi pin to "0" (input mode).
Note 2: Set the corresponding port direction bit for SCL and SDA pins to "0" (input mode).
Note 3: Set the corresponding port direction bit for each CLKi pin to "0" (input mode).

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

b7 b6 b5 b4 b3 b2 b1 b0

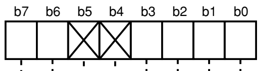
Symbol	Address	After reset
U0C0 to U2C0	03A4$_{16}$, 03AC$_{16}$, 01FC$_{16}$	00001000$_2$

Bit symbol	Bit name	Function	RW
CLK0	BRG count source select bit	b1 b0 0 0 : f1SIO or f2SIO is selected 0 1 : f8SIO is selected 1 0 : f32SIO is selected 1 1 : Must not be set	RW
CLK1			RW
CRS	$\overline{CTS}$/$\overline{RTS}$ function select bit (Note 1)	Effective when CRD = 0 0 : $\overline{CTS}$ function is selected (Note 2) 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 3)	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 4)	0 : LSB first 1 : MSB first	RW

Note 1: $\overline{CTS_1}$/$\overline{RTS_1}$ can be used when the UCON register's CLKMD1 bit = 0 (only CLK1 output) and the UCON register's RCSP bit = 0 (CTS_0/RTS_0 not separated).
Note 2: Set the corresponding port direction bit for each $\overline{CTSi}$ pin to "0" (input mode).
Note 3: SCL$_2$/P7$_1$ is N channel open-drain output. Cannot be set to the CMOS output. Set the NCH bit of the U2C0 register to "0".
Note 4: Effective for clock synchronous serial I/O mode and UART mode transfer data 8-bit long.

Figure 1.15.4 U0MR to U2MR Registers and U0C0 to U2C0 Registers

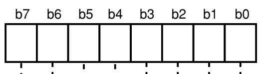
UARTj transmit/receive control register 1 (j = 0, 1)



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 UjTB register 1 : No data present in UjTB register	RO
RE	Receive enable bit	0 : Reception disabled 1 : Reception enabled	RW
RI	Receive complete flag	0 : No data present in UjRB register 1 : Data present in UjRB register	RO
— (b5-b4)	Nothing is assigned. When write, set to "0". When read, these contents are "0".		—
UjLCH	Data logic select bit	0 : No reverse 1 : Reverse	RW
UjERE	Error signal output enable bit	0 : Output disabled 1 : Output enabled	RW

UART2 transmit/receive control register 1



Symbol	Address	After reset
U2C1	$01FD_{16}$	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 1.15.5 U0C1 to U2C1 Registers

UART transmit/receive control register 2

b7 b6 b5 b4 b3 b2 b1 b0

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 enabled	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 ($\overline{CTS_0}$ supplied from the $P6_4$ pin)	RW
— (b7)	Nothing is assigned. When write, set to "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)

UARTi special mode register (i = 0 to 2)

b7 b6 b5 b4 b3 b2 b1 b0 (b3 = 0)

Symbol	Address	After reset
U0SMR to U2SMR	$01EF_{16}$, $01F3_{16}$, $01F7_{16}$	$X0000000_2$

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)	Reserved bit	Set to "0"	RW
ABSCS	Bus collision detect sampling clock select bit	0 : Rising edge of transfer clock 1 : Underflow signal of timer Aj (Note 2)	RW
ACSE	Auto clear function select bit of transmit enable bit	0 : No auto clear function 1 : Auto clear at occurrence of bus collision	RW
SSS	Transmit start condition select bit	0 : Not synchronized to RxD_i 1 : Synchronized to RxD_i (Note 3)	RW
— (b7)	Nothing is assigned. When write, set to "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.).
Note 2: Underflow signal of timer A3 in UART0, underflow signal of timer A4 in UART1, underflow signal of timer A0 in UART2.
Note 3: When a transfer begins, the SSS bit is set to "0" (Not synchronized to RxD_i).

Figure 1.15.6 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	$01EE_{16}$, $01F2_{16}$, $01F6_{16}$	$X0000000_2$

Bit symbol	Bit name	Function	RW
IICM2	I^2C mode select bit 2	Refer to "Table 1.15.11 I^2C Mode FUnctions"	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: "L" output	RW
SDHI	SDA output disable bit	0: Enabled 1: Disabled (high-impedance)	RW
— (b7)	Nothing is assigned. When write, set to "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	$01ED_{16}$, $01F1_{16}$, $01F5_{16}$	$000X0X0X_2$

Bit symbol	Bit name	Function	RW
— (b0)	Nothing is assigned. When write, set to "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 to "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 to "0". When read, its content is indeterminate.		—
DL0	SDAi digital delay setup bit (Notes 1, 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^2C mode. In other than I^2C 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 1.15.7 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: U0SMR4 to U2SMR4
Address: $01EC_{16}$, $01F0_{16}$, $01F4_{16}$
After reset: 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 1.15.8 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 1.15.1 lists the specifications of the clock synchronous serial I/O mode. Table 1.15.2 lists the registers used in clock synchronous serial I/O mode and the register values set.

Table 1.15.1 Clock Synchronous Serial I/O Mode Specifications

Item	Specification
Transfer data format	• Transfer data length: 8 bits
Transfer clock	• UiMR 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 $\overline{CTS}$ function, $\overline{RTS}$ function or $\overline{CTS}/\overline{RTS}$ function disabled
Transmission start condition	• Before transmission can start, the following requirements must be met (Note 1) – The TE bit of UiC1 register = 1 (transmission enabled) – The TI bit of UiC1 register = 0 (data present in UiTB register) – If $\overline{CTS}$ function is selected, input on the $\overline{CTS}$i pin = L
Reception start condition	• Before reception can start, the following requirements must be met (Note 1) – The RE bit of UiC1 register = 1 (reception enabled) – The TE bit of UiC1 register = 1 (transmission enabled) – The TI bit of UiC1 register = 0 (data present in the UiTB register)
Interrupt request generation timing	• For transmission, one of the following conditions can be selected – The UiIRS bit (Note 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 3) 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 selected to occur synchronously with the rising or the falling edge of the transfer clock • LSB first, MSB first selection Whether to start sending/receiving data beginning with bit 0 or beginning with bit 7 can be selected • Continuous receive mode selection Reception is enabled immediately by reading the UiRB register • Switching serial data logic This function reverses the logic value of the transmit/receive data • Transfer clock output from multiple pins selection (UART1) The output pin can be selected in a program from two UART1 transfer clock pins that have been set • Separate $\overline{CTS}/\overline{RTS}$ pins (UART0) $\overline{CTS_0}$ and $\overline{RTS_0}$ are input/output from separate pins

i = 0 to 2

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

Table 1.15.2 Registers to Be Used and Settings in Clock Synchronous Serial I/O Mode

Register	Bit	Function
UiTB (Note 1)	0 to 7	Set transmission data
UiRB (Note 1)	0 to 7	Reception data can be read
	OER	Overrun error flag
UiBRG	0 to 7	Set a transfer rate
UiMR (Note 1)	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 2)	Select the source of UART2 transmit interrupt
	U2RRM (Note 2)	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"

i = 0 to 2

Note 1: Not all register bits are described above. Set those bits to "0" when writing to the registers in clock synchronous serial I/O mode.

Note 2: 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.

Table 1.15.3 lists the functions of the input/output pins during clock synchronous serial I/O mode. Table 1.15.3 shows pin functions for the case where the multiple transfer clock output pin select function is deselected. Table 1.15.4 lists the $P6_4$ 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 TxD_i pin outputs an "H". (If the N channel open-drain output is selected, this pin is in a high-impedance state.)
Figure 1.15.9 shows the transmit/receive timings during clock synchronous serial I/O mode.

Table 1.15.3 Pin Functions (When Not Select Multiple Transfer Clock Output Pin Function)

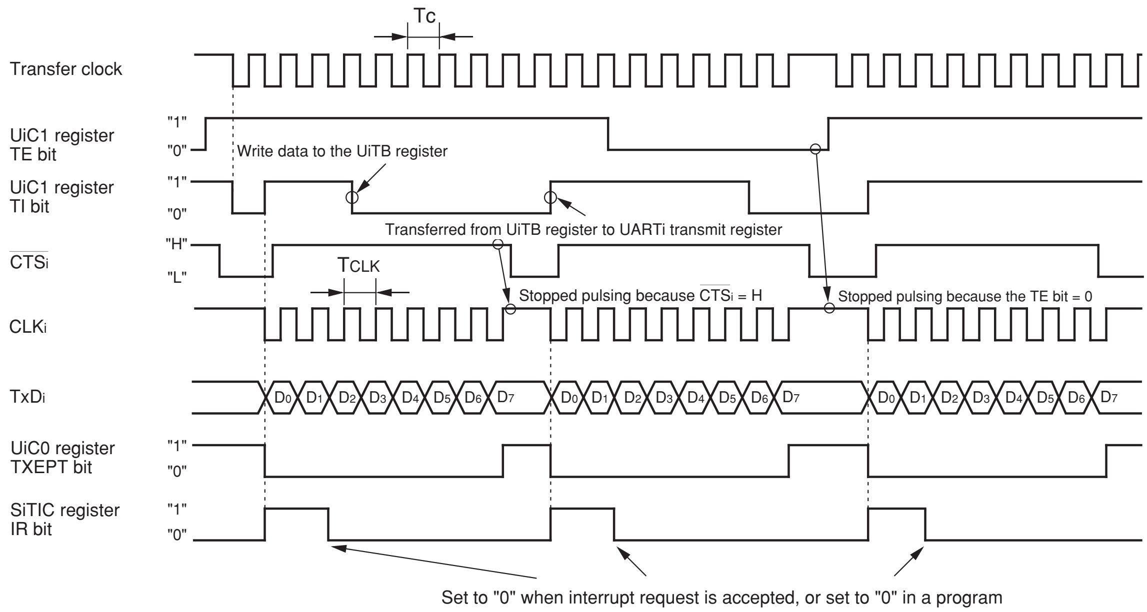
Pin name	Function	Method of selection
TxD_i ($P6_3$, $P6_7$, $P7_0$)	Serial data output	(Outputs dummy data when performing reception only)
RxD_i ($P6_2$, $P6_6$, $P7_1$)	Serial data input	PD6 register's PD6_2 bit = 0, PD6_6 bit = 0 PD7 register's PD7_1 bit = 0 (Can be used as an input port when performing transmission only)
CLK_i ($P6_1$, $P6_5$, $P7_2$)	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 PD7 register's PD7_2 bit = 0
$\overline{CTS_i}/\overline{RTS_i}$ ($P6_0$, $P6_4$, $P7_3$)	$\overline{CTS}$ 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
	$\overline{RTS}$ output	UiC0 register's CRD bit = 0 UiC0 register's CRS bit = 1
	I/O port	UiC0 register's CRD bit = 1

Table 1.15.4 $P6_4$ Pin Functions

Pin function	Bit set value					
	U1C0 register		UCON register			PD6 register
	CRD	CRS	RCSP	CLKMD1	CLKMD0	PD6_4
$P6_4$	1	-	0	0	-	Input: 0, Output: 1
$\overline{CTS_1}$	0	0	0	0	-	0
$\overline{RTS_1}$	0	1	0	0	-	-
$\overline{CTS_0}$ (Note 1)	0	0	1	0	-	0
$CLKS_1$	-	-	-	1 (Note 2)	1	-

Note 1: In addition to this, set the U0C0 register's CRD bit to "0" ($\overline{CTS_0}/\overline{RTS_0}$ enabled) and the U0C0 register's CRS bit to "1" ($\overline{RTS_0}$ 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

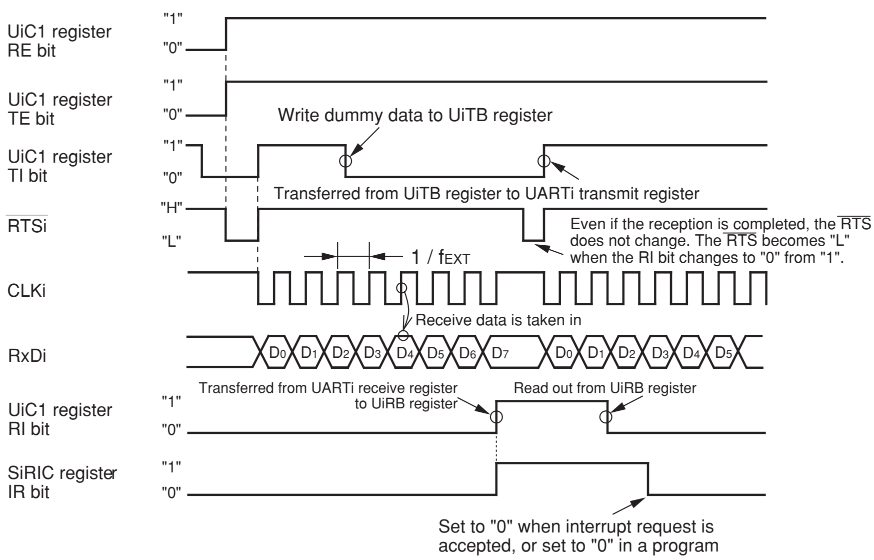
(1) Example of transmit timing (when internal clock is selected)



$Tc = T_{CLK} = 2(n + 1) / fj$
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 (CTS/RTS enabled), CRS bit = 0 (CTS selected)
- UiC0 register CKPOL bit = 0 (transmit data output at the falling edge and receive data taken in at the rising edge of the transfer clock)
- UiRS bit = 0 (an interrupt request occurs when the transmit buffer becomes empty): U0IRS bit is the UCON register bit 0, U1IRS bit is the UCON register bit 1, and U2IRS bit is the U2C1 register bit 4

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



The above timing diagram applies to the case where the register bits are set as follows:
- UiMR register CKDIR bit = 1 (external clock)
- UiC0 register CRD bit = 0 (CTS/RTS enabled), CRS bit = 1 (RTS selected)
- UiC0 register CKPOL bit = 0 (transmit data output at the falling edge and receive data taken in at the rising edge of the transfer clock)

f_{EXT}: frequency of external clock

Make sure the following conditions are met when input to the CLKi pin before receiving data is high:
- UiC1 register TE bit = 1 (transmission enabled)
- UiC1 register RE bit = 1 (reception enabled)
- Write dummy data to the UiTB registe

Figure 1.15.9 Transmit and Receive Operation

(a) CLK Polarity Select Function

Use the UiC0 register (i = 0 to 2)'s CKPOL bit to select the transfer clock polarity. Figure 1.15.10 shows the polarity of the transfer clock.

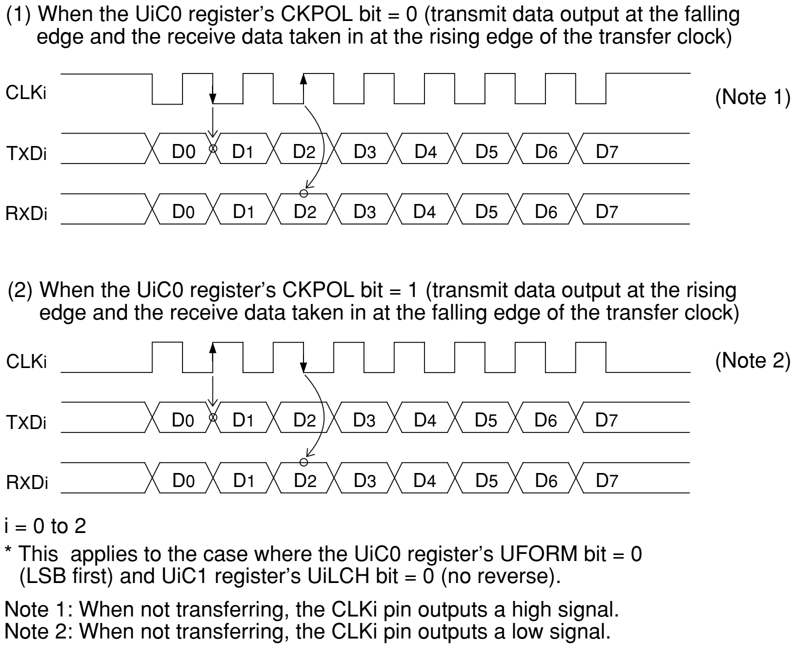


Figure 1.15.10 Transfer Clock Polarity

(b) LSB First/MSB First Select Function

Use the UiC0 register (i = 0 to 2)'s UFORM bit to select the transfer format. Figure 1.15.11 shows the transfer format.

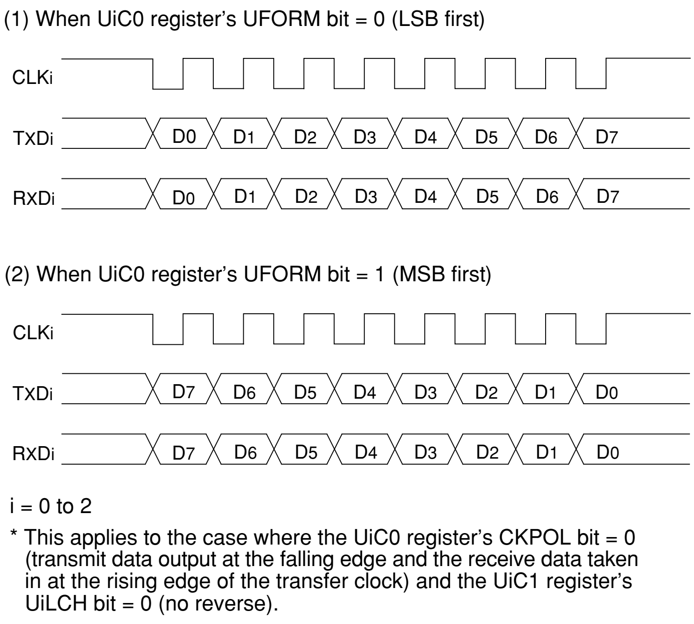


Figure 1.15.11 Transfer Format

(c) Continuous Receive Mode

When the UiRRM bit (i = 0 to 2) = 1 (continuous receive mode), the UiC1 register's TI bit is set to "0" (data present in 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, and the U2RRM bit is the U2C1 register bit 5.

(d) Serial Data Logic Switching Function

When the UiC1 register (i = 0 to 2)'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 1.15.12 shows serial data logic.

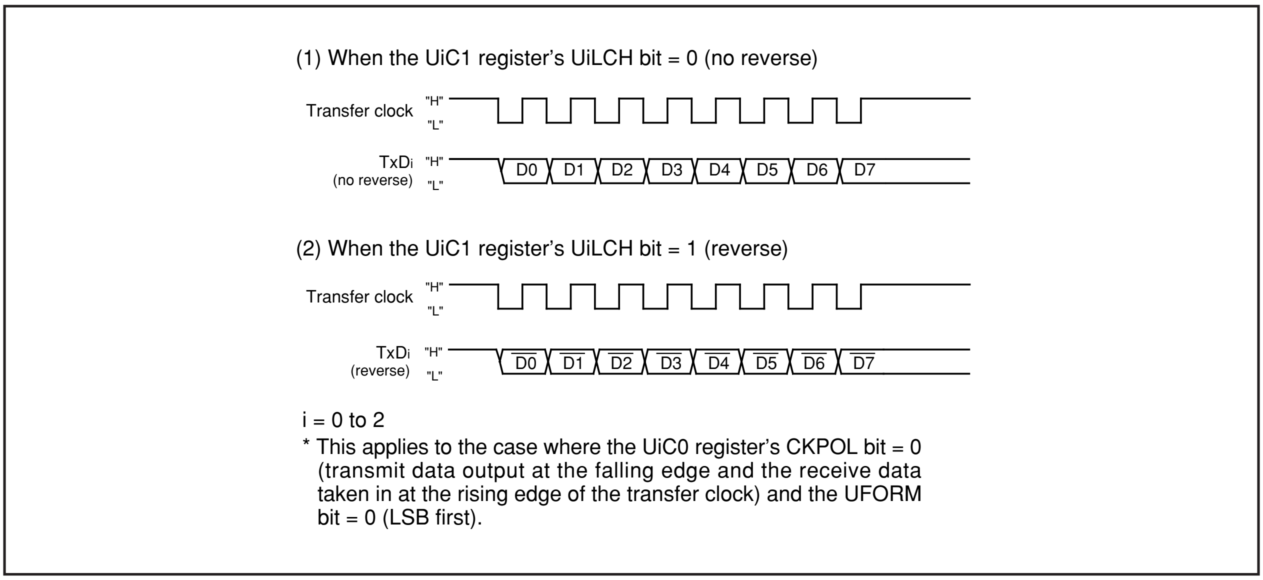


Figure 1.15.12 Serial Data Logic Switching

(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. Figure 1.15.13 shows the transfer clock output from the multiple pins function usage. This function can be used when the selected transfer clock for UART1 is an internal clock.

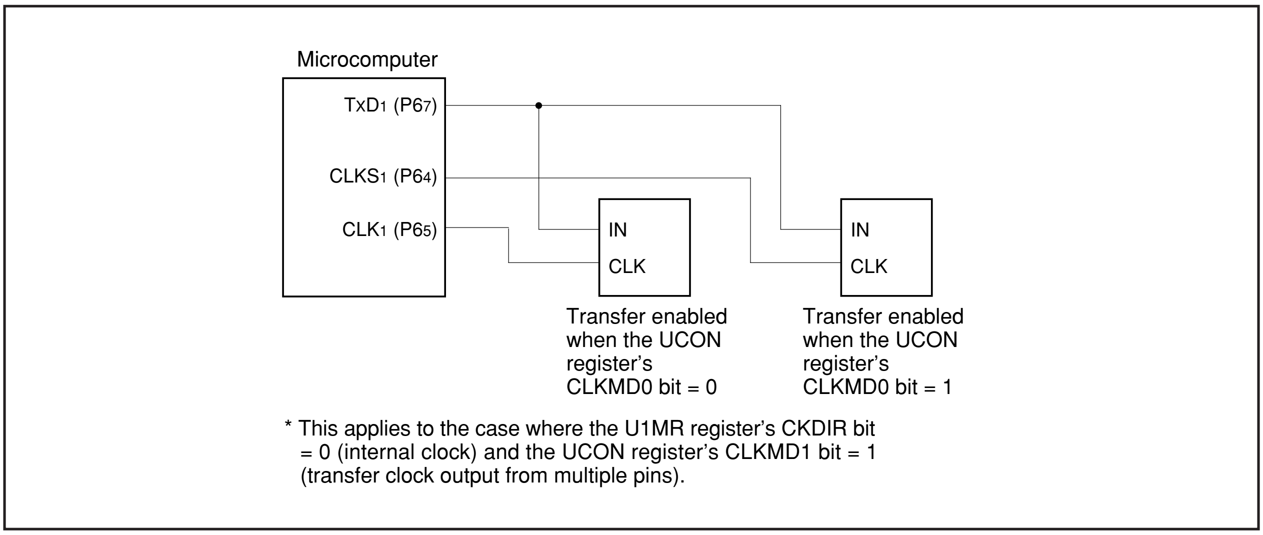


Figure 1.15.13 Transfer Clock Output From Multiple Pins

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

This function separates $\overline{CTS_0}$/$\overline{RTS_0}$, outputs $\overline{RTS_0}$ from the $P6_0$ pin, and accepts as input the $\overline{CTS_0}$ from the $P6_4$ pin. To use this function, set the register bits as shown below.

- U0C0 register's CRD bit = 0 (enables UART0 $\overline{CTS}$/$\overline{RTS}$)
- U0C0 register's CRS bit = 1 (outputs UART0 $\overline{RTS}$)
- U1C0 register's CRD bit = 0 (enables UART1 $\overline{CTS}$/$\overline{RTS}$)
- U1C0 register's CRS bit = 0 (inputs UART1 $\overline{CTS}$)
- UCON register's RCSP bit = 1 (inputs $\overline{CTS_0}$ from the $P6_4$ pin)
- UCON register's CLKMD1 bit = 0 ($CLKS_1$ not used)

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

Figure 1.15.14 shows $\overline{CTS}$/$\overline{RTS}$ separate function usage.

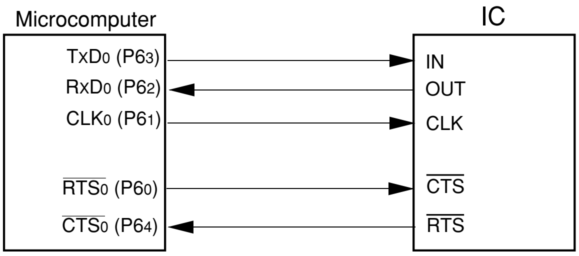


Figure 1.15.14 $\overline{CTS}$/$\overline{RTS}$ Separate Function

Clock Asynchronous Serial I/O (UART) Mode

The UART mode allows transmitting and receiving data after setting the desired transfer rate and transfer data format. Tables 1.15.5 lists the specifications of the UART mode. Table 1.15.6 lists the registers used in UART mode and the register values set.

Table 1.15.5 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 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) f_{EXT}: Input from CLKi pin. n :Setting value of UiBRG register 00_{16} to FF_{16}
Transmission, reception control	• Selectable from $\overline{CTS}$ function, $\overline{RTS}$ function or $\overline{CTS}$/$\overline{RTS}$ function disabled
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 $\overline{CTS}$ function is selected, input on the $\overline{CTS}$i pin = L
Reception start condition	• Before reception can start, the following requirements must be met – 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 1) = 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 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 to "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. • T_XD, R_XD I/O polarity switch This function reverses the polarities of the T_XD pin output and R_XD pin input. The logic levels of all I/O data is reversed. • Separate $\overline{CTS}$/$\overline{RTS}$ pins (UART0) $\overline{CTS_0}$ and $\overline{RTS_0}$ are input/output from separate pins

i = 0 to 2
Note 1: The U0IRS and U1IRS bits respectively are the UCON register bits 0 and 1; the U2IRS bit is the U2C1 register bit 4.
Note 2: If an overrun error occurs, the value of UiRB register will be indeterminate. The IR bit of SiRIC register does not change.

Table 1.15.6 Registers to Be Used and Settings in UART Mode

Register	Bit	Function
UiTB	0 to 8	Set transmission data (Note 1)
UiRB	0 to 8	Reception data can be read (Note 1)
	OER,FER,PER,SUM	Error flag
UiBRG	0 to 7	Set a transfer rate
UiMR	SMD2 to SMD0	Set these bits to “100_2” when transfer data is 7-bit long Set these bits to “101_2” when transfer data is 8-bit long Set these bits to “110_2” when transfer data is 9-bit 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
	CKPOL	Set to “0”
	UFORM	LSB first or MSB first can be selected when transfer data is 8-bit long. Set this bit to “0” when transfer data is 7- or 9-bit 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”

i = 0 to 2

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

Table 1.15.7 lists the functions of the input/output pins during UART mode. Table 1.15.8 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.)
Figure 1.15.15 shows the typical transmit timings in UART mode. Figure 1.15.16 shows the typical receive timing in UART mode.

Table 1.15.7 I/O Pin Functions

Pin name	Function	Method of selection
TxD_i ($P6_3$, $P6_7$, $P7_0$)	Serial data output	(Outputs dummy data when performing reception only)
RxD_i ($P6_2$, $P6_6$, $P7_1$)	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)
CLK_i ($P6_1$, $P6_5$, $P7_2$)	I/O 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
$\overline{CTS_i}$/$\overline{RTS_i}$ ($P6_0$, $P6_4$, $P7_3$)	$\overline{CTS}$ 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
	$\overline{RTS}$ output	UiC0 register’s CRD bit = 0 UiC0 register’s CRS bit = 1
	I/O port	UiC0 register’s CRD bit = 1

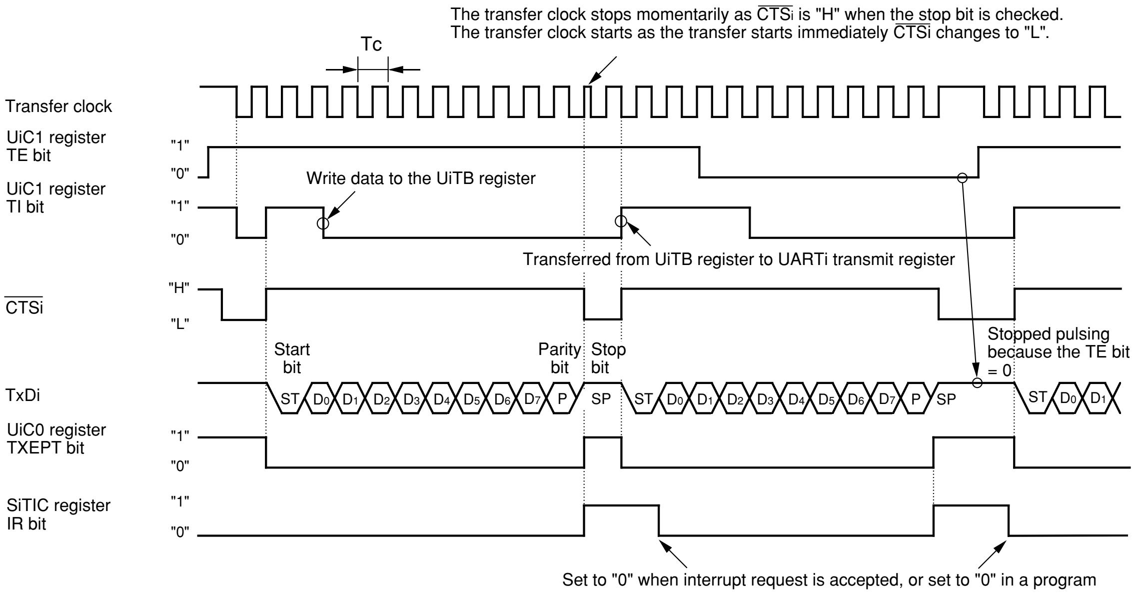
i = 0 to 2

Table 1.15.8 $P6_4$ 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
$\overline{CTS_1}$	0	0	0	0	0
$\overline{RTS_1}$	0	1	0	0	-
$\overline{CTS_0}$ (Note)	0	0	1	0	0

Note : In addition to this, set the U0C0 register’s CRD bit to “0” ($\overline{CTS_0}$/$\overline{RTS_0}$ enabled) and the U0C0 register’s CRS bit to “1” ($\overline{RTS_0}$ selected).

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

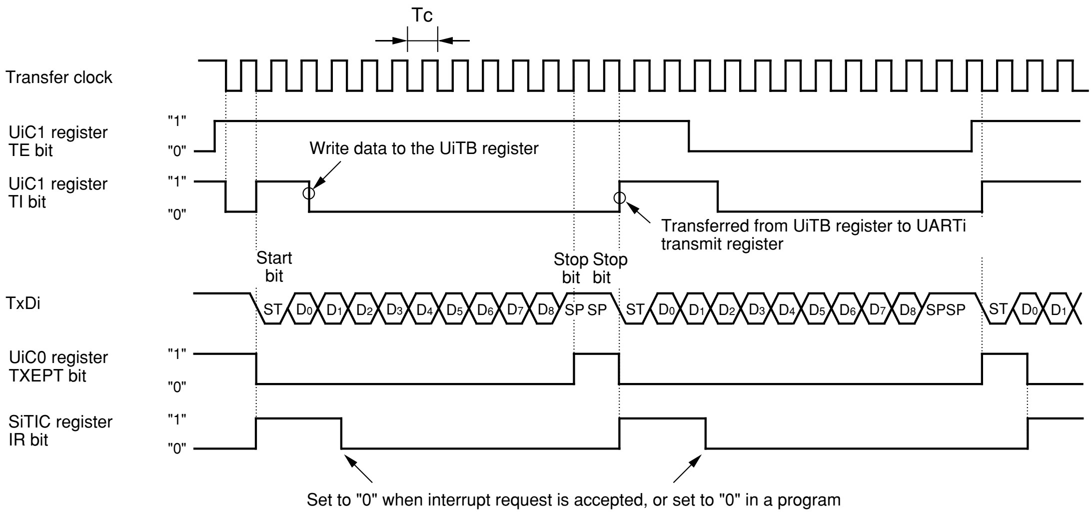


The above timing diagram applies to the case where the register bits are set as follows:

- 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)
- UiIRS 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) / fEXT
fj : frequency of UiBRG count source (f1SIO, f2SIO, f8SIO, f32SIO)
fEXT : 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-bit long (parity disabled, two stop bits)



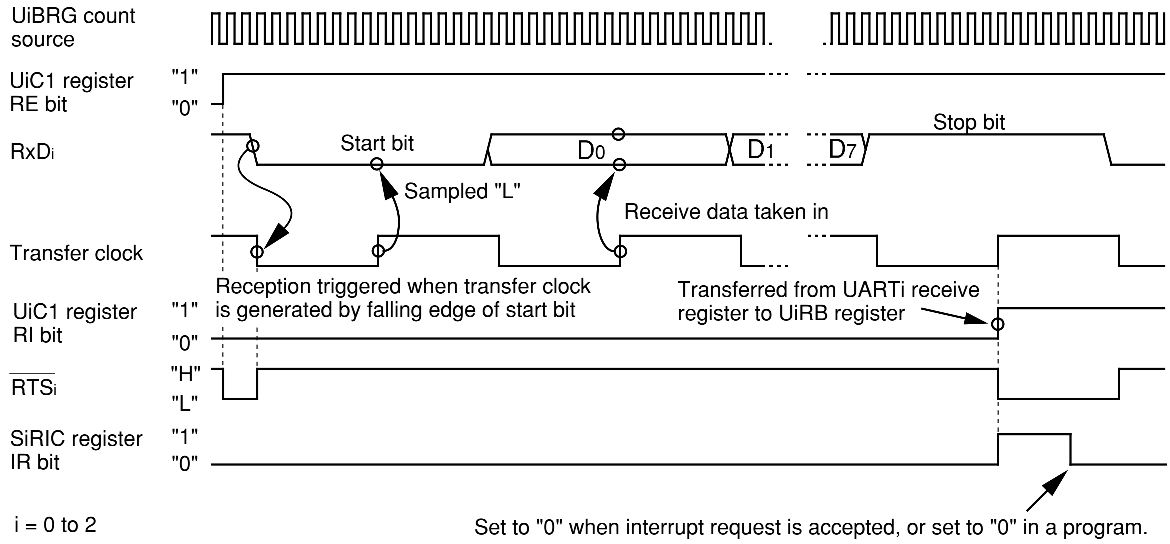
The above timing diagram applies to the case where the register bits are set as follows:

- UiMR register PRYE bit = 0 (parity disabled)
- UiMR register STPS bit = 1 (2 stop bits)
- UiC0 register CRD bit = 1 (CTS/RTS disabled)
- UiIRS 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) / fEXT
fj : frequency of UiBRG count source (f1SIO, f2SIO, f8SIO, f32SIO)
fEXT : frequency of UiBRG count source (external clock)
n : value set to UiBRG
i = 0 to 2

Figure 1.15.15 Transmit Operation

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

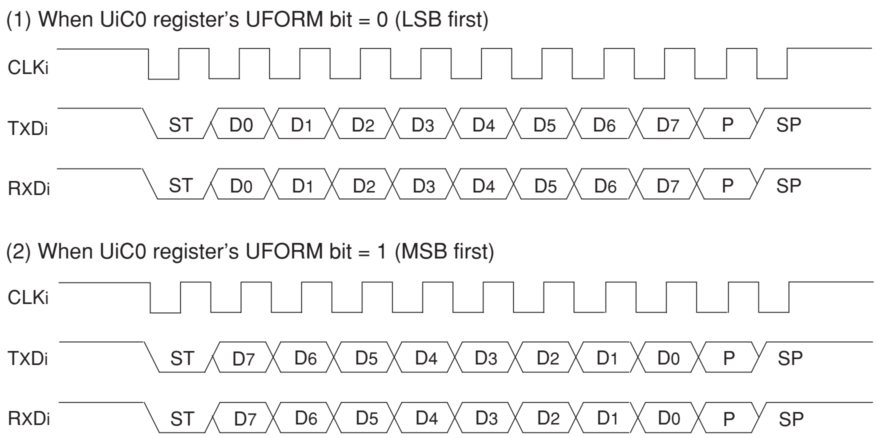


The above timing diagram applies to the case where the register bits are set as follows:
- UiMR register PRYE bit = 0 (parity disabled)
- UiMR register STPS bit = 0 (1 stop bit)
- UiC0 register CRD bit = 0 (CTSi/RTSi enabled), CRS bit = 1 (RTSi selected)

Figure 1.15.16 Receive Operation

(a) LSB First/MSB First Select Function

As shown in Figure 1.15.17, use the UiC0 register's UFORM bit to select the transfer format. This function is valid when transfer data is 8-bit long.



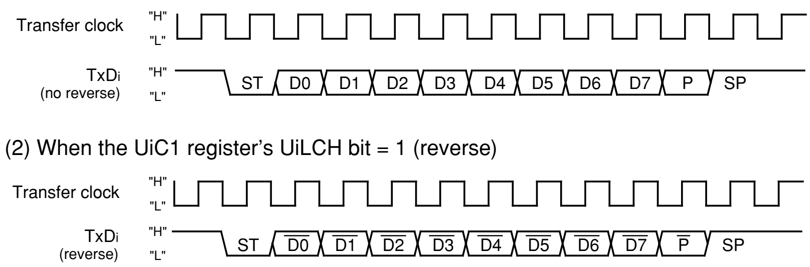
i = 0 to 2
ST: Start bit
P: Parity bit
SP: Stop bit

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).

Figure 1.15.17 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 1.15.18 shows serial data logic.



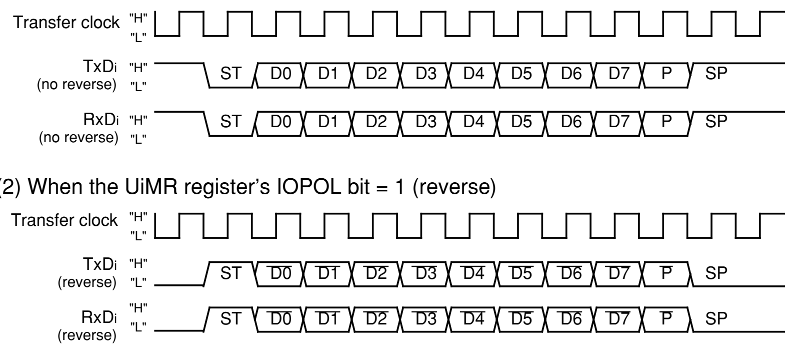
i = 0 to 2
ST: Start bit
P: Parity bit
SP: Stop bit

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).

Figure 1.15.18 Serial Data Logic Switching

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

This function inverses the polarities of the TxD_i pin output and RxD_i pin input. The logic levels of all input/output data (including the start, stop and parity bits) are inversed. Figure 1.15.19 shows the TxD and RxD input/output polarity inverse.



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

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).

Figure 1.15.19 TxD and RxD I/O Polarity Inverse

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

This function separates $\overline{CTS_0}$/$\overline{RTS_0}$, outputs $\overline{RTS_0}$ from the $P6_0$ pin, and accepts as input the $\overline{CTS_0}$ from the $P6_4$ pin. To use this function, set the register bits as shown below.

- U0C0 register's CRD bit = 0 (enables UART0 $\overline{CTS}$/$\overline{RTS}$)
- U0C0 register's CRS bit = 1 (outputs UART0 $\overline{RTS}$)
- U1C0 register's CRD bit = 0 (enables UART1 $\overline{CTS}$/$\overline{RTS}$)
- U1C0 register's CRS bit = 0 (inputs UART1 $\overline{CTS}$)
- UCON register's RCSP bit = 1 (inputs $\overline{CTS_0}$ from the $P6_4$ pin)
- UCON register's CLKMD1 bit = 0 ($CLKS_1$ not used)

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

Figure 1.15.20 shows $\overline{CTS}$/$\overline{RTS}$ separate function usage.

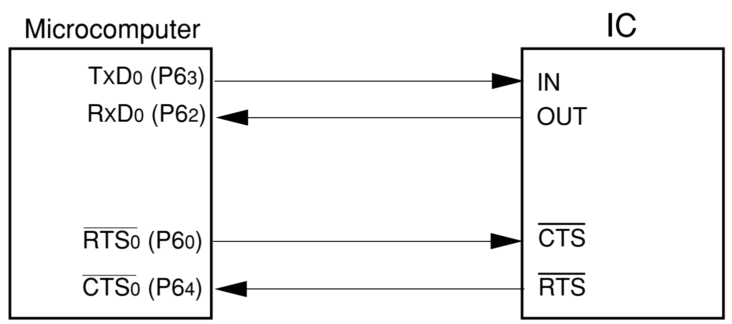


Figure 1.15.20 $\overline{CTS}$/$\overline{RTS}$ Separate Function

Special Mode 1 (I^2C Mode)

I^2C mode is provided for use as a simplified I^2C interface compatible mode. Table 1.15.9 lists the specifications of the I^2C mode. Figure 1.15.21 shows the block diagram for I^2C mode. Table 1.15.10 lists the registers used in the I^2C mode and the register values set. Table 1.15.11 lists the features in I^2C mode. Figure 1.15.22 shows SCLi timing.

As shown in Table 1.15.11, the microcomputer is placed in I^2C 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.

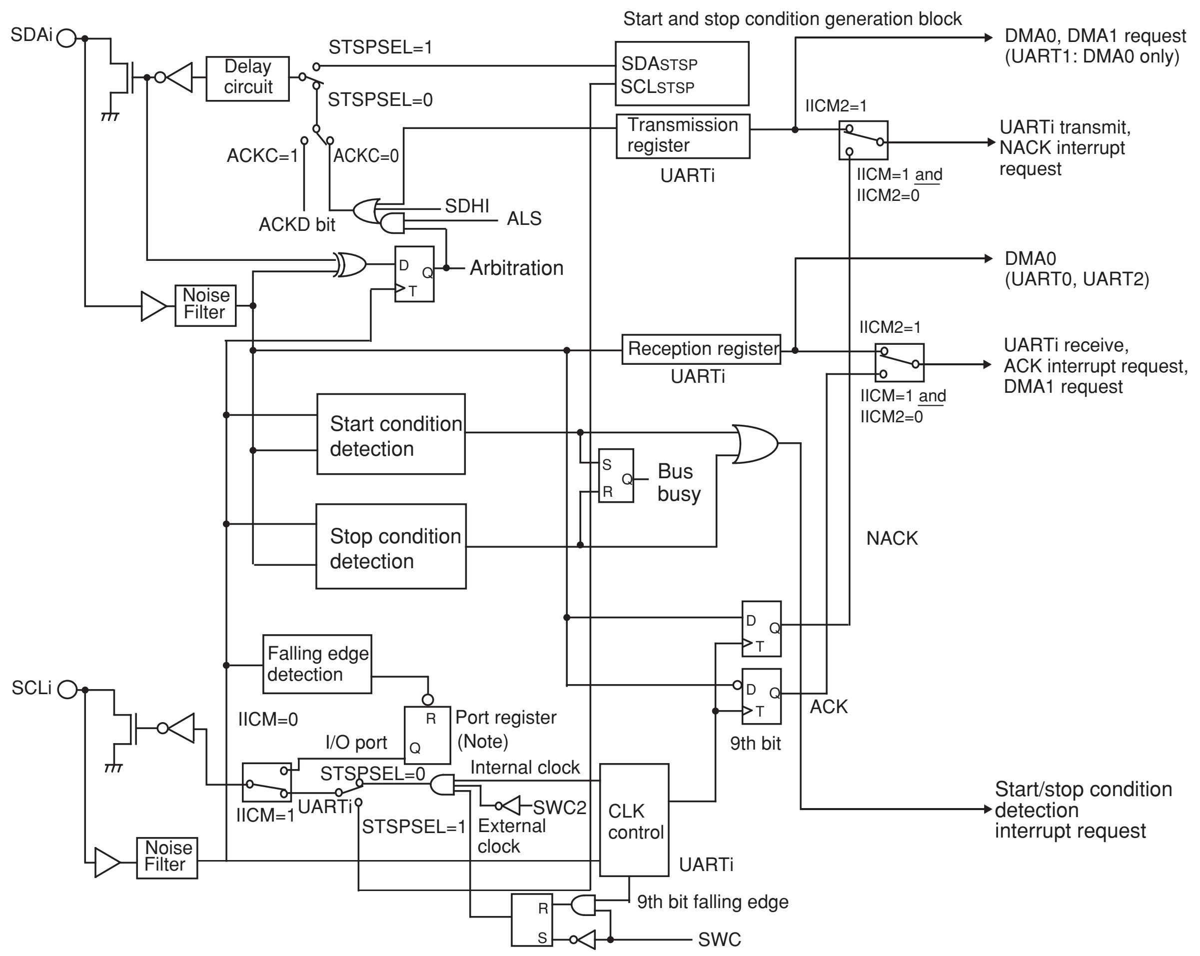
Table 1.15.9 I^2C Mode Specifications

Item	Specification
Transfer data format	• Transfer data length: 8 bits
Transfer clock	• During master UiMR 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 SCL_i 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

i = 0 to 2

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.

i = 0 to 2

IICM: UiSMR register's bit
IICM2, SWC, ALS, SWC2, SDHI: UiSMR2 register's bits
STSPSEL, ACKD, ACKC: UiSMR4 register's bits

Note: If the IICM bit =1, the pins can be read even when the PD6_2, PD6_6 or PD7_1 bit = 1 (output mode).

Figure 1.15.21 I²C Mode Block Diagram

Table 1.15.10 Registers to Be Used and Settings in I2C Mode

Register	Bit	Function	
		Master	Slave
UiTB (Note 1)	0 to 7	Set transmission data	
UiRB (Note 1)	0 to 7	Reception data can be read	
	8	ACK or NACK is set in this bit	
	ABT	Arbitration lost detection flag	Invalid
	OER	Overrun error flag	
UiBRG	0 to 7	Set a transfer rate	Invalid
UiMR (Note 1)	SMD2 to SMD0	Set to “010_2”	
	CKDIR	Set to “0”	Set to “1”
	IOPOL	Set to “0”	
UiC0	CLK1, CLK0	Select the count source for the UiBRG register	Invalid
	CRS	Invalid because CRD = 1	
	TXEPT	Transmit register empty flag	
	CRD	Set to “1”	
	NCH	Set to “1”	
	CKPOL	Set to “0”	
	UFORM	Set to “1”	
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)	Invalid	
	U2RRM (Note 2), UiLCH, UiERE	Set to “0”	
UiSMR	IICM	Set to “1”	
	ABC	Select the timing at which arbitration-lost is detected	Invalid
	BBS	Bus busy flag	
	3 to 7	Set to “0”	
UiSMR2	IICM2	Refer to “Table 1.15.11 I2C Mode Functions”	
	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	
	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	
	SDHI	Set this bit to “1” to disable SDAi output	
	7	Set to “0”	
UiSMR3	0, 2, 4 and NODC	Set to “0”	
	CKPH	Refer to Table 1.15.11 I2C Mode Functions”	
	DL2 to DL0	Set the amount of SDAi digital delay	
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	
	ACKC	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
IFSR0	IFSR06, ISFR07	Set to “1”	
UCON	U0IRS, U1IRS	Invalid	
	2 to 7	Set to “0”	

i = 0 to 2

Note 1: Not all register bits are described above. Set those bits to “0” when writing to the registers in I2C mode.

Note 2: 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.

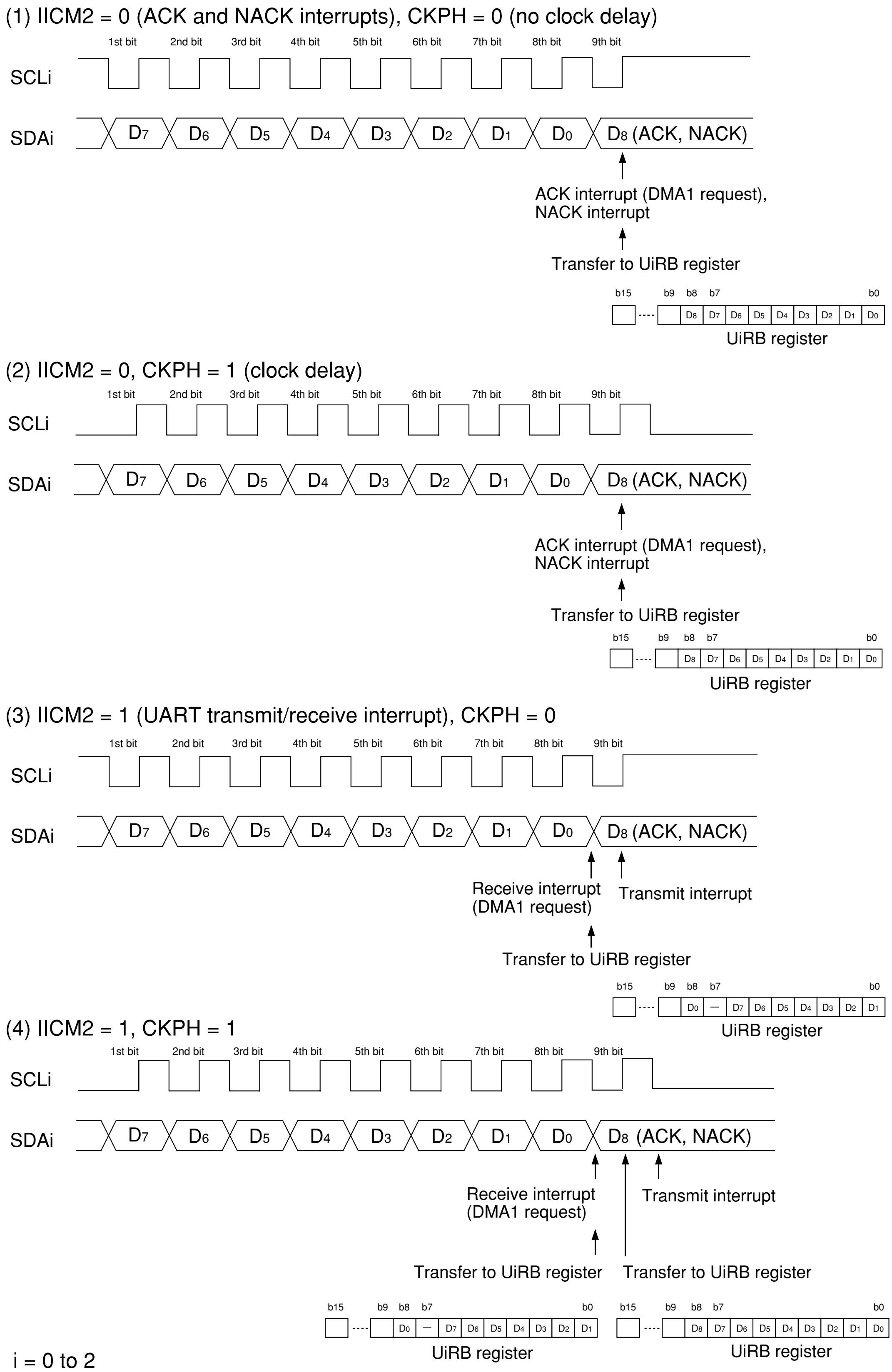
Table 1.15.11 I²C Mode Functions

Function	Clock synchronous serial I/O mode (SMD2 to SMD0 = 001_2, IICM = 0)	I²C mode (SMD2 to SMD0 = 010_2, IICM = 1) IICM2 = 0 (NACK/ACK interrupt) CKPH = 0 (No clock delay)	IICM2 = 0 (NACK/ACK interrupt) CKPH = 1 (Clock delay)	IICM2 = 1 (UART transmit/UART receive interrupt) CKPH = 0 (No clock delay)	IICM2 = 1 (UART transmit/UART receive interrupt) CKPH = 1 (Clock delay)
Factor of interrupt number 6, 7 and 10 (Notes 1, 5, 7)	-	Start condition detection or stop condition detection (Refer to "Table 1.15.12 STSPSEL Bit Functions")			
Factor of interrupt number 15, 17 and 19 (Notes 1, 6)	UARTi transmission Transmission started or completed (selected by UiIRS)	No acknowledgment detection (NACK) Rising edge of SCLi 9th bit		UARTi transmission Rising edge of SCLi 9th bit	UARTi transmission Falling edge of SCLi next to the 9th bit
Factor of interrupt number 16, 18 and 20 (Notes 1, 6)	UARTi reception When 8th bit received CKPOL = 0 (rising edge) CKPOL = 1 (falling edge)	Acknowledgment detection (ACK) Rising edge of SCLi 9th bit		UARTi reception Falling edge of SCLi 9th bit	
Timing for transferring data from the UART reception shift register to the UiRB register	CKPOL = 0 (rising edge) CKPOL = 1 (falling edge)	Rising edge of SCLi 9th bit		Falling edge of SCLi 9th bit	Falling and rising edges of SCLi 9th bit
UARTi transmission output delay	Not delayed	Delayed			
Functions of P6$_3$, P6$_7$ and P7$_0$ pins	TxD$_i$ output	SDAi input/output			
Functions of P6$_2$, P6$_6$ and P7$_1$ pins	RxD$_i$ input	SCLi input/output			
Functions of P6$_1$, P6$_5$ and P7$_2$ pins	CLK$_i$ input or output selected	- (Cannot be used in I²C mode)			
Noise filter width	15 ns	200 ns			
Read RxD$_i$ and SCLi pins levels	Possible when the corresponding port direction bit = 0	Always possible no matter how the corresponding port direction bit is set			
Initial value of TxD$_i$ and SDAi outputs	CKPOL = 0 (H) CKPOL = 1 (L)	The value set in the port register before setting I²C mode (Note 2)			
Initial and end value of SCLi	-	H	L	H	L
DMA1 factor (Note 6)	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 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 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: If the source or cause of any interrupt is changed, the IR bit in the interrupt control register for the changed interrupt may inadvertently be set to "1" (interrupt requested). (Refer to "Precautions for Interrupts" of the Usage Notes Reference Book.) If one of the bits shown below is changed, the interrupt source, the interrupt timing, etc. change. Therefore, always be sure to set the IR bit to "0" (interrupt not requested) after changing those bits.
- SMD2 to SMD0 bits in the UiMR register
- IICM bit in the UiSMR register
- IICM2 bit in the UiSMR2 register
- CKPH bit in the UiSMR3 register

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)
Note 5: Refer to "Figure 1.15.24 STSPSEL Bit Functions".
Note 6: Refer to "Figure 1.15.22 Transfer to UiRB Register and Interrupt Timing".
Note 7: When using UART0, be sure to set the IFSR06 bit in the IFSR0 register to "1" (cause of interrupt: UART0 bus collision). When using UART1, be sure to set the IFSR07 bit in the IFSR0 register to "1" (cause of interrupt: UART1 bus collision).



i = 0 to 2

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

Figure 1.15.22 Transfer to UiRB Register and Interrupt Timing

• Detection of Start and Stop Condition

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

A start condition-detected interrupt request is generated when the 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.

Figure 1.15.23 shows the detection of start and stop condition.

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

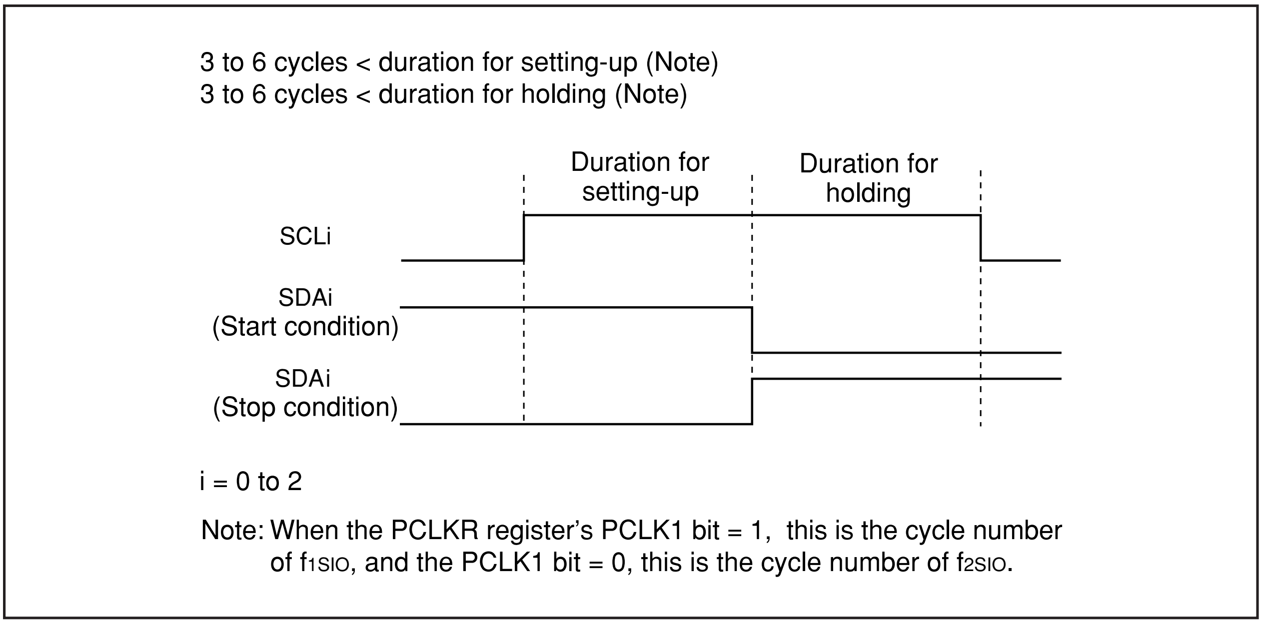


Figure 1.15.23 Detection of Start and Stop Condition

• 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).

The output procedure is described below.

(1) Set the STAREQ bit, RSTAREQ bit or STPREQ bit to "1" (start).

(2) Set the STSPSEL bit in the UiSMR4 register to "1" (output).

Table 1.15.12 and Figure 1.15.24 show the functions of the STSPSEL bit.

Table 1.15.12 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
Start/stop condition interrupt request generation timing	Start/stop condition detection	Finish generating start/stop condition

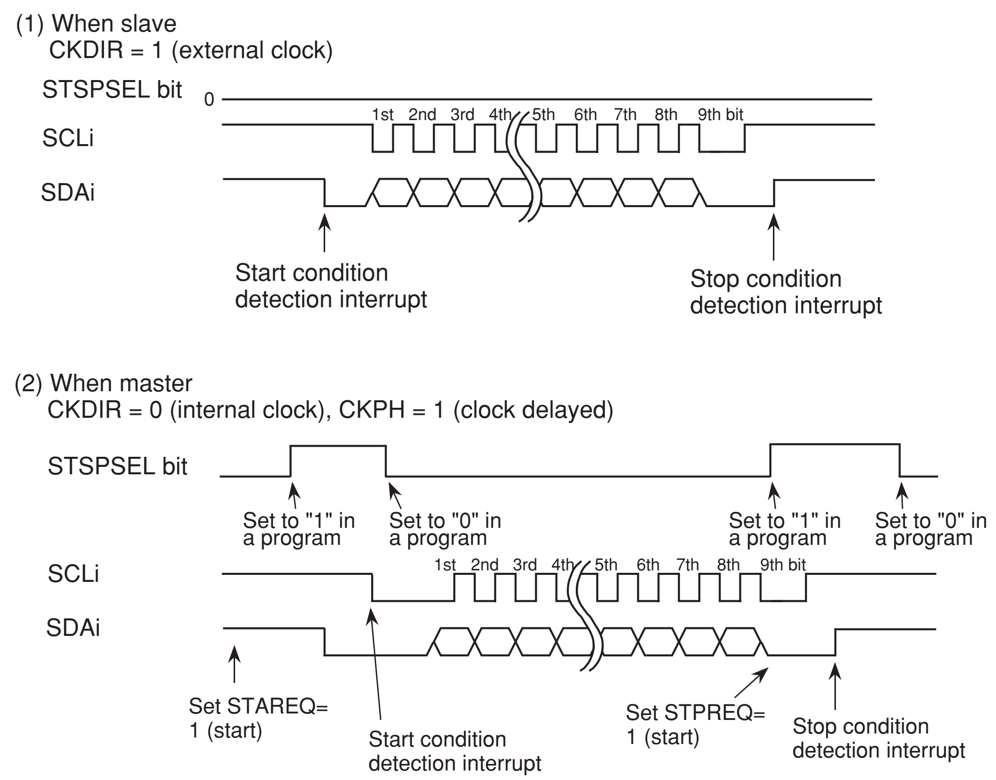


Figure 1.15.24 STSPSEL Bit Functions

- **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 set 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, set 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 1.15.24.

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. Setting 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 set to “1” (ACK data output), the value of the ACKD bit in the UiSMR4 register is output from the SDAi pin.

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

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

• **Initialization of Transmission/Reception**

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

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

Note that when UARTi transmission/reception is started using this function, the TI bit 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. Transfer clock polarity and phase are selectable. Table 1.15.13 lists the specifications of Special Mode 2. Figure 1.15.25 shows communication control example for Special Mode 2. Table 1.15.14 lists the registers used in Special Mode 2 and the register values set.

Table 1.15.13 Special Mode 2 Specifications

Item	Specification
Transfer data format	• Transfer data length: 8 bits
Transfer clock	• Master mode UiMR 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 (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 3) 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

i = 0 to 2

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

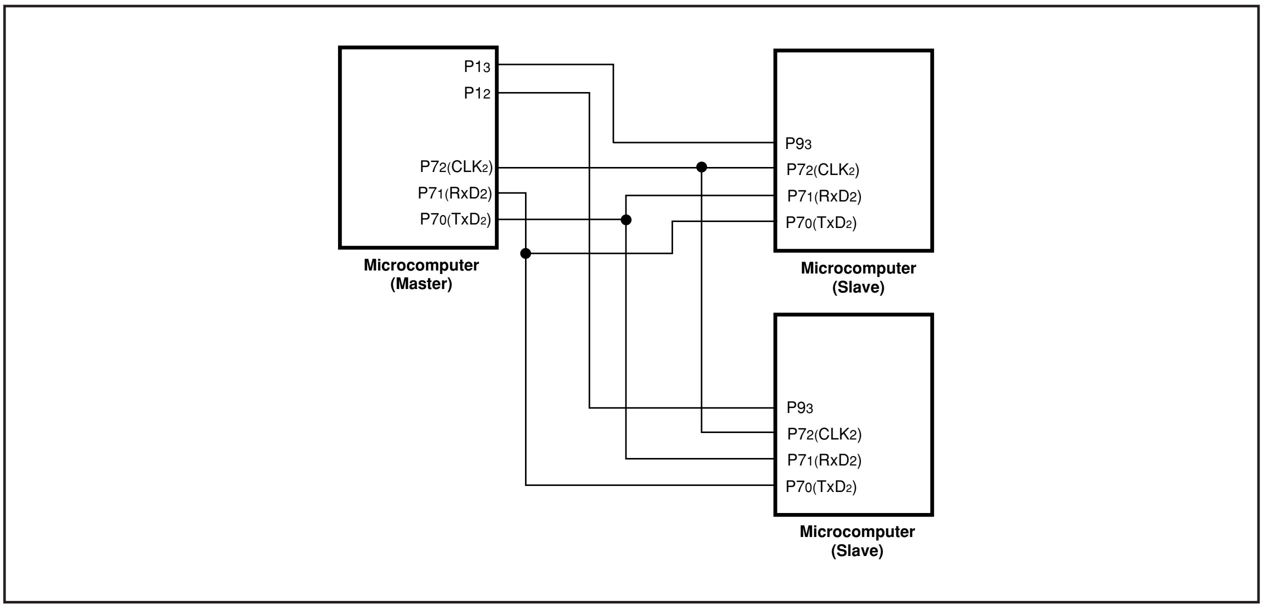


Figure 1.15.25 Serial Bus Communication Control Example (UART2)

Table 1.15.14 Registers to Be Used and Settings in Special Mode 2

Register	Bit	Function
UiTB (Note 1)	0 to 7	Set transmission data
UiRB (Note 1)	0 to 7	Reception data can be read
	OER	Overrun error flag
UiBRG	0 to 7	Set a transfer rate
UiMR (Note 1)	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 2)	Select UART2 transmit interrupt cause
	U2RRM (Note 2), 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”

i = 0 to 2

Note 1: Not all register bits are described above. Set those bits to “0” when writing to the registers in Special Mode 2.

Note 2: 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.

• **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 1.15.26 shows the transmission and reception timing in master (internal clock).

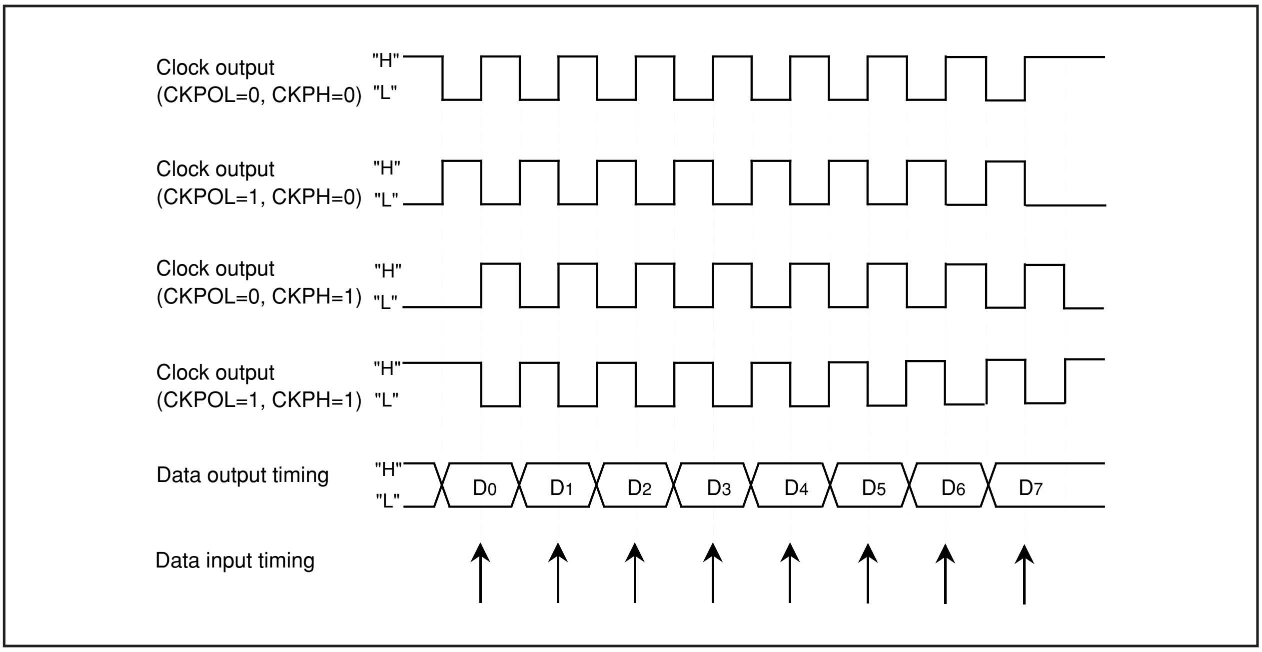


Figure 1.15.26 Transmission and Reception Timing in Master Mode (Internal Clock)

(b) Slave (External Clock)

Figure 1.15.27 shows the transmission and reception timing (CKPH = 0) in slave (external clock).
Figure 1.15.28 shows the transmission and reception timing (CKPH = 1) in slave (external clock).

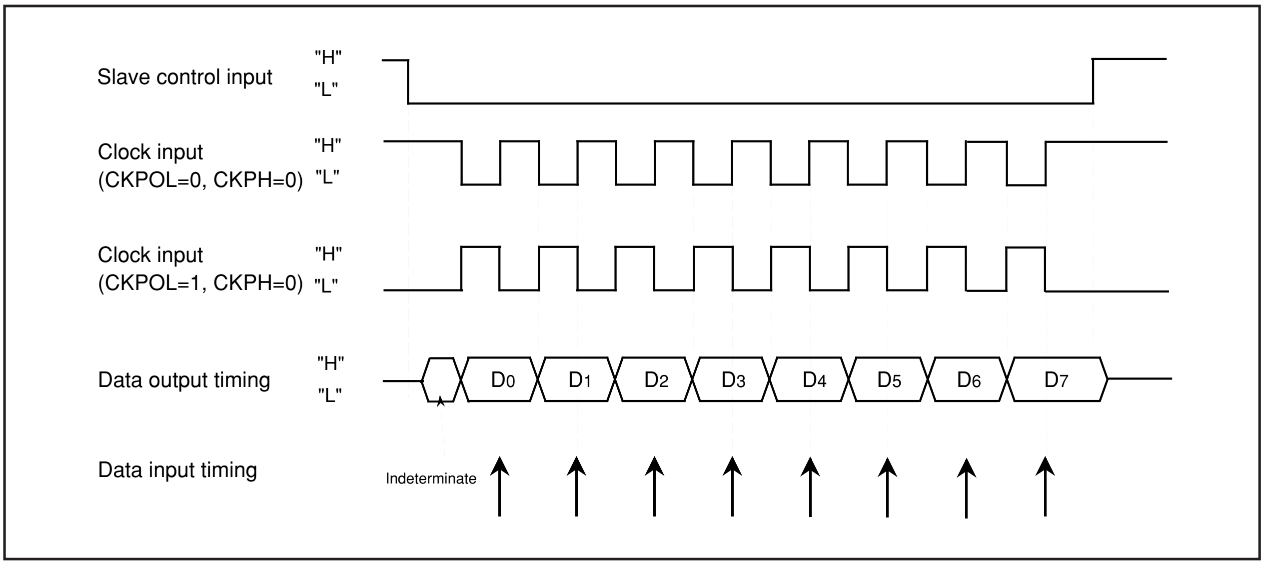


Figure 1.15.27 Transmission and Reception Timing (CKPH = 0) in Slave Mode (External Clock)

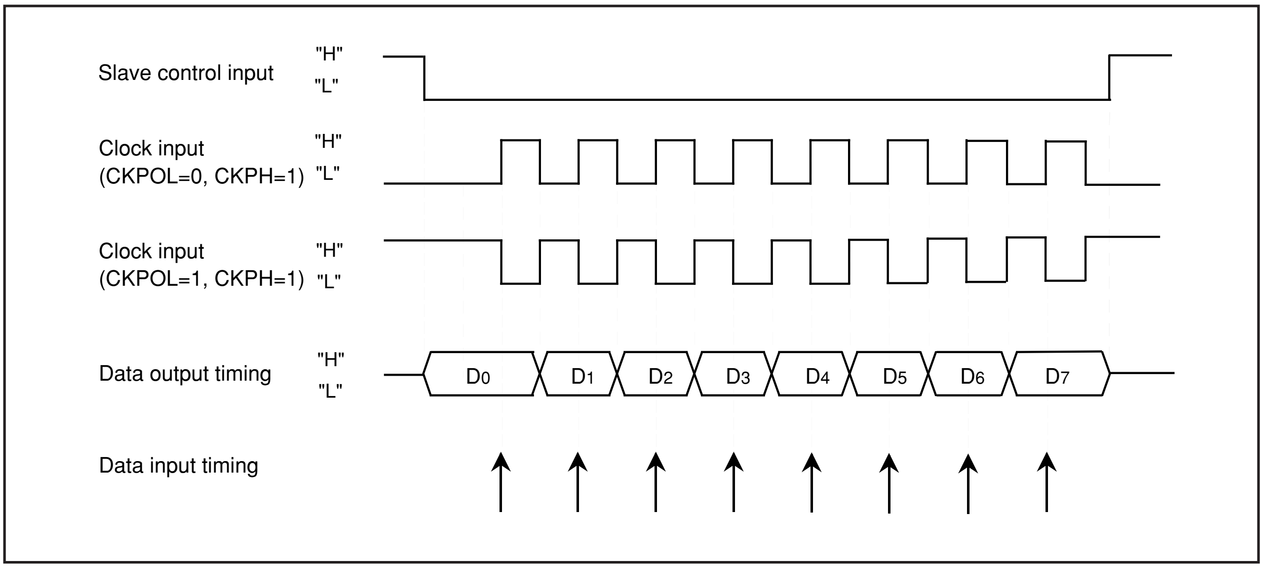


Figure 1.15.28 Transmission and Reception Timing (CKPH = 1) in Slave Mode (External Clock)

Special Mode 3 (IE Mode)

In this mode, one bit of IEBus is approximated with one byte of UART mode waveform.
Table 1.15.15 lists the registers used in IE mode and the register values set. Figure 1.15.29 shows the functions of bus collision detect function related bits.
If the TxDi pin (i = 0 to 2) output level and RxDi pin input level do not match, a UARTi bus collision detect interrupt request is generated.
Use the IFSR0 register's IFSR06 and IFSR07 bits to enable the UART0/UART1 bus collision detect function.

Table 1. 15.15 Registers to Be Used and Settings in IE Mode

Register	Bit	Function
UiTB	0 to 8	Set transmission data
UiRB	0 to 8	Reception data can be read
(Note 1)	OER,FER,PER,SUM	Error flag
UiBRG	0 to 7	Set a transfer rate
UiMR	SMD2 to SMD0	Set to "110$_2$"
	CKDIR	Select the internal clock or external clock
	STPS	Set to "0"
	PRY	Invalid because PRYE = 0
	PRYE	Set to "0"
	IOPOL	Select the TxD/RxD input/output polarity
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 mode
	CKPOL	Set to "0"
	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 2)	Select the source of UART2 transmit interrupt
	UiRRM (Note 2), UiLCH, UiERE	Set to "0"
UiSMR	0 to 3, 7	Set to "0"
	ABSCS	Select the sampling timing at which to detect a bus collision
	ACSE	Set this bit to "1" to use the auto clear function of transmit enable bit
	SSS	Select the transmit start condition
UiSMR2	0 to 7	Set to "0"
UiSMR3	0 to 7	Set to "0"
UiSMR4	0 to 7	Set to "0"
IFSR0	IFSR06, IFSR07	Set to "1"
UCON	U0IRS, U1IRS	Select the source of UART0/UART1 transmit interrupt
	U0RRM, U1RRM	Set to "0"
	CLKMD0	Invalid because CLKMD1 = 0
	CLKMD1, RCSP, 7	Set to "0"

i= 0 to 2
Note 1: Not all register bits are described above. Set those bits to "0" when writing to the registers in IE mode.
Note 2: Set the U0C1 and U1C1 registers bit 4 and bit 5 to "0". The U0IRS, U1IRS, U0RRM and U1RRM bits are in the UCON register.

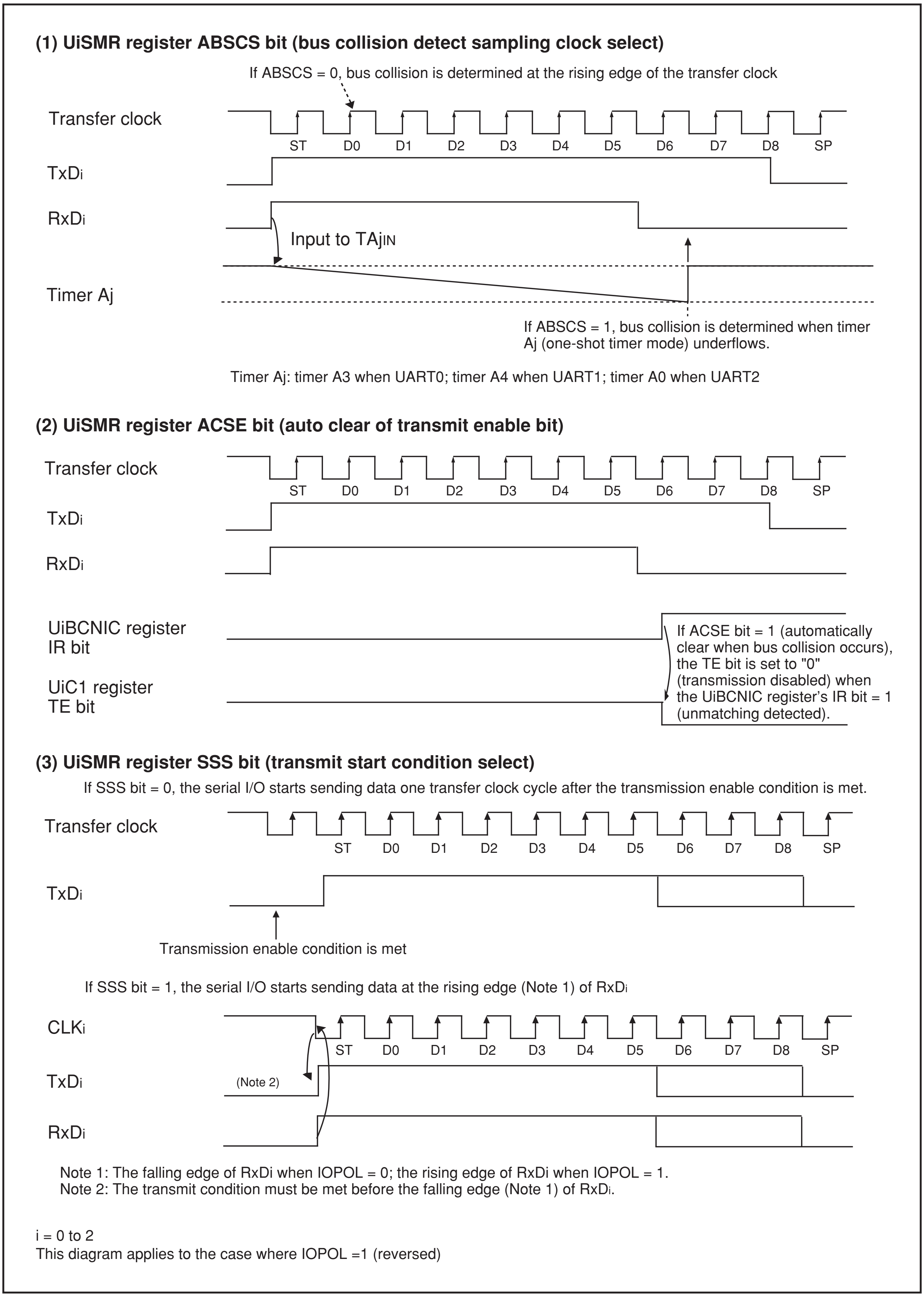


Figure 1.15.29 Bus Collision Detect Function-Related Bits

Special Mode 4 (SIM Mode) (UART2)

Based on UART mode, this is an SIM interface compatible mode. Direct and inverse formats can be implemented, and this mode allows to output a low from the TxD_2 pin when a parity error is detected. Tables 1.15.16 lists the specifications of SIM mode. Table 1.15.17 lists the registers used in the SIM mode and the register values set. Figure 1.15.30 shows the typical transmit/receive timing in SIM mode.

Table 1.15.16 SIM Mode Specifications

Item	Specification
Transfer data format	• Direct format • Inverse format
Transfer clock	• U2MR register's CKDIR bit = 0 (internal clock) : fi/ 16(n+1) fi = f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO}. n: Setting value of U2BRG register 00_{16} to FF_{16} • CKDIR bit = 1 (external clock) : f_{EXT}/16(n+1) f_{EXT}: Input from CLK_2 pin. n: Setting value of U2BRG register 00_{16} to FF_{16}
Transmission start condition	• Before transmission can start, the following requirements must be met – The TE bit of U2C1 register = 1 (transmission enabled) – The TI bit of U2C1 register = 0 (data present in U2TB register)
Reception start condition	• Before reception can start, the following requirements must be met – The RE bit of U2C1 register = 1 (reception enabled) – Start bit detection
Interrupt request generation timing (Note 2)	• For transmission When the serial I/O finished sending data from the U2TB transfer register (U2IRS bit = 1) • For reception When transferring data from the UART2 receive register to the U2RB 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 U2RB 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 During reception, if a parity error is detected, parity error signal is output from the TxD_2 pin. During transmission, a parity error is detected by the level of input to the RxD_2 pin when a transmission interrupt occurs • Error sum flag This flag is set to “1” when any of the overrun, framing, and parity errors is encountered

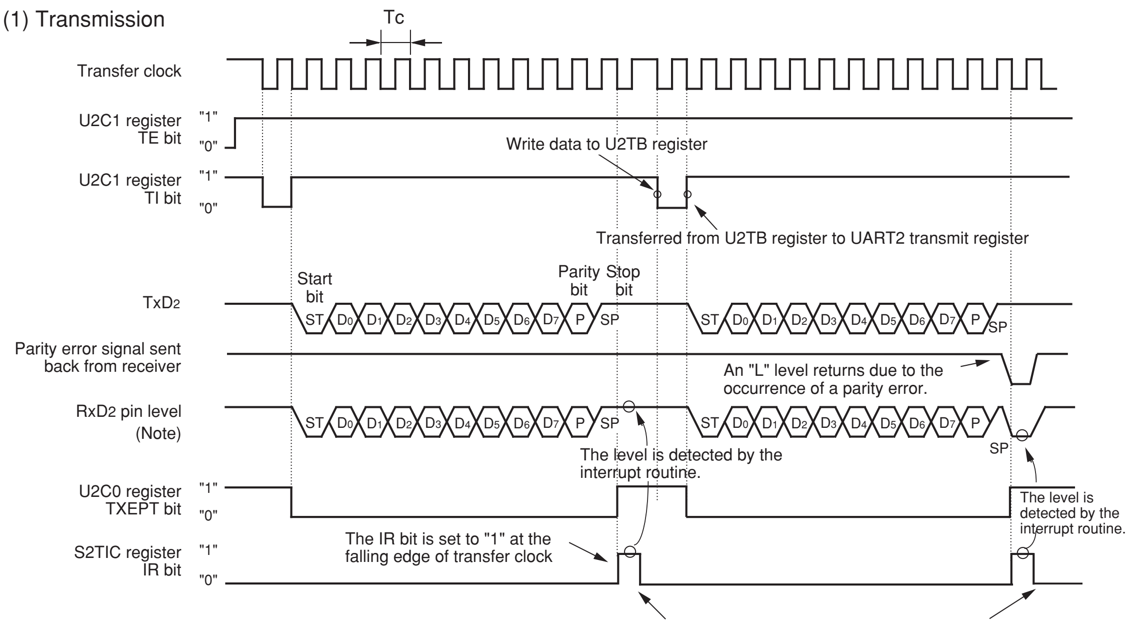
Note 1: If an overrun error occurs, the value of U2RB register will be indeterminate. The IR bit of S2RIC register does not change.

Note 2: A transmit interrupt request is generated by setting the U2IRS bit in the U2C1 register to “1” (transmit is completed) and U2ERE bit to “1” (error signal output) after reset. Therefore, when using SIM mode, be sure to set the IR bit to “0” (interrupt not requested) after setting these bits.

Table 1.15.17 Registers to Be Used and Settings in SIM Mode

Register	Bit	Function
U2TB (Note)	0 to 7	Set transmission data
U2RB (Note)	0 to 7	Reception data can be read
	OER,FER,PER,SUM	Error flag
U2BRG	0 to 7	Set a transfer rate
U2MR	SMD2 to SMD0	Set to “101_2”
	CKDIR	Select the internal clock or external clock
	STPS	Set to “0”
	PRY	Set this bit to “1” for direct format or “0” for inverse format
	PRYE	Set to “1”
	IOPOL	Set to “0”
U2C0	CLK1, CLK0	Select the count source for the U2BRG register
	CRS	Invalid because CRD = 1
	TXEPT	Transmit register empty flag
	CRD	Set to “1”
	NCH	Set to “0”
	CKPOL	Set to “0”
	UFORM	Set this bit to “0” for direct format or “1” for inverse format
U2C1	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	Set to “1”
	U2RRM	Set to “0”
	U2LCH	Set this bit to “0” for direct format or “1” for inverse format
	U2ERE	Set to “1”
U2SMR (Note)	0 to 3	Set to “0”
U2SMR2	0 to 7	Set to “0”
U2SMR3	0 to 7	Set to “0”
U2SMR4	0 to 7	Set to “0”

Note: Not all register bits are described above. Set those bits to “0” when writing to the registers in SIM mode.

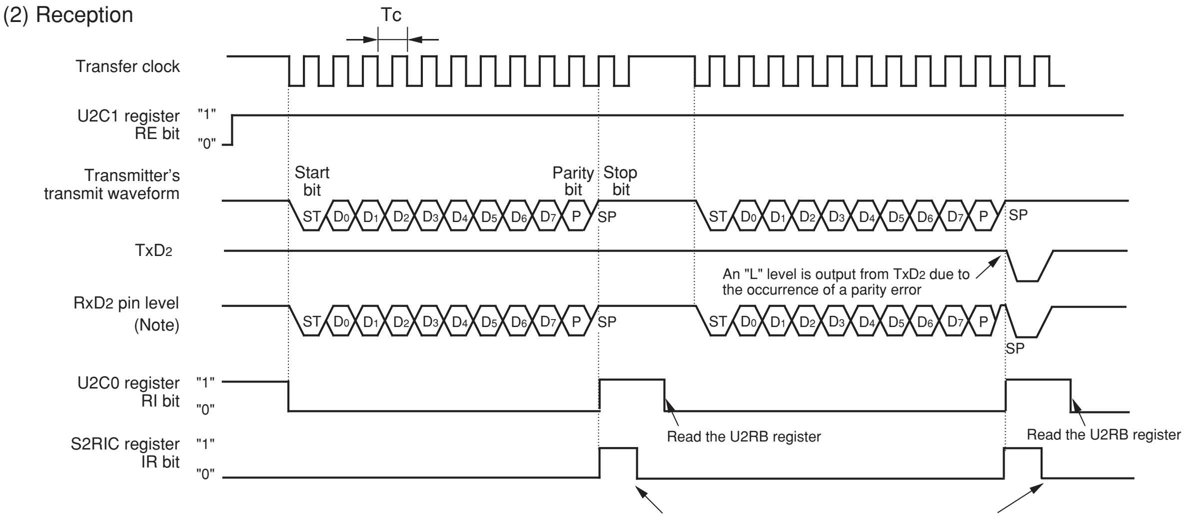


The above timing diagram applies to the case where data is transferred in the direct format.
- U2MR register STPS bit = 0 (1 stop bit)
- U2MR register PRY bit = 1 (even parity)
- U2C0 register UFORM bit = 0 (LSB first)
- U2C1 register U2LCH bit = 0 (no reverse)
- U2C1 register U2IRS bit = 1 (transmit is completed)

Set to "0" when interrupt request is accepted, or set to "0" in a program

Tc = 16 (n + 1) / fi or 16 (n + 1) / f_{EXT}
fi : frequency of U2BRG count source (f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO})
f_{EXT} : frequency of U2BRG count source (external clock)
n : value set to U2BRG

Note: Because TxD_2 and RxD_2 are connected, this is composite waveform consisting of the TxD_2 output and the parity error signal sent back from receiver.



The above timing diagram applies to the case where data is received in the direct format.
- U2MR register STPS bit = 0 (1 stop bit)
- U2MR register PRY bit = 1 (even parity)
- U2C0 register UFORM bit = 0 (LSB first)
- U2C1 register U2LCH bit = 0 (no reverse)
- U2C1 register U2IRS bit = 1 (transmit is completed)

Set to "0" when interrupt request is accepted, or set to "0" in a program

Tc = 16 (n + 1) / fi or 16 (n + 1) / f_{EXT}
fi : frequency of U2BRG count source (f_{1SIO}, f_{2SIO}, f_{8SIO}, f_{32SIO})
f_{EXT} : frequency of U2BRG count source (external clock)
n : value set to U2BRG

Note: Because TxD_2 and RxD_2 are connected, this is composite waveform consisting of the transmitter's transmit waveform and the parity error signal received.

Figure 1.15.30 Transmit and Receive Timing in SIM Mode

Figure 1.15.31 shows the example of connecting the SIM interface. Connect T_XD_2 and R_XD_2 and apply pull-up.

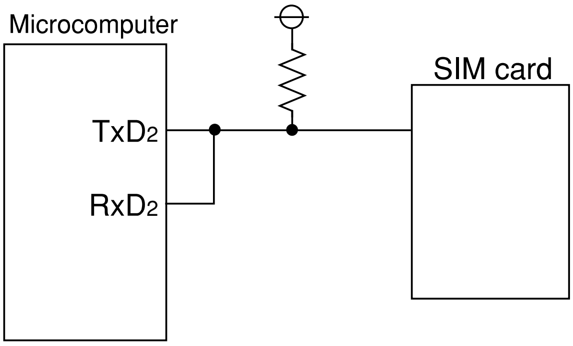


Figure 1.15.31 SIM Interface Connection

(a) Parity Error Signal Output

The parity error signal is enabled by setting the U2C1 register's U2ERE bit to "1".

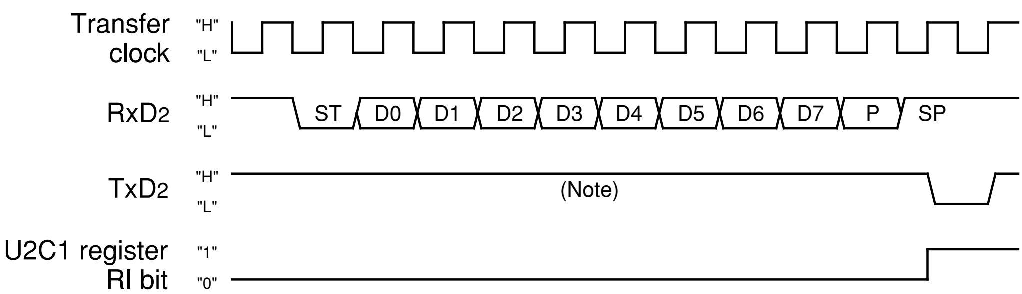
• When receiving

The parity error signal is output when a parity error is detected while receiving data. This is achieved by pulling the TxD_2 output low with the timing shown in Figure 1.15.32. If the R2RB register is read while outputting a parity error signal, the PER bit is set to "0" and at the same time the TxD_2 output is returned high.

• When transmitting

A transmission-finished interrupt request is generated at the falling edge of the transfer clock pulse that immediately follows the stop bit. Therefore, whether a parity signal has been returned can be determined by reading the port that shares the RxD_2 pin in a transmission-finished interrupt service routine.

Figure 1.15.32 shows the output timing of the parity error signal



This timing diagram applies to the case where the direct format is implemented.

Note: The output of microcomputer is in the high-impedance state (pulled up externally).

ST: Start bit
P: Even Parity
SP: Stop bit

Figure 1.15.32 Parity Error Signal Output Timing

(b) Format

• Direct Format

Set the U2MR register's PRY bit to “1”, U2C0 register's UFORM bit to “0” and U2C1 register's U2LCH bit to “0”.

• Inverse Format

Set the PRY bit to “0”, UFORM bit to “1” and U2LCH bit to “1”.

Figure 1.15.33 shows the SIM interface format.

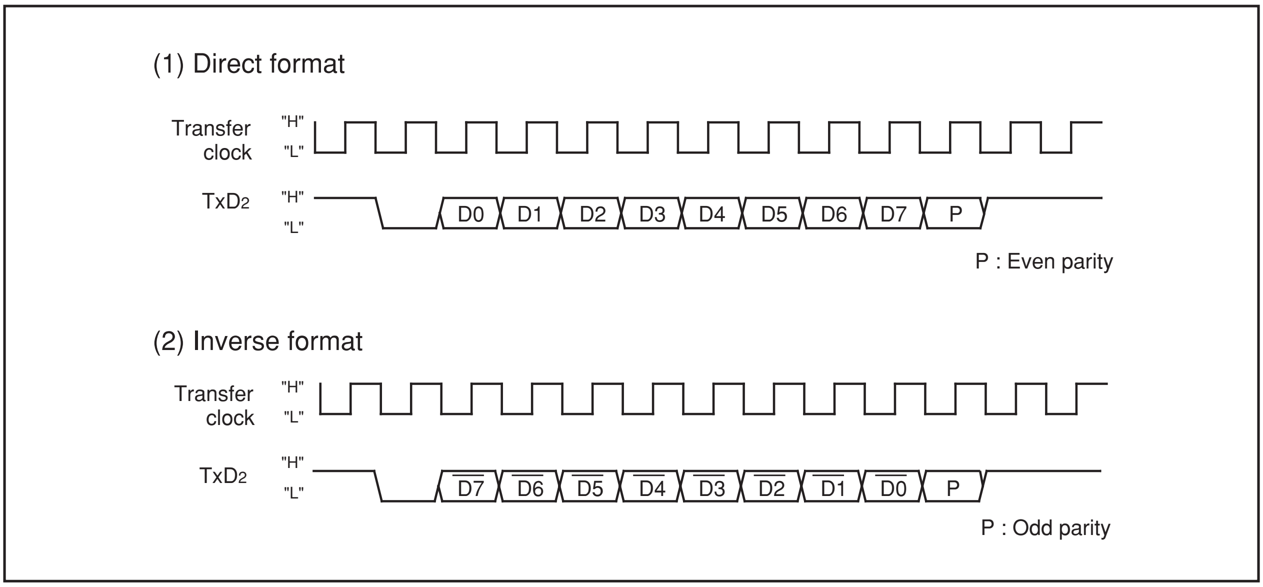


Figure 1.15.33 SIM Interface Format

SI/O3

SI/O3 is exclusive clock-synchronous serial I/O.

Figure 1.15.34 shows the block diagram of SI/O3, and Figure 1.15.35 shows the SI/O3-related registers.

Table 1.15.18 lists the specifications of SI/O3.

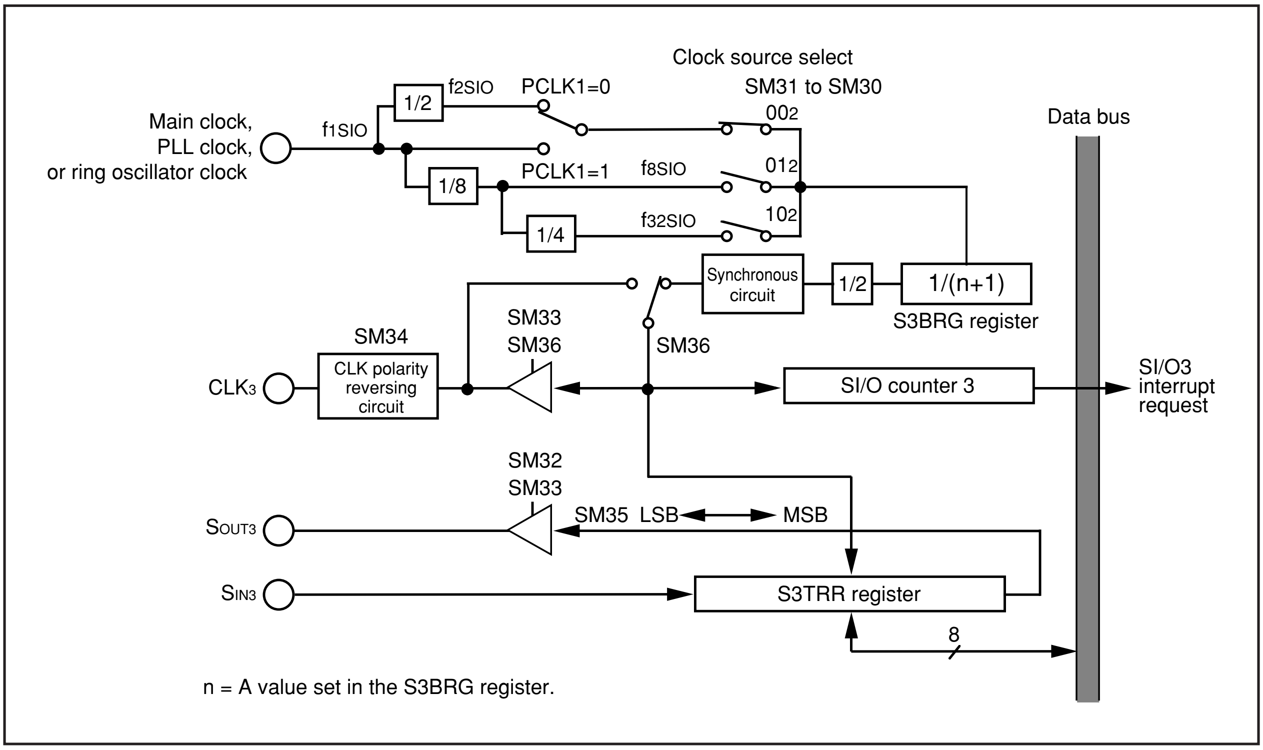


Figure 1.15.34 SI/O3 Block Diagram

SI/O3 control register (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
S3C	$01E2_{16}$	01000000_{16}

Bit symbol	Bit name	Description	RW
SM30	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
SM31			RW
SM32	S_{OUT3} output disable bit (Note 2)	0 : S_{OUT3} output 1 : S_{OUT3} output disabled (high-impedance)	RW
SM33	S I/O3 port select bit	0 : Input/output port 1 : S_{OUT3} output, CLK_3 function	RW
SM34	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
SM35	Transfer direction select bit	0 : LSB first 1 : MSB first	RW
SM36	Synchronous clock select bit	0 : External clock (Note 3) 1 : Internal clock (Note 4)	RW
SM37	S_{OUT3} initial value set bit	Effective when SM33 = 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 enabled).
Note 2: When the SM32 bit is set to "1", the target pin goes to a high-impedance state regardless of which function of the pin is being used.
Note 3: Set the SM33 bit to "1" and the corresponding port direction bit to "0" (input mode).
Note 4: Set the SM33 bit to "1" (S_{OUT3} output, CLK_3 function).

SI/O3 bit rate generator (Notes 1, 2)

b7 b0

Symbol	Address	After reset
S3BRG	$01E3_{16}$	Indeterminate

Description	Setting range	RW
Assuming that set value = n, S3BRG 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/O3 transmit/receive register (Notes 1, 2)

b7 b0

Symbol	Address	After reset
S3TRR	$01E0_{16}$	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 S_{IN3} to "0" (input mode).

Figure 1.15.35 S3C Register, S3BRG Register and S3TRR Register

Table 1.15.18 SI/O3 Specifications

Item	Specification
Transfer data format	• Transfer data length: 8 bits
Transfer clock	• S3C register's SM36 bit = 1 (internal clock) : fj/ 2(n+1) fj = f_{1SIO}, f_{8SIO}, f_{32SIO}. n = Setting value of S3BRG register 00_{16} to FF_{16}. • SM36 bit = 0 (external clock) : Input from CLK_3 pin (Note 1)
Transmission/reception start condition	• Before transmission/reception can start, the following requirements must be met Write transmit data to the S3TRR register (Notes 2, 3)
Interrupt request generation timing	• When S3C register's SM34 bit = 0 The rising edge of the last transfer clock pulse (Note 4) • When SM34 = 1 The falling edge of the last transfer clock pulse (Note 4)
CLK_3 pin function	I/O port, transfer clock input, transfer clock output
S_{OUT3} pin function	I/O port, transmit data output, high-impedance
SIN3 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 S_{OUT3} initial value set function When the S3C register's SM36 bit = 0 (external clock), the S_{OUT3} pin output level while not transmitting can be selected. • CLK polarity selection Whether transmit data is output/input timing at the rising edge or falling edge of transfer clock can be selected.

Note 1: To set the S3C register's SM36 bit to "0" (external clock), follow the procedure described below.
- If the S3C register's SM34 bit = 0, write transmit data to the S3TRR register while input on the CLK_3 pin is high. The same applies when rewriting the S3C register's SM37 bit.
- If the SM34 bit = 1, write transmit data to the S3TRR register while input on the CLK_3 pin is low. The same applies when rewriting the SM37 bit.
- Because shift operation continues as long as the transfer clock is supplied to the SI/O3 circuit, stop the transfer clock after supplying eight pulses. If the SM36 bit = 1 (internal clock), the transfer clock automatically stops.

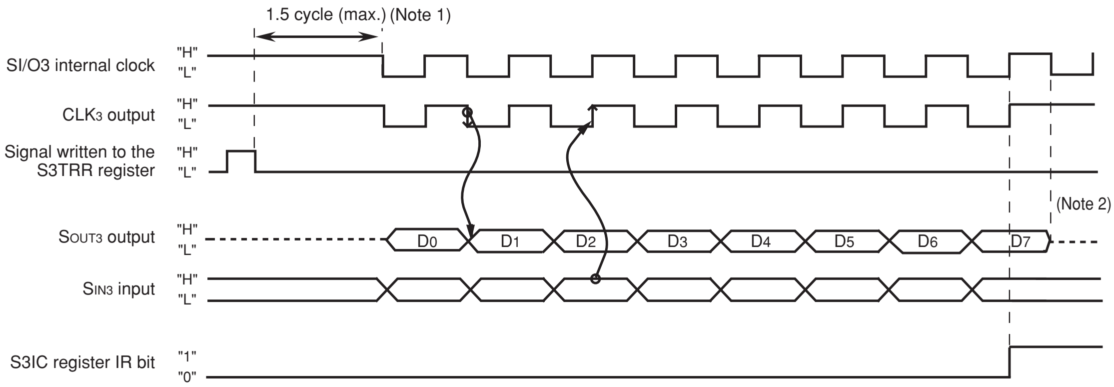
Note 2: Unlike UART0 to UART2, SI/O3 is not separated between the transfer register and buffer. Therefore, do not write the next transmit data to the S3TRR register during transmission.

Note 3: When the S3C register's SM36 bit = 1 (internal clock), S_{OUT3} 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 S3TRR register during this period, S_{OUT3} immediately goes to a high-impedance state, with the data hold time thereby reduced.

Note 4: When the S3C register's SM36 bit = 1 (internal clock), the transfer clock stops in the high state if the SM34 bit = 0, or stops in the low state if the SM34 bit = 1.

(a) SI/O3 Operation Timing

Figure 1.15.36 shows the SI/O3 operation timing.



* This diagram applies to the case where the S3C register bits are set as follows:

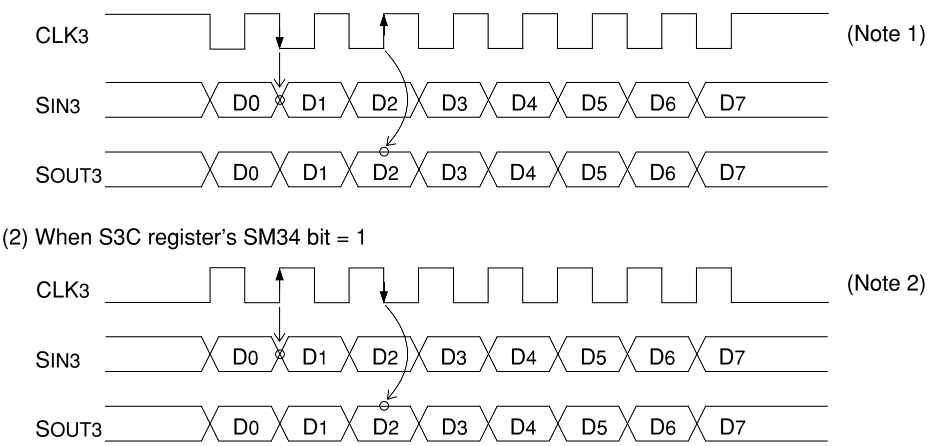
- SM32 = 0 (SOUT3 output)
- SM33 = 1 (SOUT3 output, CLK3 function)
- SM34 = 0 (transmit data output at the falling edge and receive data input at the rising edge of the transfer clock)
- SM35 = 0 (LSB first)
- SM36 = 1 (internal clock)

Note 1: If the SM36 bit = 1 (internal clock), the serial I/O starts sending or receiving data a maximum of 1.5 transfer clock cycles after writing to the S3TRR register.
Note 2: When the SM36 bit = 1 (internal clock), the SOUT3 pin is placed in the high-impedance state after the transfer finishes.

Figure 1.15.36 SI/O3 Operation Timing

(b) CLK Polarity Selection

The S3C register's SM34 bit allows selection of the polarity of the transfer clock. Figure 1.15.37 shows the polarity of the transfer clock.



*This diagram applies to the case where the S3C register bits are set as follows:

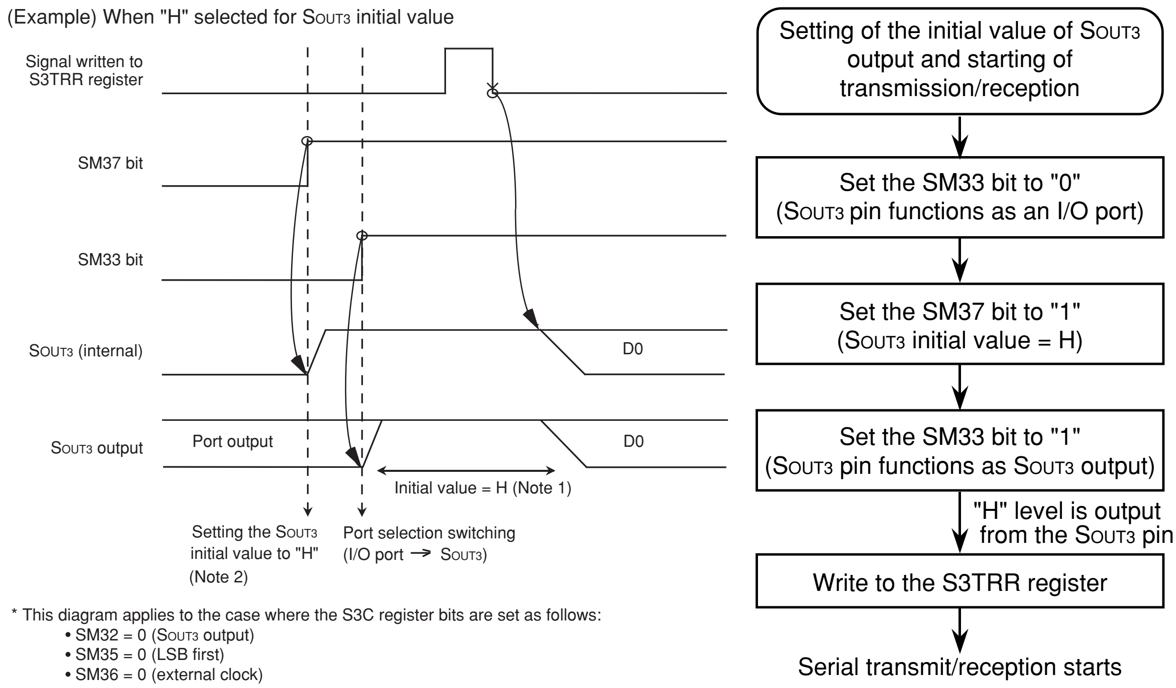
- SM35 = 0 (LSB first)
- SM36 = 1 (internal clock)

Note 1: When the SM36 bit = 1 (internal clock), a high level is output from the CLK3 pin if not transferring data.
Note 2: When the SM36 bit = 1 (internal clock), a low level is output from the CLK3 pin if not transferring data.

Figure 1.15.37 Polarity of Transfer Clock

(c) Functions for Setting an S_{OUT3} Initial Value

If the S3C register's SM36 bit = 0 (external clock), the S_{OUT3} pin output can be fixed high or low when not transferring. Figure 1.15.38 shows the timing chart for setting an S_{OUT3} initial value and how to set it.



* This diagram applies to the case where the S3C register bits are set as follows:
- SM32 = 0 (S_{OUT3} output)
- SM35 = 0 (LSB first)
- SM36 = 0 (external clock)

Note 1: If the SM36 bit = 1 (internal clock) or if the SM32 bit = 1 (S_{OUT3} output disabled), this output goes to the high-impedance state.
Note 2: S_{OUT3} can only be initialized when input on the CLK_3 pin is in the high state if the S3C register's SM34 bit = 0 (transmit data output at the falling edge of the transfer clock) or in the low state if the SM34 bit = 1 (transmit data output at the rising edge of the transfer clock).

Figure 1.15.38 S_{OUT3}'s Initial Value Setting

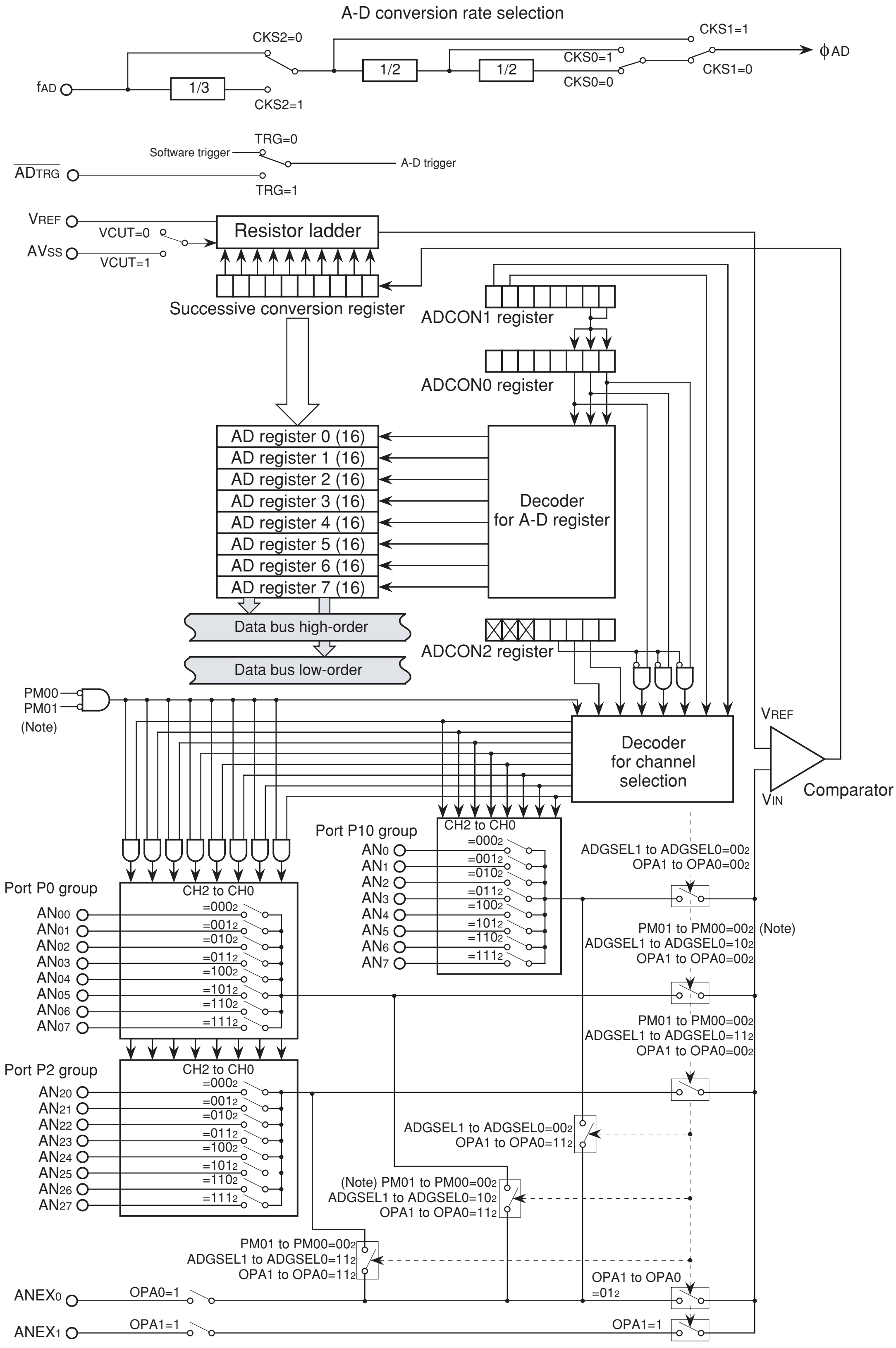
A-D Converter

The microcomputer contains one A-D converter circuit based on 10-bit successive approximation method configured with a capacitive-coupling amplifier. The analog inputs share the pins with $P10_0$ to $P10_7$, $P9_5$, $P9_6$, $P0_0$ to $P0_7$, and $P2_0$ to $P2_7$. Similarly, $\overline{AD_{TRG}}$ input shares the pin with $P9_7$. Therefore, when using these inputs, make sure the corresponding port direction bits are set to “0” (input mode).
When not using the A-D converter, set the VCUT bit to “0” (V_{REF} unconnected), so that no current will flow from the V_{REF} pin into the resistor ladder, helping to reduce the power consumption of the chip.
The A-D conversion result is stored in the ADi register bits for AN_i, AN_{0i}, and AN_{2i} pins (i = 0 to 7).
Table 1.16.1 shows the performance of the A-D converter. Figure 1.16.1 shows the block diagram of the A-D converter, and Figures 1.16.2 and 1.16.3 show the A-D converter-related registers.

Table 1.16.1 A-D Converter Performance

Item	Performance
Method of A-D conversion	Successive approximation (capacitive coupling amplifier)
Analog input voltage (Note 1)	0V to AV_{CC} (V_{CC})
Operating clock ϕ_{AD} (Note 2)	f_{AD}, divide-by-2 of f_{AD}, divide-by-3 of f_{AD}, divide-by-4 of f_{AD}, divide-by-6 of f_{AD}, divide-by-12 of f_{AD}
Resolution	8 bits or 10 bits (selectable)
Integral nonlinearity error	• With 8-bit resolution: ±2LSB • With 10-bit resolution : ±3LSB When external operation amp connection mode is selected : ±7LSB
Operating modes	One-shot mode, repeat mode, single sweep mode, repeat sweep mode 0, and repeat sweep mode 1
Analog input pins	8 pins (AN_0 to AN_7) + 2 pins (ANEX0 and ANEX1) + 8 pins (AN_{00} to AN_{07}) + 8 pins (AN_{20} to AN_{27})
A-D conversion start condition	• Software trigger The ADCON0 register's ADST bit is set to “1” (A-D conversion starts) • External trigger (retriggerable) Input on the $\overline{AD_{TRG}}$ pin changes state from high to low after the ADST bit is set to “1” (A-D conversion starts)
Conversion speed per pin	• Without sample and hold function 8-bit resolution: 49 ϕ_{AD} cycles, 10-bit resolution: 59 ϕ_{AD} cycles • With sample and hold function 8-bit resolution: 28 ϕ_{AD} cycles, 10-bit resolution: 33 ϕ_{AD} cycles

Note 1: Does not depend on use of sample and hold function.
Note 2: Operation clock frequency (ϕ_{AD} frequency) must be 10 MHz or less.
A case without sample-and-hold function, turn (ϕ_{AD} frequency) into 250 kHz or more.
A case with the sample and hold function, turn (ϕ_{AD} frequency) into 1 MHz or more.



Note: Port P0 group (AN00 to AN07) can be used as analog input pins even when the PM01 to PM00 bits are set to "01₂" (memory expansion mode) and the PM05 to PM04 bits are set to "11₂" (multiplexed bus allocated to the entire $\overline{CS}$ space)

Figure 1.16.1 A-D Converter Block Diagram

A-D control register 0 (Note)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
ADCON0	$03D6_{16}$	$00000XXX_2$

Bit symbol	Bit name	Function	RW
CH0	Analog input pin select bit	Function varies with each operation mode	RW
CH1			RW
CH2			RW
MD0	A-D operation mode select bit 0	b4 b3 0 0 : One-shot mode 0 1 : Repeat mode 1 0 : Single sweep mode 1 1 : Repeat sweep mode 0 or Repeat sweep mode 1	RW
MD1			RW
TRG	Trigger select bit	0 : Software trigger 1 : $\overline{AD_{TRG}}$ trigger	RW
ADST	A-D conversion start flag	0 : A-D conversion disabled 1 : A-D conversion started	RW
CKS0	Frequency select bit 0	See Note 2 for the ADCON2 register	RW

Note: If the ADCON0 register is rewritten during A-D conversion, the conversion result will be indeterminate.

A-D control register 1 (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

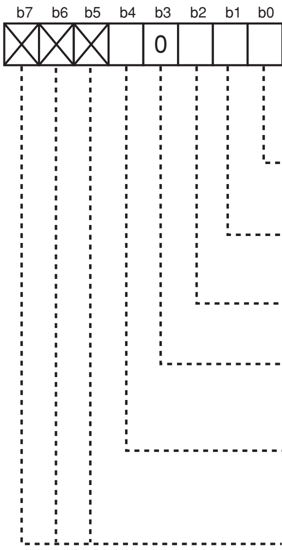
Symbol	Address	After reset
ADCON1	$03D7_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
SCAN0	A-D sweep pin select bit	Function varies with each operation mode	RW
SCAN1			RW
MD2	A-D operation mode select bit 1	0 : Any mode other than repeat sweep mode 1 1 : Repeat sweep mode 1	RW
BITS	8/10-bit mode select bit	0 : 8-bit mode 1 : 10-bit mode	RW
CKS1	Frequency select bit 1	See Note 2 for the ADCON2 register	RW
VCUT	V_{REF} connect bit (Note 2)	0 : V_{REF} not connected 1 : V_{REF} connected	RW
OPA0	External op-amp connection mode bit	Function varies with each operation mode	RW
OPA1			RW

Note 1: If the ADCON1 register is rewritten during A-D conversion, the conversion result will be indeterminate.
Note 2: If the VCUT bit is reset from "0" (V_{REF} unconnected) to "1" (V_{REF} connected), wait for 1 µs or more before starting A-D conversion.

Figure 1.16.2 ADCON0 Register and ADCON1 Register

A-D control register 2 (Note 1)



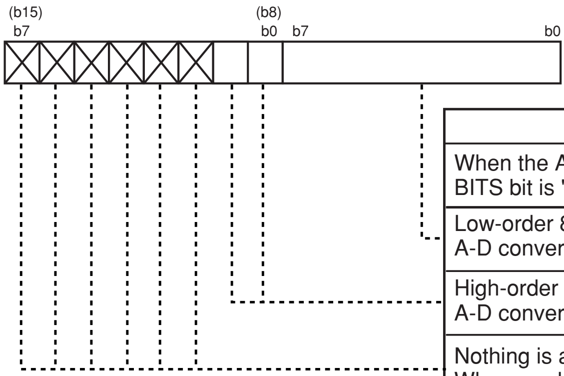
Symbol	Address	After reset
ADCON2	03D4₁₆	00₁₆

Bit symbol	Bit name	Function	RW
SMP	A-D conversion method select bit	0 : Without sample and hold 1 : With sample and hold	RW
ADGSEL0	A-D input group select bit	b2 b1 0 0 : Port P10 group is selected 0 1 : Must not be set 1 0 : Port P0 group is selected 1 1 : Port P2 group is selected	RW
ADGSEL1			RW
– (b3)	Reserved bit	Set to "0"	RW
CKS2	Frequency select bit 2 (Note 2)	0 : Selects f_{AD}, divide-by-2 of f_{AD}, or divide-by-4 of f_{AD}. 1 : Selects divide-by-3 of f_{AD}, divide-by-6 of f_{AD}, or divide-by-12 of f_{AD}.	RW
– (b7-b5)	Nothing is assigned. When write, set to "0". When read, their contents are "0".		–

Note 1: If the ADCON2 register is rewritten during A-D conversion, the conversion result will be indeterminate.
Note 2: The ϕ_{AD} frequency must be 10 MHz or less. The selected ϕ_{AD} frequency is determined by a combination of the ADCON0 register's CKS0 bit, ADCON1 register's CKS1 bit, and ADCON2 register's CKS2 bit.

CKS0	CKS1	CKS2	ϕ_{AD}
0	0	0	Divide-by-4 of f_{AD}
0	0	1	Divide-by-2 of f_{AD}
0	1	0	f_{AD}
0	1	1	
1	0	0	Divide-by-12 of f_{AD}
1	0	1	Divide-by-6 of f_{AD}
1	1	0	Divide-by-3 of f_{AD}
1	1	1	

A-D register i (i = 0 to 7)



Symbol	Address	After reset
AD0	03C1₁₆ to 03C0₁₆	Indeterminate
AD1	03C3₁₆ to 03C2₁₆	Indeterminate
AD2	03C5₁₆ to 03C4₁₆	Indeterminate
AD3	03C7₁₆ to 03C6₁₆	Indeterminate
AD4	03C9₁₆ to 03C8₁₆	Indeterminate
AD5	03CB₁₆ to 03CA₁₆	Indeterminate
AD6	03CD₁₆ to 03CC₁₆	Indeterminate
AD7	03CF₁₆ to 03CE₁₆	Indeterminate

Function		RW
When the ADCON1 register's BITS bit is "1" (10-bit mode)	When the ADCON1 register's BITS bit is "0" (8-bit mode)	RW
Low-order 8 bits of A-D conversion result	A-D conversion result	RO
High-order 2 bits of A-D conversion result	When read, the content is indeterminate.	RO
Nothing is assigned. When write, set to "0". When read, their contents are "0".		–

Figure 1.16.3 ADCON2 Register, and AD0 to AD7 Registers

(1) One-shot Mode

In this mode, the input voltage on one selected pin is A-D converted once. Table 1.16.2 lists the specifications of one-shot mode. Figure 1.16.4 shows the ADCON0 and ADCON1 registers in one-shot mode.

Table 1.16.2 One-shot Mode Specifications

Item	Specification
Function	The input voltage on one pin selected by the ADCON0 register's CH2 to CH0 bits and ADCON2 register's ADGSEL1 to ADGSEL0 bits or the ADCON1 register's OPA1 to OPA0 bits is A-D converted once.
A-D conversion start condition	• When the ADCON0 register's TRG bit is “0” (software trigger) The ADCON0 register's ADST bit is set to “1” (A-D conversion starts) • When the TRG bit is “1” ($\overline{AD_{TRG}}$ trigger) Input on the $\overline{AD_{TRG}}$ pin changes state from high to low after the ADST bit is set to “1” (A-D conversion starts)
A-D conversion stop condition	• Completion of A-D conversion (If a software trigger is selected, the ADST bit is set to “0” (A-D conversion halted).) • Set the ADST bit to “0”
Interrupt request generation timing	Completion of A-D conversion
Analog input pin	Select one pin from AN_0 to AN_7, AN_{00} to AN_{07}, AN_{20} to AN_{27}, ANEX0 to ANEX1
Reading of result of A-D converter	Read one of the AD0 to AD7 registers that corresponds to the selected pin

A-D control register 0 (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
			0	0			

Symbol	Address	After reset
ADCON0	$03D6_{16}$	$00000XXX_2$

Bit symbol	Bit name	Function	RW
CH0	Analog input pin select bit	b2 b1 b0 0 0 0 : AN_0 is selected 0 0 1 : AN_1 is selected 0 1 0 : AN_2 is selected 0 1 1 : AN_3 is selected 1 0 0 : AN_4 is selected 1 0 1 : AN_5 is selected 1 1 0 : AN_6 is selected (Note 2) 1 1 1 : AN_7 is selected (Note 3)	RW
CH1			RW
CH2			RW
MD0	A-D operation mode select bit 0	b4 b3 0 0 : One-shot mode (Note 3)	RW
MD1			RW
TRG	Trigger select bit	0 : Software trigger 1 : $\overline{AD_{TRG}}$ trigger	RW
ADST	A-D conversion start flag	0 : A-D conversion disabled 1 : A-D conversion started	RW
CKS0	Frequency select bit 0	See Note 2 for the ADCON2 register	RW

Note 1: If the ADCON0 register is rewritten during A-D conversion, the conversion result will be indeterminate.
Note 2: AN_{00} to AN_{07}, and AN_{20} to AN_{27} can be used in same way as AN_0 to AN_7. Use the ADCON2 register’s ADGSEL1 to ADGSEL0 bits to select the desired pin.
Note 3: After rewriting the MD1 to MD0 bits, set the CH2 to CH0 bits over again using another instruction.

A-D control register 1 (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
		1			0		

Symbol	Address	After reset
ADCON1	$03D7_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
SCAN0	A-D sweep pin select bit	Invalid in one-shot mode	RW
SCAN1			RW
MD2	A-D operation mode select bit 1	Set to "0" when one-shot mode is selected	RW
BITS	8/10-bit mode select bit	0 : 8-bit mode 1 : 10-bit mode	RW
CKS1	Frequency select bit 1	See Note 2 for the ADCON2 register	RW
VCUT	V_{REF} connect bit (Note 2)	1 : V_{REF} connected	RW
OPA0	External op-amp connection mode bit	b7 b6 0 0 : ANEX0 and ANEX1 are not used 0 1 : ANEX0 input is A-D converted 1 0 : ANEX1 input is A-D converted 1 1 : External op-amp connection mode	RW
OPA1			RW

Note 1: If the ADCON1 register is rewritten during A-D conversion, the conversion result will be indeterminate.
Note 2: If the VCUT bit is reset from "0" (V_{REF} unconnected) to "1" (V_{REF} connected), wait for 1 µs or more before starting A-D conversion.

Figure 1.16.4 ADCON0 Register and ADCON1 Register in One-shot Mode

(2) Repeat Mode

In this mode, the input voltage on one selected pin is A-D converted repeatedly. Table 1.16.3 lists the specifications of repeat mode. Figure 1.16.5 shows the ADCON0 and ADCON1 registers in repeat mode.

Table 1.16.3 Repeat Mode Specifications

Item	Specification
Function	The input voltage on one pin selected by the ADCON0 register's CH2 to CH0 bits and ADCON2 register's ADGSEL1 to ADGSEL0 bits or the ADCON1 register's OPA1 to OPA0 bits is A-D converted repeatedly.
A-D conversion start condition	• When the ADCON0 register's TRG bit is “0” (software trigger) The ADCON0 register's ADST bit is set to “1” (A-D conversion starts) • When the TRG bit is “1” ($\overline{AD_{TRG}}$ trigger) Input on the $\overline{AD_{TRG}}$ pin changes state from high to low after the ADST bit is set to “1” (A-D conversion starts)
A-D conversion stop condition	Set the ADST bit to “0” (A-D conversion halted)
Interrupt request generation timing	None generated
Analog input pin	Select one pin from AN_0 to AN_7, AN_{00} to AN_{07}, AN_{20} to AN_{27}, ANEX0 to ANEX1
Reading of result of A-D converter	Read one of the AD0 to AD7 registers that corresponds to the selected pin

A-D control register 0 (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0: b4 = 0, b3 = 1

Symbol	Address	After reset
ADCON0	$03D6_{16}$	$00000XXX_2$

Bit symbol	Bit name	Function	RW
CH0	Analog input pin select bit	b2 b1 b0 0 0 0 : AN_0 is selected 0 0 1 : AN_1 is selected 0 1 0 : AN_2 is selected 0 1 1 : AN_3 is selected 1 0 0 : AN_4 is selected 1 0 1 : AN_5 is selected 1 1 0 : AN_6 is selected (Note 2) 1 1 1 : AN_7 is selected (Note 3)	RW
CH1			RW
CH2			RW
MD0	A-D operation mode select bit 0	b4 b3 0 1 : Repeat mode (Note 3)	RW
MD1			RW
TRG	Trigger select bit	0 : Software trigger 1 : $\overline{AD_{TRG}}$ trigger	RW
ADST	A-D conversion start flag	0 : A-D conversion disabled 1 : A-D conversion started	RW
CKS0	Frequency select bit 0	See Note 2 for the ADCON2 register	RW

Note 1: If the ADCON0 register is rewritten during A-D conversion, the conversion result will be indeterminate.
Note 2: AN_{00} to AN_{07}, and AN_{20} to AN_{27} can be used in same way as AN_0 to AN_7. Use the ADCON2 register's ADGSEL1 to ADGSEL0 bits to select the desired pin.
Note 3: After rewriting the MD1 to MD0 bits, set the CH2 to CH0 bits over again using another instruction.

A-D control register 1 (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0: b5 = 1, b2 = 0

Symbol	Address	After reset
ADCON1	$03D7_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
SCAN0	A-D sweep pin select bit	Invalid in repeat mode	RW
SCAN1			RW
MD2	A-D operation mode select bit 1	Set to "0" when repeat mode is selected	RW
BITS	8/10-bit mode select bit	0 : 8-bit mode 1 : 10-bit mode	RW
CKS1	Frequency select bit 1	See Note 2 for the ADCON2 register	RW
VCUT	V_{REF} connect bit (Note 2)	1 : V_{REF} connected	RW
OPA0	External op-amp connection mode bit	b7 b6 0 0 : ANEX0 and ANEX1 are not used 0 1 : ANEX0 input is A-D converted 1 0 : ANEX1 input is A-D converted 1 1 : External op-amp connection mode	RW
OPA1			RW

Note 1: If the ADCON1 register is rewritten during A-D conversion, the conversion result will be indeterminate.
Note 2: If the VCUT bit is reset from "0" (V_{REF} unconnected) to "1" (V_{REF} connected), wait for 1 µs or more before starting A-D conversion.

Figure 1.16.5 ADCON0 Register and ADCON1 Register in Repeat Mode

(3) Single Sweep Mode

In this mode, the input voltages on selected pins are A-D converted, one pin at a time. Table 1.16.4 lists the specifications of single sweep mode. Figure 1.16.6 shows the ADCON0 and ADCON1 registers in single sweep mode.

Table 1.16.4 Single Sweep Mode Specifications

Item	Specification
Function	The input voltages on pins selected by the ADCON1 register's SCAN1 to SCAN0 bits and ADCON2 register's ADGSEL1 to ADGSEL0 bits are A-D converted, one pin at a time.
A-D conversion start condition	• When the ADCON0 register's TRG bit is “0” (software trigger) The ADCON0 register's ADST bit is set to “1” (A-D conversion starts) • When the TRG bit is “1” ($\overline{AD_{TRG}}$ trigger) Input on the $\overline{AD_{TRG}}$ pin changes state from high to low after the ADST bit is set to “1” (A-D conversion starts)
A-D conversion stop condition	• Completion of A-D conversion (If a software trigger is selected, the ADST bit is set to “0” (A-D conversion halted).) • Set the ADST bit to “0”
Interrupt request generation timing	Completion of A-D conversion
Analog input pin	Select from AN_0 to AN_1 (2 pins), AN_0 to AN_3 (4 pins), AN_0 to AN_5 (6 pins), AN_0 to AN_7 (8 pins) (Note)
Reading of result of A-D converter	Read one of the AD0 to AD7 registers that corresponds to the selected pin

Note: AN_{00} to AN_{07}, and AN_{20} to AN_{27} can be used in the same way as AN_0 to AN_7.

A-D control register 0 (Note)

b7	b6	b5	b4	b3	b2	b1	b0
			1	0			

Symbol	Address	After reset
ADCON0	$03D6_{16}$	$00000XXX_2$

Bit symbol	Bit name	Function	RW
CH0	Analog input pin select bit	Invalid in single sweep mode	RW
CH1			RW
CH2			RW
MD0	A-D operation mode select bit 0	b4 b3 1 0 : Single sweep mode	RW
MD1			RW
TRG	Trigger select bit	0 : Software trigger 1 : $\overline{AD_{TRG}}$ trigger	RW
ADST	A-D conversion start flag	0 : A-D conversion disabled 1 : A-D conversion started	RW
CKS0	Frequency select bit 0	See Note 2 for the ADCON2 register	RW

Note: If the ADCON0 register is rewritten during A-D conversion, the conversion result will be indeterminate.

A-D control register 1 (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
		1			0		

Symbol	Address	After reset
ADCON1	$03D7_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
SCAN0	A-D sweep pin select bit	When single sweep mode is selected b1 b0 0 0 : AN_0, AN_1 (2 pins) 0 1 : AN_0 to AN_3 (4 pins) 1 0 : AN_0 to AN_5 (6 pins) 1 1 : AN_0 to AN_7 (8 pins) (Note 2)	RW
SCAN1			RW
MD2	A-D operation mode select bit 1	Set to "0" when single sweep mode is selected	RW
BITS	8/10-bit mode select bit	0 : 8-bit mode 1 : 10-bit mode	RW
CKS1	Frequency select bit 1	See Note 2 for the ADCON2 register	RW
VCUT	V_{REF} connect bit (Note 3)	1 : V_{REF} connected	RW
OPA0	External op-amp connection mode bit	b7 b6 0 0 : ANEX0 and ANEX1 are not used 0 1 : Must not be set 1 0 : Must not be set 1 1 : External op-amp connection mode	RW
OPA1			RW

Note 1: If the ADCON1 register is rewritten during A-D conversion, the conversion result will be indeterminate.
Note 2: AN_{00} to AN_{07}, and AN_{20} to AN_{27} can be used in same way as AN_0 to AN_7. Use the ADCON2 register's ADGSEL1 to ADGSEL0 bits to select the desired pin.
Note 3: If the VCUT bit is reset from "0" (V_{REF} unconnected) to "1" (V_{REF} connected), wait for 1 µs or more before starting A-D conversion.

Figure 1.16.6 ADCON0 Register and ADCON1 Register in Single Sweep Mode

(4) Repeat Sweep Mode 0

In this mode, the input voltages on selected pins are A-D converted repeatedly. Table 1.16.5 lists the specifications of repeat sweep mode 0. Figure 1.16.7 shows the ADCON0 and ADCON1 registers in repeat sweep mode 0.

Table 1.16.5 Repeat Sweep Mode 0 Specifications

Item	Specification
Function	The input voltages on pins selected by the ADCON1 register's SCAN1 to SCAN0 bits and ADCON2 register's ADGSEL1 to ADGSEL0 bits are A-D converted repeatedly.
A-D conversion start condition	• When the ADCON0 register's TRG bit is “0” (software trigger) The ADCON0 register's ADST bit is set to “1” (A-D conversion starts) • When the TRG bit is “1” ($\overline{AD_{TRG}}$ trigger) Input on the $\overline{AD_{TRG}}$ pin changes state from high to low after the ADST bit is set to “1” (A-D conversion starts)
A-D conversion stop condition	Set the ADST bit to “0” (A-D conversion halted)
Interrupt request generation timing	None generated
Analog input pin	Select from AN_0 to AN_1 (2 pins), AN_0 to AN_3 (4 pins), AN_0 to AN_5 (6 pins), AN_0 to AN_7 (8 pins) (Note)
Reading of result of A-D converter	Read one of the AD0 to AD7 registers that corresponds to the selected pin

Note: AN_{00} to AN_{07}, and AN_{20} to AN_{27} can be used in the same way as AN_0 to AN_7.

A-D control register 0 (Note)

b7	b6	b5	b4	b3	b2	b1	b0
			1	1			

Symbol	Address	After reset
ADCON0	03D6₁₆	00000XXX₂

Bit symbol	Bit name	Function	RW
CH0	Analog input pin select bit	Invalid in repeat sweep mode 0	RW
CH1			RW
CH2			RW
MD0	A-D operation mode select bit 0	b4 b3 1 1 : Repeat sweep mode 0 or Repeat sweep mode 1	RW
MD1			RW
TRG	Trigger select bit	0 : Software trigger 1 : $\overline{AD_{TRG}}$ trigger	RW
ADST	A-D conversion start flag	0 : A-D conversion disabled 1 : A-D conversion started	RW
CKS0	Frequency select bit 0	See Note 2 for the ADCON2 register	RW

Note: If the ADCON0 register is rewritten during A-D conversion, the conversion result will be indeterminate.

A-D control register 1 (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
		1			0		

Symbol	Address	After reset
ADCON1	03D7₁₆	00₁₆

Bit symbol	Bit name	Function	RW
SCAN0	A-D sweep pin select bit	When repeat sweep mode 0 is selected b1 b0 0 0 : AN_0, AN_1 (2 pins) 0 1 : AN_0 to AN_3 (4 pins) 1 0 : AN_0 to AN_5 (6 pins) 1 1 : AN_0 to AN_7 (8 pins) (Note 2)	RW
SCAN1			RW
MD2	A-D operation mode select bit 1	Set to "0" when repeat sweep mode 0 is selected	RW
BITS	8/10-bit mode select bit	0 : 8-bit mode 1 : 10-bit mode	RW
CKS1	Frequency select bit 1	See Note 2 for the ADCON2 register	RW
VCUT	V_{REF} connect bit (Note 3)	1 : V_{REF} connected	RW
OPA0	External op-amp connection mode bit	b7 b6 0 0 : ANEX0 and ANEX1 are not used 0 1 : Must not be set 1 0 : Must not be set 1 1 : External op-amp connection mode	RW
OPA1			RW

Note 1: If the ADCON1 register is rewritten during A-D conversion, the conversion result will be indeterminate.
Note 2: AN_{00} to AN_{07}, and AN_{20} to AN_{27} can be used in same way as AN_0 to AN_7. Use the ADCON2 register's ADGSEL1 to ADGSEL0 bits to select the desired pin.
Note 3: If the VCUT bit is reset from "0" (V_{REF} unconnected) to "1" (V_{REF} connected), wait for 1 µs or more before starting A-D conversion.

Figure 1.16.7 ADCON0 Register and ADCON1 Register in Repeat Sweep Mode 0

(5) Repeat Sweep Mode 1

In this mode, the input voltages on all pins are A-D converted repeatedly, with priority given to the selected pins. Table 1.16.6 lists the specifications of repeat sweep mode 1. Figure 1.16.8 shows the ADCON0 and ADCON1 registers in repeat sweep mode 1.

Table 1.16.6 Repeat Sweep Mode 1 Specifications

Item	Specification
Function	The input voltages on all pins selected by the ADCON2 register's ADGSEL1 to ADGSEL0 bits are A-D converted repeatedly, with priority given to pins selected by the ADCON1 register's SCAN1 to SCAN0 bits and ADGSEL1 to ADGSEL0 bits. Example : If AN_0 selected, input voltages are A-D converted in order of $AN_0 \rightarrow AN_1 \rightarrow AN_0 \rightarrow AN_2 \rightarrow AN_0 \rightarrow AN_3$, and so on.
A-D conversion start condition	• When the ADCON0 register's TRG bit is “0” (software trigger) The ADCON0 register's ADST bit is set to “1” (A-D conversion starts) • When the TRG bit is “1” ($\overline{AD_{TRG}}$ trigger) Input on the $\overline{AD_{TRG}}$ pin changes state from high to low after the ADST bit is set to “1” (A-D conversion starts)
A-D conversion stop condition	Set the ADST bit to “0” (A-D conversion halted)
Interrupt request generation timing	None generated
Analog input pins to be given priority when A-D converted	Select from AN_0 (1 pin), AN_0 to AN_1 (2 pins), AN_0 to AN_2 (3 pins), AN_0 to AN_3 (4 pins) (Note)
Reading of result of A-D converter	Read one of the AD0 to AD7 registers that corresponds to the selected pin

Note: AN_{00} to AN_{07}, and AN_{20} to AN_{27} can be used in the same way as AN_0 to AN_7.

A-D control register 0 (Note)

b7	b6	b5	b4	b3	b2	b1	b0
			1	1			

Symbol: ADCON0 Address: $03D6_{16}$ After reset: $00000XXX_2$

Bit symbol	Bit name	Function	RW
CH0	Analog input pin select bit	Invalid in repeat sweep mode 1	RW
CH1			RW
CH2			RW
MD0	A-D operation mode select bit 0	b4 b3 1 1 : Repeat sweep mode 0 or Repeat sweep mode 1	RW
MD1			RW
TRG	Trigger select bit	0 : Software trigger 1 : $\overline{AD_{TRG}}$ trigger	RW
ADST	A-D conversion start flag	0 : A-D conversion disabled 1 : A-D conversion started	RW
CKS0	Frequency select bit 0	See Note 2 for the ADCON2 register	RW

Note: If the ADCON0 register is rewritten during A-D conversion, the conversion result will be indeterminate.

A-D control register 1 (Note 1)

b7	b6	b5	b4	b3	b2	b1	b0
		1			1		

Symbol: ADCON1 Address: $03D7_{16}$ After reset: 00_{16}

Bit symbol	Bit name	Function	RW
SCAN0	A-D sweep pin select bit	When repeat sweep mode 1 is selected b1 b0 0 0 : AN_0 (1 pin) 0 1 : AN_0, AN_1 (2 pins) 1 0 : AN_0 to AN_2 (3 pins) 1 1 : AN_0 to AN_3 (4 pins) (Note 2)	RW
SCAN1			RW
MD2	A-D operation mode select bit 1	Set to "1" when repeat sweep mode 1 is selected	RW
BITS	8/10-bit mode select bit	0 : 8-bit mode 1 : 10-bit mode	RW
CKS1	Frequency select bit 1	See Note 2 for the ADCON2 register	RW
VCUT	V_{REF} connect bit (Note 3)	1 : V_{REF} connected	RW
OPA0	External op-amp connection mode bit	b7 b6 0 0 : ANEX0 and ANEX1 are not used 0 1 : Must not be set 1 0 : Must not be set 1 1 : External op-amp connection mode	RW
OPA1			RW

Note 1: If the ADCON1 register is rewritten during A-D conversion, the conversion result will be indeterminate.
Note 2: AN_{00} to AN_{07}, and AN_{20} to AN_{27} can be used in same way as AN_0 to AN_7. Use the ADCON2 register's ADGSEL1 to ADGSEL0 bits to select the desired pin.
Note 3: If the VCUT bit is reset from "0" (V_{REF} unconnected) to "1" (V_{REF} connected), wait for 1 µs or more before starting A-D conversion.

Figure 1.16.8 ADCON0 Register and ADCON1 Register in Repeat Sweep Mode 1

(a) Resolution Select Function

The desired resolution can be selected using the ADCON1 register's BITS bit. If the BITS bit is set to "1" (10-bit conversion accuracy), the A-D conversion result is stored in the ADi register (i = 0 to 7)'s bit 0 to bit 9. If the BITS bit is set to "0" (8-bit conversion accuracy), the A-D conversion result is stored in the ADi register's bit 0 to bit 7.

(b) Sample and Hold

If the ADCON2 register's SMP bit is set to "1" (with sample-and-hold), the conversion speed per pin is increased to 28 ϕ_{AD} cycles for 8-bit resolution or 33 ϕ_{AD} cycles for 10-bit resolution. Sample-and-hold is effective in all operation modes. Select whether or not to use the sample-and-hold function before starting A-D conversion.

(c) Extended Analog Input Pins

In one-shot and repeat modes, the ANEX0 and ANEX1 pins can be used as analog input pins. Use the ADCON1 register's OPA1 to OPA0 bits to select whether or not use ANEX0 and ANEX1.
The A-D conversion results of ANEX0 and ANEX1 inputs are stored in the AD0 and AD1 registers, respectively.

(d) External Operation Amp Connection Mode

Multiple analog inputs can be amplified using a single external op-amp via the ANXE0 and ANEX1 pins. Set the ADCON1 register's OPA1 to OPA0 bits to "11_2" (external op-amp connection mode). The inputs from ANi (i = 0 to 7) (Note) are output from the ANEX0 pin. Amplify this output with an external op-amp before sending it back to the ANEX1 pin. The A-D conversion result is stored in the corresponding ADi register. The A-D conversion speed depends on the response characteristics of the external op-amp. Note that the ANXE0 and ANEX1 pins cannot be directly connected to each other. Figure 1.16.9 shows an example of how to connect the pins in external operation amp.

Note: AN_{0i} and AN_{2i} can be used the same as AN_i.

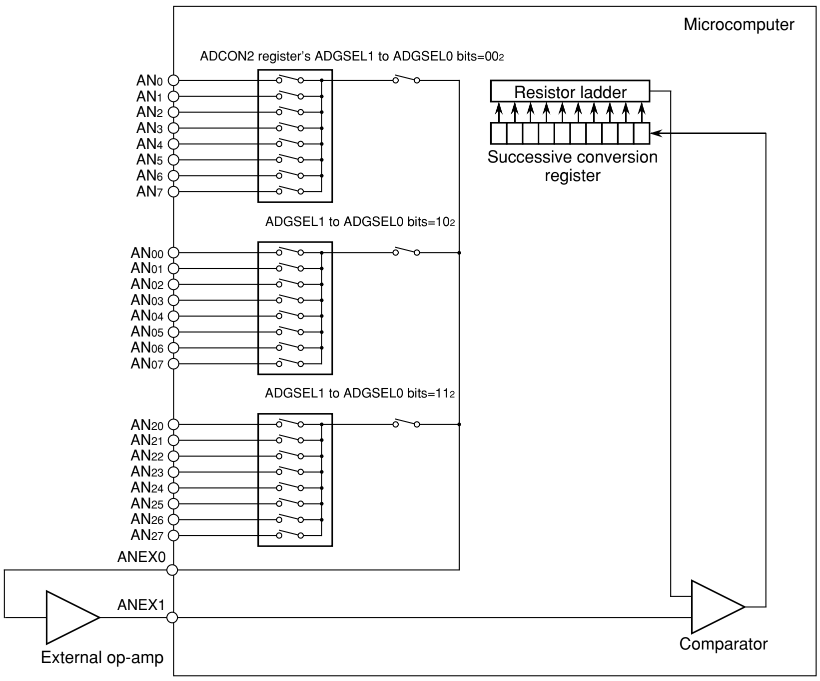


Figure 1.16.9 External Op-amp Connection

(e) Current Consumption Reducing Function

When not using the A-D converter, its resistor ladder and reference voltage input pin (V_{REF}) can be separated using the ADCON1 register's VCUT bit. When separated, no current will flow from the V_{REF} pin into the resistor ladder, helping to reduce the power consumption of the chip.
To use the A-D converter, set the VCUT bit to "1" (V_{REF} connected) and then set the ADCON0 register's ADST bit to "1" (A-D conversion start). The VCUT and ADST bits cannot be set to "1" at the same time. Nor can the VCUT bit be set to "0" (V_{REF} unconnected) during A-D conversion.
Note that this does not affect V_{REF} for the D-A converter (irrelevant).

(f) Analog Input Pin and External Sensor Equivalent Circuit Example

Figure 1.16.10 shows analog input pin and external sensor equivalent circuit example.

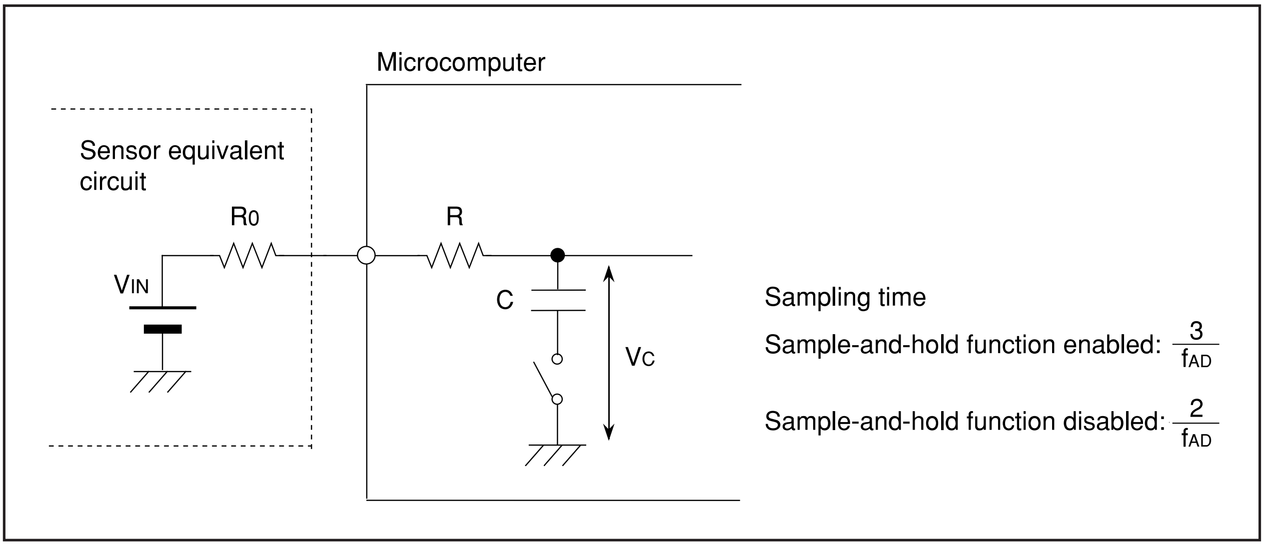


Figure 1.16.10 Analog Input Pin and External Sensor Equivalent Circuit

D-A Converter

This is an 8-bit, R-2R type D-A converter. These are two independent D-A converters.

D-A conversion is performed by writing to the DAi register (i = 0, 1). To output the result of conversion, set the DACON register's DAiE bit to "1" (output enabled). Before D-A conversion can be used, the corresponding port direction bit must be set to "0" (input mode). Setting the DAiE bit to "1" removes a pull-up from the corresponding port.

Output analog voltage (V) is determined by a set value (n : decimal) in the DAi register.

$$V = V_{REF} \times n/256 \ (n = 0 \text{ to } 255)$$

V_{REF} : reference voltage

Table 1.17.1 lists the performance of the D-A converter. Figure 1.17.1 shows the block diagram of the D-A converter. Figure 1.17.2 shows the D-A converter-related registers. Figure 1.17.3 shows the D-A converter equivalent circuit.

Table 1.17.1 D-A Converter Performance

Item	Performance
D-A conversion method	R-2R method
Resolution	8 bits
Analog output pin	2 (DA0 and DA1)

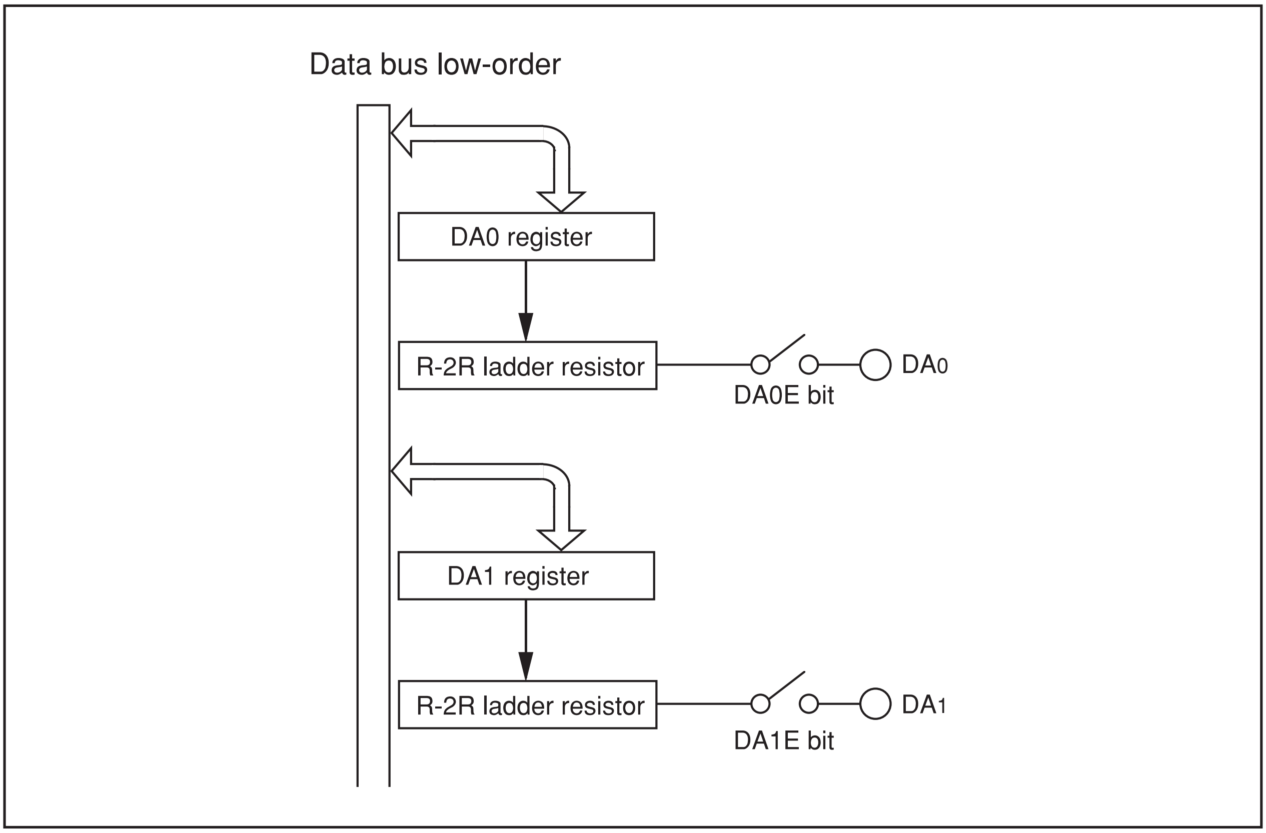


Figure 1.17.1 D-A Converter Block Diagram

D-A control register (Note)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
DACON	$03DC_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
DA0E	D-A0 output enable bit	0 : Output disabled 1 : Output enabled	RW
DA1E	D-A1 output enable bit	0 : Output disabled 1 : Output enabled	RW
– (b7-b2)	Nothing is assigned. When write, set to "0". When read, their contents are "0".		–

Note: When not using the D-A converter, set the DAiE bit (i = 0, 1) to "0" (output disabled) to reduce the unnecessary current consumption in the chip and set the DAi register to "00_{16}" to prevent current from flowing into the R-2R resistor ladder.

D-Ai register (Note) (i = 0, 1)

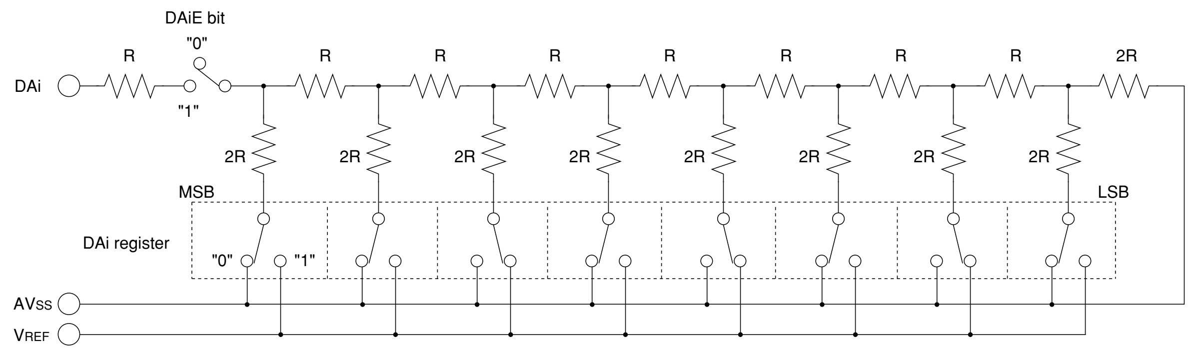
b7 b0

Symbol	Address	After reset
DA0	$03D8_{16}$	Indeterminate
DA1	$03DA_{16}$	Indeterminate

Function	RW
Output value of D-A conversion	RW

Note: When not using the D-A converter, set the DAiE bit (i = 0, 1) to "0" (output disabled) to reduce the unnecessary current consumption in the chip and set the DAi register to "00_{16}" to prevent current from flowing into the R-2R resistor ladder.

Figure 1.17.2 DACON Register, DA0 Register and DA1 Register



i = 0, 1

Note: The above diagram shows an instance in which the DAi register is assigned "$2A_{16}$".

Figure 1.17.3 D-A Converter Equivalent Circuit

CRC Calculation

The Cyclic Redundancy Check (CRC) operation detects an error in data blocks. The microcomputer uses a generator polynomial of CRC-CCITT ($X^{16} + X^{12} + X^{5} + 1$) to generate CRC code.
The CRC code consists of 16 bits which are generated for each data block in given length, separated in 8-bit unit. After the initial value is set in the CRCD register, the CRC code is set in that register each time one byte of data is written to the CRCIN register. CRC code generation for one-byte data is finished in two cycles.
Figure 1.18.1 shows the block diagram of the CRC circuit. Figure 1.18.2 shows the CRC-related registers. Figure 1.18.3 shows the calculation example using the CRC operation.

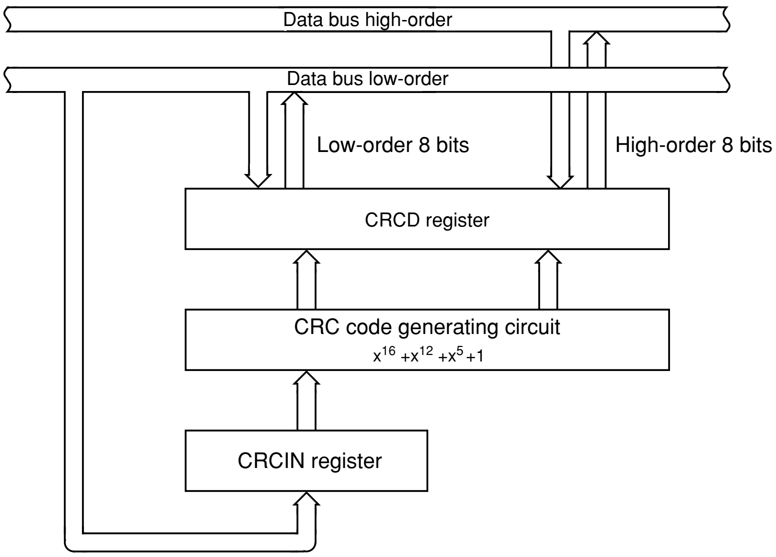


Figure 1.18.1 CRC Circuit Block Diagram

CRC data register

(b15) b7 (b8) b0 b7 b0

Symbol	Address	After reset
CRCD	$03BD_{16}$-$03BC_{16}$	Indeterminate

Function	Setting range	RW
When data is written to the CRCIN register after setting the initial value in the CRCD register, the CRC code can be read out from the CRCD register.	0000_{16} to $FFFF_{16}$	RW

CRC input register

b7 b0

Symbol	Address	After reset
CRCIN	$03BE_{16}$	Indeterminate

Function	Setting range	RW
Data input	00_{16} to FF_{16}	RW

Figure 1.18.2 CRCD Register and CRCIN Register

Setup procedure and CRC operation when generating CRC code "$80C4_{16}$"

(a) CRC operation performed by the M16C

CRC code: Remainder of a division in which the value written to the CRCIN register with its bit positions reversed is divided by the generator polynomial

Generator polynomial: $X^{16} + X^{12} + X^{5} + 1$ ($1\ 0001\ 0000\ 0010\ 0001_2$)

(b) Setting procedure

(1) Reverse the bit positions of the value "$80C4_{16}$" bytewise in a program.
"80_{16}" → "01_{16}", "$C4_{16}$" → "23_{16}"

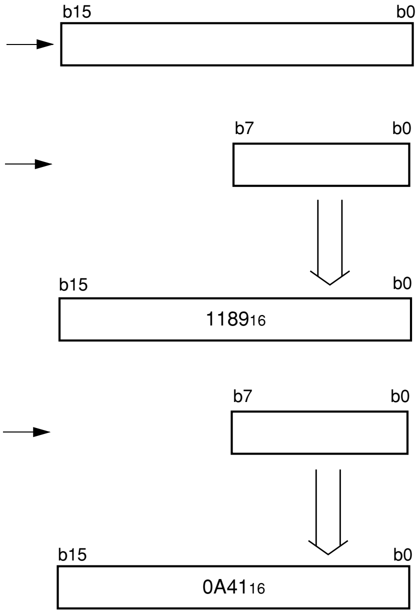
(2) Write 0000_{16} (initial value) → CRCD register

(3) Write 01_{16} → CRCIN register

Two cycles later, the CRC code for "80_{16}," i.e., 9188_{16}, has its bit positions reversed to become "1189_{16}" which is stored in the CRCD register.

(4) Write 23_{16} → CRCIN register

Two cycles later, the CRC code for "$80C4_{16}$," i.e., 8250_{16}, has its bit positions reversed to become "$0A41_{16}$" which is stored in the CRCD register.



(c) Details of CRC operation

In the case of (3) above, the value written to the CRCIN register "01_{16} (00000001_2)" has its bit positions reversed to become "10000000_2". The value "$1000\ 0000\ 0000\ 0000\ 0000\ 0000_2$" derived from that by adding 16 digits and the CRCD register s initial value "0000_{16}" are added, the result of which is divided by the generator polynomial using modulo-2 arithmetic.

```
                                              1000  1000
1 0001 0000 0010 0001 ) 1000  0000  0000  0000  0000  0000   <- Data
                        1000  1000  0001  0000  1
                              1000  0001  0000  1000  0
                              1000  1000  0001  0000  1
                                    1001  0001  1000  1000   <- CRC code
Generator polynomial
```

Modulo-2 operation is operation that complies with the law given below.

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0
-1 = 1

The value "$0001\ 0001\ 1000\ 1001_2$ (1189_{16})" derived from the remainder "$1001\ 0001\ 1000\ 1000_2$ (9188_{16})" by reversing its bit positions may be read from the CRCD register.

If operation (4) above is performed subsequently, the value written to the CRCIN register "23_{16} (00100011_2)" has its bit positions reversed to become "11000100_2". The value "$1100\ 0100\ 0000\ 0000\ 0000\ 0000_2$" derived from that by adding 16 digits and the remainder in (3) "$1001\ 0001\ 1000\ 1000_2$" which is left in the CRCD register are added, the result of which is divided by the generator polynomial using modulo-2 arithmetic.
The value "$0000\ 1010\ 0100\ 0001_2$ ($0A41_{16}$)" derived from the remainder by reversing its bit positions may be read from the CRCD register.

Figure 1.18.3 CRC Calculation

CAN Module

The CAN (Controller Area Network) module for the M16C/6N4 group of microcomputers is a communication controller implementing the CAN 2.0B protocol as defined in the BOSCH specification. The M16C/6N4 group contains two CAN modules which can transmit and receive messages in both standard (11-bit) ID and extended (29-bit) ID formats.
Figure 1.19.1 shows a block diagram of the CAN module.
External CAN bus driver and receiver are required.

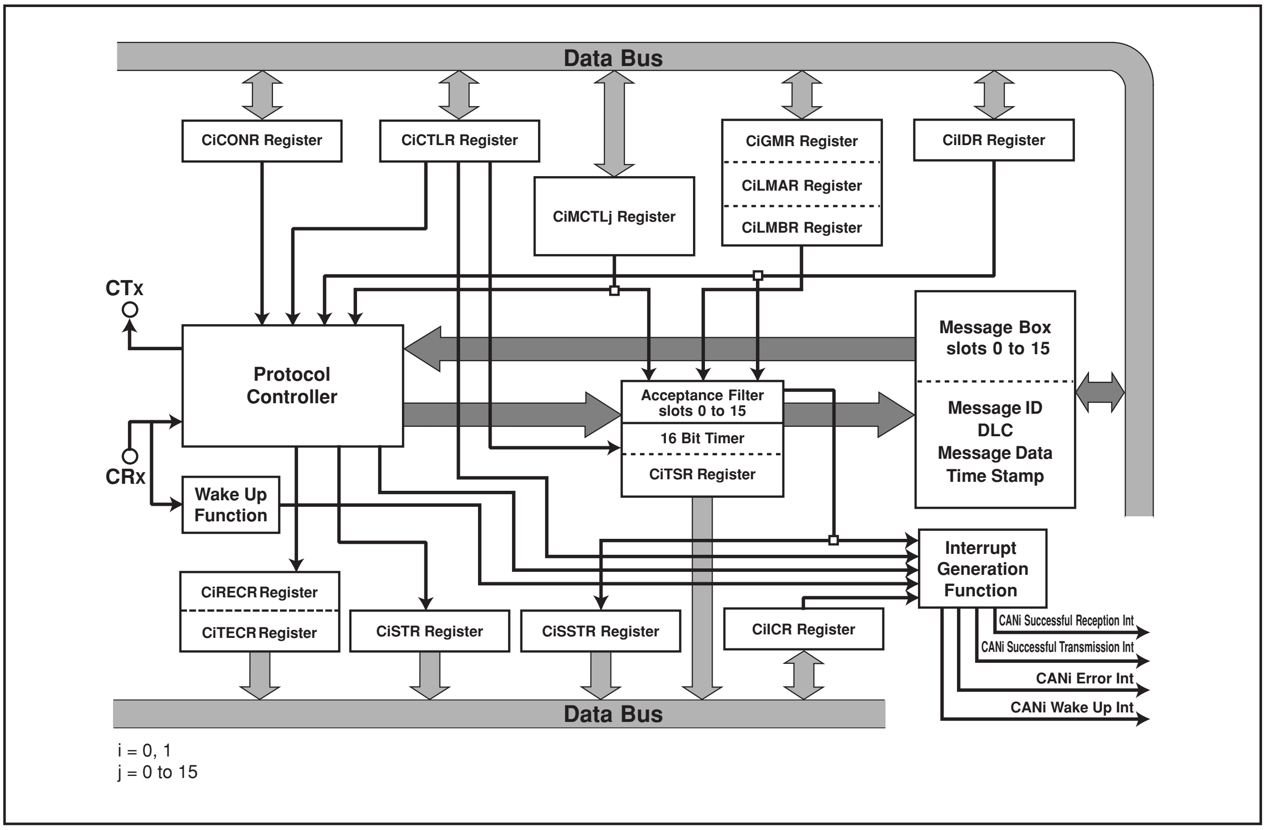


Figure 1.19.1 Block Diagram of CAN Module

CTx/CRx:	CAN I/O pins.
Protocol controller:	This controller handles the bus arbitration and the CAN protocol services, i.e. bit timing, stuffing, error status etc.
Message box:	This memory block consists of 16 slots that can be configured either as transmitter or receiver. Each slot contains an individual ID, data length code, a data field (8 bytes) and a time stamp.
Acceptance filter:	This block performs filtering operation for received messages. For the filtering operation, the CiGMR register (i = 0, 1), the CiLMAR register, or the CiLMBR register is used.
16 bit timer:	Used for the time stamp function. When the received message is stored in the message memory, the timer value is stored as a time stamp.
Wake up function:	CAN0/1 wake up interrupt is generated by a message from the CAN bus.
Interrupt generation function:	The interrupt events are provided by the CAN module. CANi successful reception interrupt, CANi successful transmission interrupt, CAN0/1 error interrupt, and CAN0/1 wake up interrupt.

CAN Module-Related Registers

The CANi (i = 0, 1) modules have the following registers.

(1) CAN Message Box

A CAN module is equipped with 16 slots (16 bytes or 8 words each). Slots 14 and 15 can be used as Basic CAN.

- Priority of the slots: The smaller the number of the slot, the higher the priority, in both transmission and reception.
- A program can define whether a slot is defined as transmitter or receiver.

(2) Acceptance Mask Registers

A CAN module is equipped with 3 masks for the acceptance filter.

- CANi global mask register (CiGMR register: 6 bytes)
  Configuration of the masking condition for acceptance filtering processing to slots 0 to 13
- CANi local mask A register (CiLMAR register: 6 bytes)
  Configuration of the masking condition for acceptance filtering processing to slot 14
- CANi local mask B register (CiLMBR register: 6 bytes)
  Configuration of the masking condition for acceptance filtering processing to slot 15

(3) CAN SFR Registers

- CANi message control register j (CiMCTLj register: 8 bits × 16) (j = 0 to 15)
  Control of transmission and reception of a corresponding slot
- CANi control register (CiCTLR register: 16 bits)
  Control of the CAN protocol
- CANi status register (CiSTR register: 16 bits)
  Indication of the protocol status
- CANi slot status register (CiSSTR register: 16 bits)
  Indication of the status of contents of each slot
- CANi interrupt control register (CiICR register: 16 bits)
  Selection of "interrupt enabled or disabled" for each slot
- CANi extended ID register (CiIDR register: 16 bits)
  Selection of ID format (standard or extended) for each slot
- CANi configuration register (CiCONR register: 16 bits)
  Configuration of the bus timing
- CANi receive error count register (CiRECR register: 8 bits)
  Indication of the error status of the CAN module in reception: the counter value is incremented or decremented according to the error occurrence.
- CANi transmit error count register (CiTECR register: 8 bits)
  Indication of the error status of the CAN module in transmission: the counter value is incremented or decremented according to the error occurrence.
- CANi time stamp register (CiTSR register: 16 bits)
  Indication of the value of the time stamp counter
- CANi acceptance filter support register (CiAFS register: 16 bits)
  Decoding the received ID for use by the acceptance filter support unit

Explanation of each register is given below.

CANi Message Box (i = 0, 1)

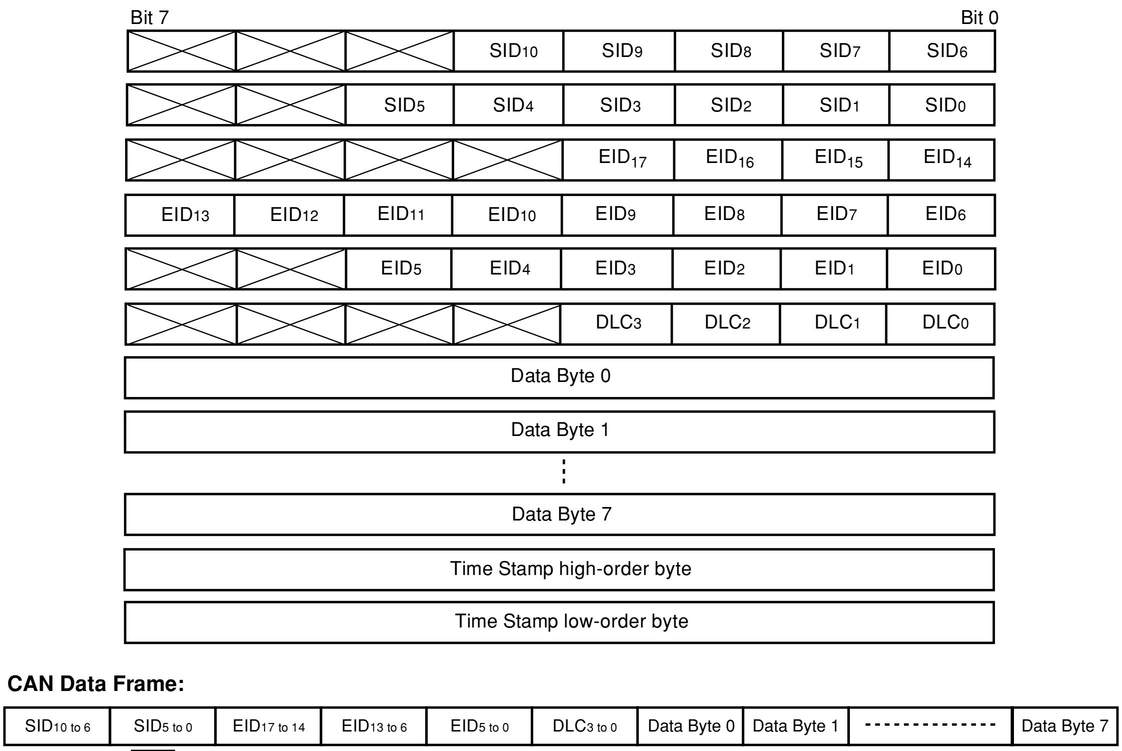
Table 1.19.1 shows the memory mapping of the CANi message box.
It is possible to access to the message box in byte or word.
Mapping of the message contents differs from byte access to word access. Byte access or word access can be selected by the MsgOrder bit of the CiCTLR register.

Table 1.19.1 Memory Mapping of CANi Message Box (n = 0 to 15: the number of the slot)

Address		Message content (Memory mapping)	
CAN0	CAN1	Byte access (8 bits)	Word access (16 bits)
0060_{16} + n •16 + 0	0260_{16} + n •16 + 0	SID_{10} to SID_6	SID_5 to SID_0
0060_{16} + n •16 + 1	0260_{16} + n •16 + 1	SID_5 to SID_0	SID_{10} to SID_6
0060_{16} + n •16 + 2	0260_{16} + n •16 + 2	EID_{17} to EID_{14}	EID_{13} to EID_6
0060_{16} + n •16 + 3	0260_{16} + n •16 + 3	EID_{13} to EID_6	EID_{17} to EID_{14}
0060_{16} + n •16 + 4	0260_{16} + n •16 + 4	EID_5 to EID_0	Data Length Code (DLC)
0060_{16} + n •16 + 5	0260_{16} + n •16 + 5	Data Length Code (DLC)	EID_5 to EID_0
0060_{16} + n •16 + 6	0260_{16} + n •16 + 6	Data byte 0	Data byte 1
0060_{16} + n •16 + 7 ⋮ 0060_{16} + n •16 + 13	0260_{16} + n •16 + 7 ⋮ 0260_{16} + n •16 + 13	Data byte 1 ⋮ Data byte 7	Data byte 0 ⋮ Data byte 6
0060_{16} + n •16 + 14	0260_{16} + n •16 + 14	Time stamp high-order byte	Time stamp low-order byte
0060_{16} + n •16 + 15	0260_{16} + n •16 + 15	Time stamp low-order byte	Time stamp high-order byte

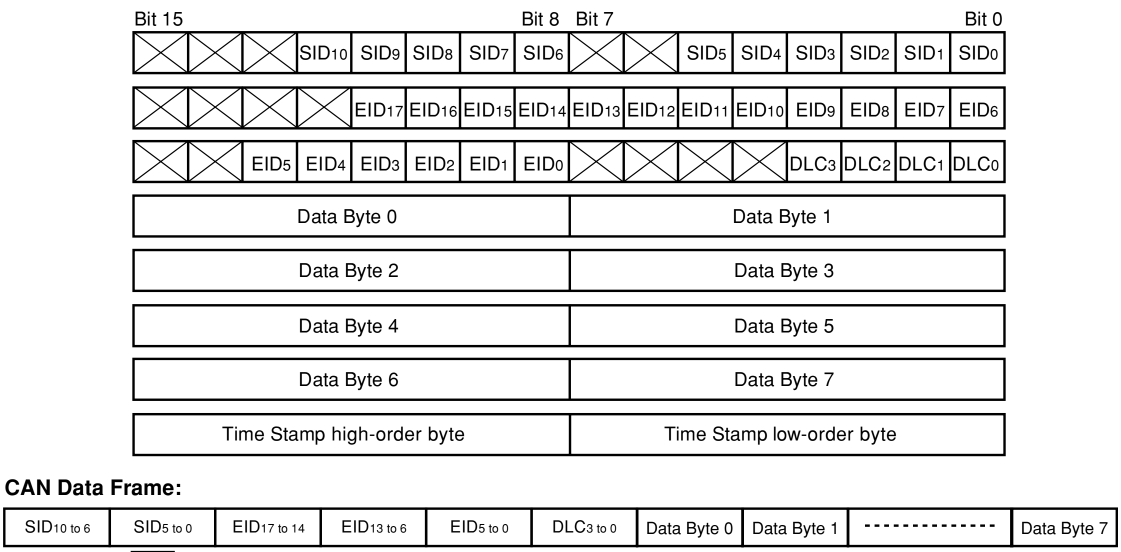
i = 0, 1

Figures 1.19.2 and 1.19.3 show the bit mapping in each slot in byte access and word access. The content of each slot remains unchanged unless transmission or reception of a new message is performed.



Note: When ☒ is read, the value is the one written upon the transmission slot configuration.
The value is "0" when read on the reception slot configuration.

Figure 1.19.2 Bit Mapping in Byte Access



Note: When ☒ is read, the value is the one written upon the transmission slot configuration.
The value is "0" when read on the reception slot configuration.

Figure 1.19.3 Bit Mapping in Word Access

Acceptance Mask Registers

Figures 1.19.4 and 1.19.5 show the CiGMR register (i = 0, 1), the CiLMAR register, and the CiLMBR register, in which bit mapping in byte access and word access are shown.

Bit 7							Bit 0	CAN0 address	CAN1 address	Register
			SID10	SID9	SID8	SID7	SID6	0160_{16}	0360_{16}	CiGMR register
		SID5	SID4	SID3	SID2	SID1	SID0	0161_{16}	0361_{16}	CiGMR register
				EID17	EID16	EID15	EID14	0162_{16}	0362_{16}	CiGMR register
EID13	EID12	EID11	EID10	EID9	EID8	EID7	EID6	0163_{16}	0363_{16}	CiGMR register
		EID5	EID4	EID3	EID2	EID1	EID0	0164_{16}	0364_{16}	CiGMR register
			SID10	SID9	SID8	SID7	SID6	0166_{16}	0366_{16}	CiLMAR register
		SID5	SID4	SID3	SID2	SID1	SID0	0167_{16}	0367_{16}	CiLMAR register
				EID17	EID16	EID15	EID14	0168_{16}	0368_{16}	CiLMAR register
EID13	EID12	EID11	EID10	EID9	EID8	EID7	EID6	0169_{16}	0369_{16}	CiLMAR register
		EID5	EID4	EID3	EID2	EID1	EID0	$016A_{16}$	$036A_{16}$	CiLMAR register
			SID10	SID9	SID8	SID7	SID6	$016C_{16}$	$036C_{16}$	CiLMBR register
		SID5	SID4	SID3	SID2	SID1	SID0	$016D_{16}$	$036D_{16}$	CiLMBR register
				EID17	EID16	EID15	EID14	$016E_{16}$	$036E_{16}$	CiLMBR register
EID13	EID12	EID11	EID10	EID9	EID8	EID7	EID6	$016F_{16}$	$036F_{16}$	CiLMBR register
		EID5	EID4	EID3	EID2	EID1	EID0	0170_{16}	0370_{16}	CiLMBR register

i = 0, 1

Figure 1.19.4 Bit Mapping of Mask Registers in Byte Access

Bit 15							Bit 8	Bit 7							Bit 0	CAN0 address	CAN1 address	Register
			SID10	SID9	SID8	SID7	SID6			SID5	SID4	SID3	SID2	SID1	SID0	0160_{16}	0360_{16}	CiGMR register
				EID17	EID16	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	EID7	EID6	0162_{16}	0362_{16}	CiGMR register
		EID5	EID4	EID3	EID2	EID1	EID0									0164_{16}	0364_{16}	CiGMR register
			SID10	SID9	SID8	SID7	SID6			SID5	SID4	SID3	SID2	SID1	SID0	0166_{16}	0366_{16}	CiLMAR register
				EID17	EID16	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	EID7	EID6	0168_{16}	0368_{16}	CiLMAR register
		EID5	EID4	EID3	EID2	EID1	EID0									$016A_{16}$	$036A_{16}$	CiLMAR register
			SID10	SID9	SID8	SID7	SID6			SID5	SID4	SID3	SID2	SID1	SID0	$016C_{16}$	$036C_{16}$	CiLMBR register
				EID17	EID16	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	EID7	EID6	$016E_{16}$	$036E_{16}$	CiLMBR register
		EID5	EID4	EID3	EID2	EID1	EID0									0170_{16}	0370_{16}	CiLMBR register

i = 0, 1

Figure 1.19.5 Bit Mapping of Mask Registers in Word Access

CAN SFR Registers

CiMCTLj Register (i = 0, 1, j = 0 to 15)

Figure 1.19.6 shows the CiMCTLj register.

CANi message control register j (i = 0, 1, j = 0 to 15) (Note 4)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
C0MCTL0 to C0MCTL15	0200_{16} to $020F_{16}$	00_{16}
C1MCTL0 to C1MCTL15	0220_{16} to $022F_{16}$	00_{16}

Bit symbol	Bit name	Function (Note 4)	RW
NewData	Successful reception flag	When set to reception slot 0: The content of the slot is read or still under processing by the CPU. 1 The CAN module has stored new data in the slot.	RO (Note 1)
SentData	Successful transmission flag	When set to transmission slot 0: Transmission is not started or completed yet. 1: Transmission is successfully completed.	RO (Note 1)
InvalData	"Under reception" flag	When set to reception slot 0: The message is valid. 1: The message is invalid. (The message is being updated.)	RO
TrmActive	"Under transmission" flag	When set to transmission slot 0: Waiting for bus idle or completion of arbitration. 1: Transmitting	RO
MsgLost	Overwrite flag	When set to reception slot 0: No message has been overwritten in this slot. 1: This slot already contained a message, but it has been overwritten by a new one.	RO (Note 1)
RemActive	Remote frame transmission/ reception status flag (Note 2)	0: Data frame transmission/reception status 1: Remote frame automatic transfer status	RW
RspLock	Transmission/ reception auto response lock mode select bit	When set to reception remote frame slot 0: After a remote frame is received, it will be answered automatically. 1: After a remote frame is received, no transmission will be started as long as this bit is set to "1". (Not responding)	RW
Remote	Remote frame corresponding slot select bit	0: Slot not corresponding to remote frame 1: Slot corresponding to remote frame	RW
RecReq	Reception slot request bit (Note 3)	0: Not reception slot 1: Reception slot	RW
TrmReq	Transmission slot request bit (Note 3)	0: Not transmission slot 1: Transmission slot	RW

Note 1: As for write, only writing "0" is possible. The value of each bit is written when the CAN module enters the respective state.
Note 2: In Basic CAN mode, they serve as data format identification flag. Refer to "Basic CAN Mode" for more details.
Note 3: One slot cannot be defined as reception slot and transmission slot at the same time.
Note 4: This register can not be set in CAN reset/initialization mode of the CAN module.

Figure 1.19.6 CiMCTLj Register

CiCTLR Register (i = 0, 1)

Figure 1.19.7 shows the CiCTLR register.

CANi control register (i = 0, 1)

Symbol	Address	After reset
C0CTLR	0210_{16}	$X0000001_2$
C1CTLR	0230_{16}	$X0000001_2$

Bit symbol	Bit name	Function	RW
Reset	CAN module reset bit	0: Operation mode 1: Reset/initialization mode	RW
LoopBack	Loop back mode select bit	0: Normal operation mode 1: Loop back mode	RW
MsgOrder	Message order select bit	0: Word access 1: Byte access	RW
BasicCAN	Basic CAN mode select bit	0: Normal operation mode 1: Basic CAN mode	RW
BusErrEn	Bus error interrupt enable bit	0: Bus error interrupt disabled 1: Bus error interrupt enabled	RW
Sleep	Sleep mode select bit	0: Sleep mode disabled 1: Sleep mode enabled; clock supply stopped	RW
PortEn	CAN port enable bit	0: I/O port function 1: CTx/CRx function (Note)	RW
— (b7)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—

Note: CTx/CRx function regardless of configuration of PD7 and PD9 registers.

(b15) ... (b8)

Symbol	Address	After reset
C0CTLR	0211_{16}	$XX0X0000_2$
C1CTLR	0231_{16}	$XX0X0000_2$

Bit symbol	Bit name	Function	RW
TSPreScale Bit1, Bit0	Time stamp prescaler	b1 b0 0 0: Period of 1 bit time 0 1: Period of 1/2 bit time 1 0: Period of 1/4 bit time 1 1: Period of 1/8 bit time	RW
TSReset	Time stamp counter reset bit (Note 1)	0: Normal operation mode 1: Force reset of the time stamp counter	RW
RetBusOff	Return from bus off command bit (Note 2)	0: Normal operation mode 1: Force return from bus off	RW
— (b4)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—
RXOnly	Listen-only mode select bit	0: Normal operation mode 1: Listen-only mode	RW
— (b7-b6)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—

Note 1: When the TSReset bit = 1, the CiTSR register is set to "0000_{16}". After this, the bit is automatically set to "0".
Note 2: When the RetBusOff bit = 1, the CiRECR register and the CiTECR register are set to "00_{16}". After this, the bit is automatically set to "0".

Figure 1.19.7 CiCTLR Register

CiSTR Register (i = 0, 1)

Figure 1.19.8 shows the CiSTR register.

CANi status register (i = 0, 1) (Note)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
C0STR	0212_{16}	00_{16}
C1STR	0232_{16}	00_{16}

Bit symbol	Bit name	Function	RW
MBOX	Active slot bits	b3 b2 b1 b0 0 0 0 0 : Slot 0 0 0 0 1 : Slot 1 0 0 1 0 : Slot 2 : 1 1 1 0 : Slot 14 1 1 1 1 : Slot 15	RO
TrmSucc	Successful transmission flag	0: No [successful] transmission 1: The CAN module has transmitted a message successfully.	RO
RecSucc	Successful reception flag	0: No [successful] reception 1: CAN module received a message successfully.	RO
TrmState	Transmission flag (Transmitter)	0: CAN module is idle or receiver. 1: CAN module is transmitter.	RO
RecState	Reception flag (Receiver)	0: CAN module is idle or transmitter. 1: CAN module is receiver.	RO

Note: This register can not be set in CAN reset/initialization mode of the CAN module.

(b15) b7 b6 b5 b4 b3 b2 b1 b0 (b8)

Symbol	Address	After reset
C0STR	0213_{16}	$X0000001_2$
C1STR	0233_{16}	$X0000001_2$

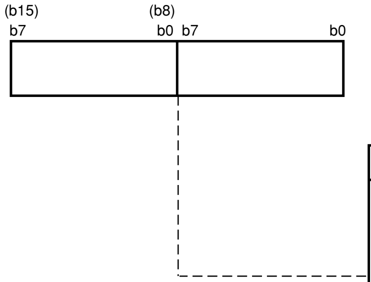
Bit symbol	Bit name	Function	RW
State_Reset	Reset state flag	0: Operation mode 1: Reset mode	RO
State_LoopBack	Loop back state flag	0: Normal operation mode 1: Loop back mode	RO
State_MsgOrder	Message order state flag	0:Word access 1: Byte access	RO
State_BasicCAN	Basic CAN mode state flag	0: Normal operation mode 1: Basic CAN mode	RO
State_BusError	Bus error state flag	0: No error has occurred. 1: A CAN bus error has occurred.	RO
State_ErrPass	Error passive state flag	0: The CAN module is not in error passive state. 1: The CAN module is in error passive state.	RO
State_BusOff	Error bus off state flag	0: The CAN module is not in error bus off state. 1: The CAN module is in error bus off state.	RO
— (b7)	Nothing is assigned. When write, set to "0". When read, its content is indeterminate.		—

Figure 1.19.8 CiSTR Register

CiSSTR Register (i = 0, 1)

Figure 1.19.9 shows the CiSSTR register.

CANi slot status register (i = 0, 1)



Symbol	Address	After reset
C0SSTR	0215_{16}, 0214_{16}	0000_{16}
C1SSTR	0235_{16}, 0234_{16}	0000_{16}

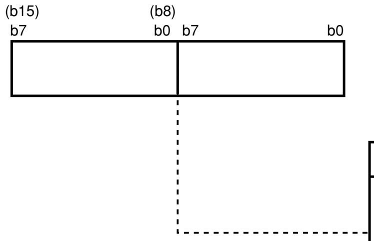
Function	Setting values	RW
Slot status bits Each bit corresponds to the slot with the same number.	0: Reception slot The message has been read. Transmission slot Transmission is not completed. 1: Reception slot The message has not been read. Transmission slot Transmission is completed.	RO

Figure 1.19.9 CiSSTR Register

CiICR Register (i = 0, 1)

Figure 1.19.10 shows the CiICR register.

CANi interrupt control register (i = 0, 1) (Note)



Symbol	Address	After reset
C0ICR	0217_{16}, 0216_{16}	0000_{16}
C1ICR	0237_{16}, 0236_{16}	0000_{16}

Function	Setting values	RW
Interrupt enable bits: Each bit corresponds with a slot with the same number. Enabled/disabled of successful transmission interrupt or successful reception interrupt can be selected.	0: Interrupt disabled 1: Interrupt enabled	RW

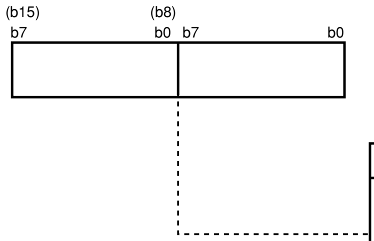
Note: This register can not be set in CAN reset/initialization mode of the CAN module.

Figure 1.19.10 CiICR Register

CiIDR Register

Figure 1.19.11 shows the CiIDR register.

CANi extended ID register (i = 0, 1) (Note)



Symbol	Address	After reset
C0IDR	0219_{16}, 0218_{16}	0000_{16}
C1IDR	0239_{16}, 0238_{16}	0000_{16}

Function	Setting values	RW
Extended ID bits: Each bit corresponds with a slot with the same number. Selection of the ID format that each slot handles.	0: Standard ID 1: Extended ID	RW

Note: This bit can not be set in CAN reset/initialization mode of the CAN module.

Figure 1.19.11 CiIDR Register

CiCONR Register (i = 0, 1)

Figure 1.19.12 shows the CiCONR register.

CANi configuration register (i = 0, 1)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
C0CONR	021A16	Indeterminate
C1CONR	023A16	Indeterminate

Bit symbol	Bit name	Function	RW
BRP	Prescaler division ratio select bits	b3 b2 b1 b0 0 0 0 0 : Divide-by-1 of fCAN 0 0 0 1 : Divide-by-2 of fCAN 0 0 1 0 : Divide-by-3 of fCAN ⋮ 1 1 1 0 : Divide-by-15 of fCAN 1 1 1 1 : Divide-by-16 of fCAN (Note)	RW
SAM	Sampling control bit	0 : One time sampling 1 : Three times sampling	RW
PTS	Propagation time segment control bits	b7 b6 b5 0 0 0 : 1Tq 0 0 1 : 2Tq 0 1 0 : 2Tq ⋮ 1 1 0 : 7Tq 1 1 1 : 8Tq	RW

Note: fCAN serves for the CAN clock. The period is decided by configuration of the CCLKi bits (i = 0 to 2 and 4 to 6) of CCLKR register.

(b15) b7 b6 b5 b4 b3 b2 b1 (b8) b0

Symbol	Address	After reset
C0CONR	021B16	Indeterminate
C1CONR	023B16	Indeterminate

Bit symbol	Bit name	Function	RW
PBS1	Phase buffer segment 1 control bits	b2 b1b0 0 0 0 : Inhibited 0 0 1 : 2Tq 0 1 0 : 3Tq ⋮ 1 1 0 : 7Tq 1 1 1 : 8Tq	RW
PBS2	Phase buffer segment 2 control bits	b5 b4 b3 0 0 0 : Inhibited 0 0 1 : 2Tq 0 1 0 : 3Tq ⋮ 1 1 0 : 7Tq 1 1 1 : 8Tq	RW
SJW	Resynchronization jump width control bits	b7 b6 0 0 : 1Tq 0 1 : 2Tq 1 0 : 3Tq 1 1 : 4Tq	RW

Figure 1.19.12 CiCONR Register

CiRECR Register (i = 0, 1)

Figure 1.19.13 shows the CiRECR register.

CANi receive error count register (i = 0, 1) (Note 2)

b7 b0

Symbol	Address	After reset
C0RECR	$021C_{16}$	00_{16}
C1RECR	$023C_{16}$	00_{16}

Function	Counter value	RW
Reception error counting function The value is incremented or decremented according to the CAN module's error status.	00_{16} to FF_{16} (Note 1)	RO

Note 1: The value is indeterminate in bus off state.
Note 2: This register can not be set in CAN reset/initialization mode of the CAN module.

Figure 1.19.13 CiRECR Register

CiTECR Register (i = 0, 1)

Figure 1.19.14 shows the CiTECR register.

CANi transmit error count register (i = 0, 1) (Note)

b7 b0

Symbol	Address	After reset
C0TECR	$021D_{16}$	00_{16}
C1TECR	$023D_{16}$	00_{16}

Function	Counter value	RW
Transmission error counting function The value is incremented or decremented according to the CAN module's error status.	00_{16} to FF_{16}	RO

Note: This register can not be set in CAN reset/initialization mode of the CAN module.

Figure 1.19.14 CiTECR Register

CiTSR Register (i = 0, 1)

Figure 1.19.15 shows the CiTSR register.

CANi time stamp register (i = 0, 1) (Note)

(b15) b7 (b8) b0 b7 b0

Symbol	Address	After reset
C0TSR	$021F_{16}$, $021E_{16}$	0000_{16}
C1TSR	$023F_{16}$, $023E_{16}$	0000_{16}

Function	Setting range	RW
Time stamp function	0000_{16} to $FFFF_{16}$	RO

Note: This register can not be set in CAN reset/initialization mode of the CAN module.

Figure 1.19.15 CiTSR Register

CiAFS Register (i = 0, 1)

Figure 1.19.16 shows the CiAFS register.

CANi acceptance filter support register (i = 0, 1)

(b15) b7 (b8) b0 b7 b0

Symbol	Address	After reset
C0AFS	0243_{16}, 0242_{16}	Indeterminate
C1AFS	0245_{16}, 0244_{16}	Indeterminate

Function	Setting values	RW
Write the content equivalent to the standard frame ID of the received message. The value is "converted standard frame ID" when read.	Standard frame ID	RW

Figure 1.19.16 CiAFS Register

Operational Modes

The CAN module has the following three operational modes.

- CAN Reset/Initialization Mode
- CAN Sleep Mode
- CAN Operation Mode

Figure 1.19.17 shows transition between operational modes.

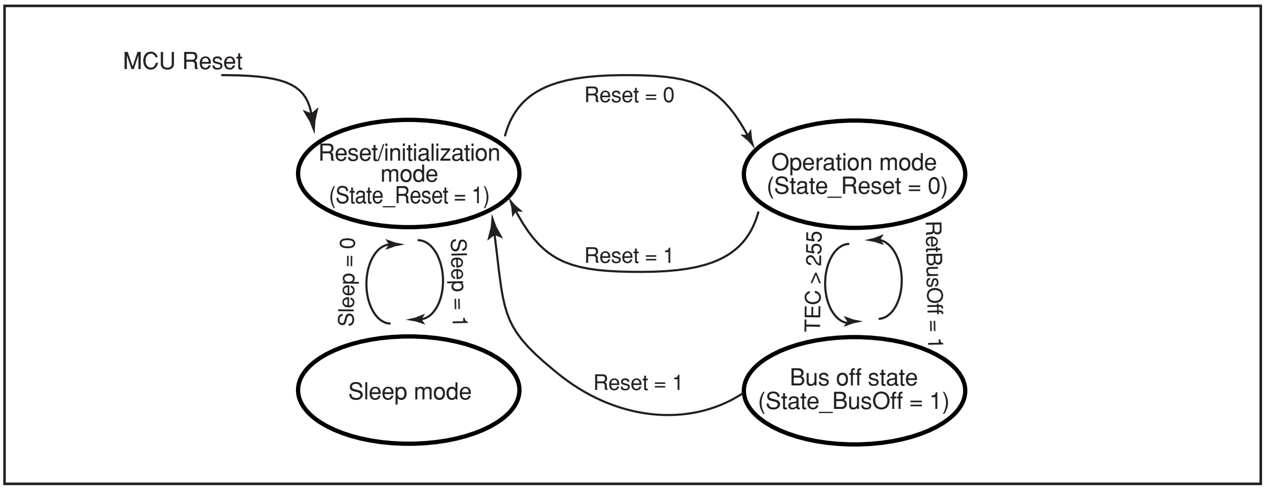


Figure 1.19.17 Transition Between Operational Modes

CAN Reset/Initialization Mode

The CAN reset/initialization mode is activated upon MCU reset or by setting the Reset bit of the CiCTLR register (i = 0, 1). It can be observed by reading the State_Reset bit of the CiSTR register. Entering the CAN reset/initialization mode initiates the following functions by the module:

- Suspend all communication functions. When the CAN reset/initialization mode is activated during an ongoing transmission in operation mode, the module suspends the mode transition until completion of the transmission (successful, arbitration loss, or error detection) and then sets the State_Reset bit.
- Initialization of CiMCTLj (j = 0 to 15), CiSTR, CiICR, CiIDR,CiRECR, CiTECR and CiTSR registers to their reset values. All these registers are locked to prevent CPU modification.
- The CiCTLR and CiCONR registers and the message box retain their contents and are available for CPU access.

CAN Operation Mode

The CAN operation mode is activated by clearing the Reset bit of the CiCTLR register. Entering the operation mode initiates the following functions by the module:

- The module's communication functions are released and it becomes an active node on the network and may transmit and receive CAN messages.
- Release the internal fault confinement logic including receive and transmit error counters. The module may leave the CAN operation mode depending on the error counts.

Within the CAN operation mode the module may be in three different sub modes, depending on which type of communication functions are performed:

- Module idle: The modules receive and transmit sections are inactive.
- Module receives: The module receives a CAN message sent by another node.
- Module transmits: The module transmits a CAN message. The module may receive its own message simultaneously when the loopback function is enabled.

Figure 1.19.18 shows sub modes of the CAN operation mode.

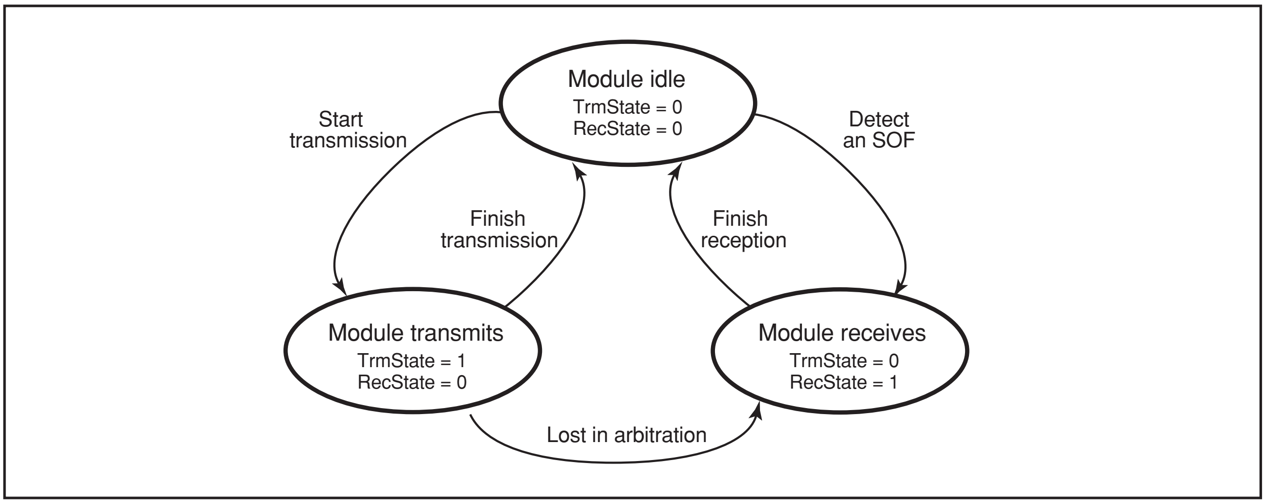


Figure 1.19.18 Sub Modes of CAN Operation Mode

CAN Sleep Mode

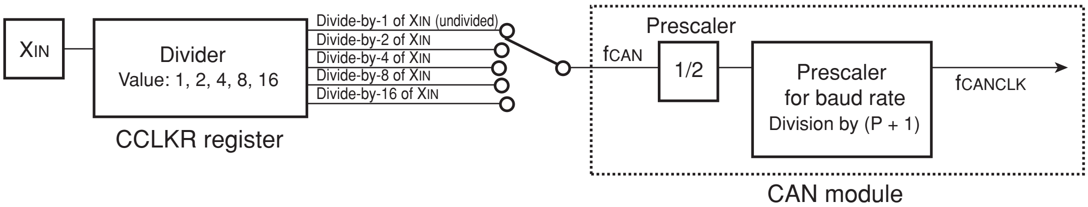
The CAN sleep mode is activated by setting the Sleep bit of the CiCTLR register. It should never be activated from the CAN operation mode but only via the CAN reset/initialization mode. Entering the CAN sleep mode instantly stops the modules clock supply and thereby reduces power dissipation.

Bus off State

The bus off state is entered according to the fault confinement rules of the CAN specification. It can be quit instantly to error active state by setting the RetBusOff bit of the CiCTLR register to “1” (force return from buss off) and CAN communication becomes possible again. This does not alter any CAN registers, except CiRECR and CiTECR registers.

Configuration of the CAN Module System Clock

The M16C/6N4 group has a CAN module system clock select circuit dedicated to each channel.
Configuration of the CAN module system clock can be done through manipulating the CCLKR register and the BRP bits of the CiCONR registers (i = 0, 1).
For the CCLKR register, refer to “Clock Generation Circuit”.
Figure 1.19.19 shows a block diagram of the clock generation circuit of the CAN module system.



i = 0, 1
f_{CAN}: CAN module system clock
P: The value written in the BRP bit of the CiCONR register. P = 0 to 15
f_{CANCLK}: CAN communication clock $f_{CANCLK} = f_{CAN}/2(P + 1)$

Figure 1.19.19 Block Diagram of CAN Module System Clock Generation Circuit

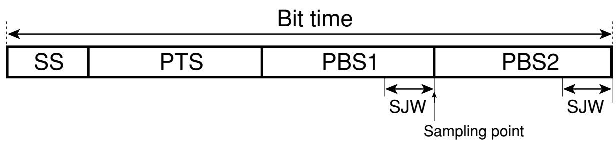
CAN Bus Timing Control

Bit Timing Configuration

The bit time consists of the following four segments:

- Synchronization segment (SS)
  This serves for monitoring a falling edge for synchronization.
- Propagation time segment (PTS)
  This segment absorbs physical delay on the CAN network which amounts to double the total sum of delay on the CAN bus, the input comparator delay, and the output driver delay.
- Phase buffer segment 1 (PBS1)
  This serves for compensating the phase error. When the falling edge of the bit falls later than expected, the segment can become longer by the maximum of the value defined in SJW.
- Phase buffer segment 2 (PBS2)
  This segment has the same function as the phase buffer segment 1. When the falling edge of the bit falls earlier than expected, the segment can become shorter by the maximum of the value defined in SJW.

Figure 1.19.20 shows the bit timing.



The range of each segment: Bit time = 8 to 25Tq
SS = 1Tq
PTS = 1Tq to 8Tq
PBS1 = 2Tq to 8Tq
PBS2 = 2Tq to 8Tq
SJW = 1Tq to 4Tq

Configuration of PBS1 and PBS2: PBS1 ≥ PBS2
PBS1 ≥ SJW
PBS2 ≥ 2 when SJW = 1
PBS2 ≥ SJW when 2 ≤ SJW ≤ 4

Figure 1.19.20. Bit Timing

Baud Rate

Baud rate depends on X_{IN}, the division value of the CAN module system clock, the division value of the prescaler for baud rate, and the number of Tq of one bit.
Table 1.19.2 shows the examples of baud rate.

Table 1.19.2 Examples of Baud Rate

Baud rate	20 MHz	16 MHz	10 MHz	8 MHz
1 Mbps	10Tq (1)	8Tq (1)	–	–
500 kbps	10Tq (2) 20Tq (1)	8Tq (2) 16Tq (1)	10Tq (1) –	8Tq (1) –
125 kbps	10Tq (8) 20Tq (4)	8Tq (8) 16Tq (4)	10Tq (4) 20Tq (2)	8Tq (4) 16Tq (2)
83.3 kbps	10Tq (12) 20Tq (6)	8Tq (12) 16Tq (6)	10Tq (6) 20Tq (3)	8Tq (6) 16Tq (3)
33.3 kbps	10Tq (30) 20Tq (15)	8Tq (30) 16Tq (15)	10Tq (15) –	8Tq (15) –

Note: The number in ( ) indicates a value of "f_{CAN} division value" multiplied by "division value of the prescaler for baud rate".

■ Calculation of Baud Rate

$$\frac{X_{IN}}{2 \times \text{"}f_{CAN}\text{ division value (Note 1)"} \times \text{"division value of prescaler for baud rate (Note 2)"} \times \text{"number of Tq of one bit"}}$$

Note 1: f_{CAN} division value = 1, 2, 4, 8, 16
f_{CAN} division value: a value selected in the CCLKR register
Note 2: Division value of prescaler for baud rate = P + 1 (P: 0 to 15)
P: a value selected in the BRP bit of the CiCONR register (i = 0, 1)

Acceptance Filtering Function and Masking Function

These functions serve the users to select and receive a facultative message. The CiGMR register (i =0, 1), the CiLMAR register, and the CiLMBR register can perform masking to the standard ID and the extended ID of 29 bits. The CiGMR register corresponds to slots 0 to 13, the CiLMAR register corresponds to slot 14, and the CiLMBR register corresponds to slot 15. The masking function becomes valid to 11 bits or 29 bits of a received ID according to the value in the corresponding slot of the CiIDR register upon acceptance filtering operation. When the masking function is employed, it is possible to receive a certain range of IDs. Figure 1.19.21 shows correspondence of the mask registers and slots, Figure 1.19.22 shows the acceptance function.

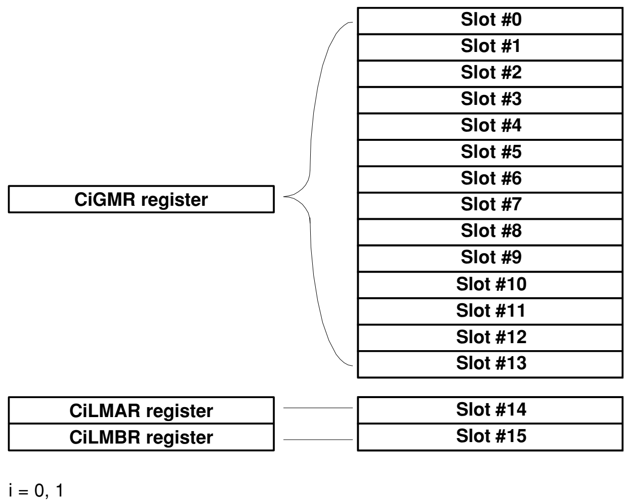


Figure 1.19.21 Correspondence of Mask Registers to Slots

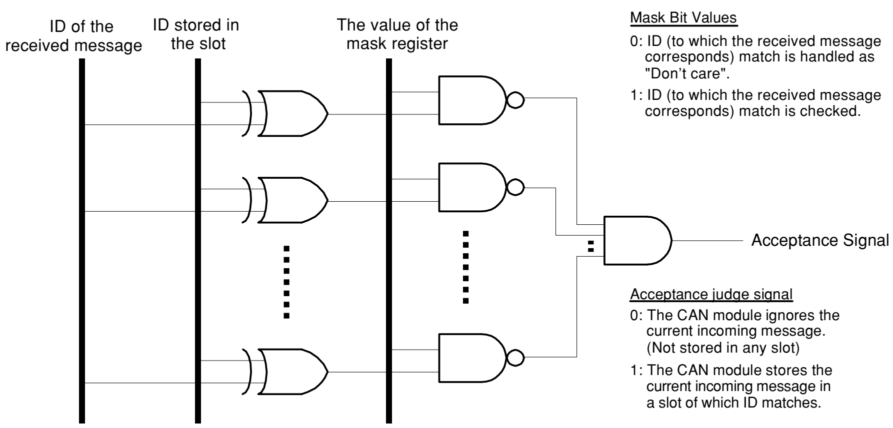


Figure 1.19.22 Acceptance Function

When using the acceptance function, note the following points.

(1) When one ID is defined in two slots, the one with a smaller number alone is valid.

(2) When it is configured that slots 14 and 15 receive all IDs with Basic CAN mode, slots 14 and 15 receive all IDs which are not stored into slots 0 to 13.

Acceptance Filter Support Unit (ASU)

The acceptance filter support unit has a function to judge valid/invalid of a received ID through table search. The IDs to receive are registered in the data table; a received ID is stored in the CiAFS register (i = 0, 1), and table search is performed with a decoded received ID. The acceptance filter support unit can be used for the IDs of the standard frame only.

The acceptance filter support unit is valid in the following cases.

- When the ID to receive cannot be masked by the acceptance filter.
  (Example) IDs to receive: 078_{16}, 087_{16}, 111_{16}
- When there are too many IDs to receive; it would take too much time to filter them by software.

Figure 1.19.23 shows the write and read of CiAFS register in word access.

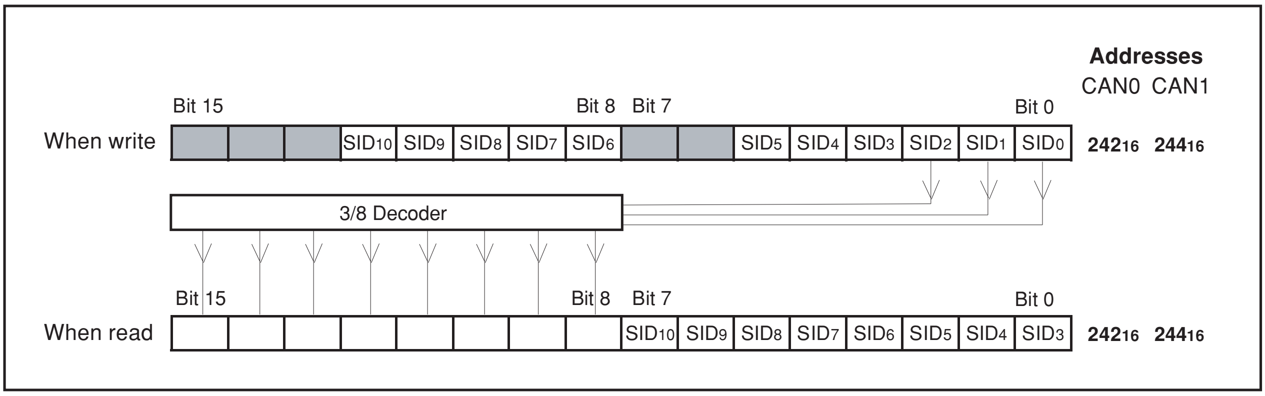


Figure 1.19.23 Write/read of CiAFS Register in Word Access

Basic CAN Mode

When the BasicCAN bit of the CiCTLR register (i = 0, 1) is set to "1", slots 14 and 15 correspond to Basic CAN mode. When slots 14 and 15 are defined as reception slots in Basic CAN mode, received messages are stored in slots 14 and 15 alternately.

Figure 1.19.24 shows the operation of slots 14 and 15 in Basic CAN mode.

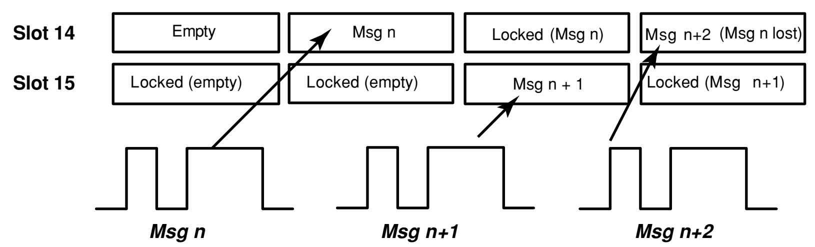


Figure 1.19.24 Operation of Slots 14 and 15 in Basic CAN Mode

When configuring Basic CAN mode, note the following points.

(1) Selection of Basic CAN mode has to be done in reset/initialization mode.

(2) Select the same ID for slots 14 and 15. Also, configuration of the CiLMAR register and that of the CiLMBR register has to be the same.

(3) Define slots 14 and 15 as reception slot only.

(4) There is no protection available against message overwrite. A message can be overwritten by a new message.

(5) Slots 0 to 13 can be used in the same way as in normal CAN operation mode.

Return from Bus off Function

When the protocol controller enters bus off state, it is possible to make it forced return from bus off state by the return from bus off function of the CiCTLR register (i = 0, 1). At this time, the error state changes from bus off state to error active state. Implementation of this function initializes the protocol controller. However, registers of the CAN module such as CiCONR register and the content of each slot are not initialized.

Time Stamp Counter and Time Stamp Function

When the CiTSR register is read, the value of the time stamp counter at the moment is read. The period of the time stamp counter reference clock is the same as that of 1 bit time that is configured by the CiCONR register. The time stamp counter functions as a free run counter.
The 1 bit time period can be divided by 1 (undivided), 2, 4 or 8 to produce the time stamp counter reference clock. Use the TSPreScale bits 1 and 0 of the CiCTLR register to select the divide-by-n value.
The time stamp counter is equipped with a register that captures the counter value when the protocol controller regards it as a successful reception. The captured value is stored when a time stamp value is stored in a reception slot.

Listen-Only Mode

When the RXOnly bit of the CiCTLR register is set to "1", the module enters listen-only mode.
In listen-only mode, no transmission -- data frames, error frames, and ACK response -- is performed to bus.

Reception and Transmission

Configuration of CAN Reception and Transmission Mode

Table 1.19.3 shows configuration of CAN reception and transmission mode.

Table 1.19.3 Configuration of CAN Reception and Transmission Mode

TrmReq	RecReq	Remote	RspLock	Communication mode of the slot
0	0	——	——	Communication environment configuration mode: configure the communication mode of the slot.
0	1	0	0	Configured as a reception slot for a data frame.
1	0	1	0	Configured as a transmission slot for a remote frame. (At this time the RemActive bit is "1".) After completion of transmission, this functions as a reception slot for a data frame. (At this time the RemActive bit is "0".) However, when an ID that matches on the CAN bus is detected before remote frame transmission, this immediately functions as a reception slot for a data frame.
1	0	0	0	Configured as a transmission slot for a data frame.
0	1	1	1/0	Configured as a reception slot for a remote frame. (At this time the RemActive bit is "1".) After completion of reception, this functions as a transmission slot for a data frame. (At this time the RemActive bit is "0".) However, transmission does not start as long as RspLock bit remains "1"; thus no automatic remote frame response. Response (transmission) starts when RspLock bit is set to "0".

RemActive bit, RspLock bit: CiMCTLj register's bits (i = 0, 1, j = 0 to 15)

When configuring a slot as a reception slot, note the following points.

(1) Before configuring a slot as a reception slot, be sure to set the CiMCTLj registers (i = 0, 1, j = 0 to 15) to "00_{16}".

(2) A received message is stored in a slot that matches the condition first according to the result of reception mode configuration and acceptance filtering operation. Upon deciding in which slot to store, the smaller the number of the slot is, the higher priority it has.

(3) In normal CAN operation mode, when a CAN module transmits a message of which ID matches, the CAN module never receives the transmitted data. In loop back mode, however, the CAN module receives back the transmitted data. In this case, the module does not return ACK.

When configuring a slot as a transmission slot, note the following points.

(1) Before configuring a slot as a transmission slot, be sure to set the CiMCTLj registers to "00_{16}".

(2) Set the TrmReq bit in the CiMCTLj register to "0" (not transmission slot) before rewriting a transmission slot.

(3) A transmission slot should not be rewritten when the TrmActive bit in the CiMCTLj register is "1" (transmitting).
If it is rewritten, an indeterminate data will be transmitted.

Reception

Figure 1.19.25 shows the behavior of the module when receiving two consecutive CAN messages, that fit into the slot of the shown CiMCTLj register (i =0, 1, j = 0 to 15) and leads to losing/overwriting of the first message.

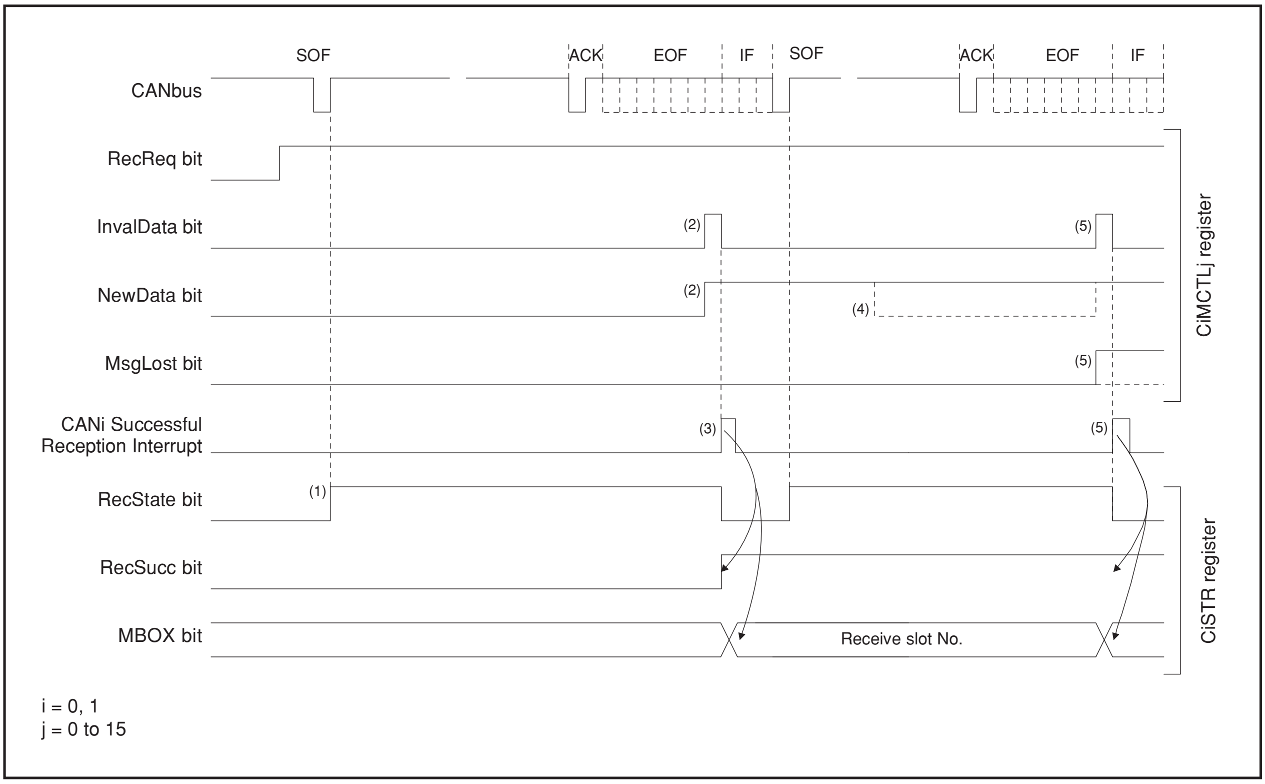


Figure 1.19.25 Timing of Receive Data Frame Sequence

1) On monitoring a SOF on the CAN bus the RecState bit in the CiSTR register (i = 0, 1) becomes “1” (CAN module is receiver) immediately, given the module has no transmission pending (refer to “Transmission”).
2) After successful reception of the message the NewData bit in the CiMCTLj register (j = 0 to 15) of the receiving slot becomes “1” (stored new data in slot). The InvalData bit in the CiMCTLj register becomes “1” (message is being updated) at the same time and the InvalData bit becomes “0” (message is valid) again after the complete message was transferred to the slot.
3) When the interrupt enable bit in the CiICR register of the receiving slot = 1 (interrupt enabled), the successful reception interrupt request is occurred and the MBOX bit in the CiSTR register changes. It shows the slot number where the message was stored and the RecSucc bit in the CiSTR register is active.
4) After reading out the message out of the slot, the CPU should set the New Data bit to “0” (the content of the slot is read or still under processing by the CPU).
5) If the NewData bit is not set to “0” by the CPU and the Receive request for the slot is not disabled before the next successful reception of a CAN message that is fitting in this slot the MsgLost bit in the CiMCTLj register becomes “1” (message has been overwritten). The new received message is transferred to the slot. The interrupt request and change of the CiSTR register is same as in 3).

Transmission

Figure 1.19.26 shows the timing of the transmit sequence.

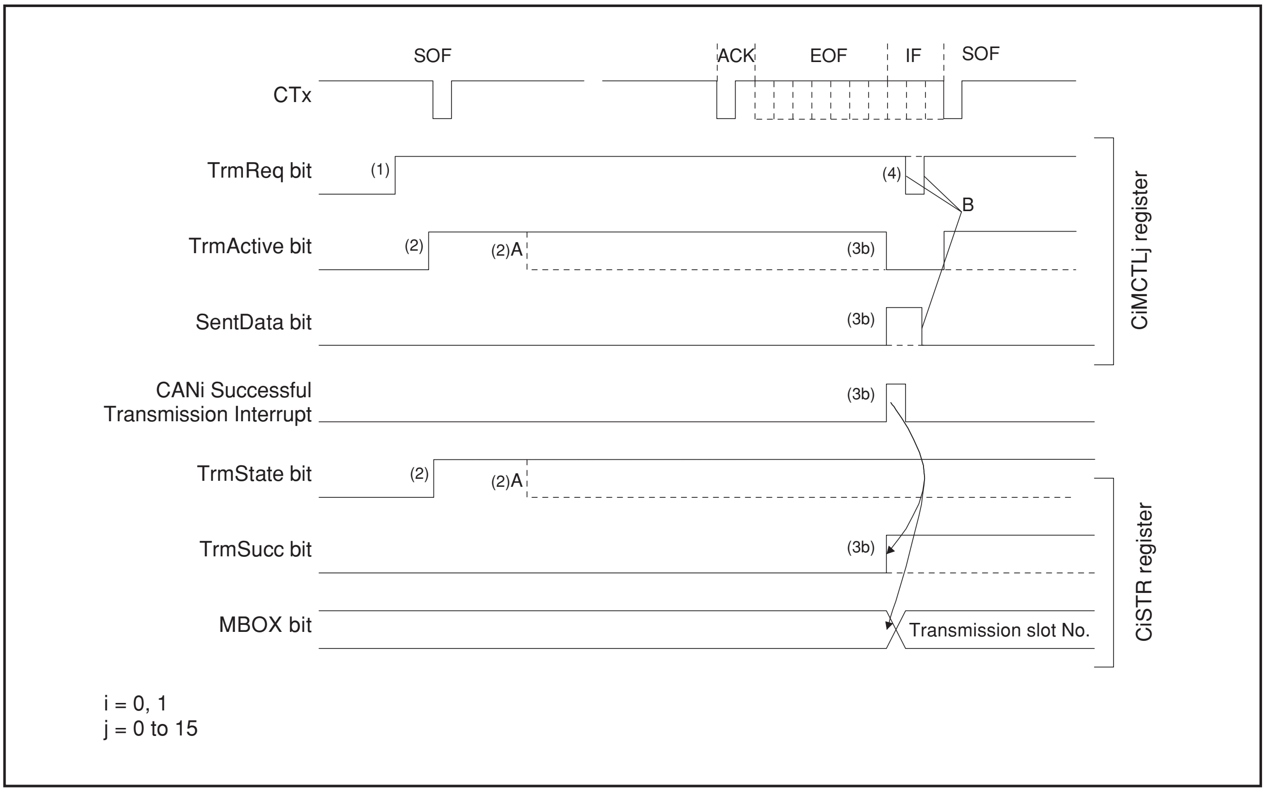


Figure 1.19.26 Timing of Transmit Sequence

1) If one or more of the slots of a module has a request for transmission, the module attempts to start the transmission at the next possible time (depending on the bus condition).
2) The TrmActive bit in the CiMCTLj register (i = 0, 1, j = 0 to 15) of the lowest slot with transmit request is set to "1" (transmitting). Also the TrmState bit in the CiSTR register is set to "1" (transmitter). If the arbitration is lost against another CAN node both bits are set to "0" (idle) again (A).
3a) When the arbitration was won, but the transmission was not successful;
The module will attempt to re-transmit.
3b) When the arbitration was won and the transmission has been successful;
The SentData bit in the CiMCTLj register is set to "1" (transmission is successfully completed) and TrmSucc bit in the CiSTR register is set to "1" (transmitted a message successfully). If the according interrupt enable bit in the CiICR register is "1", the successful transmission interrupt request is occurred. The number of the slot that was transmitted can be found in MBOX bit in the CiSTR register.
4) After a successful transmission, the module will not attempt to send the slot again until it is reactivated. To reactivate a slot for transmission, first the TrmReq bit in the CiMCTLj register has to be set to "0" (not transmission slot). Then the Sent Data bit in the CiMCTLj register can be set to "0" (transmission is not started or completed yet) and the TrmReq bit is can be set to "1" (transmission slot) again (B). Note that the SentData bit is locked and cannot be set to "0" as long as TrmReq bit =1.

CAN Interrupts

The CAN module provides the following CAN interrupts.

- CAN0 Successful Reception Interrupt
- CAN0 Successful Transmission Interrupt
- CAN1 Successful Reception Interrupt
- CAN1 Successful Transmission Interrupt
- CAN0/1 Error Interrupt
  - Error Passive State
  - Error BusOff State
  - Bus Error (this feature can be disabled separately)
- CAN0/1 Wake Up Interrupt

When the CPU detects a successful reception/transmission interrupt request, the MBOX bit in the CiSTR register (i = 0, 1) must be read to determine which slot has generated the interrupt request.

Programmable I/O Ports

The programmable input/output ports (hereafter referred to simply as "I/O ports") consist of 87 lines P0 to P10 (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 1.20.1 to 1.20.5 show the I/O ports. Figure 1.20.6 shows the I/O pins.

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

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 or D-A converter output pin, set the direction bit for that pin to "0" (input mode). Any pin used as an output pin for peripheral functions other than the D-A converter is directed for output no matter how the corresponding direction bit is set.

When using any pin as a bus control pin, refer to "Bus Control."

(1) Port Pi Direction Register (PDi Register, i = 0 to 10)

Figure 1.20.7 shows the PDi register.

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.

During memory expansion and microprocessor modes, the PDi registers for the pins functioning as bus control pins (A_0 to A_{19}, D_0 to D_{15}, CS_0 to CS_3, $\overline{RD}$, $\overline{WRL}/\overline{WR}$, $\overline{WRH}/\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$, $\overline{HLDA}$, and BCLK) cannot be modified.

No direction register bit for $P8_5$ is available.

(2) Port Pi Register (Pi Register, i = 0 to 10)

Figure 1.20.8 shows the Pi register.

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.

During memory expansion and microprocessor modes, the PDi registers for the pins functioning as bus control pins (A_0 to A_{19}, D_0 to D_{15}, CS_0 to CS_3, $\overline{RD}$, $\overline{WRL}/\overline{WR}$, $\overline{WRH}/\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$, $\overline{HLDA}$, and BCLK) cannot be modified.

(3) Pull-up Control Register j (PURj Register, j = 0 to 2)

Figure 1.20.9 shows the PURj register.

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

However, the pull-up control register has no effect on P0 to P3, $P4_0$ to $P4_3$, and P5 during memory expansion and microprocessor modes. Although the register contents can be modified, no pull-up resistors are connected.

(4) Port Control Register (PCR Register)

Figure 1.20.10 shows the PCR 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.

Tables 1.20.1 and 1.20.2 list an example connection of unused pins. Figure 1.20.11 shows an example connection of unused pins.

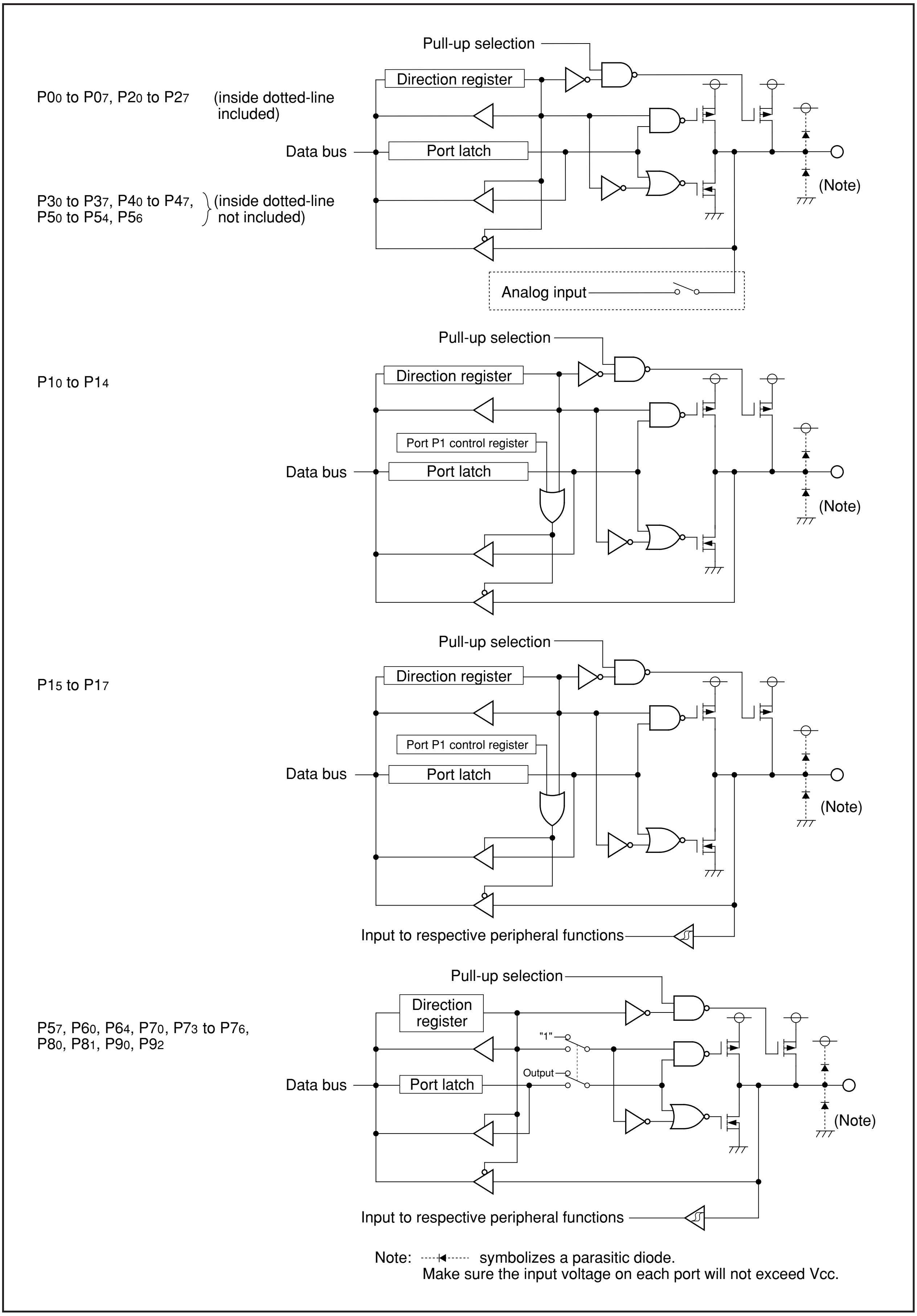


Figure 1.20.1 I/O Ports (1)

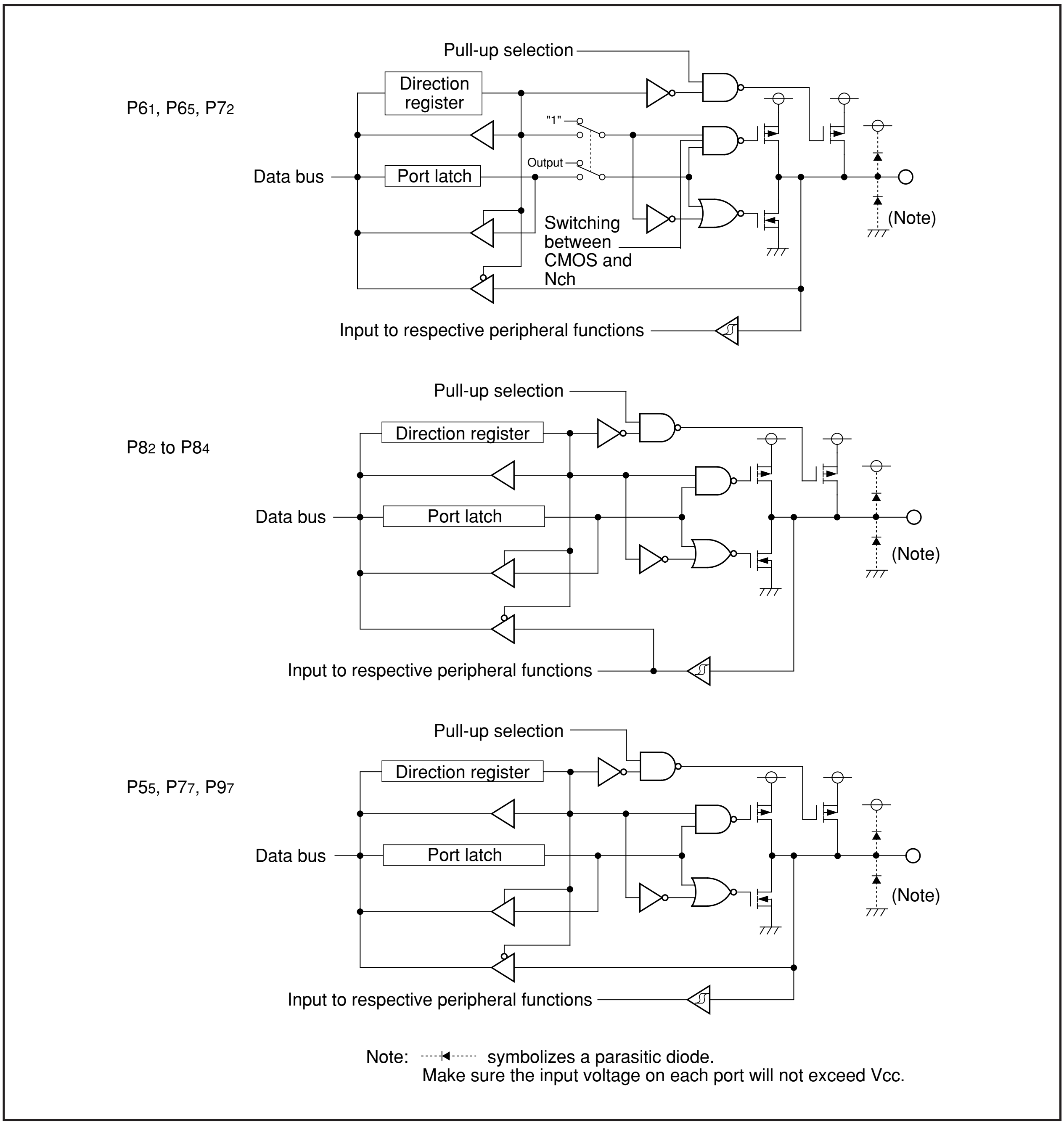


Figure 1.20.2 I/O Ports (2)

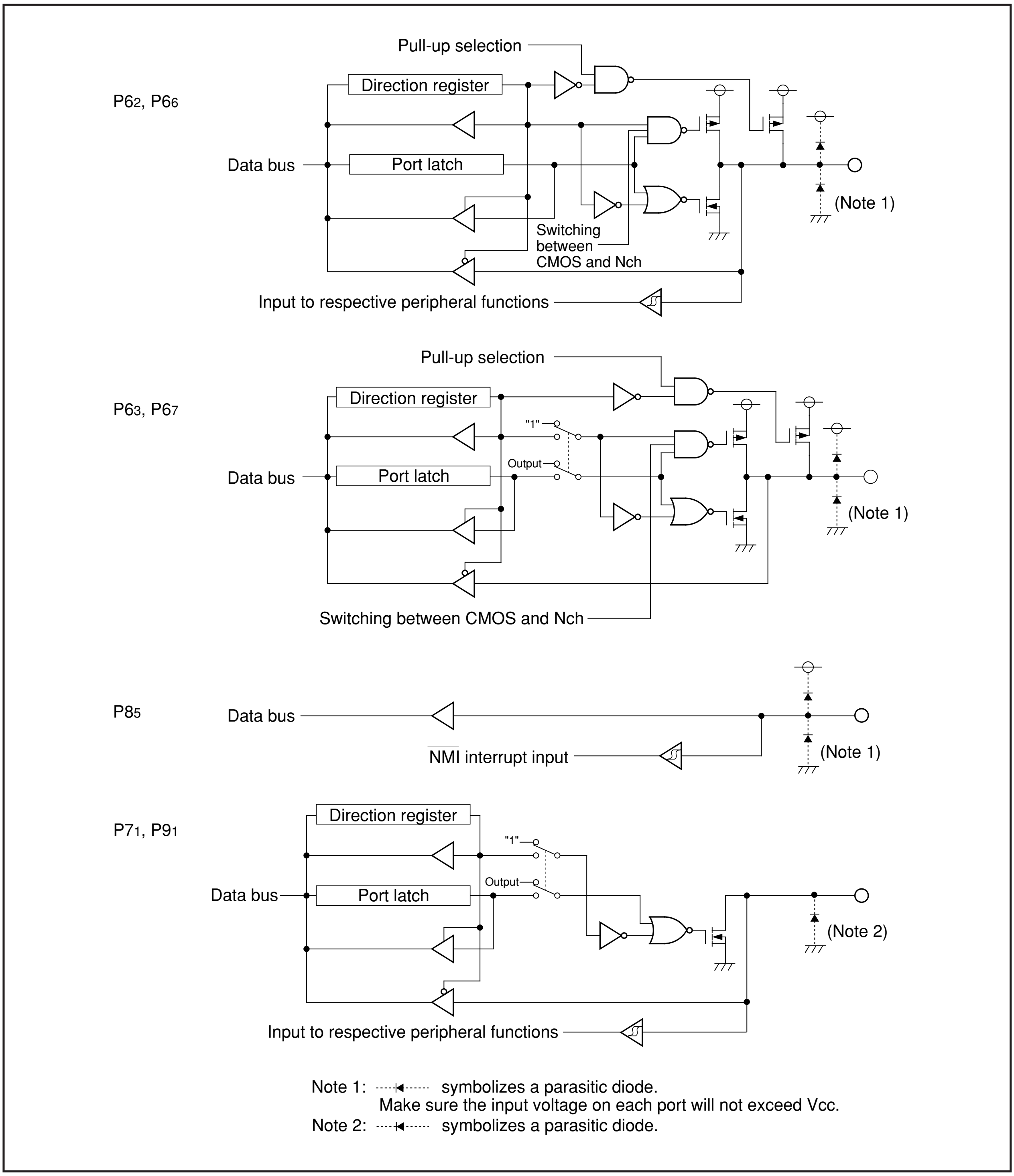


Figure 1.20.3 I/O Ports (3)

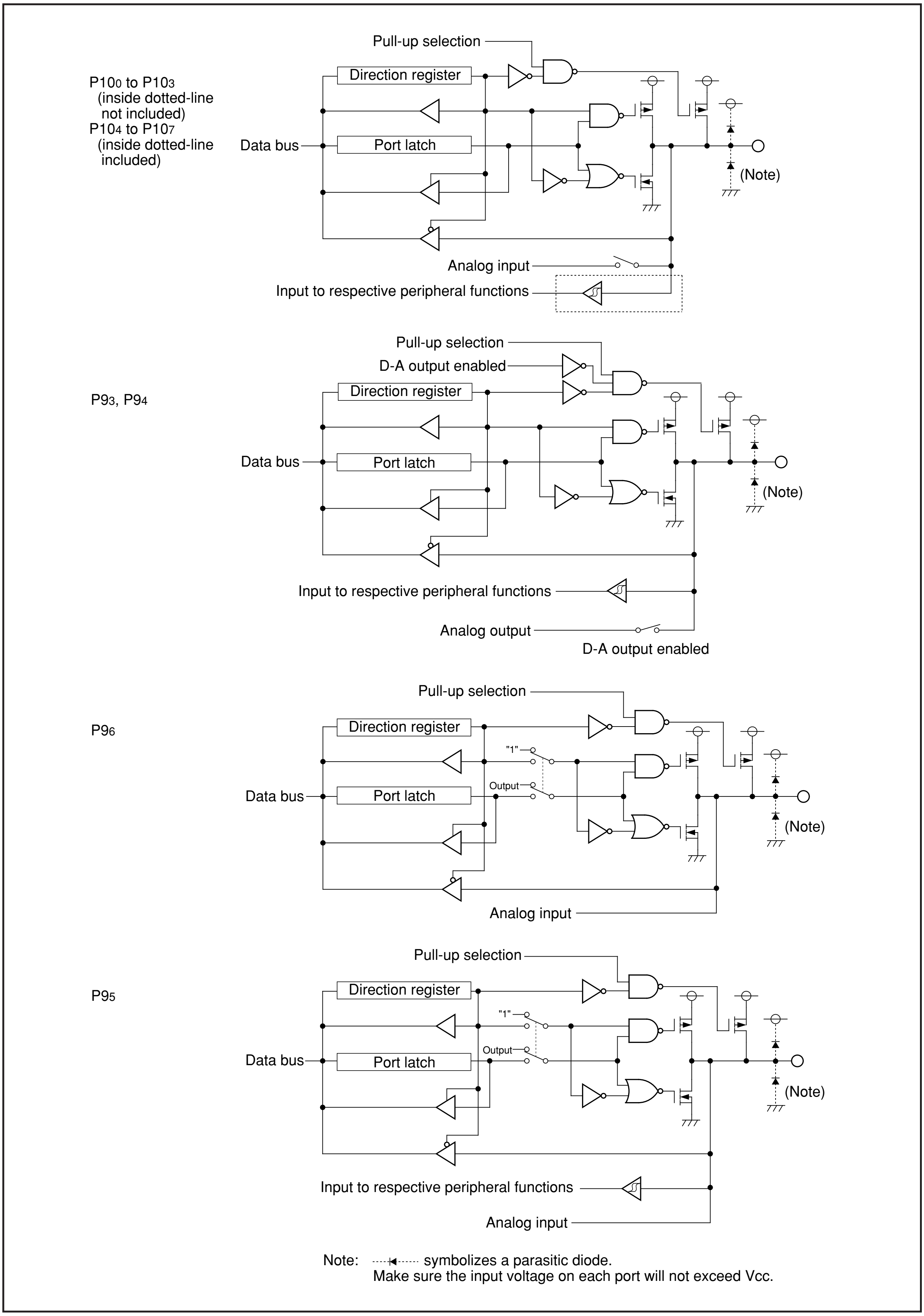


Figure 1.20.4 I/O Ports (4)

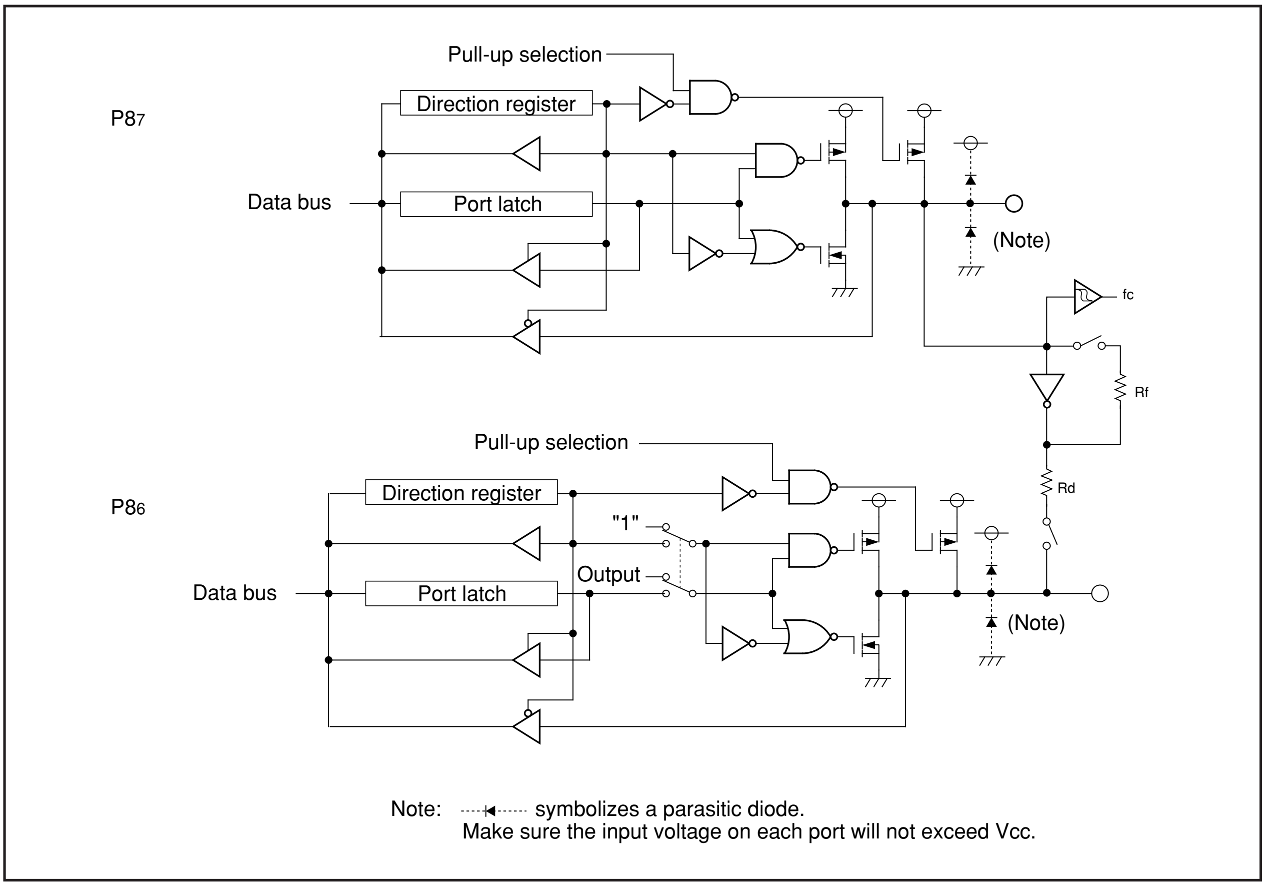


Figure 1.20.5 I/O Ports (5)

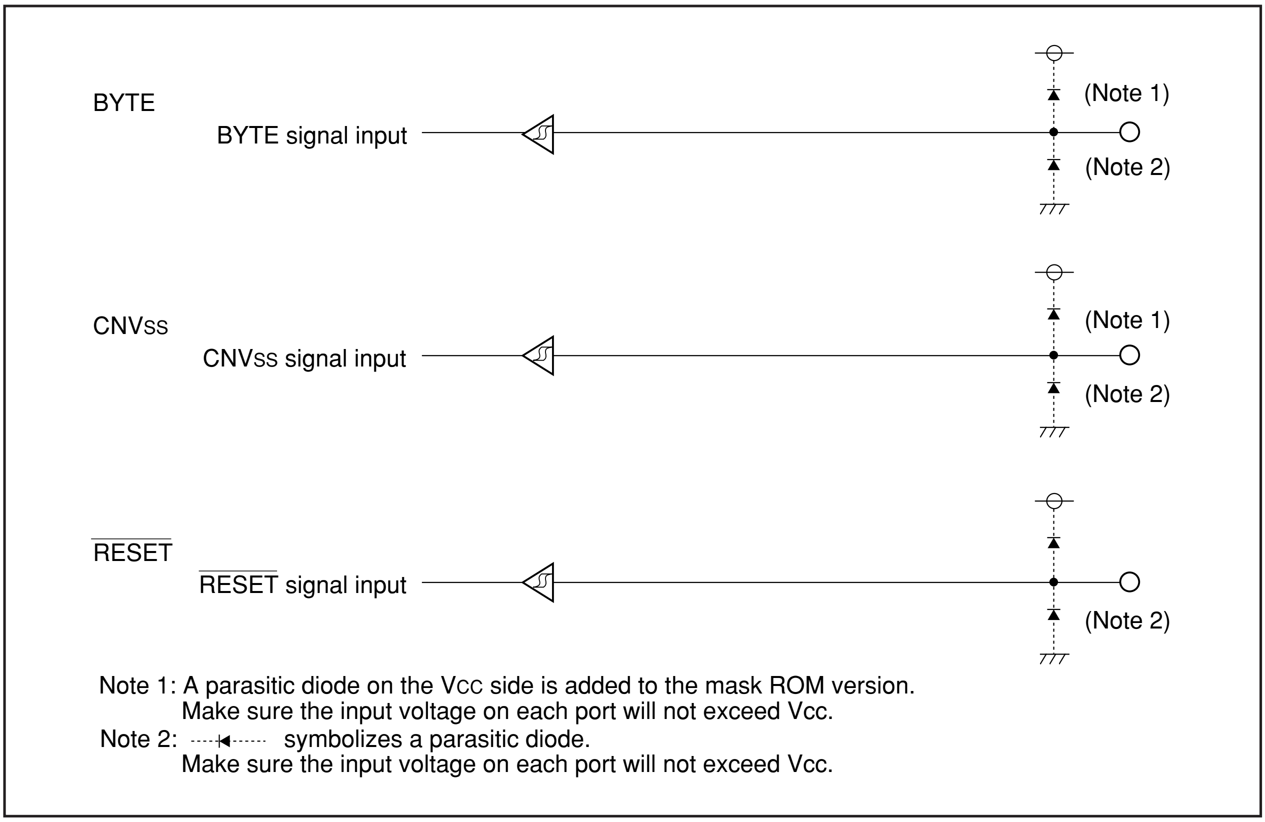


Figure 1.20.6 I/O Pins

Port Pi direction register (i = 0 to 7, 9, 10) (Notes 1, 2)

Symbol	Address	After reset
PD0 to PD3	$03E2_{16}$, $03E3_{16}$, $03E6_{16}$, $03E7_{16}$	00_{16}
PD4 to PD7	$03EA_{16}$, $03EB_{16}$, $03EE_{16}$, $03EF_{16}$	00_{16}
PD9, PD10	$03F3_{16}$, $03F6_{16}$	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, 9, 10)	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 PD7 and PD9 registers are written to by the next instruction after setting the PRCR register's PRC2 bit to "1" (write enabled).

Note 2: During memory expansion and microprocessor modes, the PD register for the pins functioning as bus control pins (A_0 to A_{19}, D_0 to D_{15}, $\overline{CS_0}$ to $\overline{CS_3}$, $\overline{RD}$, $\overline{WRL}/\overline{WR}$, $\overline{WRH}/\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$, $\overline{HLDA}$ and BCLK) cannot be modified.

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. When write, set to "0". When read, its content is 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 1.20.7 PD0 to PD10 Registers

Port Pi register (i = 0 to 7, 9, 10) (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
P0 to P3	$03E0_{16}$, $03E1_{16}$, $03E4_{16}$, $03E5_{16}$	Indeterminate
P4 to P7	$03E8_{16}$, $03E9_{16}$, $03EC_{16}$, $03ED_{16}$	Indeterminate
P9, P10	$03F1_{16}$, $03F4_{16}$	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 2) (i = 0 to 7, 9, 10)	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: During memory expansion and microprocessor modes, the Pi register for the pins functioning as bus control pins (A_0 to A_{19}, D_0 to D_{15}, $\overline{CS_0}$ to $\overline{CS_3}$, $\overline{RD}$, $\overline{WRL}/\overline{WR}$, $\overline{WRH}/\overline{BHE}$, ALE, $\overline{RDY}$, $\overline{HOLD}$, $\overline{HLDA}$ and BCLK) cannot be modified.

Note 2: Since $P7_1$ and $P9_1$ are N channel open-drain ports, the data is high-impedance.

Port P8 register

b7 b6 b5 b4 b3 b2 b1 b0

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
Pi8_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 1.20.8 P0 to P10 Registers

Pull-up control register 0 (Note 1)

b7 b6 b5 b4 b3 b2 b1 b0

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

Bit symbol	Bit name	Function	RW
PU00	$P0_0$ to $P0_3$ pull-up	0 : Not pulled high 1 : Pulled high (Note 2)	RW
PU01	$P0_4$ to $P0_7$ pull-up		RW
PU02	$P1_0$ to $P1_3$ pull-up		RW
PU03	$P1_4$ to $P1_7$ pull-up		RW
PU04	$P2_0$ to $P2_3$ pull-up		RW
PU05	$P2_4$ to $P2_7$ pull-up		RW
PU06	$P3_0$ to $P3_3$ pull-up		RW
PU07	$P3_4$ to $P3_7$ pull-up		RW

Note 1: During memory expansion 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

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset (Note 1)
PUR1	03FD$_{16}$	00000000$_2$ 00000010$_2$

Bit symbol	Bit name	Function	RW
PU10	$P4_0$ to $P4_3$ pull-up (Note 2)	0 : Not pulled high 1 : Pulled high (Note 5)	RW
PU11	$P4_4$ to $P4_7$ pull-up (Note 3)		RW
PU12	$P5_0$ to $P5_3$ pull-up (Note 2)		RW
PU13	$P5_4$ to $P5_7$ pull-up (Note 2)		RW
PU14	$P6_0$ to $P6_3$ pull-up		RW
PU15	$P6_4$ to $P6_7$ pull-up		RW
PU16	$P7_0$, $P7_2$ and $P7_3$ pull-up (Note 4)		RW
PU17	$P7_4$ to $P7_7$ pull-up		RW

Note 1: The values after hardware reset is 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).

Note 2: During memory expansion and microprocessor modes, the pins are not pulled high although their corresponding register contents can be modified.
Note 3: 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 4: The $P7_1$ pin does not have pull-up.
Note 5: 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 2

b7 b6 b5 b4 b3 b2 b1 b0

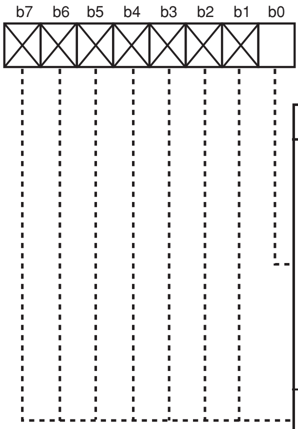
Symbol	Address	After reset
PUR2	03FE$_{16}$	00$_{16}$

Bit symbol	Bit name	Function	RW
PU20	$P8_0$ to $P8_3$ pull-up	0 : Not pulled high 1 : Pulled high (Note 3)	RW
PU21	$P8_4$ to $P8_7$ pull-up (Note 1)		RW
PU22	$P9_0$, $P9_2$ and $P9_3$ pull-up (Note 2)		RW
PU23	$P9_4$ to $P9_7$ pull-up		RW
PU24	$P10_0$ to $P10_3$ pull-up		RW
PU25	$P10_4$ to $P10_7$ pull-up		RW
– (b7-b6)	Nothing is assigned. When write, set to "0". When read, its content is "0".		–

Note 1: The $P8_5$ pin does not have pull-up.
Note 2: The $P9_1$ pin does not have pull-up.
Note 3: The pin for which this bit is "1" (pulled high) and the direction bit is "0" (input mode) is pulled high.

Figure 1.20.9 PUR0 to PUR2 Registers

Port control register



Symbol	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 $P1_0$ to $P1_7$ 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. When write, set to "0". When read, its content is "0".		–

Figure 1.20.10 PCR Register

Table 1.20.1 Unassigned Pin Handling in Single-chip Mode

Pin name	Connection
Ports P0 to P7, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, P9, P10	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. (Notes 1, 2, 3)
X_{OUT} (Note 4)	Open
$\overline{NMI}$($P8_5$)	Connect via resistor to V_{CC} (pull-up)
AV_{CC}	Connect to V_{CC}
AV_{SS}, V_{REF}, BYTE	Connect to V_{SS}

Note 1: When setting the port for output mode and leave it open, be aware that the port remains in input mode until it is switched to output mode in a program after reset. For this reason, the voltage level on the pin becomes indeterminate, causing the power supply current to increase while the port remains in input mode. Furthermore, by considering a possibility that the contents of the direction registers could be changed by noise or noise-induced runaway, it is recommended that the contents of the direction registers be periodically reset in software, for the increased reliability of the program.
Note 2: Make sure the unused pins are processed with the shortest possible wiring from the microcomputer pins (within 2 cm).
Note 3: When the ports $P7_1$ and $P9_1$ are set for output mode, make sure a low-level signal is output from the pins. The ports $P7_1$ and $P9_1$ are N-channel open-drain outputs.
Note 4: With external clock input to X_{IN} pin.

Table 1.20.2 Unassigned Pin Handling in Memory Expansion Mode and Microprocessor Mode

Pin name	Connection
Ports P0 to P7, $P8_0$ to $P8_4$, $P8_6$, $P8_7$, P9, P10	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. (Notes 1, 2, 3, 4)
$P4_5/\overline{CS_1}$ to $P4_7/\overline{CS_3}$	Connect to V_{CC} via a resistor (pulled high) by setting the PD4 register's corresponding direction bit for $\overline{CS_i}$ (i = 1 to 3) to "0" (input mode) and the CSR register's $\overline{CS_i}$ bit to "0" (chip select disabled).
$\overline{BHE}$, ALE, $\overline{HLDA}$, X_{OUT} (Note 5), BCLK (Note 6)	Open
$\overline{HOLD}$, $\overline{RDY}$, $\overline{NMI}$($P8_5$)	Connect via resistor to V_{CC} (pull-up)
AV_{CC}	Connect to V_{CC}
AV_{SS}, V_{REF}	Connect to V_{SS}

Note 1: When setting the port for output mode and leave it open, be aware that the port remains in input mode until it is switched to output mode in a program after reset. For this reason, the voltage level on the pin becomes indeterminate, causing the power supply current to increase while the port remains in input mode. Furthermore, by considering a possibility that the contents of the direction registers could be changed by noise or noise-induced runaway, it is recommended that the contents of the direction registers be periodically reset in software, for the increased reliability of the program.
Note 2: Make sure the unused pins are processed with the shortest possible wiring from the microcomputer pins (within 2 cm).
Note 3: If the CNV_{SS} pin has the V_{SS} level applied to it, these pins are set for input ports until the processor mode is switched over in a program after reset. For this reason, the voltage levels on these pins become indeterminate, causing the power supply current to increase while they remain set for input ports.
Note 4: When the ports $P7_1$ and $P9_1$ are set for output mode, make sure a low-level signal is output from the pins. The ports $P7_1$ and $P9_1$ are N-channel open-drain outputs.
Note 5: With external clock input to X_{IN} pin.
Note 6: If the PM0 register's PM07 bit is set to "1" (BCLK not output), connect this pin to V_{CC} via a resistor. (pulled high).

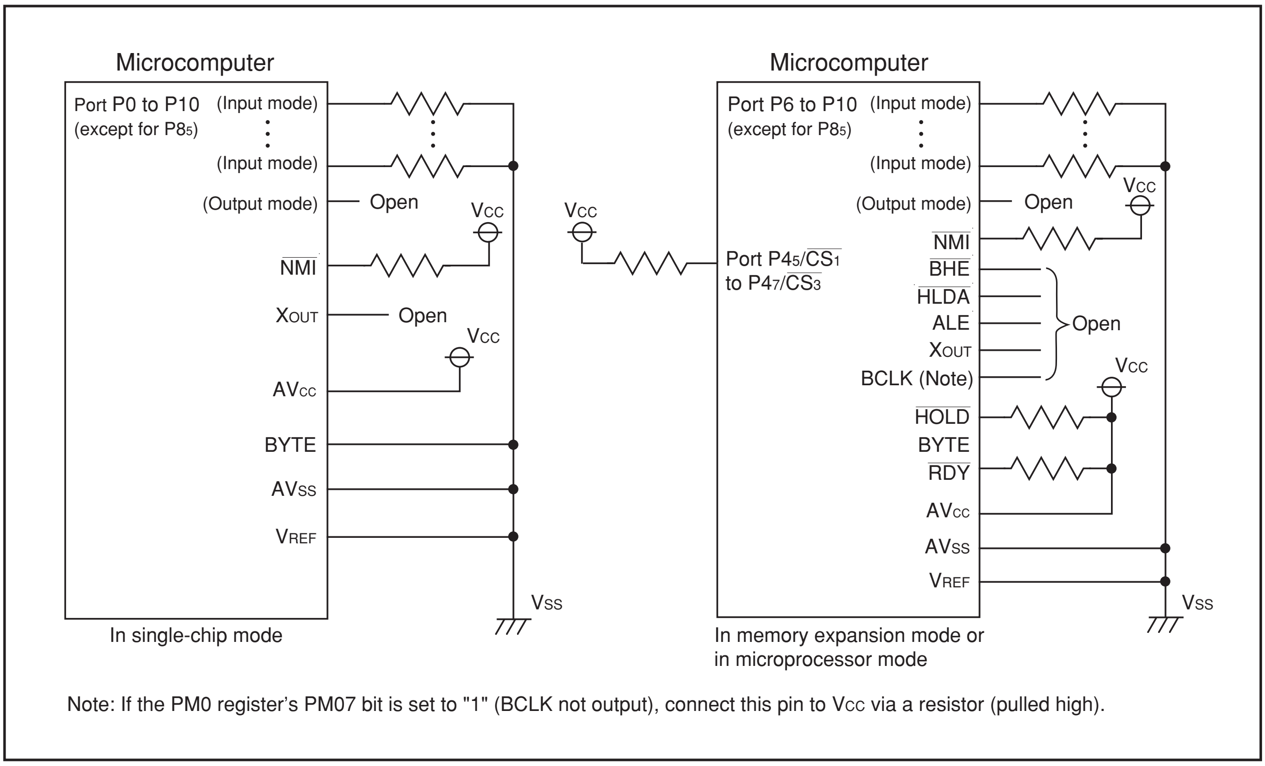


Figure 1.20.11 Unassigned Pins Handling

Electrical Characteristics

Table 1.21.1 Absolute Maximum Ratings

Symbol	Parameter		Condition	Rated value	Unit
V_{CC1}	Supply voltage		V_{CC1}=AV_{CC}	–0.3 to 6.5	V
V_{CC2}	Supply voltage		V_{CC2}	–0.3<V_{CC2}=V_{CC1}	V
AV_{CC}	Analog supply voltage		V_{CC1}=AV_{CC}	–0.3 to 6.5	V
V_I	Input voltage	$\overline{RESET}$, CNV_{SS}, BYTE, $P6_0$~$P6_7$, $P7_0$, $P7_2$~$P7_7$, $P8_0$~$P8_7$, $P9_0$, $P9_2$~$P9_7$, $P10_0$~$P10_7$, V_{REF}, X_{IN}		–0.3 to V_{CC1}+0.3	V
		$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$~$P3_7$, $P4_0$~$P4_7$, $P5_0$~$P5_7$		–0.3 to V_{CC2}+0.3	V
		$P7_1$, $P9_1$		–0.3 to 6.5	V
V_O	Output voltage	$P6_0$~$P6_7$, $P7_0$, $P7_2$~$P7_7$, $P8_0$~$P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$~$P9_7$, $P10_0$~$P10_7$, X_{OUT}		–0.3 to V_{CC1}+0.3	V
		$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$~$P3_7$, $P4_0$~$P4_7$, $P5_0$~$P5_7$		–0.3 to V_{CC2}+0.3	V
		$P7_1$, $P9_1$		–0.3 to 6.5	V
P_d	Power dissipation		T_{opr}=25°C	700	mW
T_{opr}	Operating ambient temperature			–40 to 85/–40 to 125 (option)	°C
T_{stg}	Storage temperature			–65 to 150	°C

option: If you desire this option, please so specify.

Table 1.21.2 Recommended Operating Conditions (Note 1)

Symbol	Parameter		Standard Min.	Standard Typ.	Standard Max.	Unit
V_{CC1}, V_{CC2}	Supply voltage (V_{CC1}=V_{CC2})		4.2	5.0	5.5	V
AV_{CC}	Analog supply voltage			V_{CC}		V
V_{SS}	Supply voltage			0		V
AV_{SS}	Analog supply voltage			0		V
V_{IH}	HIGH input voltage	$P3_1$~$P3_7$, $P4_0$~$P4_7$, $P5_0$~$P5_7$, $P6_0$~$P6_7$, $P7_0$, $P7_2$~$P7_7$, $P8_0$~$P8_7$, $P9_0$, $P9_2$~$P9_7$, $P10_0$~$P10_7$, X_{IN}, $\overline{RESET}$, CNV_{SS}, BYTE	$0.8V_{CC}$		V_{CC}	V
		$P7_1$, $P9_1$	$0.8V_{CC}$		6.5	V
		$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$ (During single-chip mode)	$0.8V_{CC}$		V_{CC}	V
		$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$ (Data input during memory expansion and microprocessor modes)	$0.5V_{CC}$		V_{CC}	V
V_{IL}	LOW input voltage	$P3_1$~$P3_7$, $P4_0$~$P4_7$, $P5_0$~$P5_7$, $P6_0$~$P6_7$, $P7_0$~$P7_7$, $P8_0$~$P8_7$, $P9_0$~$P9_7$, $P10_0$~$P10_7$, X_{IN}, $\overline{RESET}$, CNV_{SS}, BYTE	0		$0.2V_{CC}$	V
		$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$ (During single-chip mode)	0		$0.2V_{CC}$	V
		$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$ (Data input during memory expansion and microprocessor modes)	0		$0.16V_{CC}$	V
$I_{OH (peak)}$	HIGH peak output current	$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$~$P3_7$, $P4_0$~$P4_7$, $P5_0$~$P5_7$, $P6_0$~$P6_7$, $P7_0$, $P7_2$~$P7_7$, $P8_0$~$P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$~$P9_7$, $P10_0$~$P10_7$			–10.0	mA
$I_{OH (avg)}$	HIGH average output current	$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$~$P3_7$, $P4_0$~$P4_7$, $P5_0$~$P5_7$, $P6_0$~$P6_7$, $P7_0$, $P7_2$~$P7_7$, $P8_0$~$P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2$~$P9_7$, $P10_0$~$P10_7$			–5.0	mA
$I_{OL (peak)}$	LOW peak output current	$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$~$P3_7$, $P4_0$~$P4_7$, $P5_0$~$P5_7$, $P6_0$~$P6_7$, $P7_0$~$P7_7$, $P8_0$~$P8_4$, $P8_6$, $P8_7$, $P9_0$~$P9_7$, $P10_0$~$P10_7$			10.0	mA
$I_{OL (avg)}$	LOW average output current	$P0_0$~$P0_7$, $P1_0$~$P1_7$, $P2_0$~$P2_7$, $P3_0$~$P3_7$, $P4_0$~$P4_7$, $P5_0$~$P5_7$, $P6_0$~$P6_7$, $P7_0$~$P7_7$, $P8_0$~$P8_4$, $P8_6$, $P8_7$, $P9_0$~$P9_7$, $P10_0$~$P10_7$			5.0	mA
$f(X_{IN})$	Main clock input oscillation frequency (Notes 4, 5 and 6)	No wait; Mask ROM version, Flash memory version; V_{CC}=4.2 to 5.5V	0		16	MHz
$f(X_{CIN})$	Sub clock oscillation frequency			32.768	50	kHz
f(Ring)	Ring oscillation frequency			1		MHz
f(PLL)	PLL clock oscillation frequency				20	MHz
f(BCLK)	CPU operation clock	V_{CC}=4.2 to 5.5V	0		20	MHz
t_{su}(PLL)	PLL frequency synthesizer stabilization wait time				20	ms

Note 1: Referenced to V_{CC} = 4.2 to 5.5 V at Topr = –40 to 85 °C unless otherwise specified.

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

Note 3: The total I_{OL} (peak) for ports P0, P1, P2, $P8_6$, $P8_7$, P9 and P10 must be 80mA max. The total I_{OL} (peak) for ports P3, P4, P5, P6, P7 and $P8_0$ to $P8_4$ must be 80mA max. The total I_{OH} (peak) for ports P0, P1, and P2 must be –40mA max. The total I_{OH} (peak) for ports P3, P4 and P5 must be –40mA max. The total I_{OH} (peak) for ports P6, P7 and $P8_0$ to $P8_4$ must be –40mA max. The total I_{OH} (peak) for ports $P8_6$, $P8_7$, P9 and P10 must be –40mA max.

Note 4: Relationship between main clock oscillation frequency and supply voltage is shown right.

Note 5: Execute program /erase of flash memory by V_{CC} = 5.0 ± 0.5 V.

Note 6: When using 16 MHz or more, use PLL clock.

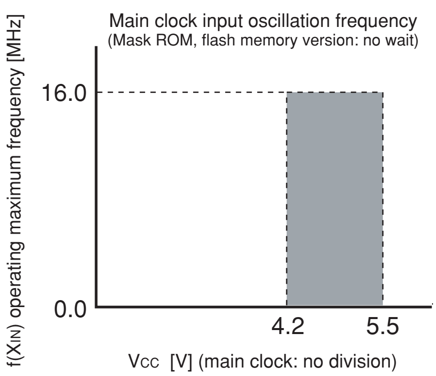


Table 1.21.3 Electrical Characteristics (Note 1)

Symbol	Parameter			Measuring condition		Standard Min.	Standard Typ.	Standard Max.	Unit
V_{OH}	HIGH output voltage	$P0_0 \sim P0_7$, $P1_0 \sim P1_7$, $P2_0 \sim P2_7$, $P3_0 \sim P3_7$, $P4_0 \sim P4_7$, $P5_0 \sim P5_7$, $P6_0 \sim P6_7$, $P7_0$, $P7_2 \sim P7_7$, $P8_0 \sim P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2 \sim P9_7$, $P10_0 \sim P10_7$		I_{OH}=–5mA		V_{CC}-2.0		V_{CC}	V
V_{OH}	HIGH output voltage	$P0_0 \sim P0_7$, $P1_0 \sim P1_7$, $P2_0 \sim P2_7$, $P3_0 \sim P3_7$, $P4_0 \sim P4_7$, $P5_0 \sim P5_7$, $P6_0 \sim P6_7$, $P7_0$, $P7_2 \sim P7_7$, $P8_0 \sim P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2 \sim P9_7$, $P10_0 \sim P10_7$		I_{OH}=–200μA		V_{CC}-0.3		V_{CC}	V
V_{OH}	HIGH output voltage	X_{OUT}	HIGHPOWER	I_{OH}=–1mA		3.0		V_{CC}	V
			LOWPOWER	I_{OH}=–0.5mA		3.0		V_{CC}	
	HIGH output voltage	X_{COUT}	HIGHPOWER	With no load applied			2.5		V
			LOWPOWER	With no load applied			1.6		
V_{OL}	LOW output voltage	$P0_0 \sim P0_7$, $P1_0 \sim P1_7$, $P2_0 \sim P2_7$, $P3_0 \sim P3_7$, $P4_0 \sim P4_7$, $P5_0 \sim P5_7$, $P6_0 \sim P6_7$, $P7_0 \sim P7_7$, $P8_0 \sim P8_4$, $P8_6$, $P8_7$, $P9_0 \sim P9_7$, $P10_0 \sim P10_7$		I_{OL}=5mA				2.0	V
V_{OL}	LOW output voltage	$P0_0 \sim P0_7$, $P1_0 \sim P1_7$, $P2_0 \sim P2_7$, $P3_0 \sim P3_7$, $P4_0 \sim P4_7$, $P5_0 \sim P5_7$, $P6_0 \sim P6_7$, $P7_0 \sim P7_7$, $P8_0 \sim P8_4$, $P8_6$, $P8_7$, $P9_0 \sim P9_7$, $P10_0 \sim P10_7$		I_{OL}=200μA				0.45	V
V_{OL}	LOW output voltage	X_{OUT}	HIGHPOWER	I_{OL}=1mA				2.0	V
			LOWPOWER	I_{OL}=0.5mA				2.0	
	LOW output voltage	X_{COUT}	HIGHPOWER	With no load applied			0		V
			LOWPOWER	With no load applied			0		
V_{T+}–V_{T-}	Hysteresis	$\overline{HOLD}$, $\overline{RDY}$, $TA0_{IN} \sim TA4_{IN}$, $TB0_{IN} \sim TB5_{IN}$, $\overline{INT_0} \sim \overline{INT_5}$, $\overline{NMI}$, $\overline{AD_{TRG}}$, $\overline{CTS_0} \sim \overline{CTS_2}$, SCL, SDA, $CLK_0 \sim CLK_4$, $TA2_{OUT} \sim TA4_{OUT}$, $\overline{KI_0} \sim \overline{KI_3}$, $RxD_0 \sim RxD_2$, S_{IN3}				0.2		1.0	V
V_{T+}–V_{T-}	Hysteresis	$\overline{RESET}$				0.2		2.2	V
I_{IH}	HIGH input current	$P0_0 \sim P0_7$, $P1_0 \sim P1_7$, $P2_0 \sim P2_7$, $P3_0 \sim P3_3$, $P4_0 \sim P4_7$, $P5_0 \sim P5_7$, $P6_0 \sim P6_7$, $P7_0 \sim P7_7$, $P8_0 \sim P8_7$, $P9_0 \sim P9_7$, $P10_0 \sim P10_7$, X_{IN}, $\overline{RESET}$, CNV_{SS}, BYTE		V_I=5V				5.0	μA
I_{IL}	LOW input current	$P0_0 \sim P0_7$, $P1_0 \sim P1_7$, $P2_0 \sim P2_7$, $P3_0 \sim P3_3$, $P4_0 \sim P4_7$, $P5_0 \sim P5_7$, $P6_0 \sim P6_7$, $P7_0 \sim P7_7$, $P8_0 \sim P8_7$, $P9_0 \sim P9_7$, $P10_0 \sim P10_7$, X_{IN}, $\overline{RESET}$, CNV_{SS}, BYTE		V_I=0V				–5.0	μA
R_{PULLUP}	Pull-up resistance	$P0_0 \sim P0_7$, $P1_0 \sim P1_7$, $P2_0 \sim P2_7$, $P3_0 \sim P3_7$, $P4_0 \sim P4_7$, $P5_0 \sim P5_7$, $P6_0 \sim P6_7$, $P7_0$, $P7_2 \sim P7_7$, $P8_0 \sim P8_4$, $P8_6$, $P8_7$, $P9_0$, $P9_2 \sim P9_7$, $P10_0 \sim P10_7$		V_I=0V		30	50	170	kΩ
R_{fXIN}	Feedback resistance	X_{IN}					1.5		MΩ
R_{fXCIN}	Feedback resistance	X_{CIN}					15		MΩ
V_{RAM}	RAM retention voltage			At stop mode		2.0			V
I_{CC}	Power supply current (V_{CC} = 4.2 to 5.5V)	In single-chip mode, the output pins are open and other pins are V_{SS}.		Mask ROM	f(BCLK)=20MHz, No division, PLL operation		18	32	mA
					No division, Ring oscillation		1		mA
				Flash memory	f(BCLK)=20MHz, No division, PLL operation		20	34	mA
					No division, Ring oscillation		1.8		mA
				Flash memory Program	f(BCLK)=10MHz, V_{CC}=5V		15		mA
				Flash memory Erase	f(BCLK)=10MHz, V_{CC}=5V		25		mA
				Mask ROM	$f(X_{CIN})$=32kHz, Low power dissipation mode, ROM (Note 2)		25		μA
				Flash memory	f(BCLK)=32kHz, Low power dissipation mode, RAM (Note 2)		25		μA
					f(BCLK)=32kHz, Low power dissipation mode, Flash memory (Note 2)		420		μA
				Mask ROM	Ring oscillation, Wait mode		50		μA
				Flash memory	f(BCLK)=32kHz, Wait mode (Note 3), Oscillation capacity High		8.5		μA
					f(BCLK)=32kHz, Wait mode (Note 3), Oscillation capacity Low		3.0		μA
					Stop mode, T_{opr} = 25 °C		0.8	3.0	μA

Note 1: Referenced to V_{CC} = 4.2 to 5.5 V, Vss = 0 V at Topr = –40 to 85 °C, f(BCLK) = 20 MHz unless otherwise specified.
Note 2: This indicates the memory in which the program to be executed exists.
Note 3: With one timer operated using f_{C32}.

Table 1.21.4 A-D Conversion Characteristics (Note 1)

Symbol	Parameter		Measuring condition		Standard Min.	Standard Typ.	Standard Max.	Unit
–	Resolution		$V_{REF}=V_{CC}$				10	Bits
INL	Integral non-linearity error	10 bits	$V_{REF}=V_{CC}$ =5V	ANEX0, ANEX1 input, AN_0 to AN_7 input, AN_{00} to AN_{07} input, AN_{20} to AN_{27} input			±3	LSB
				External operation amp connection mode			±7	LSB
		8 bits	$V_{REF}=AV_{CC}=V_{CC}$=5V				±2	LSB
–	Absolute accuracy	10 bits	$V_{REF}=V_{CC}$ =5V	ANEX0, ANEX1 input, AN_0 to AN_7 input, AN_{00} to AN_{07} input, AN_{20} to AN_{27} input			±3	LSB
				External operation amp connection mode			±7	LSB
		8 bits	$V_{REF}=AV_{CC}=V_{CC}$=5V				±2	LSB
DNL	Differential non-linearity error						±1	LSB
–	Offset error						±3	LSB
–	Gain error						±3	LSB
R_{LADDER}	Ladder resistance		$V_{REF}=V_{CC}$		10		40	kΩ
t_{CONV}	Conversion time (10 bits), Sample & hold function available		$V_{REF}=V_{CC}$=5V, ϕ_{AD}=10MHz		3.3			μs
	Conversion time (8 bits), Sample & hold function available		$V_{REF}=V_{CC}$=5V, ϕ_{AD}=10MHz		2.8			μs
t_{SAMP}	Sampling time				0.3			μs
V_{REF}	Reference voltage				2.0		V_{CC}	V
V_{IA}	Analog input voltage				0		V_{REF}	V

Note 1: Referenced to V_{CC} = AV_{CC} = V_{REF} = 4.2 to 5.5 V, V_{SS} = AV_{SS} = 0 V, –40 to 85 °C unless otherwise specified.
Note 2: AD operation clock frequency (ϕ_{AD} frequency) must be 10 MHz or less.
Note 3: A case without sample & hold function turn ϕ_{AD} frequency into 250 kHz or more in addition to a limit of Note 2.
A case with sample & hold function turn ϕ_{AD} frequency into 1 MHz or more in addition to a limit of Note 2.

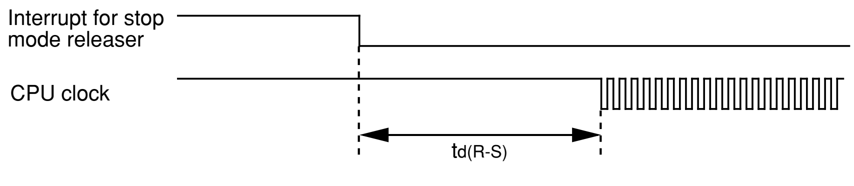
Table 1.21.5 D-A conversion Characteristics (Note 1)

Symbol	Parameter	Measuring condition	Standard Min.	Standard Typ.	Standard Max.	Unit
–	Resolution				8	Bits
–	Absolute accuracy				1.0	%
t_{su}	Setup time				3	μs
R_O	Output resistance		4	10	20	kΩ
I_{VREF}	Reference power supply input current	(Note 2)			1.5	mA

Note 1: Referenced to V_{CC} = AV_{CC} = V_{REF} = 4.2 to 5.5 V, V_{SS} = AV_{SS} = 0 V, –40 to 85 °C unless otherwise specified.
Note 2: This applies when using one D-A converter, with the DAi register (i = 0, 1) for the unused D-A converter set to "00_{16}". The A-D converter's ladder resistance is not included. Also, the current I_{VREF} always flows even though V_{REF} may have been set to be unconnected by the ADCON1 register.

Table 1.21.6 Power Supply Circuit Timing Characteristics

Symbol	Parameter	Measuring condition	Standard Min.	Standard Typ.	Standard Max.	Unit
$t_{d(P-R)}$	Time for internal power supply stabilization during powering-on	V_{CC} = 4.2 to 5.5 V			2	ms
$t_{d(R-S)}$	STOP release time				150	μs
$t_{d(W-S)}$	Low power dissipation mode wait mode release time				150	μs
$t_{d(M-L)}$	Time for internal power supply stabilization when main clock oscillation status				50	μs



Timing Requirements
(Referenced to V_{CC} = 5 V, V_{SS} = 0 V, at Topr = –40 to 85 °C unless otherwise specified)

Table 1.21.7 External Clock Input (X_{IN} Input)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
t_C	External clock input cycle time	62.5		ns
$t_{w(H)}$	External clock input HIGH pulse width	25		ns
$t_{w(L)}$	External clock input LOW pulse width	25		ns
t_r	External clock rise time		15	ns
t_f	External clock fall time		15	ns

Table 1.21.8 Memory Expansion Mode and Microprocessor Mode

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{ac1(RD-DB)}$	Data input access time (for setting with no wait)		(Note 1)	ns
$t_{ac2(RD-DB)}$	Data input access time (for setting with wait)		(Note 2)	ns
$t_{ac3(RD-DB)}$	Data input access time (when accessing multiplexed bus area)		(Note 3)	ns
$t_{su(DB-RD)}$	Data input setup time	40		ns
$t_{su(RDY-BCLK)}$	$\overline{RDY}$ input setup time	30		ns
$t_{su(HOLD-BCLK)}$	$\overline{HOLD}$ input setup time	40		ns
$t_{h(RD-DB)}$	Data input hold time	0		ns
$t_{h(BCLK-RDY)}$	$\overline{RDY}$ input hold time	0		ns
$t_{h(BCLK-HOLD)}$	$\overline{HOLD}$ input hold time	0		ns
$t_{d(BCLK-HLDA)}$	$\overline{HLDA}$ output delay time		40	ns

Note 1: Calculated according to the BCLK frequency as follows:

$$\frac{0.5 \times 10^9}{f(BCLK)} - 45 \text{ [ns]}$$

Note 2: Calculated according to the BCLK frequency as follows:

$$\frac{(n - 0.5) \times 10^9}{f(BCLK)} - 45 \text{ [ns]}$$

n is “2” for 1-wait setting, “3” for 2-wait setting and “4” for 3-wait setting.

Note 3: Calculated according to the BCLK frequency as follows:

$$\frac{(n - 0.5) \times 10^9}{f(BCLK)} - 45 \text{ [ns]}$$

n is “2” for 2-wait setting, “3” for 3-wait setting.

Timing Requirements
(Referenced to V_{CC} = 5 V, V_{SS} = 0 V, at Topr = –40 to 85 °C unless otherwise specified)

Table 1.21.9 Timer A Input (Counter Input in Event Counter Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TAi_{IN} input cycle time	100		ns
$t_{w(TAH)}$	TAi_{IN} input HIGH pulse width	40		ns
$t_{w(TAL)}$	TAi_{IN} input LOW pulse width	40		ns

Table 1.21.10 Timer A Input (Gating Input in Timer Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TAi_{IN} input cycle time	400		ns
$t_{w(TAH)}$	TAi_{IN} input HIGH pulse width	200		ns
$t_{w(TAL)}$	TAi_{IN} input LOW pulse width	200		ns

Table 1.21.11 Timer A Input (External Trigger Input in One-shot Timer Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TAi_{IN} input cycle time	200		ns
$t_{w(TAH)}$	TAi_{IN} input HIGH pulse width	100		ns
$t_{w(TAL)}$	TAi_{IN} input LOW pulse width	100		ns

Table 1.21.12 Timer A Input (External Trigger Input in Pulse Width Modulation Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{w(TAH)}$	TAi_{IN} input HIGH pulse width	100		ns
$t_{w(TAL)}$	TAi_{IN} input LOW pulse width	100		ns

Table 1.21.13 Timer A Input (Counter Increment/decrement Input in Event Counter Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(UP)}$	TAi_{OUT} input cycle time	2000		ns
$t_{w(UPH)}$	TAi_{OUT} input HIGH pulse width	1000		ns
$t_{w(UPL)}$	TAi_{OUT} input LOW pulse width	1000		ns
$t_{su(UP-TIN)}$	TAi_{OUT} input setup time	400		ns
$t_{h(TIN-UP)}$	TAi_{OUT} input hold time	400		ns

Table 1.21.14 Timer A Input (Two-phase Pulse Input in Event Counter Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TA)}$	TAi_{IN} input cycle time	800		ns
$t_{su(TAIN-TAOUT)}$	TAi_{OUT} input setup time	200		ns
$t_{su(TAOUT-TAIN)}$	TAi_{IN} input setup time	200		ns

Timing Requirements
(Referenced to V_{CC} = 5 V, V_{SS} = 0 V, at Topr = –40 to 85 °C unless otherwise specified)

Table 1.21.15 Timer B Input (Counter Input in Event Counter Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TB)}$	TBi_{IN} input cycle time (counted on one edge)	100		ns
$t_{w(TBH)}$	TBi_{IN} input HIGH pulse width (counted on one edge)	40		ns
$t_{w(TBL)}$	TBi_{IN} input LOW pulse width (counted on one edge)	40		ns
$t_{c(TB)}$	TBi_{IN} input cycle time (counted on both edges)	200		ns
$t_{w(TBH)}$	TBi_{IN} input HIGH pulse width (counted on both edges)	80		ns
$t_{w(TBL)}$	TBi_{IN} input LOW pulse width (counted on both edges)	80		ns

Table 1.21.16 Timer B Input (Pulse Period Measurement Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TB)}$	TBi_{IN} input cycle time	400		ns
$t_{w(TBH)}$	TBi_{IN} input HIGH pulse width	200		ns
$t_{w(TBL)}$	TBi_{IN} input LOW pulse width	200		ns

Table 1.21.17 Timer B Input (Pulse Width Measurement Mode)

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(TB)}$	TBi_{IN} input cycle time	400		ns
$t_{w(TBH)}$	TBi_{IN} input HIGH pulse width	200		ns
$t_{w(TBL)}$	TBi_{IN} input LOW pulse width	200		ns

Table 1.21.18 A-D Trigger Input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{C(AD)}$	$\overline{AD_{TRG}}$ input cycle time (trigger able minimum)	1000		ns
$t_{w(ADL)}$	$\overline{AD_{TRG}}$ input LOW pulse width	125		ns

Table 1.21.19 Serial I/O

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{c(CK)}$	CLK_i input cycle time	200		ns
$t_{w(CKH)}$	CLK_i input HIGH pulse width	100		ns
$t_{w(CKL)}$	CLK_i input LOW pulse width	100		ns
$t_{d(C-Q)}$	TxD_i output delay time		80	ns
$t_{h(C-Q)}$	TxD_i hold time	0		ns
$t_{su(D-C)}$	RxD_i input setup time	30		ns
$t_{h(C-D)}$	RxD_i input hold time	90		ns

Table 1.21.20 External Interrupt $\overline{INTi}$ Input

Symbol	Parameter	Standard		Unit
		Min.	Max.	
$t_{w(INH)}$	$\overline{INTi}$ input HIGH pulse width	250		ns
$t_{w(INL)}$	$\overline{INTi}$ input LOW pulse width	250		ns

Switching Characteristics
(Referenced to V_{CC} = 5 V, V_{SS} = 0 V, at Topr = –40 to 85 °C unless otherwise specified)

Table 1.21.21 Memory Expansion Mode and Microprocessor Mode (for setting with no wait)

Symbol	Parameter	Measuring condition	Standard		Unit
			Min.	Max.	
$t_{d(BCLK-AD)}$	Address output delay time	Figure 1.21.1		25	ns
$t_{h(BCLK-AD)}$	Address output hold time (refers to BCLK)		4		ns
$t_{h(RD-AD)}$	Address output hold time (refers to RD)		0		ns
$t_{h(WR-AD)}$	Address output hold time (refers to WR)		(Note 1)		ns
$t_{d(BCLK-CS)}$	Chip select output delay time			25	ns
$t_{h(BCLK-CS)}$	Chip select output hold time (refers to BCLK)		4		ns
$t_{d(BCLK-ALE)}$	ALE signal output delay time			25	ns
$t_{h(BCLK-ALE)}$	ALE signal output hold time		–4		ns
$t_{d(BCLK-RD)}$	RD signal output delay time			25	ns
$t_{h(BCLK-RD)}$	RD signal output hold time		0		ns
$t_{d(BCLK-WR)}$	WR signal output delay time			25	ns
$t_{h(BCLK-WR)}$	WR signal output hold time		0		ns
$t_{d(BCLK-DB)}$	Data output delay time (refers to BCLK)			40	ns
$t_{h(BCLK-DB)}$	Data output hold time (refers to BCLK) (Note 3)		4		ns
$t_{d(DB-WR)}$	Data output delay time (refers to WR)		(Note 2)		ns
$t_{h(WR-DB)}$	Data output hold time (refers to WR) (Note 3)		(Note 1)		ns

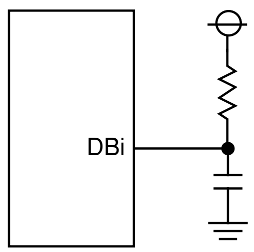
Note 1: Calculated according to the BCLK frequency as follows:

$$\frac{0.5 \times 10^9}{f(BCLK)} - 10 \text{ [ns]}$$

Note 2: Calculated according to the BCLK frequency as follows:

$$\frac{0.5 \times 10^9}{f(BCLK)} - 40 \text{ [ns]}$$ f(BCLK) is 12.5 MHz or less.

Note 3: This standard value shows the timing when the output is off, and does not show hold time of data bus.
Hold time of data bus varies with capacitor volume and pull-up (pull-down) resistance value.
Hold time of data bus is expressed in
$t = -CR \times \ln(1 - V_{OL} / V_{CC})$
by a circuit of the right figure.
For example, when V_{OL} = 0.2 V_{CC}, C = 30 pF,
R =1 kΩ, hold time of output "L" level is
$t = -30\text{ pF} \times 1\text{ k}\Omega \times \ln(1 - 0.2 V_{CC} / V_{CC}) = 6.7\text{ ns}$.



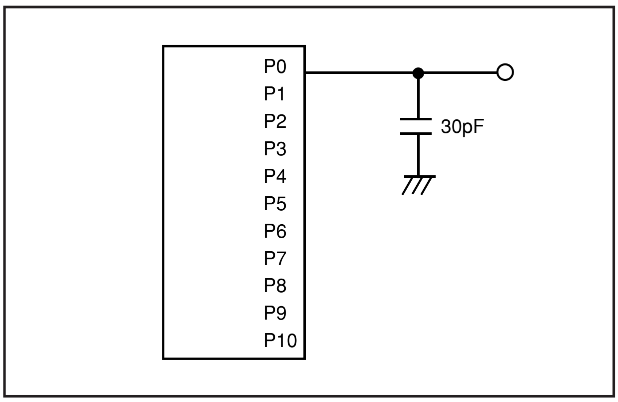


Figure 1.21.1. Port P0 to P10 Measurement Circuit

Switching Characteristics
(Referenced to V_{CC} = 5 V, V_{SS} = 0 V, at Topr = –40 to 85 °C unless otherwise specified)

Table 1.21.22 Memory Expansion Mode and Microprocessor Mode (for 1- to 3-wait setting and external area access)

Symbol	Parameter	Measuring condition	Standard Min.	Standard Max.	Unit
$t_{d(BCLK-AD)}$	Address output delay time	Figure 1.21.1		25	ns
$t_{h(BCLK-AD)}$	Address output hold time (refers to BCLK)		4		ns
$t_{h(RD-AD)}$	Address output hold time (refers to RD)		0		ns
$t_{h(WR-AD)}$	Address output hold time (refers to WR)		(Note 1)		ns
$t_{d(BCLK-CS)}$	Chip select output delay time			25	ns
$t_{h(BCLK-CS)}$	Chip select output hold time (refers to BCLK)		4		ns
$t_{d(BCLK-ALE)}$	ALE signal output delay time			25	ns
$t_{h(BCLK-ALE)}$	ALE signal output hold time		–4		ns
$t_{d(BCLK-RD)}$	RD signal output delay time			25	ns
$t_{h(BCLK-RD)}$	RD signal output hold time		0		ns
$t_{d(BCLK-WR)}$	WR signal output delay time			25	ns
$t_{h(BCLK-WR)}$	WR signal output hold time		0		ns
$t_{d(BCLK-DB)}$	Data output delay time (refers to BCLK)			40	ns
$t_{h(BCLK-DB)}$	Data output hold time (refers to BCLK) (Note 3)		4		ns
$t_{d(DB-WR)}$	Data output delay time (refers to WR)		(Note 2)		ns
$t_{h(WR-DB)}$	Data output hold time (refers to WR) (Note 3)		(Note 1)		ns

Note 1: Calculated according to the BCLK frequency as follows:

$$\frac{0.5 \times 10^9}{f(BCLK)} - 10 \text{ [ns]}$$

Note 2: Calculated according to the BCLK frequency as follows:

$$\frac{(n - 0.5) \times 10^9}{f(BCLK)} - 40 \text{ [ns]}$$

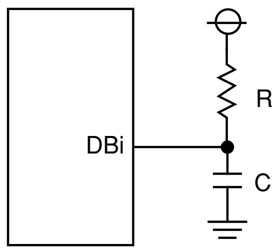
n is “1” for 1-wait setting, “2” for 2-wait setting and “3” for 3-wait setting.
When n = 1, f(BCLK) is 12.5 MHz or less.

Note 3: This standard value shows the timing when the output is off, and does not show hold time of data bus.
Hold time of data bus varies with capacitor volume and pull-up (pull-down) resistance value.
Hold time of data bus is expressed in

$t = -CR \times \ln (1 - V_{OL} / V_{CC})$

by a circuit of the right figure.
For example, when V_{OL} = 0.2 V_{CC}, C = 30 pF,
R =1 kΩ, hold time of output “L” level is

$t = -30 \text{ pF} \times 1 \text{ k}\Omega \times \ln (1 - 0.2 V_{CC} / V_{CC}) = 6.7 \text{ ns}$.



Switching Characteristics
(Referenced to V_{CC} = 5 V, V_{SS} = 0 V, at Topr = –40 to 85 °C unless otherwise specified)

Table 1.21.23 Memory Expansion Mode and Microprocessor Mode (for 2- to 3-wait setting, external area access and multiplexed bus selection)

Symbol	Parameter	Measuring condition	Standard		Unit
			Min.	Max.	
$t_{d(BCLK-AD)}$	Address output delay time	Figure 1.21.1		25	ns
$t_{h(BCLK-AD)}$	Address output hold time (refers to BCLK)		4		ns
$t_{h(RD-AD)}$	Address output hold time (refers to RD)		(Note 1)		ns
$t_{h(WR-AD)}$	Address output hold time (refers to WR)		(Note 1)		ns
$t_{d(BCLK-CS)}$	Chip select output delay time			25	ns
$t_{h(BCLK-CS)}$	Chip select output hold time (refers to BCLK)		4		ns
$t_{h(RD-CS)}$	Chip select output hold time (refers to RD)		(Note 1)		ns
$t_{h(WR-CS)}$	Chip select output hold time (refers to WR)		(Note 1)		ns
$t_{d(BCLK-RD)}$	RD signal output delay time			25	ns
$t_{h(BCLK-RD)}$	RD signal output hold time		0		ns
$t_{d(BCLK-WR)}$	WR signal output delay time			25	ns
$t_{h(BCLK-WR)}$	WR signal output hold time		0		ns
$t_{d(BCLK-DB)}$	Data output delay time (refers to BCLK)			40	ns
$t_{h(BCLK-DB)}$	Data output hold time (refers to BCLK)		4		ns
$t_{d(DB-WR)}$	Data output delay time (refers to WR)		(Note 2)		ns
$t_{h(WR-DB)}$	Data output hold time (refers to WR)		(Note 1)		ns
$t_{d(BCLK-ALE)}$	ALE signal output delay time (refers to BCLK)			25	ns
$t_{h(BCLK-ALE)}$	ALE signal output hold time (refers to BCLK)		–4		ns
$t_{d(AD-ALE)}$	ALE signal output delay time (refers to Address)		(Note 3)		ns
$t_{h(ALE-AD)}$	ALE signal output hold time (refers to Address)		(Note 4)		ns
$t_{d(AD-RD)}$	RD signal output delay from the end of Address		0		ns
$t_{d(AD-WR)}$	WR signal output delay from the end of Address		0		ns
$t_{dZ(RD-AD)}$	Address output floating start time			8	ns

Note 1: Calculated according to the BCLK frequency as follows:

$$\frac{0.5 \times 10^9}{f(BCLK)} - 10 \text{ [ns]}$$

Note 2: Calculated according to the BCLK frequency as follows:

$$\frac{(n - 0.5) \times 10^9}{f(BCLK)} - 40 \text{ [ns]}$$ n is “2” for 2-wait setting, “3” for 3-wait setting.

Note 3: Calculated according to the BCLK frequency as follows:

$$\frac{0.5 \times 10^9}{f(BCLK)} - 25 \text{ [ns]}$$

Note 4: Calculated according to the BCLK frequency as follows:

$$\frac{0.5 \times 10^9}{f(BCLK)} - 15 \text{ [ns]}$$

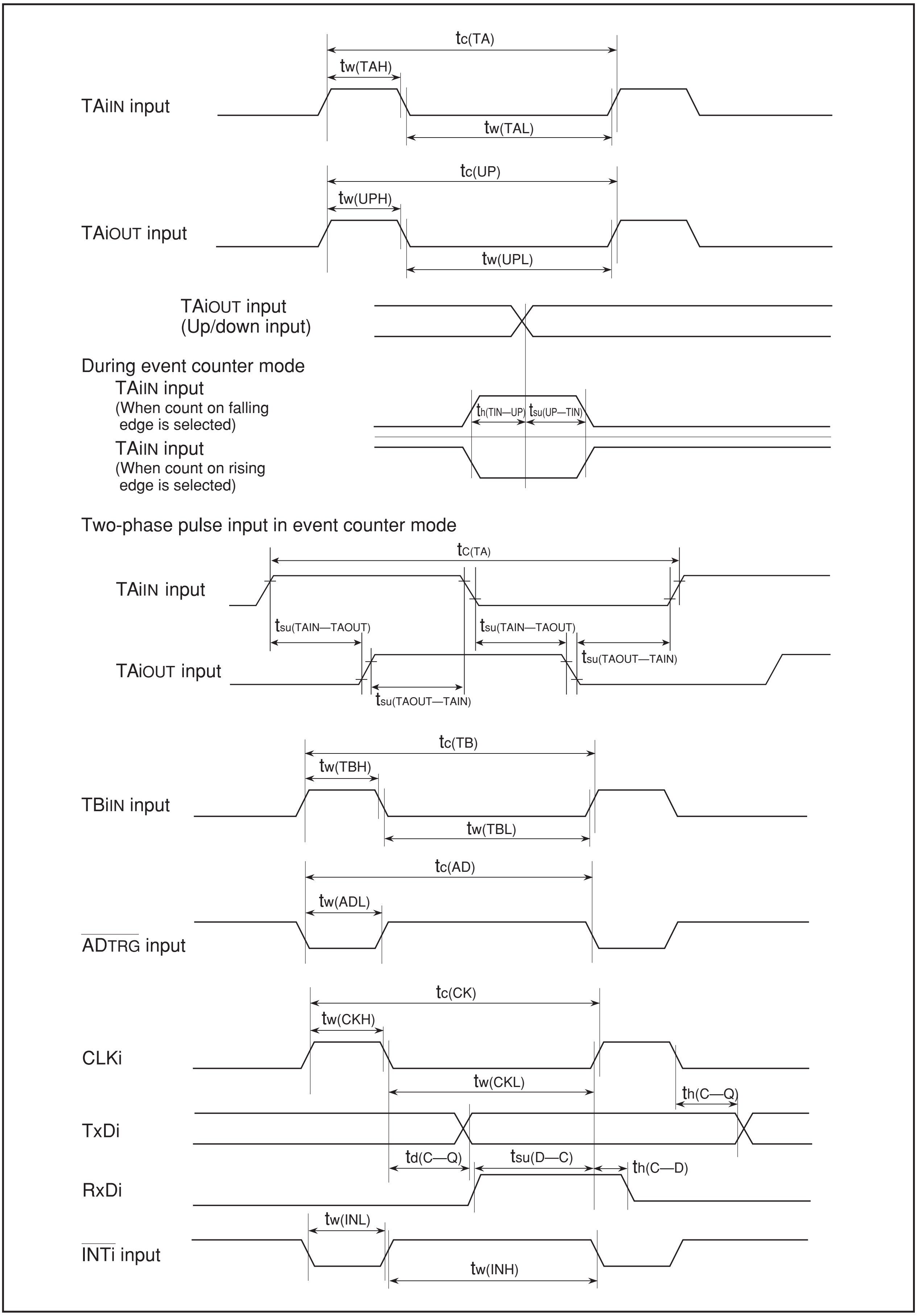
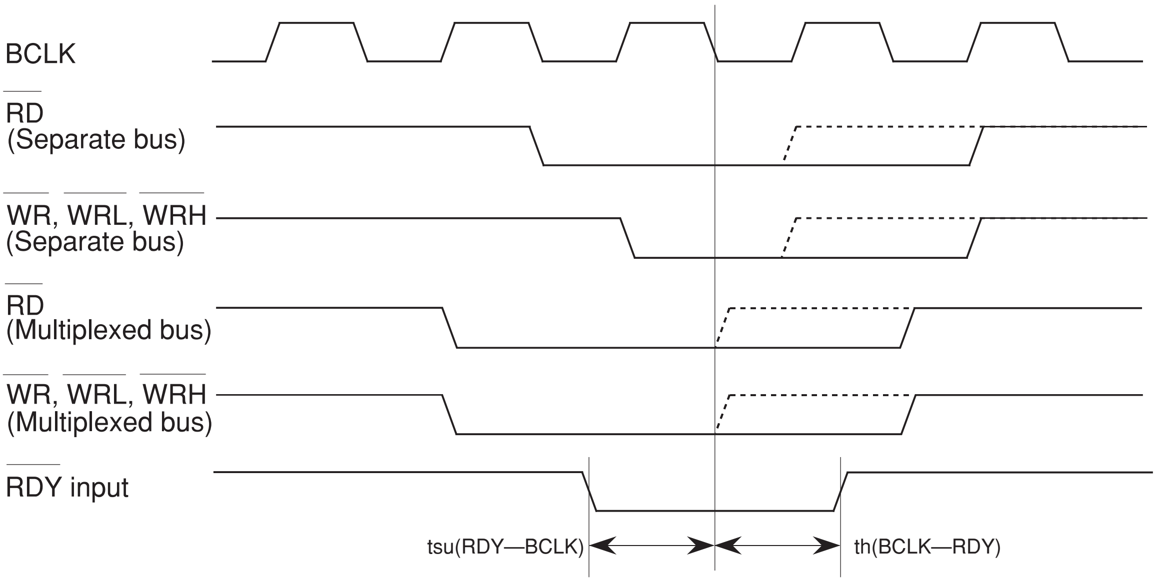


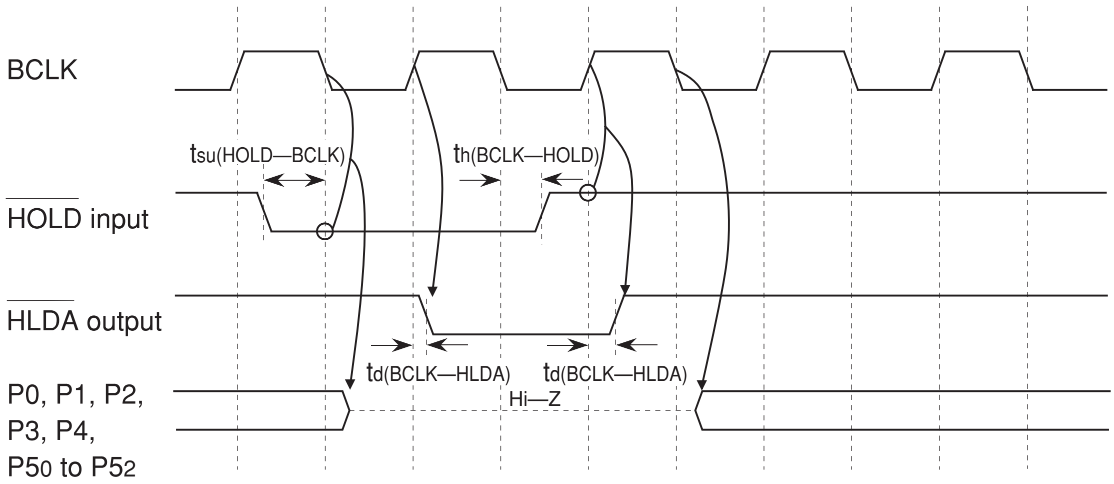
Figure 1.21.2 Timing Diagram (1)

Memory Expansion Mode and Microprocessor Mode

(Effective for setting with wait**)**



(Common to setting with wait and setting without wait**)**



Note: The above pins are set to high-impedance regardless of the input level of the BYTE pin, PM06 bit of PM0 register and PM11 bit of PM1 register.

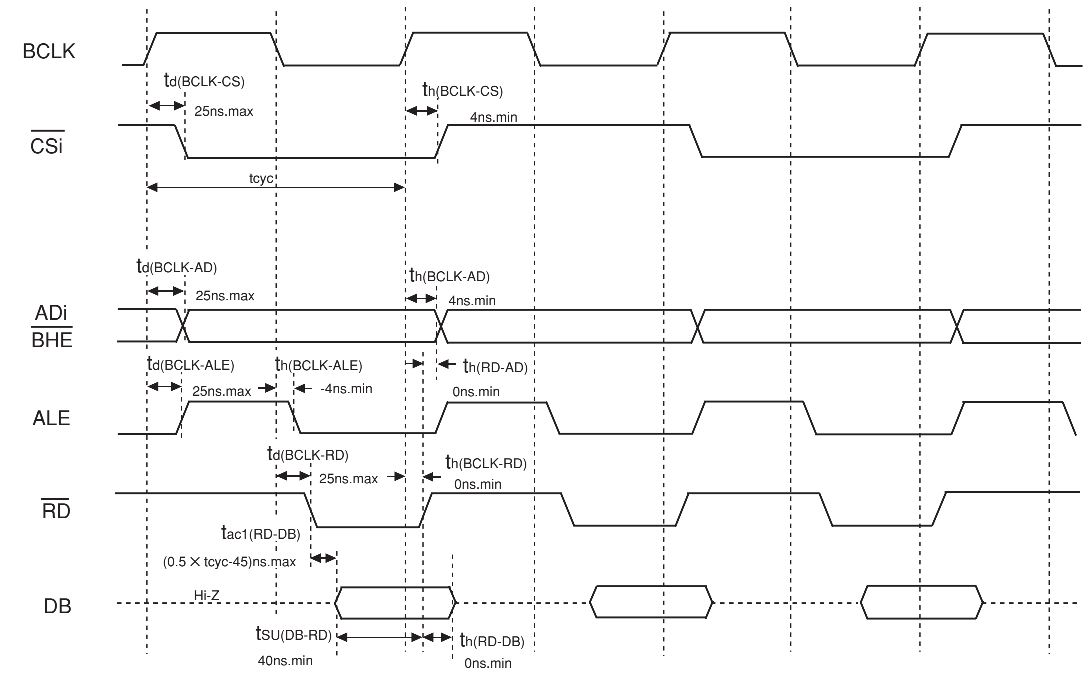
Measuring conditions :

- V_{CC} = 5 V
- Input timing voltage : Determined with V_{IL} = 1.0 V, V_{IH} = 4.0 V
- Output timing voltage: Determined with V_{OL} = 2.5 V, V_{OH} = 2.5 V

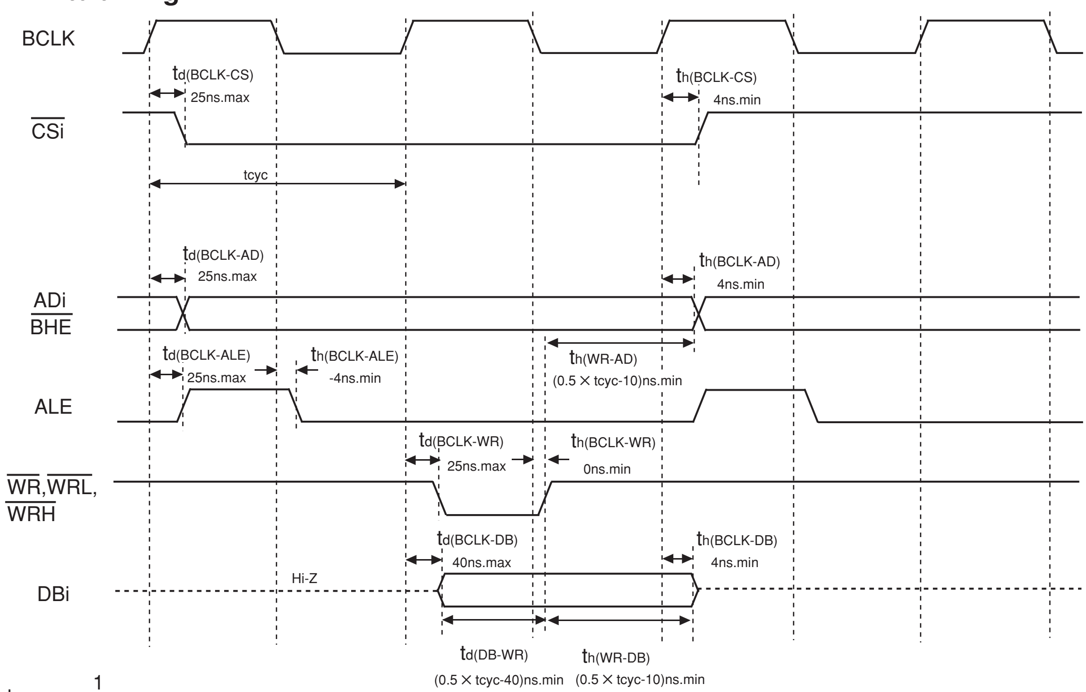
Figure 1.21.3 Timing Diagram (2)

Memory Expansion Mode and Microprocessor Mode
(For setting with no wait**)**

Read timing



Write timing



$$tcyc = \frac{1}{f(BCLK)}$$

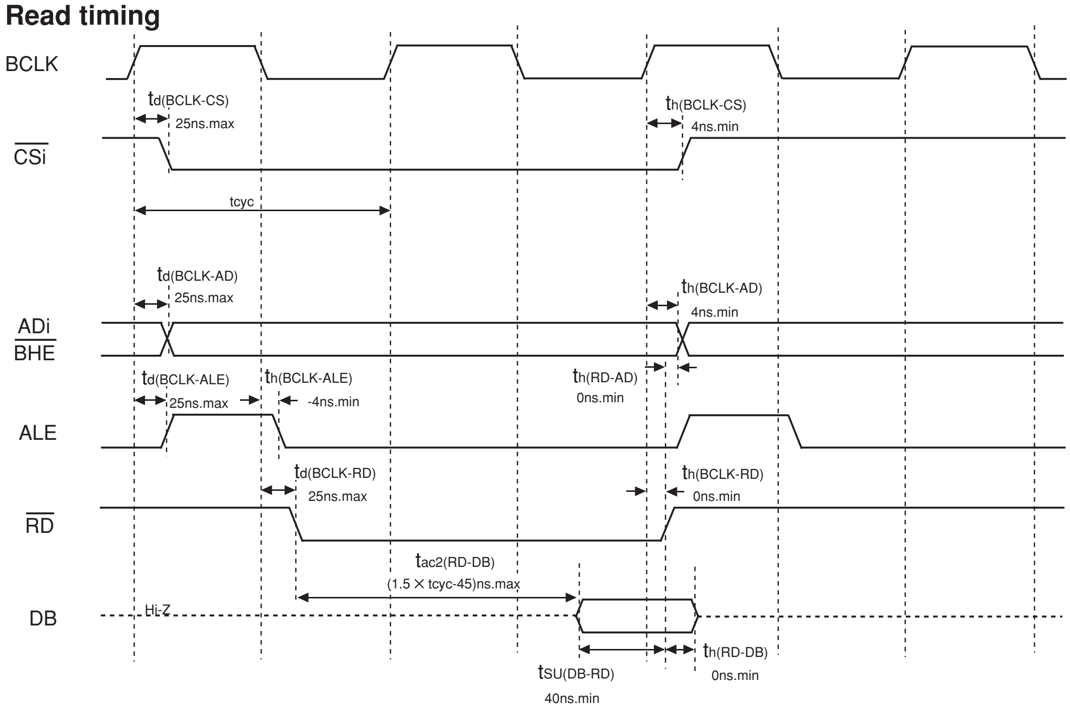
Measuring conditions :
- V_{CC} = 5 V
- Input timing voltage : V_{IL} = 0.8 V, V_{IH} = 2.0 V
- Output timing voltage : V_{OL} = 0.4 V, V_{OH} = 2.4 V

Figure 1.21.4 Timing Diagram (3)

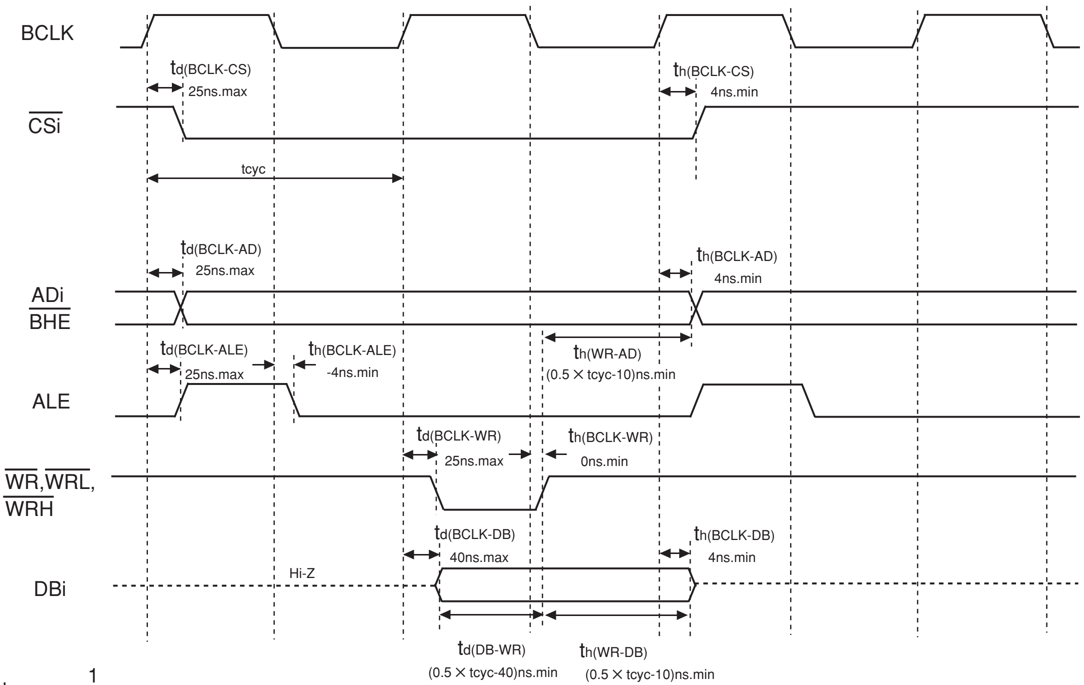
Memory Expansion Mode and Microprocessor Mode

(For 1-wait setting and external area access)

Read timing



Write timing



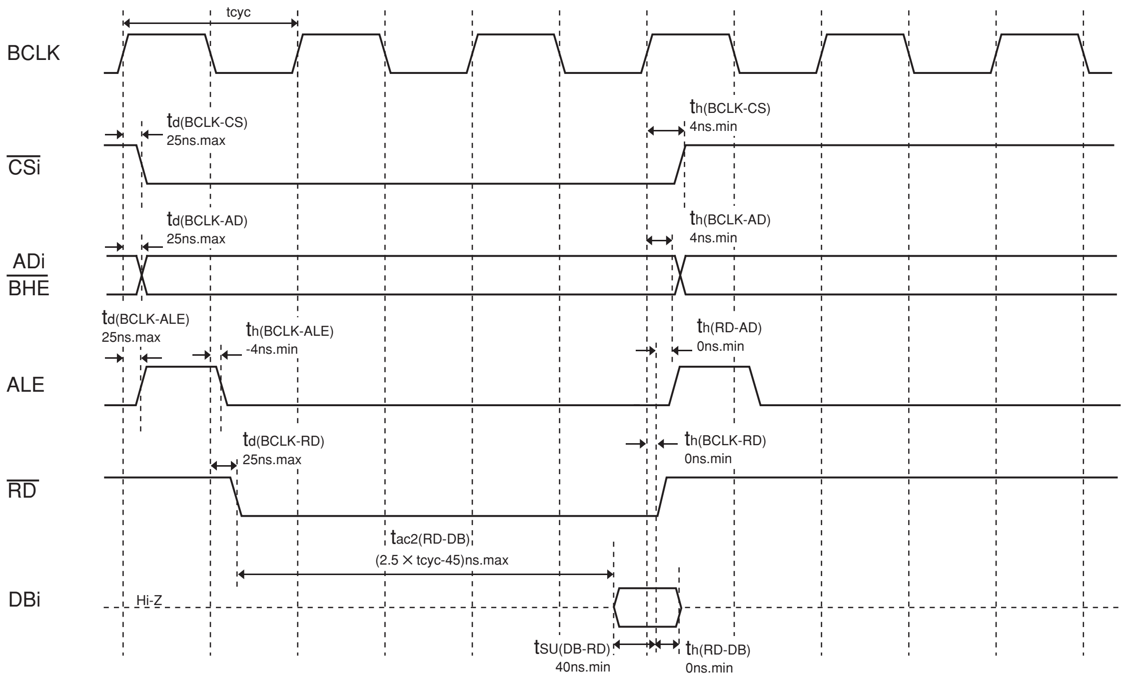
$$tcyc = \frac{1}{f(BCLK)}$$

Measuring conditions :
- V_{CC} = 5 V
- Input timing voltage : V_{IL} = 0.8 V, V_{IH} = 2.0 V
- Output timing voltage : V_{OL} = 0.4 V, V_{OH} = 2.4 V

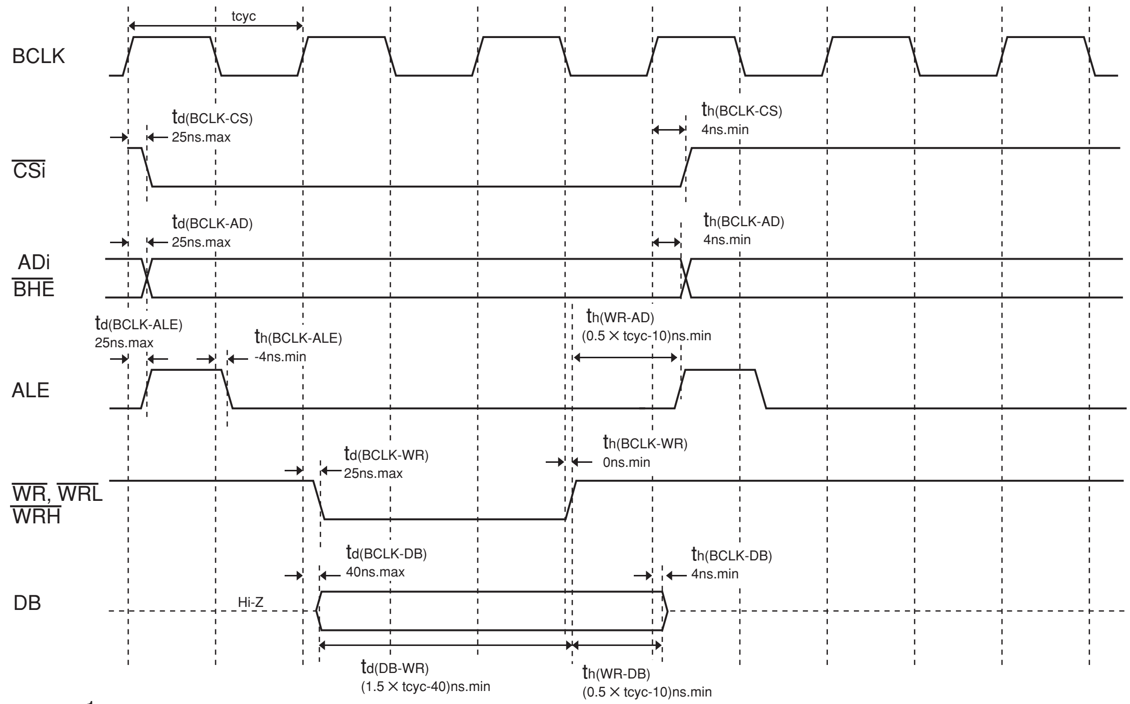
Figure 1.21.5 Timing Diagram (4)

Memory Expansion Mode and Microprocessor Mode
(For 2-wait setting and external area access)

Read timing



Write timing



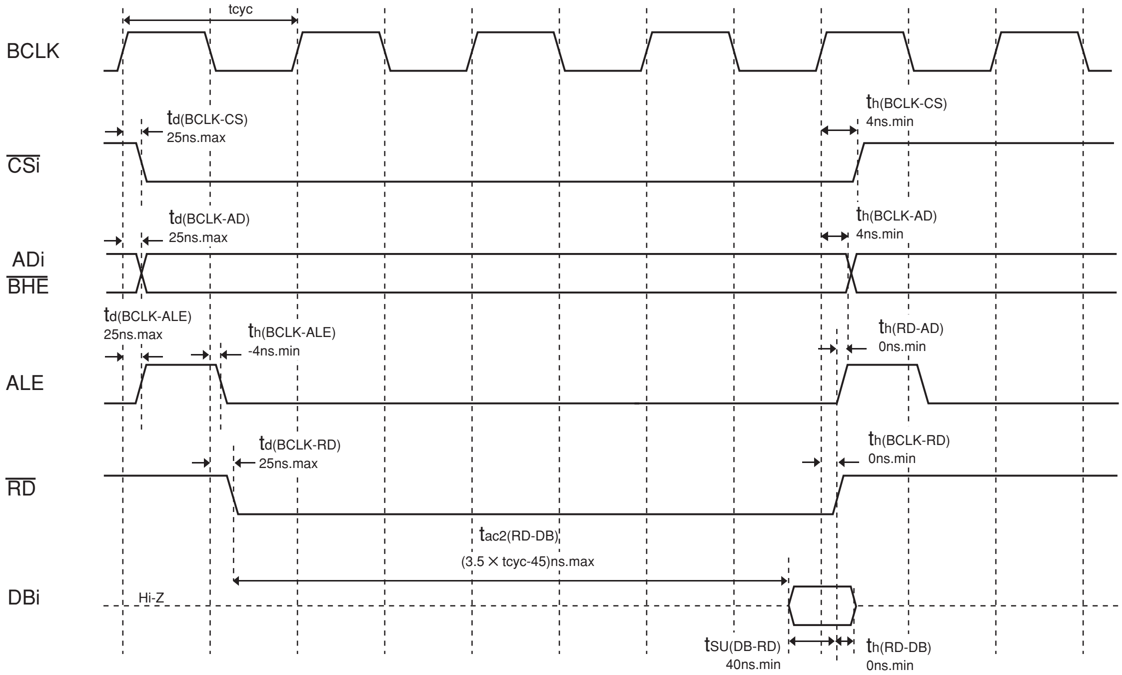
$$tcyc = \frac{1}{f(BCLK)}$$

Measuring conditions :
- V_{CC} = 5 V
- Input timing voltage : V_{IL} = 0.8 V, V_{IH} = 2.0 V
- Output timing voltage : V_{OL} = 0.4 V, V_{OH} = 2.4 V

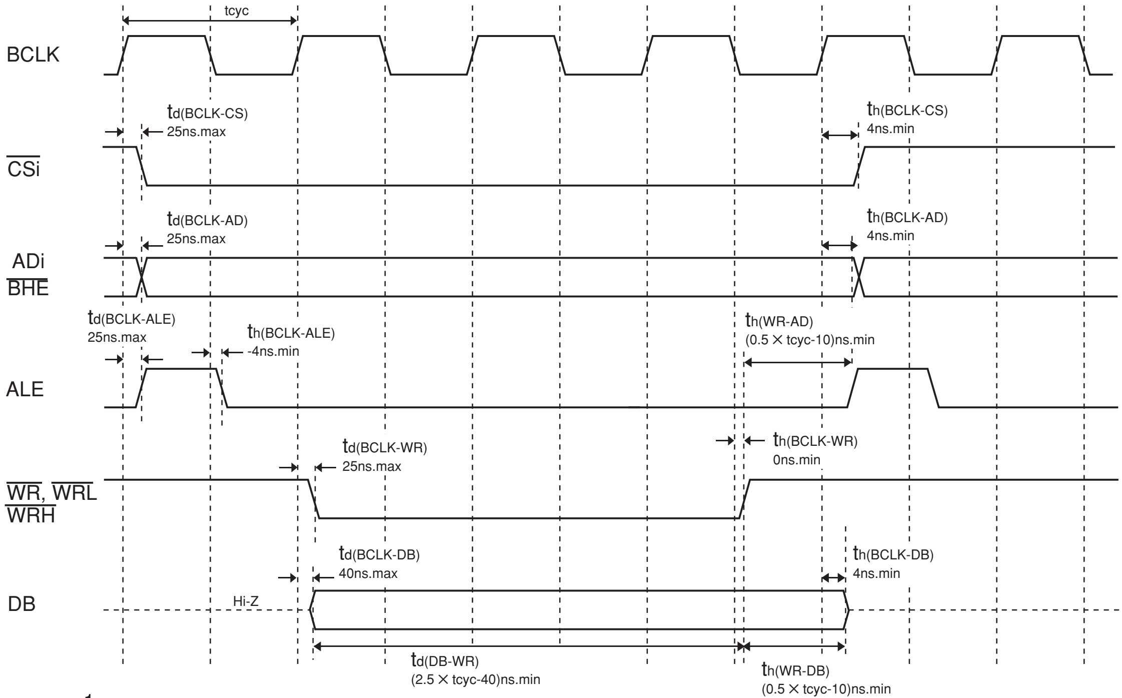
Figure 1.21.6 Timing Diagram (5)

Memory Expansion Mode and Microprocessor Mode
(For 3-wait setting and external area access)

Read timing



Write timing



$$tcyc = \frac{1}{f(BCLK)}$$

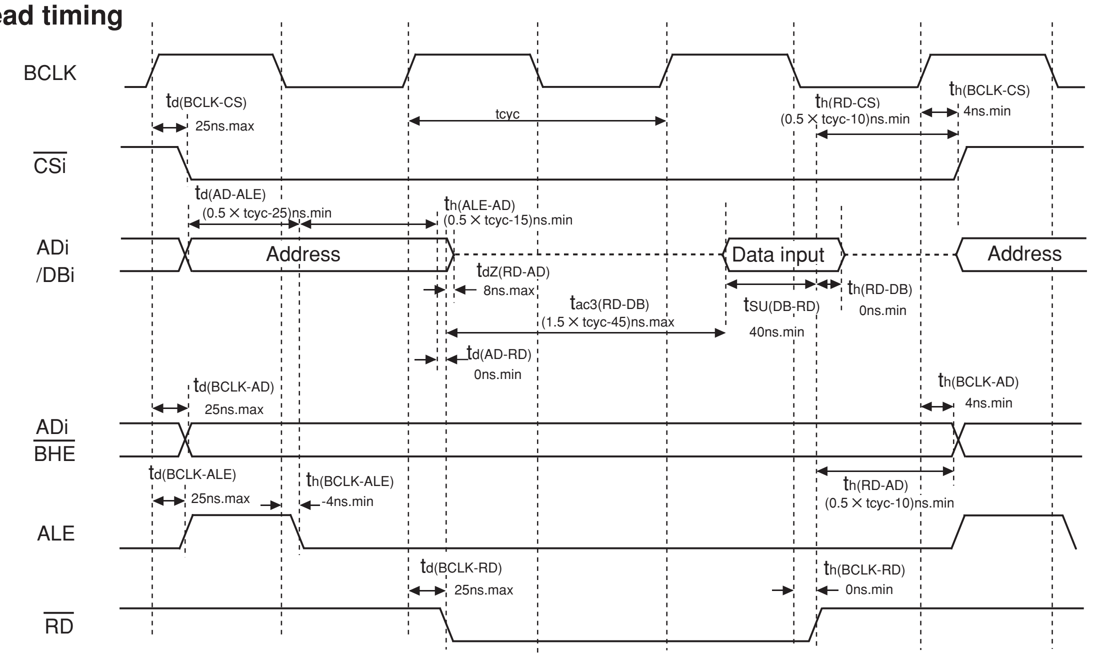
Measuring conditions :
- V_{CC} = 5 V
- Input timing voltage : V_{IL} = 0.8 V, V_{IH} = 2.0 V
- Output timing voltage : V_{OL} = 0.4 V, V_{OH} = 2.4 V

Figure 1.21.7 Timing Diagram (6)

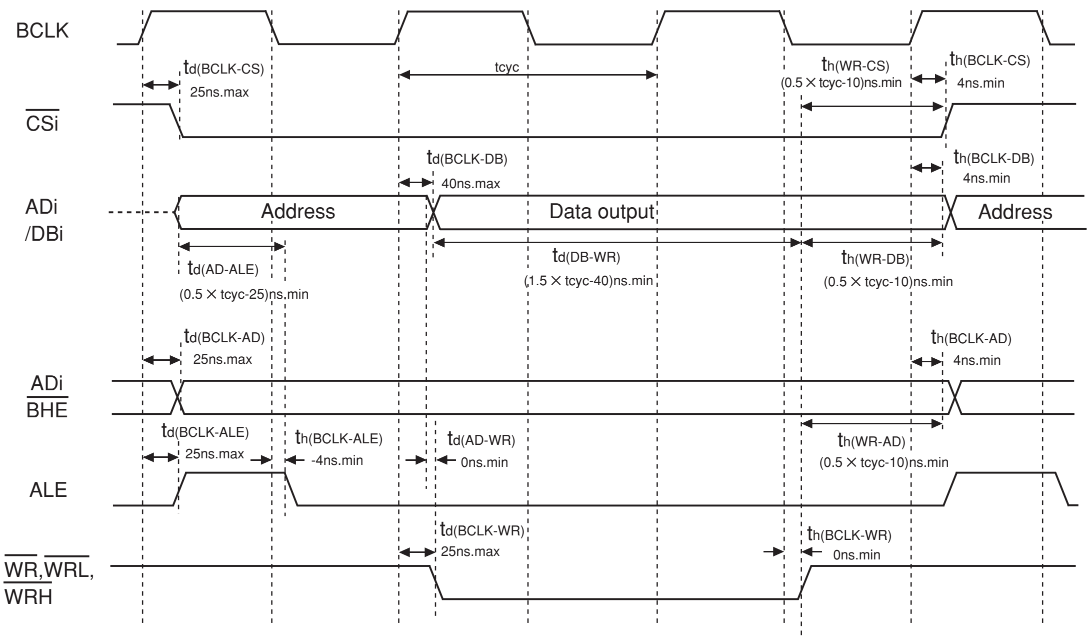
Memory Expansion Mode and Microprocessor Mode

(For 1- or 2-wait setting, external area access and multiplexed bus selection)

Read timing



Write timing



$$tcyc = \frac{1}{f(BCLK)}$$

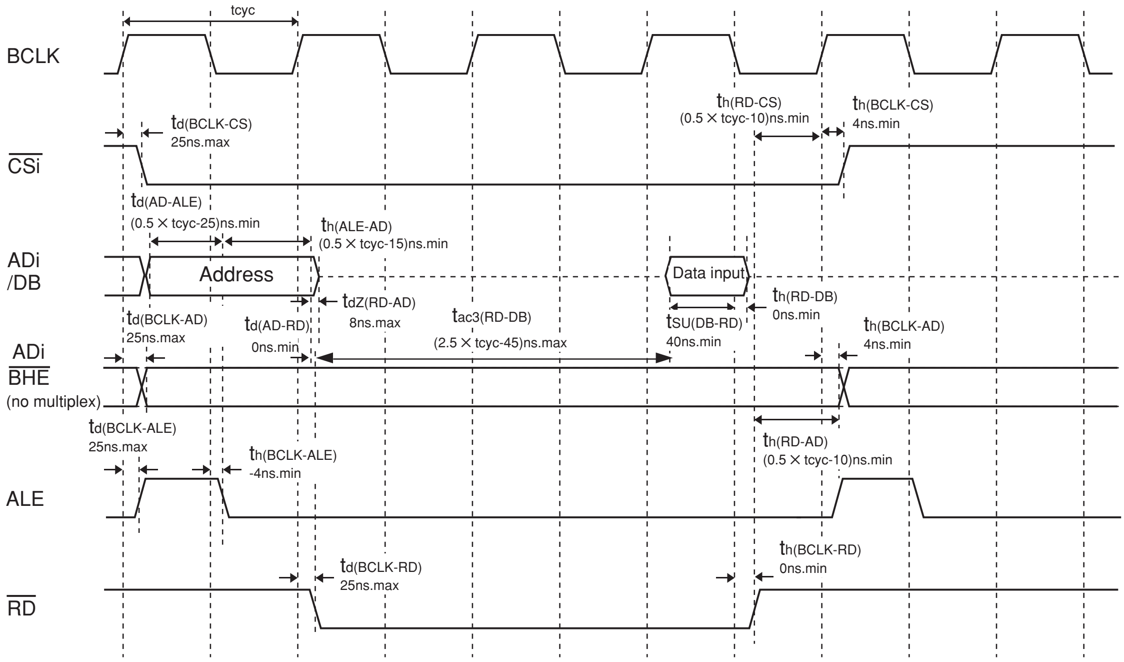
Measuring conditions :
- V_{CC} = 5 V
- Input timing voltage : V_{IL} = 0.8 V, V_{IH} = 2.0 V
- Output timing voltage : V_{OL} = 0.4 V, V_{OH} = 2.4 V

Figure 1.21.8 Timing Diagram (7)

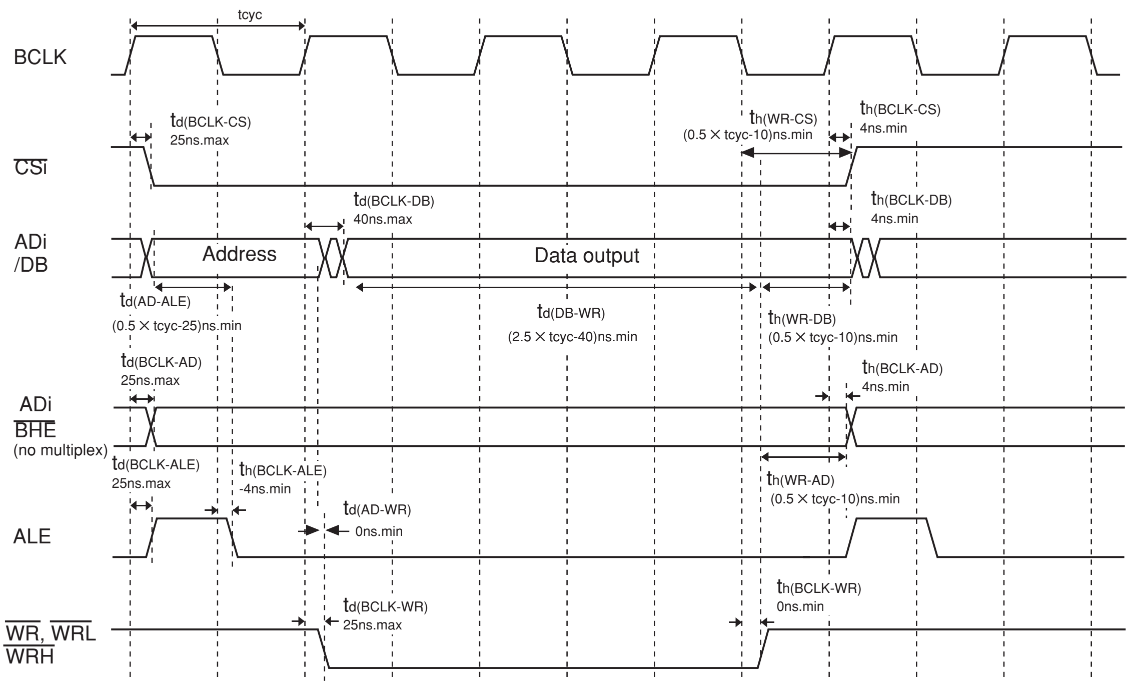
Memory Expansion Mode and Microprocessor Mode

(For 3-wait setting, external area access and multiplexed bus selection)

Read timing



Write timing



$$tcyc = \frac{1}{f(BCLK)}$$

Measuring conditions :
- V_{CC} = 5 V
- Input timing voltage : V_{IL} = 0.8 V, V_{IH} = 2.0 V
- Output timing voltage : V_{OL} = 0.4 V, V_{OH} = 2.4 V

Figure 1.21.9 Timing Diagram (8)

Flash Memory

Flash Memory Performance

The flash memory version is functionally the same as the mask ROM version except that it internally contains flash memory.

The flash memory version has four modes — CPU rewrite mode, standard serial I/O mode, parallel I/O mode and CAN I/O mode — in which its internal flash memory can be operated on.

Table 1.22.1 shows the outline performance of flash memory version (refer to "Table 1.1.1 Performance outline of M16C/6N4 Group" for the items not listed in Table 1.22.1). Table 1.22.2 shows the outline of flash memory rewrite mode.

Table 1.22.1 Flash Memory Version Specifications

Item		Specifications
Flash memory operating mode		4 modes (CPU rewrite, standard serial I/O, parallel I/O, CAN I/O)
Erase block division	User ROM area	Refer to "Figure 1.22.1 Flash Memory Block Diagram"
	Boot ROM area	1 block (4 Kbytes) (Note 1)
Method for program		In units of word, in units of byte (Note 2)
Method for erasure		Collective erase, block erase
Program, erase control method		Program and erase controlled by software command
Protect method		Protected for each block by lock bit
Number of commands		8 commands
Number of program and erasure (Note 3)		100 times
ROM code protection		Parallel I/O , standard serial I/O and CAN I/O modes are supported.

Note 1: The boot ROM area contains a standard serial I/O mode and CAN I/O mode rewrite control program which is stored in it when shipped from the factory. This area can only be rewritten in parallel I/O mode.
Note 2: Can be programmed in byte units in only parallel I/O mode.
Note 3: Definition of programming and erasure times
The programming and erasure times are defined to be per-block erasure times. For example, assume a case where a 4K-byte block A is programmed in 2,048 operations by writing one word at a time and erased thereafter. In this case, the block is reckoned as having been programmed and erased once.
If a product is guaranteed of 100 times of programming and erasure, each block in it can be erased up to 100 times.

Table 1.22.2 Flash Memory Rewrite Modes Overview

Flash memory rewrite mode	CPU rewrite mode (Note 1)	Standard serial I/O mode	Parallel I/O mode	CAN 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 (Note 2) 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 synchronous serial I/O Standard serial I/O mode 2: UART (Note 3)	The boot ROM and user ROM areas are rewritten by using a dedicated parallel programmer.	The user ROM area is rewritten by using a dedicated CAN programmer.
Areas which can be rewritten	User ROM area	User ROM area	User ROM area Boot ROM area	User ROM area
Operation mode	Single chip mode Memory expansion mode (EW0 mode) Boot mode (EW0 mode)	Boot mode	Parallel I/O mode	Boot mode
ROM programmer	None	Serial programmer	Parallel programmer	CAN programmer

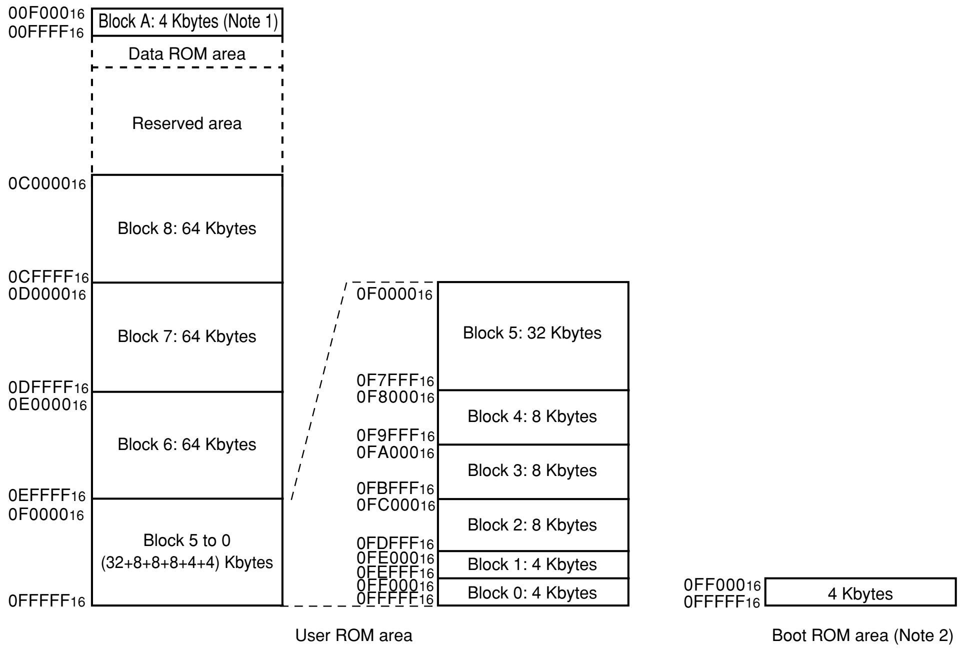
Note 1: The PM13 bit remains set to "1" while the FMR01 bit in the FMR0 register = 1 (CPU rewrite mode enabled). The PM13 bit is reverted to its original value by setting the FMR01 bit to "0" (CPU rewrite mode disabled). However, if the PM13 bit is changed during CPU rewrite mode, its changed value is not reflected until after the FMR01 bit is set to "0".
Note 2: When in CPU rewrite mode, the PM10 and PM13 bits in the PM1 register are set to "1". The rewrite control program can only be executed in the internal RAM or in an external area that is enabled for use when the PM13 bit = 1.
Note 3: When using the standard serial I/O mode 2, make sure a main clock input oscillation frequency is set to 5 MHz, 10 MHz or 16 MHz .

Memory Map

The ROM in the flash memory version is separated between a user ROM area and a boot ROM area. Figure 1.22.1 shows the block diagram of flash memory. The user ROM area has a 4-Kbyte block A, in addition to the area that stores a program for microcomputer operation during singe-chip or memory expansion mode.

The user ROM area is divided into several blocks, each of which can individually be protected (locked) against programming or erasure. The user ROM area can be rewritten in all of CPU rewrite, standard serial I/O mode, parallel I/O mode and CAN I/O mode. Block A is enabled for use by setting the PM1 register's PM10 bit to "1" (block A enabled, $\overline{CS_2}$ area at addresses 10000_{16} to $26FFF_{16}$).

The boot ROM area is located at addresses that overlap the user ROM area, and can only be rewritten in parallel I/O mode. After a hardware reset that is performed by applying a high-level signal to the CNV_{SS} and $P5_0$ pins and a low-level signal to the $P5_5$ pin, the program in the boot ROM area is executed. After a hardware reset that is performed by applying a low-level signal to the CNV_{SS} pin, the program in the user ROM area is executed (but the boot ROM area cannot be read).



* Shown here is a block diagram during single-chip mode.

Note 1: Block A can be made usable by setting the PM1 register's PM10 bit to "1" (block A enabled, $\overline{CS_2}$ area allocated at addresses 10000_{16} to $26FFF_{16}$).
Block A cannot be erased by the Erase All Unlocked Block command. Use the Block Erase command to erase it.
Note 2: The boot ROM area can only be rewritten in parallel input/output mode.
Note 3: To specify a block, use an even address in that block.

Figure 1.22.1 Flash Memory Block Diagram

Boot Mode

After a hardware reset which is performed by applying a low-level signal to the $P5_5$ pin and a high-level signal to the CNV_{SS} and $P5_0$ pins, the microcomputer is placed in boot mode, thereby executing the program in the boot ROM area.
During boot mode, the boot ROM and user ROM areas are switched over by the FMR05 bit in the FMR0 register.
The boot ROM area contains a standard serial I/O mode and CAN I/O mode based rewrite control program which was stored in it when shipped from the factory.
The boot ROM area can be rewritten in parallel input/output mode. Prepare an EW0 mode based rewrite control program and write it in the boot ROM area, and the flash memory can be rewritten as suitable for the system.

Functions to Prevent Flash Memory from Rewriting

To prevent the flash memory from being read or rewritten easily, parallel I/O mode has a ROM code protect and standard serial I/O mode and CAN I/O mode have 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 I/O mode. Figure 1.22.2 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 setting 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 I/O mode. Therefore, use standard serial I/O mode or other modes to rewrite the flash memory.

• ID Code Check Function

Use this function in standard serial I/O mode and CAN I/O 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.
Figure 1.22.3 shows the ID code store addresses.

ROM code protect control address

b7	b6	b5	b4	b3	b2	b1	b0
				1	1	1	1

Symbol	Address	Value when shipped
ROMCP	$0FFFFF_{16}$	FF_{16} (Note 1)

Bit symbol	Bit name	Function	RW
- (b3-b0)	Reserved bit	Set to "1"	RW
ROMCR	ROM code protect reset bit (Notes 1, 2)	b5 b4 0 0 : Removes protect 0 1 : 1 0 : Enables ROOMCP1 bit 1 1 :	RW RW
ROMCP1	ROM code protect level 1 set bit (Notes 1, 3, 4)	b7 b6 0 0 : 0 1 : Protect enabled 1 0 : 1 1 : Protect disabled	RW RW

Note 1: Once any of these bits is set 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}".
Note 2: If the ROMCR bits are set to "00_2", ROM code protect level 1 is removed. However, because the ROMCR bits cannot be modified during parallel I/O mode, they need to be modified in standard serial I/O or other modes.
Note 3: 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 4: The ROMCP1 bits are effective when the ROMCR bits are "01_2", "10_2" or "11_2".

Figure 1.22.2 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	Oscillation stop and re-oscillation detection/Watchdog timer vector
$0FFFF7_{16}$ to $0FFFF4_{16}$	ID6	$\overline{DBC}$ vector
$0FFFFB_{16}$ to $0FFFF8_{16}$	ID7	$\overline{NMI}$ vector
$0FFFFF_{16}$ to $0FFFFC_{16}$	ROMCP	Reset vector

4 bytes

Figure 1.22.3 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 1.22.1 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 1.22.3 lists the differences between EW0 and EW1 modes.

Table 1.22.3 EW0 Mode and EW1 Mode

Item	EW0 mode	EW1 mode
Operation mode	• Single chip mode • Memory expansion mode • Boot mode	Single chip mode
Areas in which a rewrite control program can be located	• User ROM area • Boot 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 (Note 2)	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 • Erase All Unlocked Block command Cannot be executed when the lock bit for any block in which a rewrite control program exists is set to “1” (unlocked) or the FMR0 register’s FMR02 bit is set to “1” (lock bit disabled) • 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 1)
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 1: Make sure no interrupts (except $\overline{\text{NMI}}$ and watchdog timer interrupts) and DMA transfers will occur.

Note 2: When in CPU rewrite mode, the PM10 and PM13 bits in the PM1 register are set to “1”. The rewrite control program can only be executed in the internal RAM or in an external area that is enabled for use when the PM13 bit = 1.

• **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 1.22.4 shows the FMR0 register and FMR1 register.

FMR00 Bit

This bit indicates the operating status of the flash memory. The bit is “0” when the Program, Erase, or Lock Bit program 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). During boot mode, make sure the FMR05 bit also is “1” (user ROM area access).

FMR02 Bit

The lock bit set for each block can be disabled by setting the FMR02 bit to “1” (lock bit disabled). (Refer to “Data Protect Function”.) The lock bits set are enabled by setting the FMR02 bit to “0”.

The FMR02 bit only disables the lock bit function and does not modify the lock bit data (lock bit status flag). However, if the Erase command is executed while the FMR02 bit is set to “1”, the lock bit data changes state from “0” (locked) to “1” (unlocked) after Erase is completed.

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. Setting the FMSTP bit to “1” makes the internal flash memory inaccessible. Therefore, make sure the FMSTP bit is modified in other than the flash memory area.

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 dissipation mode or ring oscillator low power dissipation mode

Figure 1.22.7 shows a flow chart to be followed before and after entering low power dissipation mode.

Note that when going to stop or wait mode, the FMR0 register does not need to be set because the power for the internal flash memory is automatically turned off and is turned back on again after returning from stop or wait mode.

FMR05 Bit

This bit switches between the boot ROM and user ROM areas during boot mode. Set this bit to “0” when accessing the boot ROM area (for read) or “1” (user ROM access) when accessing the user ROM area (for read, write, or erase).

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 set to “0”. For details, refer to “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 set to “0”. For details, refer to “Full Status Check”.

FMR11 Bit

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

FMR16 Bit

This is a read-only bit indicating the execution result of the Read Lock Bit Status command.

Figures 1.22.5 and 1.22.6 show the setting and resetting of EW0 mode and EW1 mode, respectively.

Flash memory control register 0

b7	b6	b5	b4	b3	b2	b1	b0
			0				

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

Bit symbol	Bit name	Function	RW
FMR00	RY/$\overline{BY}$ status flag	0 : Busy (being written or erased) (Note 1) 1 : Ready	RO
FMR01	CPU rewrite mode select bit (Note 2)	0 : Disables CPU rewrite mode 1 : Enables CPU rewrite mode	RW
FMR02	Lock bit disable select bit (Note 3)	0: Enables lock bit 1: Disables lock bit	RW
FMSTP	Flash memory stop bit (Notes 4, 5)	0 Enables flash memory operation 1: Stops flash memory operation (placed in low power dissipation mode, flash memory initialized)	RW
- (b4)	Reserved bit	Set to "0"	RW
FMR05	User ROM area select bit (Effective in only boot mode) (Note 4)	0 : Boot ROM area is accessed 1 : User ROM area is accessed	RW
FMR06	Program status flag (Note 6)	0 : Terminated normally 1 : Terminated in error	RO
FMR07	Erase status flag (Note 6)	0 : Terminated normally 1 : Terminated in error	RO

Note 1: This status includes writing or reading with the Lock Bit Program or Read Lock Bit Status command.
Note 2: To set this 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 this bit when the $\overline{NMI}$ pin is in the high state. Also, while in EW0 mode, make sure this bit is modified in other than the flash memory area.
To set this bit to "0", in a read array mode.
Note 3: 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 4: Make sure this bit is modified in other than the flash memory area.
Note 5: Effective when the FMR01 bit = 1 (CPU rewrite mode). If the FMR01 bit = 0, although the FMSTP bit can be set to "1" by writing "1" in a program, the flash memory is neither placed in low power mode nor initialized.
Note 6: This flag is set to "0" by executing the Clear Status command.

Flash memory control register 1

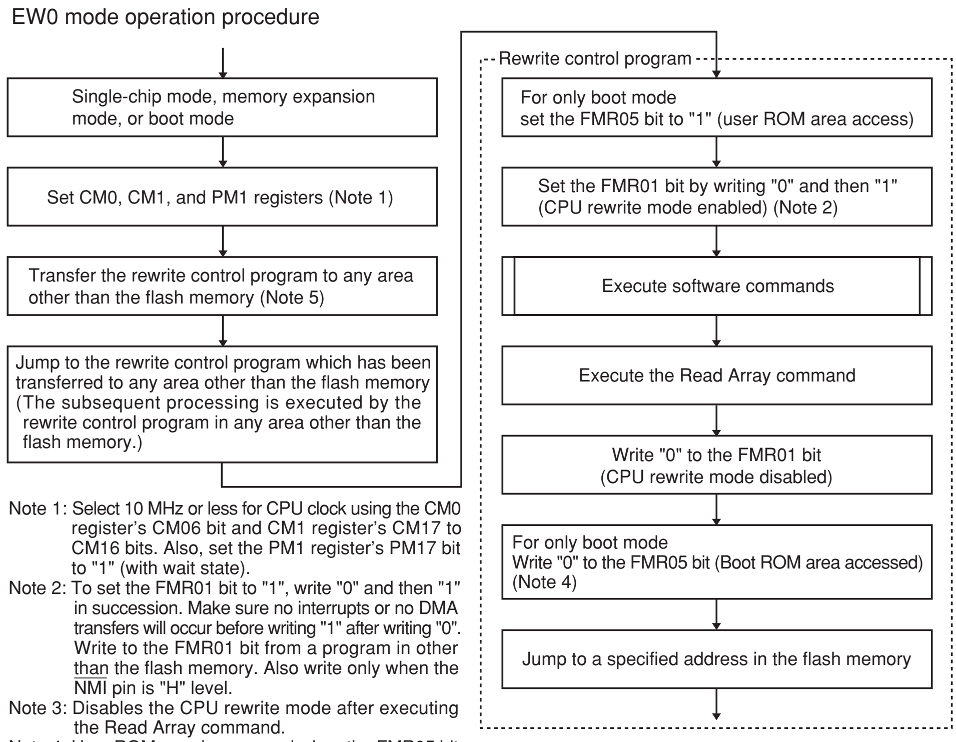
b7	b6	b5	b4	b3	b2	b1	b0
0		0	0				

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	Set to "0"	RW
FMR16	Lock bit status flag	0 : Lock 1 : Unlock	RO
- (b7)	Reserved bit	Set to "0"	RW

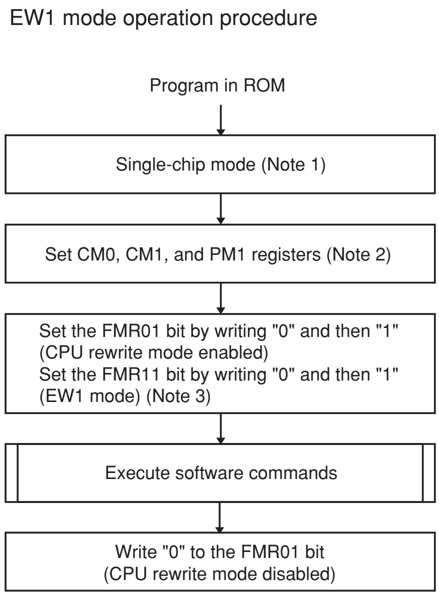
Note: To set this bit to "1", write "0" and then "1" in succession when the FMR01 bit = 1. Make sure no interrupts or no DMA transfers will occur before writing "1" after writing "0".
Write to this bit when the $\overline{NMI}$ pin is in the high state.
Note that the FMR01 and FMR11 bits both are set to "0" by setting the FMR01 bit to "0".

Figure 1.22.4 FMR0 Register and FMR1 Register



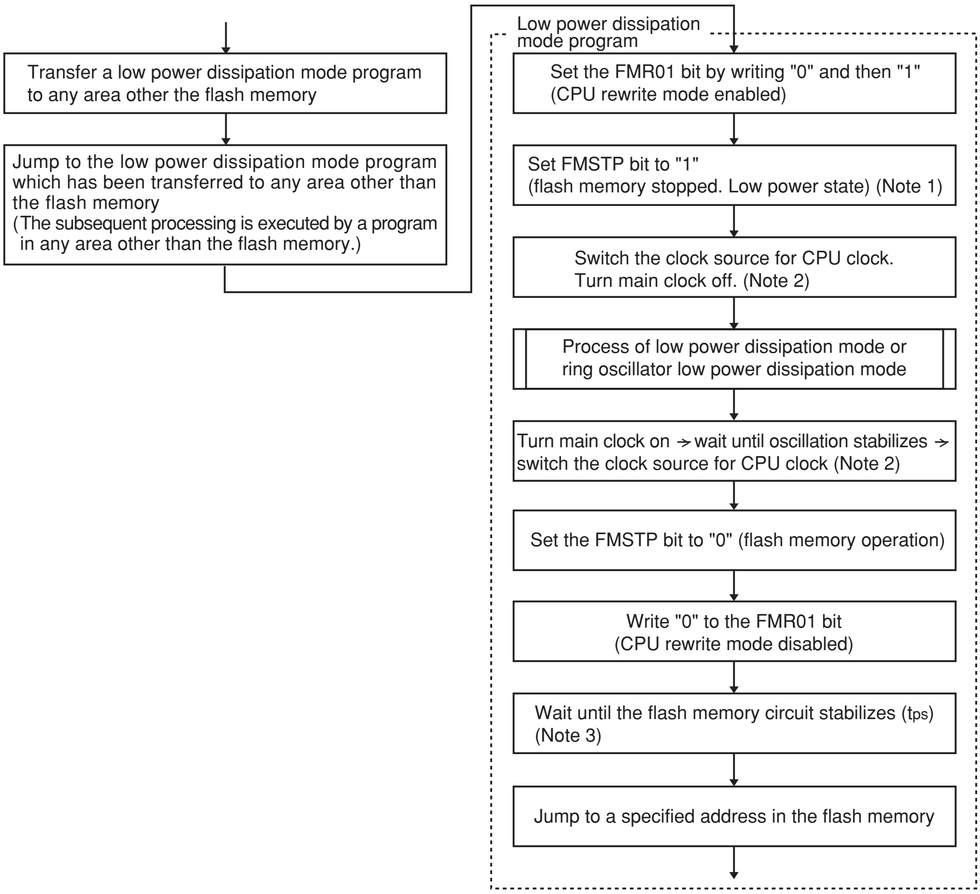
Note 1: Select 10 MHz or less for CPU clock using the CM0 register's CM06 bit and CM1 register's CM17 to CM16 bits. Also, set the PM1 register's PM17 bit to "1" (with wait state).
Note 2: To set the FMR01 bit to "1", write "0" and then "1" in succession. Make sure no interrupts or no DMA transfers will 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 $\overline{NMI}$ pin is "H" level.
Note 3: Disables the CPU rewrite mode after executing the Read Array command.
Note 4: User ROM area is accessed when the FMR05 bit is set to "1".
Note 5: When in CPU rewrite mode, the PM10 and PM13 bits in the PM1 register are set to "1". The rewrite control program can only be executed in the internal RAM or in an external area that is enabled for use when the PM13 bit = 1.

Figure 1.22.5 Setting and Resetting of EW0 Mode



Note 1: In EW1 mode, do not set the microcomputer in memory expansion or boot mode.
Note 2: Select 10 MHz or less for CPU clock using the CM0 register's CM06 bit and CM1 register's CM17 to CM16 bits. Also, set the PM1 register's PM17 bit to "1" (with wait state).
Note 3: 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". Also write only when the $\overline{NMI}$ pin is "H" level.

Figure 1.22.6 Setting and Resetting of EW1 Mode



Note 1: Set the FMSTP bit to "1" after setting the FMR01 bit to "1" (CPU rewrite mode).
Note 2: Before the clock source for CPU clock can be changed to main clock or sub clock, the clock to which to be changed must be stable.
Note 3: Insert t_{ps} wait time in a program. The flash memory cannot be accessed during this wait time.

Figure 1.22.7 Processing Before and After Low Power Dissipation 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 CPU clock 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 $\overline{NMI}$ 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 $\overline{NMI}$ 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 $\overline{NMI}$ 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 $\overline{NMI}$ 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 no DMA transfers will occur before writing “1” after writing “0”. Also only when $\overline{NMI}$ pin is “H” level.

(5) Writing in 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, parallel I/O or CAN 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
```

(10) Low Power Dissipation Mode and Ring Oscillator Low Power Dissipation Mode

If the CM05 bit is set to "1" (main clock stop), the following commands must not be executed.

- Program
- Block erase
- Erase all unlocked blocks
- Lock bit program

Software Commands

Software commands are described below. The command code and data must be read and written in 16-bit unit, to and from even addresses in the user ROM area. When writing command code, the high-order 8 bits (D_{15} to D_8) are ignored. Table 1.22.4 lists the software commands.

Table 1.22.4 Software Commands

Software command	First bus cycle			Second bus cycle		
	Mode	Address	Data (D_{15} to D_0)	Mode	Address	Data (D_{15} to D_0)
Read array	Write	×	$xxFF_{16}$	-	-	-
Read status register	Write	×	$xx70_{16}$	Read	×	SRD
Clear status register	Write	×	$xx50_{16}$	-	-	-
Program	Write	WA	$xx40_{16}$	Write	WA	WD
Block erase	Write	×	$xx20_{16}$	Write	BA	$xxD0_{16}$
Erase all unlocked block (Note 1)	Write	×	$xxA7_{16}$	Write	×	$xxD0_{16}$
Lock bit program	Write	BA	$xx77_{16}$	Write	BA	$xxD0_{16}$
Read lock bit status	Write	×	$xx71_{16}$	Write	BA	$xxD0_{16}$(Note 2)

Note 1: It is only blocks 0 to 8 that can be erased by the Erase All Unlocked Block command.
Block A cannot be erased. Use the Block Erase command to erase block A.

Note 2: Note that the commands in the second bus cycle are different from those of the existing M16C/6N0 group.
The lock bit status is output to the FMR16 bit of the FMR1 register. Read this bit: “0” (locked), “1” (unlocked)

SRD: Status register data (D_7 to D_0)

WA: Write address (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.)

WD: Write data (16 bits)

BA: Uppermost block address (even address, however)

×: Any even address in the user ROM area

x: High-order 8 bits of command (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 unit.

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

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.

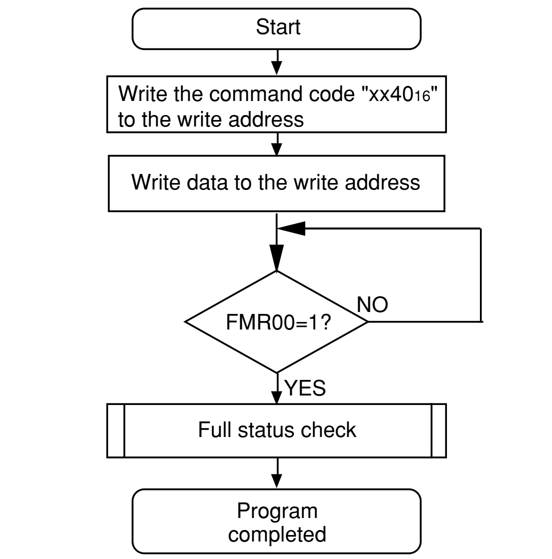
Clear Status Register Command (50_{16})

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 set to “0”.

Program Command (40_{16})

This command writes data to the flash memory in 1-word (2-byte) unit.
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”.)
Figure 1.22.8 shows an example of program flowchart.
Note that each block can be disabled from being programmed by a lock bit. (Refer to “Data Protect Function”.)
Be careful not to write over already programmed addresses.

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 set 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.



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

Figure 1.22.8 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 erasing is completed.
Check the FMR0 register’s FMR07 bit after auto erasing has finished, and the result of auto erasing can be known. (Refer to “Full Status Check”.)
Figure 1.22.9 shows an example of a block erase flowchart.
Each block can be protected against erasing by a lock bit. (Refer to “Data Protect Function”.)
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 set to “0” at the same time auto erasing starts, and set back to “1” when auto erasing finishes. In this case, the microcomputer remains in read status register mode until the Read Array or Read Lock Bit Status command is written next.

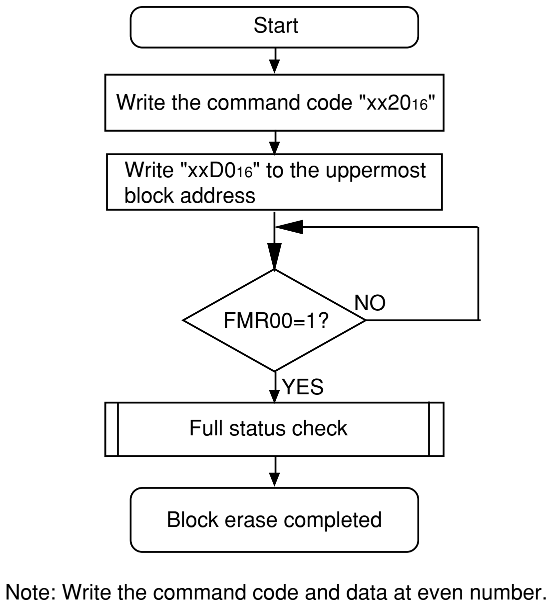


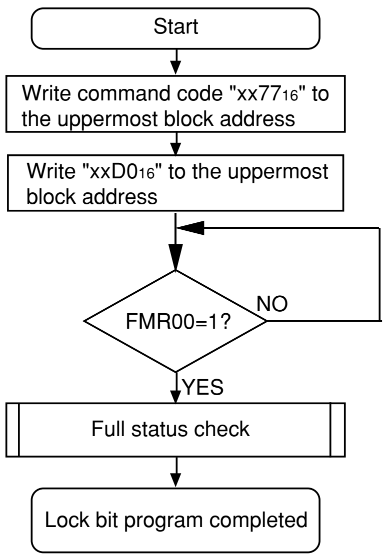
Figure 1.22.9 Block Erase Command

Erase All Unlocked Block

Write “xxA7$_{16}$” in the first bus cycle and write “xxD0$_{16}$” in the second bus cycle, and all blocks except block A will be erased successively, one block at a time.
Check the FMR0 register’s FMR00 bit to see if auto erasing has finished. The result of the auto erase operation can be known by inspecting the FMR0 register’s FMR07 bit.
Each block can be protected against erasing by a lock bit. (Refer to “Data Protect Function”.)
In EW1 mode, do not execute this command when the lock bit for any block = 1 (unlocked) in which the rewrite control program is stored, or when the FMR0 register’s FMR02 bit = 1 (lock bit disabled).
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 set 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.
Note that only blocks 0 to 8 can be erased by the Erase All Unlocked Block command. Block A cannot be erased. Use the Block Erase command to erase block A.

Lock Bit Program Command (77$_{16}$/D0$_{16}$)

This command sets the lock bit for a specified block to “0” (locked).
Write “xx77$_{16}$” in the first bus cycle and write “xxD0$_{16}$” to the uppermost address of a block (even address, however) in the second bus cycle, and the lock bit for the specified block is set to “0”. Make sure the address value specified in the first bus cycle is the same uppermost block address that is specified in the second bus cycle.
Figure 1.22.10 shows an example of a lock bit program flowchart.
The lock bit status (lock bit data) can be read using the Read Lock Bit Status command.
Check the FMR0 register’s FMR00 bit to see if writing has finished.
For details about the lock bit function, and on how to set the lock bit to “1”, refer to “Data Protect Function”.



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

Figure 1.22.10 Lock Bit Program Command

Read Lock Bit Status Command (71_{16})

This command reads the lock bit status of a specified block.

Write "$xx71_{16}$" in the first bus cycle and write "$xxD0_{16}$" to the uppermost address of a block (even address, however) in the second bus cycle, and the lock bit status of the specified block is stored in the FMR1 register's FMR16 bit. Read the FMR16 bit after the FMR0 register's FMR00 bit is set to "1" (ready).

Figure 1.22.11 shows an example of a read lock bit status flowchart.

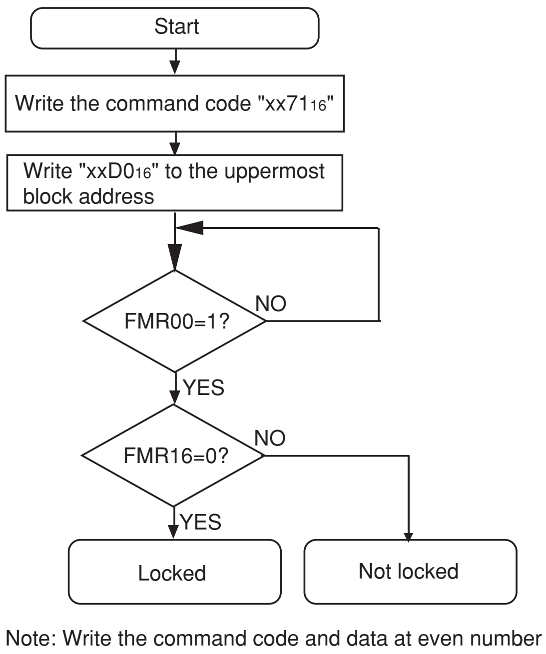


Figure 1.22.11 Read Lock Bit Status Command

Data Protect Function

Each block in the flash memory has a nonvolatile lock bit. The lock bit is effective when the FMR02 bit = 0 (lock bit enabled). The lock bit allows each block to be individually protected (locked) against programming and erasure. This helps to prevent data from inadvertently written to or erased from the flash memory. The following shows the relationship between the lock bit and the block status.

- When the lock bit = 0, the block is locked (protected against programming and erasure).
- When the lock bit = 1, the block is not locked (can be programmed or erased).

The lock bit is set to “0” (locked) by executing the Lock Bit Program command, and is set to “1” (unlocked) by erasing the block. The lock bit cannot be set to “1” by a command.
The lock bit status can be read using the Read Lock Bit Status command

The lock bit function is disabled by setting the FMR02 bit to “1”, with all blocks placed in an unlocked state. (The lock bit data itself does not change state.) Setting the FMR02 bit to “0” enables the lock bit function (lock bit data retained).
If the Block Erase or Erase All Unlocked Block command is executed while the FMR02 bit = 1, the target block or all blocks are erased irrespective of how the lock bit is set. The lock bit for each block is set to “1” after completion of erasure.
For details about the commands, refer to “Software Commands.”

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 1.22.5 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, auto erase, and lock bit write, and 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 1.22.5 Status Register

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	-	-	-

Note: The FMR07 bit (SR5) and FMR06 bit (SR4) are set to "0" by executing the Clear Status Register command. When the FMR07 bit (SR5) or FMR06 bit (SR4) = 1, the Program, Block Erase, Erase All Unlocked Block, and Lock Bit Program commands are not accepted.

D_7 to D_0: Indicates the data bus which is read out when the Read Status Register command is executed.

Full Status Check

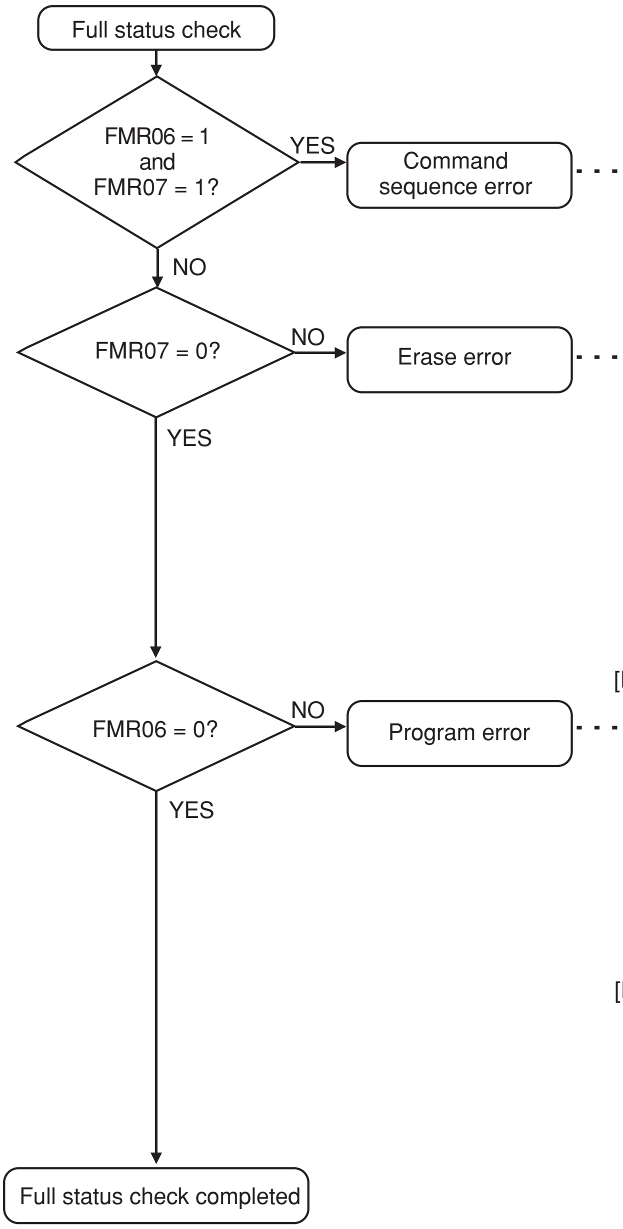
When an error occurs, the FMR0 register's FMR06 or FMR07 bits are set to "1", indicating occurrence of each specific error. Therefore, execution results can be verified by checking these status bits (full status check). Table 1.22.6 lists errors and FMR0 register status. Figure 1.22.12 shows a full status check flowchart and the action to be taken when each error occurs.

Table 1.22.6 Errors and FMR0 Register Status

FRM00 register (status register) status		Error	Error occurrence 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 Lock Bit Program, Block Erase, or Erase All Unlocked Block command (i.e., other than "xxD0₁₆" or "xxFF₁₆") (Note 1)
1	0	Erase error	•When the Block Erase command was executed on locked blocks (Note 2) •When the Block Erase or Erase All Unlocked Block command was executed on unlocked blocks but the blocks were not automatically erased correctly
0	1	Program error	•When the Block Erase command was executed on locked blocks (Note 2) •When the Program command was executed on unlocked blocks but the blocks were not automatically programmed correctly. •When the Lock Bit Program command was executed but not programmed correctly

Note 1: Writing "xxFF$_{16}$" 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.

Note 2: When the FMR02 bit of FMR0 register = 1 (lock bit disabled), no error will occur under this condition.



Command sequence error:

(1) Execute the Clear Status Register command to set status flags to "0".
(2) Reexecute the command after checking that it is entered correctly.

Erase error:

(1) Execute the Clear Status Register command to set the erase status flag to "0".
(2) Execute the Read Lock Bit Status command to see if the lock bit for the block in error is "0". If so, set the FMR0 register s FMR02 bit to "1".
(3) Reexecute the Block Erase Command or Erase All Unlocked Block command.

Note 1: If the error still occurs, the block in error cannot be used.
Furthermore, if the lock bit = 1 in (2) above, the block in error cannot be used either.

Program error:

[During programming]
(1) Execute the Clear Status Register command to set the erase status flag to "0" .
(2) Execute the Read Lock Bit Status command to see if the lock bit for the block in error is "0". If so, set the FMR0 register s FMR02 bit to "1".
(3) Reexecute the Program command.

Note 2: If the error still occurs, the block in error cannot be used.
Furthermore, if the lock bit = 1 in (2) above, the block in error cannot be used either.

[During lock bit programming]
(1) Execute the Clear Status Register command to set the erase status flag to "0" .
(2) Set the FMR0 register s FMR02 bit to "1".
(3) Execute the Block Erase command to erase the block in error.
(4) Reexecute the Lock Bit command.

Note 3: If the error still occurs, the block in error cannot be used.

Note 4: If FMR06 or FMR07 = 1, any of the Program, Block Erase, Erase All Unlocked Block, Lock Bit Program, or Read Lock Bit Status command is not accepted. Execute the Clear Status Register command before executing those commands.

Figure 1.22.12 Full Status Check and Handling Procedure for Each Error

Standard Serial I/O Mode

In standard serial I/O mode, the user ROM area can be rewritten while the microcomputer is mounted on-board by using a serial programmer suitable for the M16C/6N4 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 1.22.7 lists pin functions for standard serial I/O mode. Figures 1.22.13 shows pin connections for standard serial I/O mode.

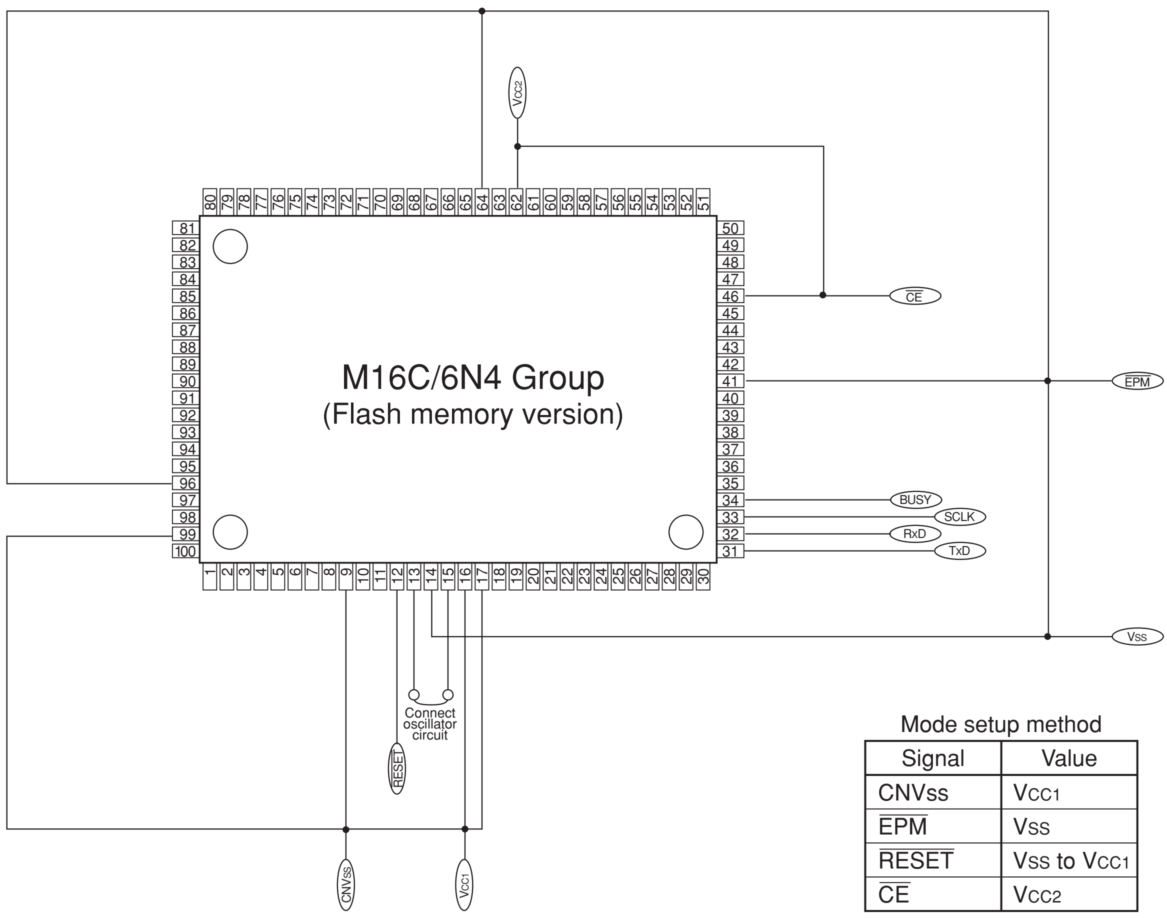
ID Code Check Function

This function determines whether the ID codes sent from the serial programmer and those written in the flash memory match. (Refer to "Functions to Prevent Flash Memory from Rewriting".)

Table 1.22.7 Pin Functions for Standard Serial I/O Mode

Pin	Name	I/O	Description
V_{CC}, V_{SS}	Power input		Apply the voltage guaranteed for Program and Erase to V_{CC} pin and 0 V to V_{SS} pin.
CNV_{SS}	CNV_{SS}	I	Connect to V_{CC} pin.
$\overline{RESET}$	Reset input	I	Reset input pin. While $\overline{RESET}$ pin is "L" level, input 20 cycles or longer clock to X_{IN} pin.
X_{IN}	Clock input	I	Connect a ceramic resonator or crystal oscillator between X_{IN} and X_{OUT} pins. To input an externally generated clock, input it to X_{IN} pin and open X_{OUT} pin.
X_{OUT}	Clock output	O	
BYTE	BYTE	I	Connect this pin to V_{CC} or V_{SS}.
AV_{CC}, AV_{SS}	Analog power supply input		Connect AV_{SS} to V_{SS} and AV_{CC} to V_{CC}, respectively.
V_{REF}	Reference voltage input	I	Enter the reference voltage for A-D and D-A converters from this pin.
$P0_0$ to $P0_7$	Input port P0	I	Input “H” or “L” level signal or open.
$P1_0$ to $P1_7$	Input port P1	I	Input “H” or “L” level signal or open.
$P2_0$ to $P2_7$	Input port P2	I	Input “H” or “L” level signal or open.
$P3_0$ to $P3_7$	Input port P3	I	Input “H” or “L” level signal or open.
$P4_0$ to $P4_7$	Input port P4	I	Input “H” or “L” level signal or open.
$P5_0$	$\overline{CE}$ input	I	Input “H” level signal.
$P5_1$ to $P5_4$, $P5_6$, $P5_7$	Input port P5	I	Input “H” or “L” level signal or open.
$P5_5$	$\overline{EPM}$ input	I	Input “L” level signal.
$P6_0$ to $P6_3$	Input port P6	I	Input “H” or “L” level signal or open.
$P6_4/\overline{RTS_1}$	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.
$P6_5/CLK_1$	SCLK input	I	Standard serial I/O mode 1: Serial clock input pin. Standard serial I/O mode 2: Input “L”.
$P6_6/RxD_1$	RxD input	I	Serial data input pin
$P6_7/TxD_1$	TxD output	O	Serial data output pin (Note)
$P7_0$ to $P7_7$	Input port P7	I	Input “H” or “L” level signal or open.
$P8_0$ to $P8_4$, $P8_6$, $P8_7$	Input port P8	I	Input “H” or “L” level signal or open.
$P8_5/\overline{NMI}$	$\overline{NMI}$ input	I	Connect this pin to V_{CC}.
$P9_0$ to $P9_4$, $P9_7$	Input port P9	I	Input “H” or “L” level signal or open.
$P9_5/CRx_0$	CRx input	I	Input “H” or “L” level signal or connect to a CAN transceiver.
$P9_6/CTx_0$	CTx output	O	Input “H” level signal, open or connect to a CAN transceiver.
$P10_0$ to $P10_7$	Input port P10	I	Input “H” or “L” level signal or open.

Note: When using standard serial input/output mode 1, the TxD pin must be held high while the $\overline{RESET}$ pin is pulled low. Therefore, connect this pin to V_{CC} 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.



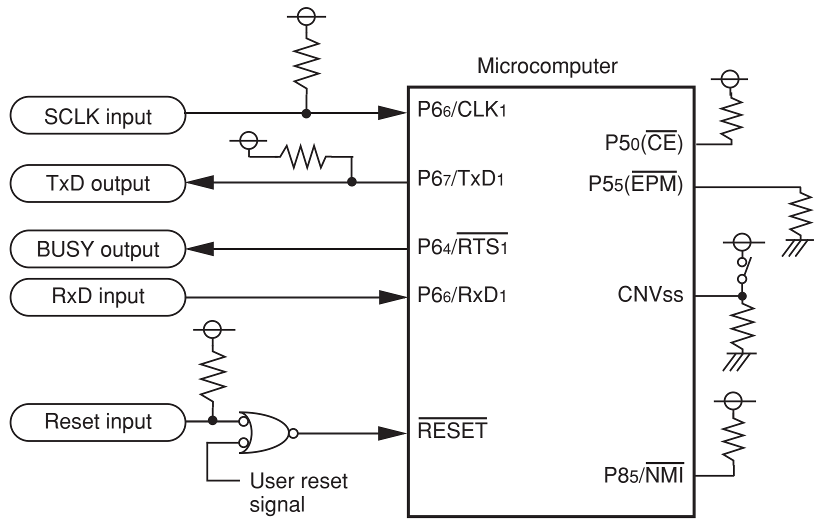
Mode setup method

Signal	Value
CNVss	VCC1
$\overline{EPM}$	VSS
$\overline{RESET}$	VSS to VCC1
$\overline{CE}$	VCC2

Figure 1.22.13 Pin Connections for Serial I/O Mode

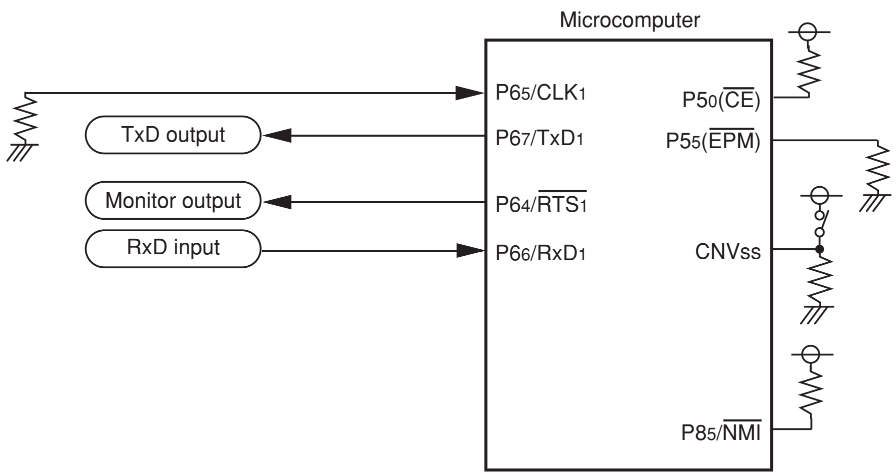
Example of Circuit Application in Standard Serial I/O Mode

Figures 1.22.14 and 1.22.15 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. Note that when using the standard serial I/O mode 2, make sure a main clock input oscillation frequency is set to 5 MHz, 10 MHz or 16 MHz .



- Control pins and external circuitry will vary according to programmer. For more information, refer to the programmer manual.
- In this example, modes are switched between single-chip mode and standard serial input/output mode by controlling the CNVss input with a switch.
- 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 1.22.14 Circuit Application in Standard Serial I/O Mode 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 1.22.15 Circuit Application in Standard Serial I/O Mode 2

Parallel I/O Mode

In parallel I/O mode, the user ROM and boot ROM areas can be rewritten by using a parallel programmer suitable for the M16C/6N4 group. For more information about parallel programmers, contact the manufacturer of your parallel programmer. For details on how to use, refer to the user's manual included with your parallel programmer.

User ROM and Boot ROM Areas

In the boot ROM area, an erase block operation is applied to only one 4-Kbyte block. The boot ROM area contains a standard serial I/O and CAN I/O modes based rewrite control program which was written in it when shipped from the factory. Therefore, when using a serial programmer or a CAN programmer, be careful not to rewrite the boot ROM area.
When in parallel I/O mode, the boot ROM area is located at addresses $0FF000_{16}$ to $0FFFFF_{16}$. When rewriting the boot ROM area, make sure that only this address range is rewritten. (Do not access other than the addresses $0FF000_{16}$ to $0FFFFF_{16}$.)

ROM Code Protect Function

The ROM code protect function inhibits the flash memory from being read or rewritten. (Refer to "Functions to Prevent Flash Memory from Rewriting".)

CAN I/O Mode

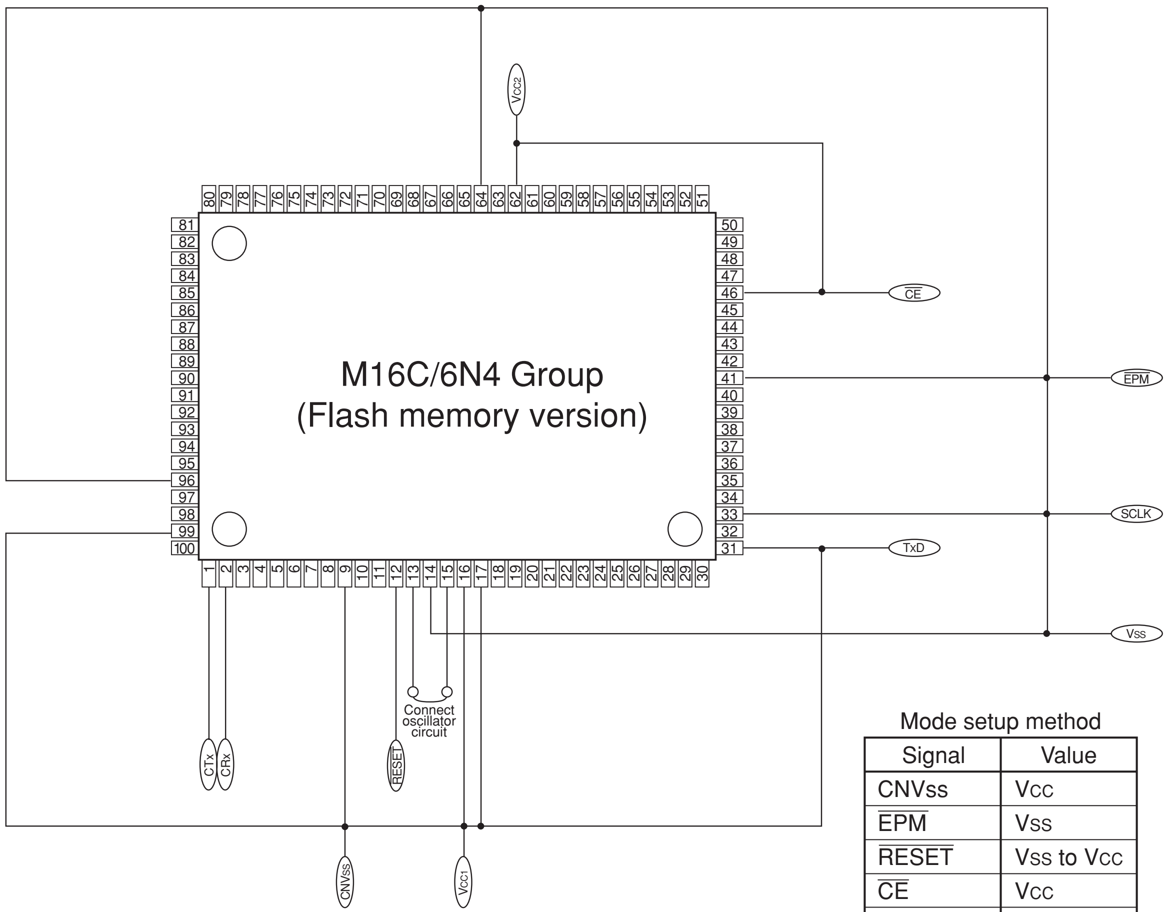
In CAN I/O mode, the user ROM area can be rewritten while the microcomputer is mounted on-board by using a CAN programmer suitable for the M16C/6N4 group. For more information about CAN programmers, contact the manufacturer of your CAN programmer. For details on how to use, refer to the user's manual included with your CAN programmer.
Table 1.22.8 lists pin functions for CAN I/O mode. Figures 1.22.16 shows pin connections for CAN I/O mode.

ID code check function

This function determines whether the ID codes sent from the CAN programmer and those written in the flash memory match. (Refer to "Functions to Prevent Flash Memory from Rewriting".)

Table 1.22.8 Pin Functions for CAN I/O Mode

Pin	Name	I/O	Description
V_{CC}, V_{SS}	Power input		Apply the voltage guaranteed for Program and Erase to V_{CC} pin and 0 V to V_{SS} pin.
CNV_{SS}	CNV_{SS}	I	Connect to V_{CC} pin.
$\overline{\text{RESET}}$	Reset input	I	Reset input pin. While $\overline{\text{RESET}}$ pin is "L" level, input 20 cycles or longer clock to X_{IN} pin.
X_{IN}	Clock input	I	Connect a ceramic resonator or crystal oscillator between X_{IN} and X_{OUT} pins. To input an externally generated clock, input it to X_{IN} pin and open X_{OUT} pin.
X_{OUT}	Clock output	O	
BYTE	BYTE	I	Connect this pin to V_{CC} or V_{SS}.
AV_{CC}, AV_{SS}	Analog power supply input		Connect AV_{SS} to V_{SS} and AV_{CC} to V_{CC}, respectively.
V_{REF}	Reference voltage input	I	Enter the reference voltage for A-D and D-A converters from this pin.
$P0_0$ to $P0_7$	Input port P0	I	Input "H" or "L" level signal or open.
$P1_0$ to $P1_7$	Input port P1	I	Input "H" or "L" level signal or open.
$P2_0$ to $P2_7$	Input port P2	I	Input "H" or "L" level signal or open.
$P3_0$ to $P3_7$	Input port P3	I	Input "H" or "L" level signal or open.
$P4_0$ to $P4_7$	Input port P4	I	Input "H" or "L" level signal or open.
$P5_0$	$\overline{\text{CE}}$ input	I	Input "H" level signal.
$P5_1$ to $P5_4$, $P5_6$, $P5_7$	Input port P5	I	Input "H" or "L" level signal or open.
$P5_5$	$\overline{\text{EPM}}$ input	I	Input "L" level signal.
$P6_0$ to $P6_4$, $P6_6$	Input port P6	I	Input "H" or "L" level signal or open.
$P6_5/CLK_1$	SCLK input	I	Input "L" level signal.
$P6_7/TxD_1$	TxD output	O	Input "H" level signal.
$P7_0$ to $P7_7$	Input port P7	I	Input "H" or "L" level signal or open.
$P8_0$ to $P8_4$, $P8_6$, $P8_7$	Input port P8	I	Input "H" or "L" level signal or open.
$P8_5/\overline{\text{NMI}}$	$\overline{\text{NMI}}$ input	I	Connect this pin to V_{CC}.
$P9_0$ to $P9_4$, $P9_7$	Input port P9	I	Input "H" or "L" level signal or open.
$P9_5/CRx_0$	CRx input	I	Connect to a CAN transceiver.
$P9_6/CTx_0$	CTx output	O	Connect to a CAN transceiver.
$P10_0$ to $P10_7$	Input port P10	I	Input "H" or "L" level signal or open.



Mode setup method

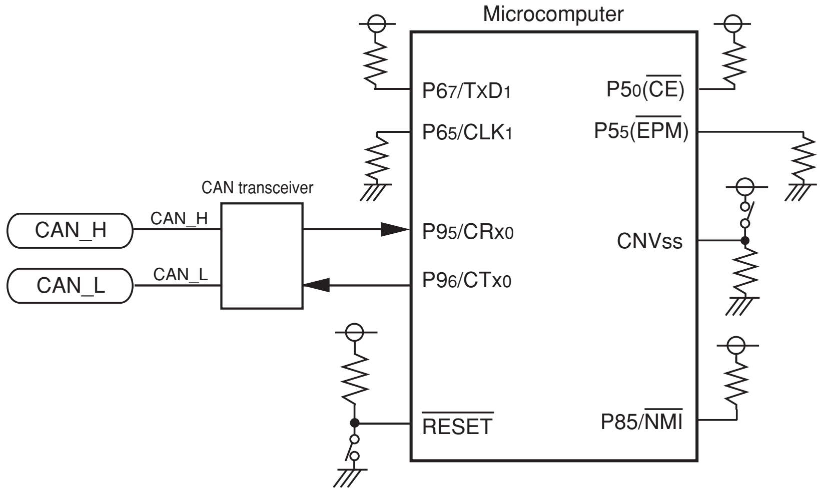
Signal	Value
CNVss	Vcc
$\overline{EPM}$	Vss
$\overline{RESET}$	Vss to Vcc
$\overline{CE}$	Vcc
SCLK	Vss
TxD	Vcc

Package: 100P6S-A

Figure 1.22.16 Pin Connections for CAN I/O Mode

Example of Circuit Application in CAN I/O Mode

Figure 1.22.17 shows example of circuit application in CAN I/O mode. Refer to the user's manual for CAN writer to handle pins controlled by a CAN writer.



- Control pins and external circuitry will vary according to programmer. For more information, refer to the programmer manual.
- In this example, modes are switched between single-chip mode and CAN input/output mode by controlling the CNVss input with a switch.

Figure 1.22.17 Circuit Application in CAN I/O Mode

Electrical Characteristics

Table 1.22.9 lists the flash memory electrical characteristics. Table 1.22.10 lists the flash memory version program/erase voltage and read operation voltage characteristics.

Table 1.22.9 Flash Memory Electrical Characteristics (Note 1)

Symbol	Parameter	Standard			Unit
		Min.	Typ.	Max.	
-	Word program time		30	200	µs
-	Block erase time		1	4	s
-	Erase all unlocked blocks time		1 × n (Note 2)	4 × n	s
-	Lock bit program time		30	200	µs
tps	Flash memory circuit stabilization wait time			15	µs

Note 1: Referenced to V_{CC} = 4.5 to 5.5 V, T_{opr} = 0 to 60 °C unless otherwise specified.

Note 2: n denotes the number of blocks to erase.

Table 1.22.10 Flash Memory Version Program/Erase Voltage and Read Operation Voltage Characteristics (at T_{opr} = 0 to 60 °C)

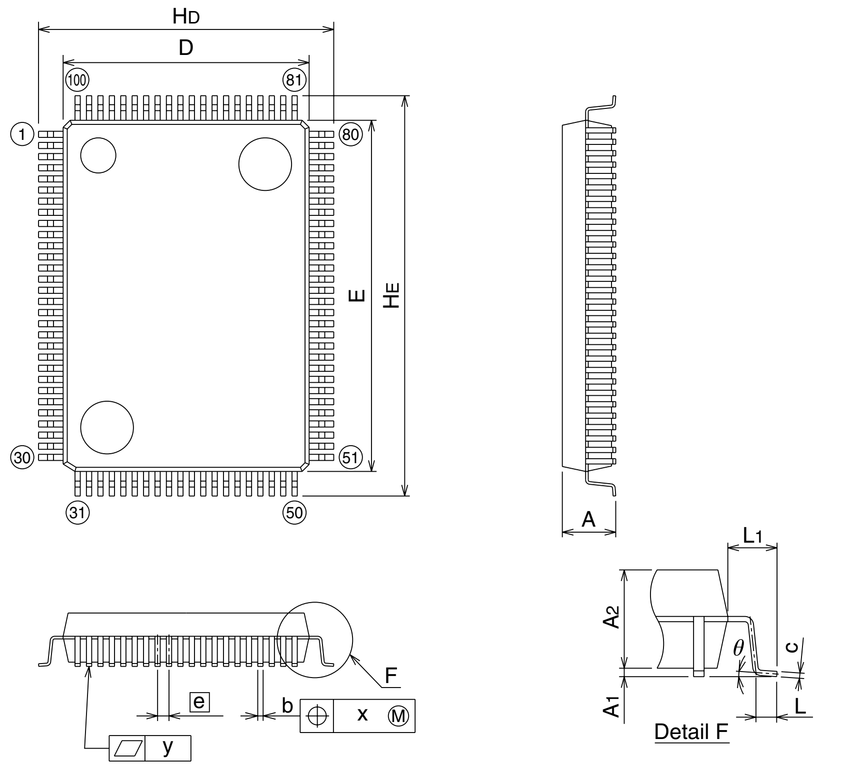
Flash program, erase voltage	Flash read operation voltage
V_{CC} = 5.0 ± 0.5 V	V_{CC} = 4.2 to 5.5 V

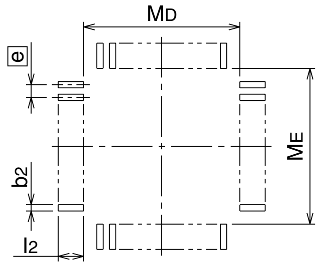
Package Dimension

100P6S-A (MMP)

Plastic 100pin 14×20mm body QFP

EIAJ Package Code	JEDEC Code	Weight(g)	Lead Material
QFP100-P-1420-0.65	–	1.58	Alloy 42





Recommended Mount Pad

Symbol	Dimension in Millimeters		
	Min	Nom	Max
A	–	–	3.05
A_1	0	0.1	0.2
A_2	–	2.8	–
b	0.25	0.3	0.4
c	0.13	0.15	0.2
D	13.8	14.0	14.2
E	19.8	20.0	20.2
e	–	0.65	–
H_D	16.5	16.8	17.1
H_E	22.5	22.8	23.1
L	0.4	0.6	0.8
L_1	–	1.4	–
x	–	–	0.13
y	–	–	0.1
θ	0°	–	10°
b_2	–	0.35	–
l_2	1.3	–	–
M_D	–	14.6	–
M_E	–	20.6	–

Register Index

R

S

T

U

W

REVISION HISTORY	M16C/6N4 Group Hardware Manual

Rev.	Date	Description	
		Page	Summary
1.00	May 30, 2003	–	First edition issued

RENESAS 16-BIT SINGLE-CHIP MICROCOMPUTER
HARDWARE MANUAL
M16C/6N4 Group Rev.1.00

Editioned by
Committee of editing of RENESAS Semiconductor Hardware Manual

M16C/6N4 Group
Hardware Manual

Renesas Technology Corp.
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan

REJ09B0010-0100Z

RENESAS

16

M16C/6N4 Group

Usage Notes Reference Book

RENESAS 16-BIT SINGLE-CHIP MICROCOMPUTER

M16C FAMILY / M16C/60 SERIES

For the most current **Usage Notes Reference Book**, please visit our website.

Usage Notes Reference Book

Before using this material, please visit our website to confirm that this is the most current document available.

Rev. 1.00
Revision date: May 30, 2003

Renesas Technology
www.renesas.com

Keep safety first in your circuit designs!

- Renesas Technology Corporation 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

- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation 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 Corporation or a third party.
- Renesas Technology Corporation 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.
- 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 Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation 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 Corporation 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 Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
- 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 Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology Corporation 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 Corporation or an authorized Renesas Technology Corporation 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.
- The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- 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.
- Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.

Preface

The “Usage Notes Reference Book” is a compilation of usage notes from the Hardware Manual as well as technical news related to this product.

Table of Contents

1. Usage Precaution

1. Usage Precaution

1.1 Precautions for External Bus

1. The external ROM version can operate only in the microprocessor mode, connect the CNV_{SS} pin to V_{CC}.

2. When resetting CNV_{SS} pin with "H" input, contents of internal ROM cannot be read out.

1.2 Precautions for PLL Frequency Synthesizer

Make the supply voltage stable to use the PLL frequency synthesizer.
For ripple with the supply voltage 5 V, keep below 10 kHz as frequency, below 0.5 V (peak to peak) as voltage fluctuation band and below 1 V/mS as voltage fluctuation rate.

1.3 Precautions for Power Control

1. When exiting stop mode by hardware reset, set $\overline{RESET}$ pin to “L” until a main clock oscillation is stabilized.

2. Insert more than four NOP instructions after an WAIT instruction or a instruction to set the CM10 bit of the CM1 register to “1” (all clock stopped). When shifting to wait mode or stop mode, an instruction queue reads ahead to the next instruction to halt a program by an WAIT instruction and an instruction to set the CM10 bit to “1”. The next instruction may be executed before entering wait mode or stop mode, depending on a combination of instruction and an execution timing.

3. Wait until the $t_{d(M-L)}$ elapses or main clock oscillation stabilization time, whichever is longer, before switching the clock source for CPU clock to the main clock.
Similarly, wait until the sub clock oscillates stably before switching the clock source for CPU clock to the sub clock.

4. Suggestions to reduce power consumption

(a) Ports

The processor retains the state of each I/O port even when it goes to wait mode or to stop mode. A current flows in active I/O ports. A pass current flows in input ports that high-impedance state. When entering wait mode or stop mode, set non-used ports to input and stabilize the potential.

(b) A-D converter

When A-D conversion is not performed, set the VCUT bit of the ADCON1 register to “0” (V_{REF} not connection). When A-D conversion is performed, start the A-D conversion at least 1 µs or longer after setting the VCUT bit to “1” (V_{REF} connection).

(c) D-A converter

When not performing D-A conversion, set the DAiE bit (i = 0, 1) of the DACON register to “0” (input inhibited) and DAi register to “00_{16}”.

(d) Stopping peripheral functions

Use the CM02 bit of the CM0 register to stop the unnecessary peripheral functions during wait mode. However, because the peripheral function clock (f_{C32}) generated from the sub clock does not stop, this measure is not conducive to reducing the power consumption of the chip. If low speed mode or low power dissipation mode is to be changed to wait mode, set the CM02 bit to “0” (do not peripheral function clock stopped when in wait mode), before changing wait mode.

(e) Switching the oscillation-driving capacity

Set the driving capacity to “LOW” when oscillation is stable.

(f) External clock

When using an external clock input for the CPU clock, set the CM05 bit of the CM0 register to “1” (stop). Setting the CM05 bit to “1” disables the X_{OUT} pin from functioning, which helps to reduce the amount of current drawn in the chip. (When using an external clock input, note that the clock remains fed into the chip regardless of how the CM05 bit is set.)

1.4 Precautions for Protection

Set the PRC2 bit to “1” (write enabled) and then write to any address, and the PRC2 bit will be set 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 no DMA transfers will occur between the instruction in which the PRC2 bit is set to “1” and the next instruction.

1.5 Precautions for Interrupts

1.5.1 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 set 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 set to “0”. This causes a problem that the interrupt is canceled, or an unexpected interrupt is generated.

1.5.2 SP Setting

Set any value in the SP (USP, ISP) before accepting an interrupt. The SP (USP, ISP) is set to “0000_{16}” after reset. Therefore, if an interrupt is accepted before setting any value in the SP (USP, ISP), the program may go out of control.
Especially when using $\overline{\text{NMI}}$ interrupt, set a value in the ISP at the beginning of the program. For the first and only the first instruction after reset, all interrupts including $\overline{\text{NMI}}$ interrupt are disabled.

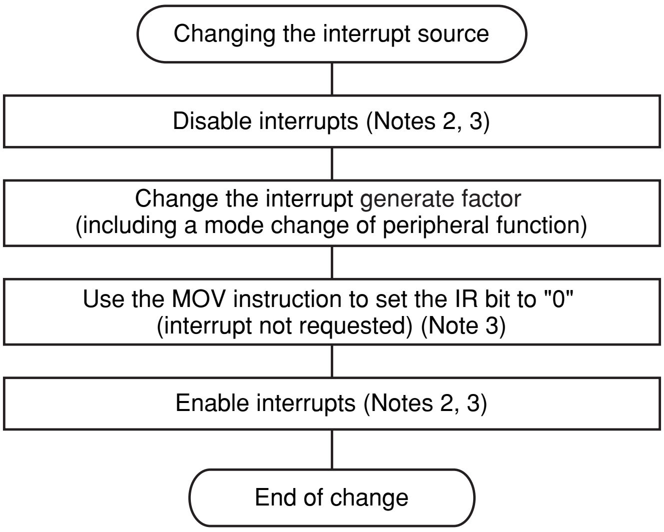
1.5.3 $\overline{\text{NMI}}$ Interrupt

1. The $\overline{\text{NMI}}$ interrupt cannot be disabled. If this interrupt is unused, connect the $\overline{\text{NMI}}$ pin to V_{CC} via a resistor (pull-up).
2. The input level of the $\overline{\text{NMI}}$ pin can be read by accessing the P8_5 bit of the P8 register. Note that the P8_5 bit can only be read when determining the pin level in $\overline{\text{NMI}}$ interrupt routine.
3. Stop mode cannot be entered into while input on the $\overline{\text{NMI}}$ pin is low. This is because while input on the $\overline{\text{NMI}}$ pin is low the CM10 bit of the CM1 register is fixed to “0”.
4. Do not go to wait mode while input on the $\overline{\text{NMI}}$ pin is low. This is because when input on the $\overline{\text{NMI}}$ pin goes low, the CPU stops but CPU clock remains active; therefore, the current consumption in the chip does not drop. In this case, normal condition is restored by an interrupt generated thereafter.
5. The low and high level durations of the input signal to the $\overline{\text{NMI}}$ pin must each be 2 CPU clock cycles + 300 ns or more.

1.5.4 Changing the Interrupt Generate Factor

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

“Changing the interrupt generate factor” referred to here means any act of changing the source, polarity or timing of the interrupt assigned to each software interrupt number. Therefore, if a mode change of any peripheral function involves changing the generate factor, polarity or timing of an interrupt, be sure to set the IR bit for that interrupt to “0” (interrupt not requested) after making such changes. Refer to the description of each peripheral function for details about the interrupts from peripheral functions. Figure 1.5.1 shows the procedure for changing the interrupt generate factor.



R bit: A bit in the interrupt control register for the interrupt whose interrupt generate factor is to be changed

Note 1: The above settings must be executed individually. Do not execute two or more settings simultaneously (using one instruction).
Note 2: Use the I flag for the $\overline{\text{INTi}}$ interrupt (i = 0 to 5).
For the interrupts from peripheral functions other than the $\overline{\text{INTi}}$ interrupt, turn off the peripheral function that is the source of the interrupt in order not to generate an interrupt request before changing the interrupt generate factor. In this case, if the maskable interrupts can all be disabled without causing a problem, use the I flag. Otherwise, use the corresponding
ILVL2 to ILVL0 bit for the interrupt whose interrupt generate factor is to be changed.
Note 3: Refer to "Rewrite the Interrupt Control Register" for details about the instructions to use and the notes to be taken for instruction execution.

Figure 1.5.1 Procedure for Changing Interrupt Generate Factor

1.5.5 $\overline{\text{INT}}$ Interrupt

1. Either an “L” level of at least $t_{W(INH)}$ or an “H” level of at least $t_{W(INL)}$ width is necessary for the signal input to pins $\overline{\text{INT}_0}$ through $\overline{\text{INT}_5}$ regardless of the CPU operation clock.
2. If the POL bit in the INT0IC to INT5IC registers or the IFSR17 to IFSR10 bits in the IFSR1 register are changed, the IR bit may inadvertently set to 1 (interrupt requested). Be sure to clear the IR bit to “0” (interrupt not requested) after changing any of those register bits.

1.5.6 Rewrite the Interrupt Control Register

(1) The interrupt control register for any interrupt should be modified in places where no requests for that interrupt may occur. Otherwise, disable the interrupt before rewriting the interrupt control register.
(2) To rewrite the interrupt control register for any interrupt after disabling that interrupt, be careful with the instruction to be used.

Changing any bit other than the IR bit

If while executing an instruction, a request for an interrupt controlled by the register being modified occurs, the IR bit in the register may not be set to "1" (interrupt requested), with the result that the interrupt request is ignored. If such a situation presents a problem, use the instructions shown below to modify the register.
Usable instructions: AND, OR, BCLR, BSET

Changing the IR bit

Depending on the instruction used, the IR bit may not always be set to "0" (interrupt not requested). Therefore, be sure to use the MOV instruction to set the IR bit to "0".

(3) When using the I flag to disable an interrupt, refer to the sample program fragments shown below as you set the I flag. (Refer to (2) for details about rewrite the interrupt control registers in the sample program fragments.)

Examples 1 through 3 show how to prevent the I flag from being set to "1" (interrupts enabled) before the interrupt control register is rewrited, owing to the effects of the internal bus and the instruction queue buffer.

Example 1: Using the NOP instruction to keep the program waiting until the interrupt control register is modified

```
INT_SWITCH1:
    FCLR I                      ; Disable interrupts.
    AND.B   #00H, 0055H         ; Set the TA0IC register to "0016".
    NOP                         ;
    NOP
    FSET I                      ; Enable interrupts.
```

The number of NOP instruction is as follows.
- PM20 of the PM2 register = 1 (1 wait) : 2
- PM20 = 0 (2 waits) : 3
- When using HOLD function : 4.

Example 2: Using the dummy read to keep the FSET instruction waiting

```
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: Using the POPC instruction to changing the I flag

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

1.5.7 Watchdog Timer Interrupt

Initialize the watchdog timer after the watchdog timer interrupt occurs.

1.6 Precautions for DMAC

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

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

Conditions

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

Step 1: Write “1” to the DMAE bit and DMAS bit in DMiCON register simultaneously (Note 1) .
Step 2: Make sure that the DMAi is in an initial state (Note 2) in a program.
If the DMAi is not in an initial state, the above steps should be repeated.

Note 1: The DMAS bit remains unchanged even if “1” is written. However, if “0” is written to this bit, it is set to “0” (DMA not requested). In order to prevent the DMAS bit from being modified to “0, “1” should be written to the DMAS bit when “1” is written to the DMAE bit. In this way the state of the DMAS bit immediately before being written can be maintained.
Similarly, when writing to the DMAE bit with a read-modify-write instruction, “1” should be written to the DMAS bit in order to maintain a DMA request which is generated during execution.

Note 2: Read the TCRi register to verify whether the DMAi is in an initial state. If the read value is equal to a value which was written to the TCRi register before DMA transfer start, the DMAi is in an initial state. (If a DMA request occurs after writing to the DMAE bit, the value written to the TCRi register is “1”.) If the read value is a value in the middle of transfer, the DMAi is not in an initial state.

1.7 Precautions for Timers

1.7.1 Timer A

1.7.1.1 Timer A (Timer Mode)

1. The timer remains idle after reset. Set the mode, count source, counter value, etc. using the TAiMR (i = 0 to 4) register and the TAi register before setting the TAiS bit in the TABSR register to “1” (count starts).
   Always make sure the TAiMR register is modified while the TAiS bit remains “0” (count stops) regardless whether after reset or not.

2. While counting is in progress, the counter value can be read out at any time by reading the TAi register. However, if the counter is read at the same time it is reloaded, the value “$FFFF_{16}$” is read. Also, if the counter is read before it starts counting after a value is set in the TAi register while not counting, the set value is read.

3. If a low-level signal is applied to the $\overline{NMI}$ pin when the IVPCR1 bit = 1 (three-phase output forcible cutoff by input on $\overline{NMI}$ pin enabled) of the TB2SC register, the $TA1_{OUT}$, $TA2_{OUT}$ and $TA4_{OUT}$ pins go to a high-impedance state.

1.7.1.2 Timer A (Event Counter Mode)

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

2. While counting is in progress, the counter value can be read out at any time by reading the TAi register. However, “$FFFF_{16}$” can be read in underflow, while reloading, and “0000_{16}” in overflow. When setting TAi register to a value during a counter stop, the setting value can be read before a counter starts counting. Also, if the counter is read before it starts counting after a value is set in the TAi register while not counting, the set value is read.

3. If a low-level signal is applied to the $\overline{NMI}$ pin when the IVPCR1 bit = 1 (three-phase output forcible cutoff by input on $\overline{NMI}$ pin enabled) of the TB2SC register, the $TA1_{OUT}$, $TA2_{OUT}$ and $TA4_{OUT}$ pins go to a high-impedance state.

1.7.1.3 Timer A (One-shot Timer Mode)

1. The timer remains idle after reset. Set the mode, count source, counter value, etc. using the TAiMR (i = 0 to 4) register, the TAi register, the ONSF register TA0TGL and TA0TGH bits and the TRGSR register before setting the TAiS bit in the TABSR register to “1” (count starts).
   Always make sure the TAiMR register, the ONSF register TA0TGL and TA0TGH bits and the TRGSR register are modified while the TAiS bit remains “0” (count stops) regardless whether after reset or not.

2. When setting the TAiS bit of the TABSR register to “0” (count stop), the followings occur:
   - A counter stops counting and a content of reload register is reloaded.
   - TAi_{OUT} pin outputs “L”.
   - After one cycle of the CPU clock, the IR bit of the TAiIC register is set to “1” (interrupt request).

3. Output in one-shot timer mode synchronizes with a count source internally generated. When an external trigger has been selected, one-cycle delay of a count source as maximum occurs between a trigger input to TAi_{IN} pin and output in one-shot timer mode.

4. The IR bit is set to “1” when timer operation mode is set with any of the following procedures:
   - Select one-shot timer mode after reset.
   - Change an operation mode from timer mode to one-shot timer mode.
   - Change an operation mode from event counter mode to one-shot timer mode.

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

5. When a trigger occurs, while counting, a counter reloads the reload register to continue counting after generating a re-trigger and counting down once. To generate a trigger while counting, generate a second trigger between occurring the previous trigger and operating longer than one cycle of a timer count source.

6. If a low-level signal is applied to the $\overline{NMI}$ pin when the IVPCR1 bit = 1 (three-phase output forcible cutoff by input on $\overline{NMI}$ pin enabled) of the TB2SC register, the $TA1_{OUT}$, $TA2_{OUT}$ and $TA4_{OUT}$ pins go to a high-impedance state.

1.7.1.4 Timer A (Pulse Width Modulation Mode)

1. The timer remains idle after reset. Set the mode, count source, counter value, etc. using the TAiMR (i = 0 to 4) register, the TAi register, the ONSF register TA0TGL and TA0TGH bits and the TRGSR register before setting the TAiS bit in the TABSR register to “1” (count starts).
   Always make sure the TAiMR register, the ONSF register TA0TGL and TA0TGH bits and the TRGSR register are modified while the TAiS bit remains “0” (count stops) regardless whether after reset or not.

2. The IR bit is set to “1” when setting a timer operation mode with any of the following procedures:
   • Select the PWM mode after reset.
   • Change an operation mode from timer mode to PWM mode.
   • Change an operation mode from event counter mode to PWM mode.
   To use the timer Ai interrupt (the IR bit), set the IR bit to “0” by program after the above listed changes have been made.

3. When setting TAiS bit to “0” (count stop) during PWM pulse output, the following action occurs:
   • Stop counting.
   • When TAi_{OUT} pin is output “H”, output level is set to “L” and the IR bit is set to “1”.
   • When TAi_{OUT} pin is output “L”, both output level and the IR bit remain unchanged.

4. If a low-level signal is applied to the $\overline{NMI}$ pin when the IVPCR1 bit = 1 (three-phase output forcible cutoff by input on $\overline{NMI}$ pin enabled) of the TB2SC register, the $TA1_{OUT}$, $TA2_{OUT}$ and $TA4_{OUT}$ pins go to a high-impedance state.

1.7.2 Timer B

1.7.2.1 Timer B (Timer Mode)

1. The timer remains idle after reset. Set the mode, count source, counter value, etc. using the TBiMR (i = 0 to 5) register and TBi register before setting the TBiS bit in the TABSR or the TBSR register to “1” (count starts).
   Always make sure the TBiMR register is modified while the TBiS bit remains “0” (count stops) regardless whether after reset or not.

   The TB0S to TB2S bits are the bits 5 to 7 of the TABSR register, the TB3S to TB5S bits are the bits 5 to 7 of the TBSR register.

2. A value of a counter, while counting, can be read in the TBi register at any time. “$FFFF_{16}$” is read while reloading. Setting value is read between setting values in TBi register at count stop and starting a counter.

1.7.2.2 Timer B (Event Counter Mode)

1. The timer remains idle after reset. Set the mode, count source, counter value, etc. using the TBiMR (i = 0 to 5) register and TBi register before setting the TBiS bit in the TABSR or the TBSR register to “1” (count starts).
   Always make sure the TBiMR register is modified while the TBiS bit remains “0” (count stops) regardless whether after reset or not.

   The TB0S to TB2S bits are the bits 5 to 7 of the TABSR register, the TB3S to TB5S bits are the bits 5 to 7 of the TBSR register.

2. A value of a counter, while counting, can be read in the TBi register at any time. “$FFFF_{16}$” is read while reloading. Setting value is read between setting values in TBi register at count stop and starting a counter.

1.7.2.3 Timer B (Pulse Period/pulse Width Measurement Mode)

1. The timer remains idle after reset. Set the mode, count source, etc. using the TBiMR (i = 0 to 5) register before setting the TBiS bit in the TABSR or the TBSR register to “1” (count starts).
   Always make sure the TBiMR register is modified while the TBiS bit remains “0” (count stops) regardless whether after reset or not. To clear the MR3 bit to “0” by writing to the TBiMR register while the TBiS bit = “1” (count starts), be sure to write the same value as previously written to the TM0D0, TM0D1, MR0, MR1, TCK0 and TCK1 bits and a 0 to the MR2 bit.

2. The IR bit of TBiIC register (i = 0 to 5) goes to “1” (interrupt request), when an effective edge of a measurement pulse is input or timer Bi is overflowed. The factor of interrupt request can be determined by use of the MR3 bit of TBiMR register within the interrupt routine.

3. If the source of interrupt cannot be identified by the MR3 bit such as when the measurement pulse input and a timer overflow occur at the same time, use another timer to count the number of times timer B has overflowed.

4. To set the MR3 bit to “0” (no overflow), set TBiMR register with setting the TBiS bit to “1” and counting the next count source after setting the MR3 bit to “1” (overflow).

5. Use the IR bit of the TBiIC register to detect only overflows. Use the MR3 bit only to determine the interrupt factor within the interrupt routine.

6. When a count is started and the first effective edge is input, an indeterminate value is transferred to the reload register. At this time, timer Bi interrupt request is not generated.

7. A value of the counter is indeterminate at the beginning of a count. The MR3 bit may be set to “1” and timer Bi interrupt request may be generated between a count start and an effective edge input.

8. For pulse width measurement, pulse widths are successively measured. Use program to check whether the measurement result is an “H” level width or an “L” level width.

1.8 Precautions for Serial I/O (Clock Synchronous Serial I/O Mode)

1.8.1 Transmission/reception

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

2. If a low-level signal is applied to the $\overline{NMI}$ pin when the IVPCR1 bit = 1 (three-phase output forcible cutoff by input on $\overline{NMI}$ pin enabled) of the TB2SC register, the $\overline{RTS_2}$ and CLK_2 pins go to a high-impedance state.

1.8.2 Transmission

When an external clock is selected, the conditions must be met while if the CKPOL bit of the UiC0 register = 0 (transmit data output at the falling edge and the receive data taken in at the rising edge of the transfer clock), the external clock is in the high state; if the CKPOL bit of the UiC0 register = 1 (transmit data output at the rising edge and the receive data taken in at the falling edge of the transfer clock), the external clock is in the low state.
• The TE bit of the UiC1 register = 1 (transmission enabled)
• The TI bit of the UiC1 register = 0 (data present in UiTB register)
• If $\overline{CTS}$ function is selected, input on the $\overline{CTS_i}$ pin = L

1.8.3 Reception

1. In operating the clock synchronous serial I/O, operating a transmitter generates a shift clock. Fix settings for transmission even when using the device only for reception. Dummy data is output to the outside from the TxD_i (i = 0 to 2) pin when receiving data.

2. When an internal clock is selected, set the TE bit of the UiC1 register to “1” (transmission enabled) and write dummy data to the UiTB register, and the shift clock will thereby be generated. When an external clock is selected, set the TE bit to “1” and write dummy data to the UiTB register, and the shift clock will be generated when the external clock is fed to the CLK_i input pin.

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

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

5. When an external clock is selected, the conditions must be met while if the CKPOL bit = 0, the external clock is in the high state; if the CKPOL bit = 1, the external clock is in the low state.
   • The RE bit of the UiC1 register = 1 (reception enabled)
   • The TE bit of the UiC1 register = 1 (transmission enabled)
   • The TI bit of the UiC1 register = 0 (data present in the UiTB register)

1.9 Precaution for Serial I/O (Special Modes)

1.9.1 Special Mode 2

If a low-level signal is applied to the $\overline{NMI}$ pin when the TB2SC register IVPCR1 bit = 1 (three-phase output forcible cutoff by input on $\overline{NMI}$ pin enabled), the RTS_2 and CLK_2 pins go to a high-impedance state.

1.9.2 Special Mode 4 (SIM Mode)

A transmit interrupt request is generated by setting the U2C1 register U2IRS bit to “1” (transmission complete) and U2ERE bit to “1” (error signal output) after reset. Therefore, when using SIM mode, be sure to set the IR bit to “0” (no interrupt request) after setting these bits.

1.10 Precautions for A-D Converter

1. Set the ADCON0 (except bit 6), ADCON1 and ADCON2 registers when A-D conversion is stopped (before a trigger occurs).

2. When the VCUT bit of the ADCON1 register is changed from “0” (V_{REF} not connected) to “1” (V_{REF} connected), start A-D conversion after passing 1 µs or longer.

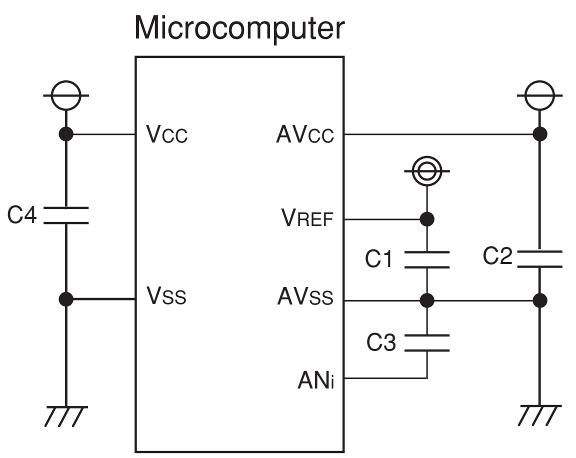
3. To prevent noise-induced device malfunction or latchup, as well as to reduce conversion errors, insert capacitors between the AV_{CC}, V_{REF}, and analog input pins (AN_i (i = 0 to 7), AN_{0i}, and AN_{2i}) each and the AV_{SS} pin. Similarly, insert a capacitor between the V_{CC} pin and the V_{SS} pin. Figure 1.10.1 is an example connection of each pin.

4. Make sure the port direction bits for those pins that are used as analog inputs are set to “0” (input mode). Also, if the TGR bit of the ADCON0 register = 1 (external trigger), make sure the port direction bit for the $\overline{AD_{TRG}}$ pin is set to “0” (input mode).

5. When using key input interrupts, do not use any of the four AN_4 to AN_7 pins as analog inputs. (A key input interrupt request is generated when the A-D input voltage goes low.)

6. The ϕ_{AD} frequency must be 10 MHz or less. Without sample-and-hold function, limit the ϕ_{AD} frequency to 250 kHz or more. With the sample and hold function, limit the ϕ_{AD} frequency to 1 MHz or more.

7. When changing an A-D operation mode, select analog input pin again in the CH2 to CH0 bits of the ADCON0 register and the SCAN1 to SCAN0 bits of the ADCON1 register.



AN_i: AN_i, AN_{0i}, and AN_{2i} (i =0 to 7)

Note 1: C1 ≥ 0.47 µF, C2 ≥ 0.47 µF, C3 ≥ 100 pF, C4 ≥ 0.1 µF (reference).
Note 2: Use thick and shortest possible wiring to connect capacitors.

Figure 1.10.1 Use of capacitors to reduce noise

8. If the CPU reads the ADi register at the same time the conversion result is stored in the ADi register after completion of A-D conversion, an incorrect value may be stored in the ADi register. This problem occurs when a divide-by-n clock derived from the main clock or a sub clock is selected for CPU clock.
   • When operating in one-shot or single-sweep mode
     Check to see that A-D conversion is completed before reading the target ADi register. (Check the IR bit of the ADIC register to see if A-D conversion is completed.)
   • When operating in repeat mode or repeat sweep mode 0 or 1
     Use the main clock for CPU clock directly without dividing it.

9. If A-D conversion is forcibly terminated while in progress by setting the ADST bit of the ADCON0 register to “0” (A-D conversion halted), the conversion result of the A-D converter is indeterminate. The contents of ADi registers irrelevant to A-D conversion may also become indeterminate. If while A-D conversion is underway the ADST bit is set to “0” in a program, ignore the values of all ADi registers.

1.11 Precautions for CAN Module

1.11.1 Reading CiSTR Register (i = 0, 1)

The CAN module on the M16C/6N4 group updates the status of the CiSTR register in a certain period. When the CPU and the CAN module access to the CiSTR register at the same time, the CPU has the access priority; the access from the CAN module is disabled. Consequently, when the updating period of the CAN module matches the access period from the CPU, the status of the CAN module cannot be updated. (Refer to Figure 1.11.1.)
Accordingly, be careful about the following points so that the access period from the CPU should not match the updating period of the CAN module:

(1) There should be a wait time of $3f_{CAN}$ or longer (refer to Table 1.11.1) before the CPU reads the CiSTR register. (Refer to Figure 1.11.2.)
(2) When the CPU polls the CiSTR register, the polling period must be $3f_{CAN}$ or longer. (Refer to Figure 1.11.3.)

Table 1.11.1 CAN Module Status Updating Period

$3f_{CAN}$ period = 3 × X_{IN} (Original oscillation period) × Division value of the CAN clock (CCLK)	
(Example 1) Condition X_{IN} 16 MHz CCLK: Divided by 1	$3f_{CAN}$ period = 3 × 62.5 ns × 1 = 187.5 ns
(Example 2) Condition X_{IN} 16 MHz CCLK: Divided by 2	$3f_{CAN}$ period = 3 × 62.5 ns × 2 = 375 ns
(Example 3) Condition X_{IN} 16 MHz CCLK: Divided by 4	$3f_{CAN}$ period = 3 × 62.5 ns × 4 = 750 ns
(Example 4) Condition X_{IN} 16 MHz CCLK: Divided by 8	$3f_{CAN}$ period = 3 × 62.5 ns × 8 = 1.5 µs
(Example 5) Condition X_{IN} 16 MHz CCLK: Divided by 16	$3f_{CAN}$ period = 3 × 62.5 ns × 16 = 3 µs

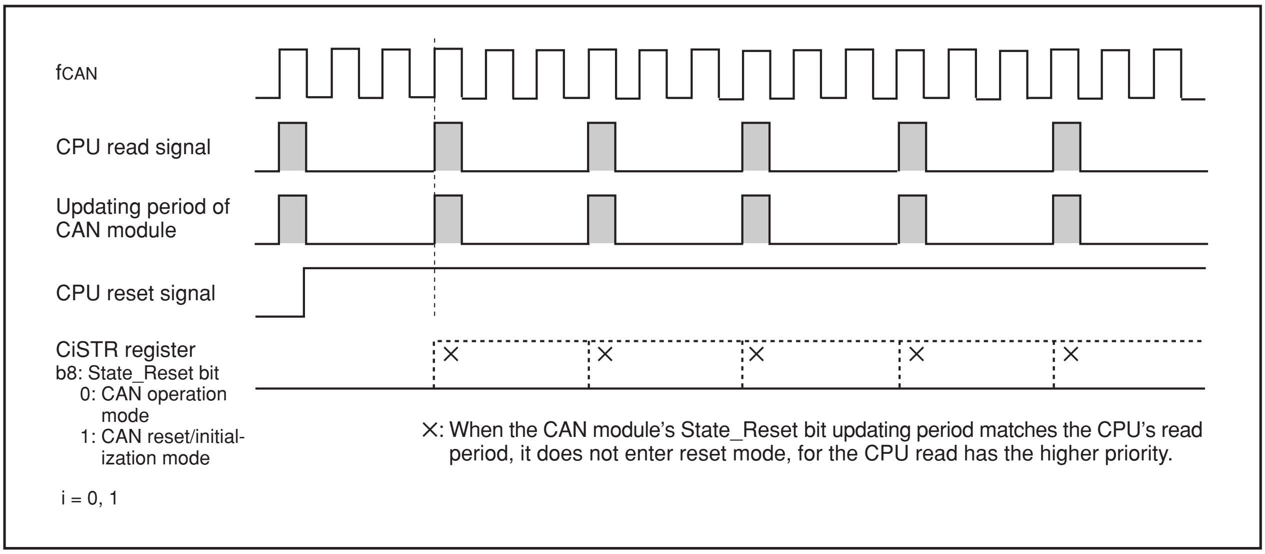


Figure 1.11.1 When Updating Period of CAN Module Matches Access Period from CPU

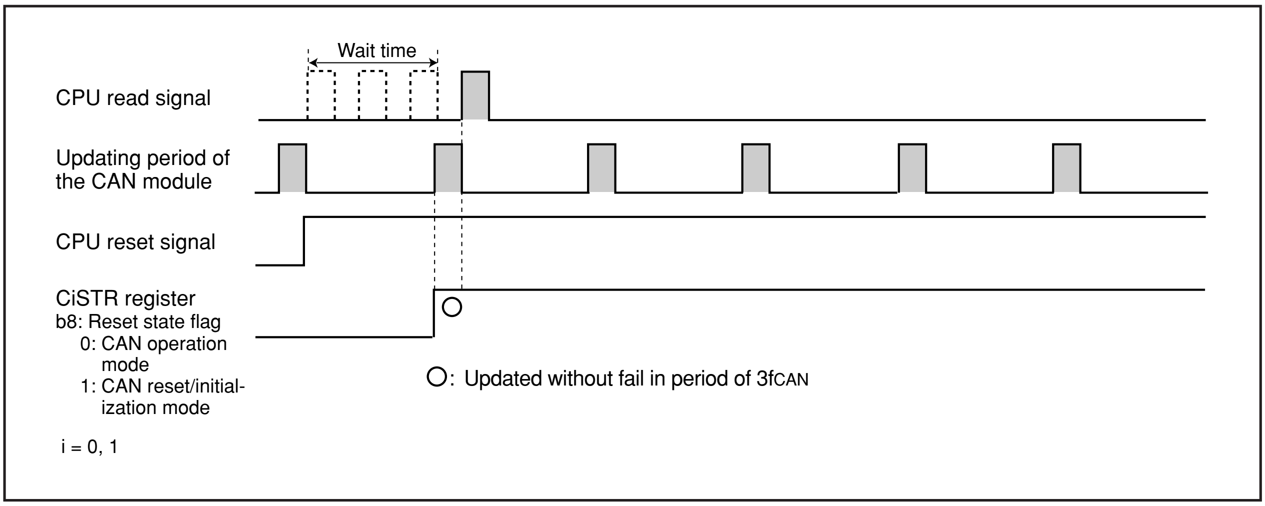


Figure 1.11.2 With a Wait Time of 3fCAN Before CPU Read

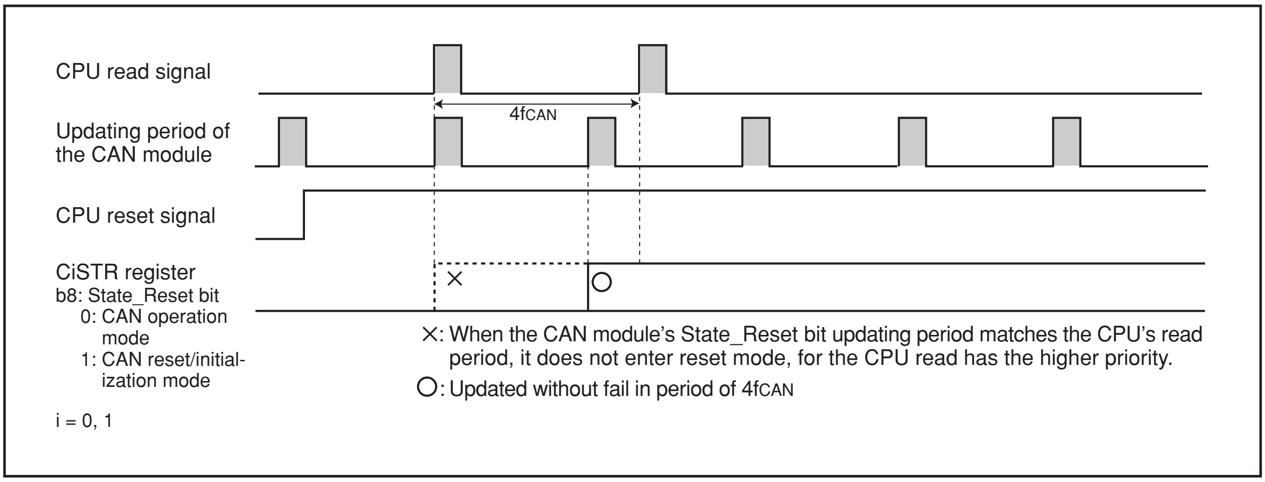
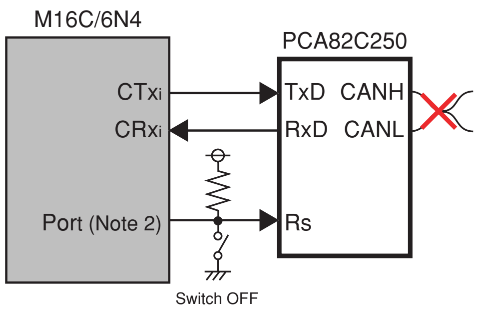
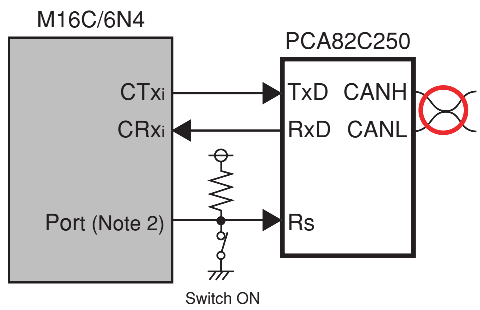


Figure 1.11.3 When Polling Period of CPU is 3fCAN or Longer

1.11.2 CAN Transceiver in Boot Mode

When programming the flash memory in boot mode via CAN bus, the operation mode of CAN transceiver should be set to “high-speed mode” or “normal operation mode”. If the operation mode is controlled by the microcomputer, CAN transceiver must be set the operation mode to “high-speed mode” or “normal operation mode” before programming the flash memory by changing the switch etc. Table 1.11.2 and 1.11.3 show pin connections of CAN transceiver.

Table 1.11.2 Pin Connections of CAN Transceiver (In case of PCA82C250: Philips product)

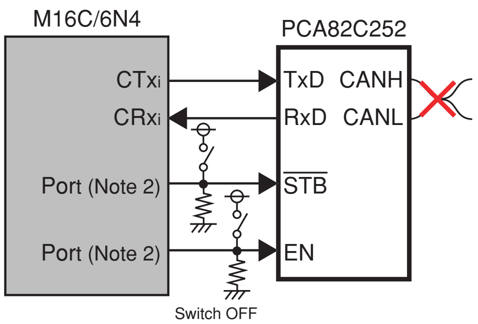
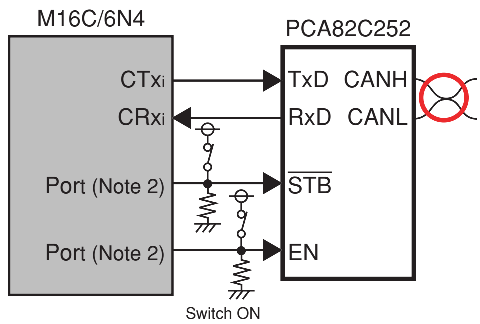
	Standby mode	High-speed mode
Rs pin (Note 1)	“H”	“L”
CAN communication	impossible	possible
Connection		

i = 0, 1

Note 1: The pin which controls the operation mode of CAN transceiver.

Note 2: Connect to enabled port to control CAN transceiver.

Table 1.11.3 Pin Connections of CAN Transceiver (In case of PCA82C252: Philips product)

	Sleep mode	Normal operation mode
$\overline{STB}$ pin (Note 1)	“L”	“H”
EN pin (Note 1)	“L”	“H”
CAN communication	impossible	possible
Connection		

i = 0, 1

Note 1: The pin which controls the operation mode of CAN transceiver.

Note 2: Connect to enabled port to control CAN transceiver.

1.12 Precautions for Programmable I/O Ports

1. If a low-level signal is applied to the $\overline{NMI}$ pin when the TB2SC register IVPCR1 bit = "1" (three-phase output forcible cutoff by input on $\overline{NMI}$ pin enabled), the $P7_2$ to $P7_5$, $P8_0$ and $P8_1$ pins go to a high-impedance state.

2. Setting the SM32 bit in the S3C register to "1" causes the $P9_2$ pin to go to a high-impedance state.

3. The input threshold voltage of pins differs between programmable input/output ports and peripheral functions.
   Therefore, if any pin is shared by a programmable input/output port and a peripheral function and the input level at this pin is outside the range of recommended operating conditions V_{IH} and V_{IL} (neither "high" nor "low"), the input level may be determined differently depending on which side—the programmable input/output port or the peripheral function—is currently selected.

1.13 Precautions for Electrical Characteristic Differences Between Mask ROM and Flash Memory Version Microcomputers

Flash memory version and mask ROM version may have different characteristics, operating margin, noise tolerated dose, noise width dose in electrical characteristics due to internal ROM, different layout pattern, etc. When switching to the mask ROM version, conduct equivalent tests as system evaluation tests conducted in the flash memory version.

1.14 Precautions for Flash Memory Version

1.14.1 Precautions for Functions to Prevent Flash Memory from Rewriting

ID codes are stored in addresses $0FFFDF_{16}$, $0FFFE3_{16}$, $0FFFEB_{16}$, $0FFFEF_{16}$, $0FFFF3_{16}$, $0FFFF7_{16}$, and $0FFFFB_{16}$. If wrong data are written to theses addresses, the flash memory cannot be read or written in standard serial I/O mode and CAN I/O mode.
The ROMCP register is mapped in address $0FFFFF_{16}$. If wrong data is written to this address, the flash memory cannot be read or written in parallel I/O mode.
In the flash memory version of microcomputer, these addresses are allocated to the vector addresses (H) of fixed vectors.

1.14.2 Precautions for 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
```

1.14.3 Precautions for Wait Mode

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

1.14.4 Precautions for Low Power Dissipation Mode and Ring Oscillator Low Power Dissipation Mode

If the CM05 bit is set to “1” (main clock stop), the following commands must not be executed.

- Program
- Block erase
- Erase all unlocked blocks
- Lock bit program

1.14.5 Writing command and data

Write the command code and data at even addresses.

1.14.6 Precautions for Program Command

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.

1.14.7 Precautions for Lock Bit Program Command

Write “$xx77_{16}$” in the first bus cycle and write “$xxD0_{16}$” to the uppermost address of a block (even address, however) in the second bus cycle, and the lock bit for the specified block is set to “0”. Make sure the address value specified in the first bus cycle is the same uppermost block address that is specified in the second bus cycle.

1.14.8 Operation speed

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

1.14.9 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

1.14.10 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 $\overline{NMI}$ 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 $\overline{NMI}$ 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 $\overline{NMI}$ 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 $\overline{NMI}$ interrupt occurs, the rewrite program must be executed again after exiting the interrupt service routine.

1.14.11 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 no DMA transfers will occur before writing "1" after writing "0". Also only when $\overline{NMI}$ pin is "H" level.

1.14.12 Writing in user ROM area

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, parallel I/O or CAN I/O mode should be used.

EW1 Mode

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

1.14.13 DMA transfer

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

REVISION HISTORY	M16C/6N4 Group Usage Notes

Rev.	Date	Description	
		Page	Summary
1.00	May 30, 2003	–	First edition issued

RENESAS 16-BIT SINGLE-CHIP MICROCOMPUTER
USAGE NOTES REFERENCE BOOK
M16C/6N4 Group Rev.1.00

Editioned by
Committee of editing of RENESAS Semiconductor Usage Notes Reference Book

M16C/6N4 Group
Usage Notes Reference Book

Renesas Technology Corp.
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan