Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	-
Core Size	-
Speed	-
Connectivity	-
Peripherals	-
Number of I/O	-
Program Memory Size	-
Program Memory Type	-
EEPROM Size	-
RAM Size	-
Voltage - Supply (Vcc/Vdd)	-
Data Converters	-
Oscillator Type	-
Operating Temperature	-
Mounting Type	-
Package / Case	-
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/renesas-electronics-america/m37641f8hp-u0

FUNCTIONAL BLOCK DIAGRAM (Package: PRQP0080GB-A)

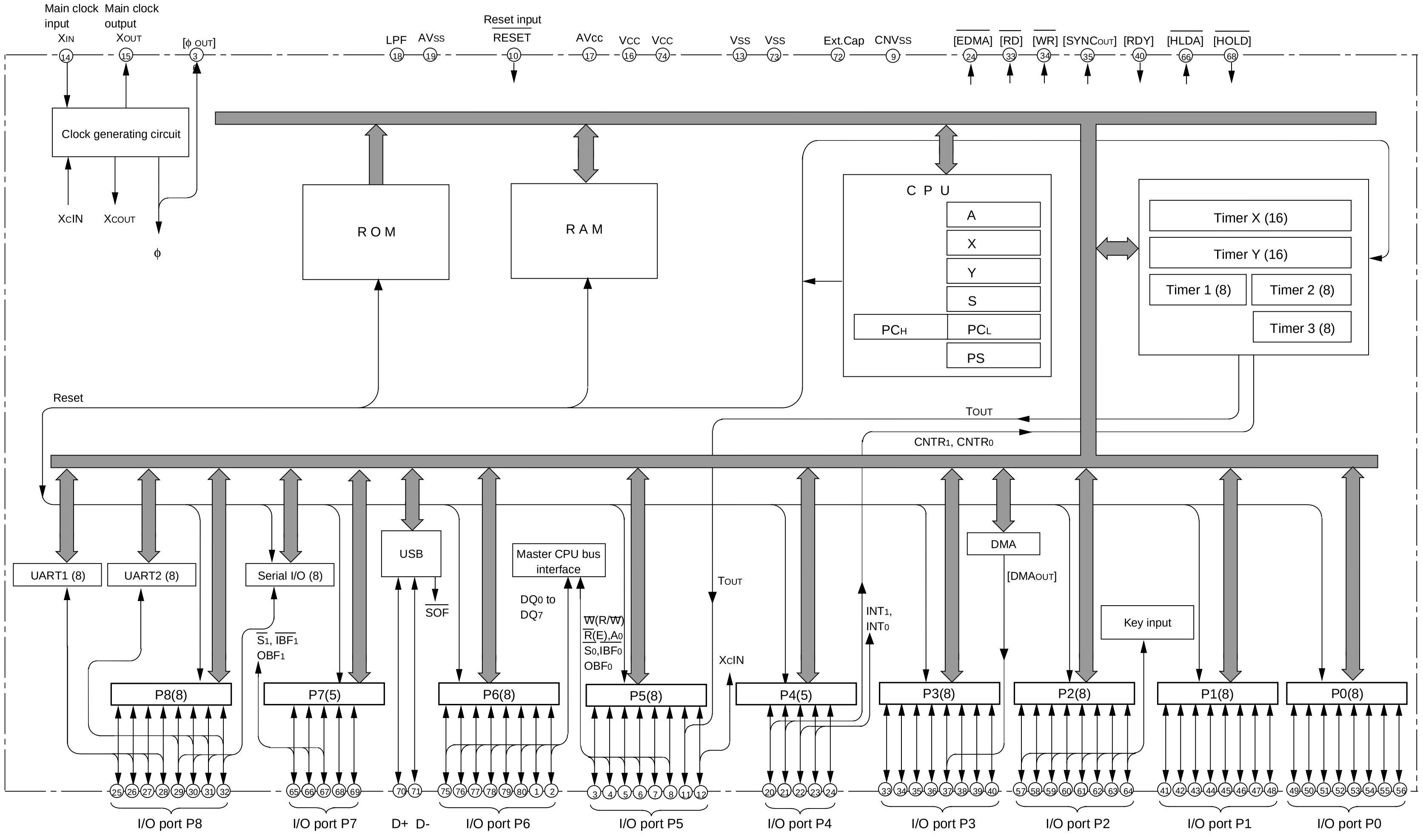


Fig. 3 Functional block diagram

MEMORY

Special Function Register (SFR) Area

The Special Function Register area in the zero page contains control registers such as I/O ports and timers.

RAM

RAM is used for data storage and for stack area of subroutine calls and interrupts.

ROM

The first 128 bytes and the last 4 bytes of ROM are reserved for device testing and the rest is user area for storing programs. In the flash memory version, program and erase can be performed in the reserved area.

Interrupt Vector Area

The interrupt vector area contains reset and interrupt vectors.

Zero Page

Access to this area with only 2 bytes is possible in the zero page addressing mode.

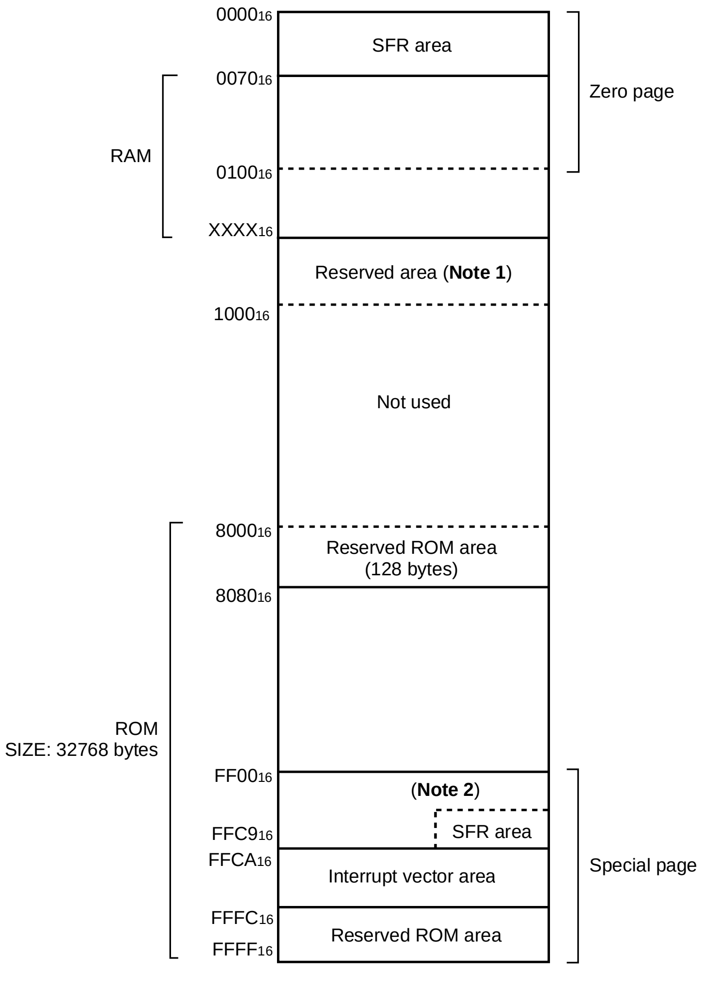
Special Page

Access to this area with only 2 bytes is possible in the special page addressing mode.

Refer to page 74 for the memory map of memory expansion and microprocessor modes.

RAM area

	RAM size (bytes)	Address $XXXX_{16}$
M37641M8	1024	$046F_{16}$
M37641F8	2560	$0A6F_{16}$



Notes 1: Reserved area in M37641F8.
2: SFR area in M37641F8.

Fig. 9 Memory map diagram

Table 6 List of I/O port function

Pin	Name	Input/Output	I/O format	Non-port function	Related SFRs	Ref. No.
P0$_0$/AB$_0$–P0$_7$/AB$_7$	Port P0	Input/Output, individual bits	CMOS input level CMOS 3-state output	Lower address output	CPU mode register A Port control register	(1)
P1$_0$/AB$_8$–P1$_7$/AB$_{15}$	Port P1			Higher address output		
P2$_0$/DB$_0$–P2$_7$/DB$_7$	Port P2		CMOS input level/V$_{IHL}$ input level CMOS 3-state output	Data bus I/O	CPU mode register A Port control register Port P2 pull-up control register	(2)
P3$_0$/RDY–P3$_7$/$\overline{RD}$	Port P3		CMOS input level CMOS 3-state output	Control signal I/O	CPU mode register A CPU mode register B Port control register	(1) (3)
P4$_0$/$\overline{EDMA}$,	Port P4					
P4$_1$/INT$_0$, P4$_2$/INT$_1$,				Control signal I/O External interrupt	CPU mode register A CPU mode register B Port control register Timer X mode register Timer Y mode register Interrupt polarity select register	(4) (5)
P4$_3$/CNTR$_0$, P4$_4$/CNTR$_1$						
P5$_0$/X$_{CIN}$, P5$_1$/T$_{OUT}$/X$_{COUT}$	Port P5		CMOS input level CMOS 3-state output	Timer 1, Timer 2 output pin Sub-clock generating input pin	CPU mode register A Port control register Clock control register Timer 123 mode register	(6) (7)
P5$_2$/OBF$_0$, P5$_3$/IBF$_0$, P5$_4$/$\overline{S}_0$, P5$_5$/A$_0$, P5$_6$/$\overline{R}$(E), P5$_7$/$\overline{W}$(R/$\overline{W}$)			CMOS input level CMOS 3-state output CMOS input level/TTL input level in Master CPU bus inferface function	Master CPU bus interface I/O pin	Data bus buffer control register 0 Port control register	(8) (9) (10)
P6$_0$/DQ$_0$–P6$_7$/DQ$_7$	Port P6		CMOS input level/TTL input level CMOS 3-state output	Master CPU bus interface I/O pin	Data bus buffer control register 0 Port control register	(11)
P7$_0$/$\overline{SOF}$,	Port P7		CMOS input level CMOS 3-state output	USB function output pin	USB control register Port control register	(12)
P7$_1$/$\overline{HOLD}$, P7$_2$/S$_1$, P7$_3$/IBF$_1$/$\overline{HLDA}$, P7$_4$/OBF$_1$			CMOS input level CMOS 3-state output CMOS input level/TTL input level in Master CPU bus inferface function	Control signal I/O Master CPU bus interface I/O pin	Data bus buffer control register 1 Port control register CPU mode register B	(13) (14) (15) (16)
P8$_0$/UTXD$_2$/$\overline{SRDY}$, P8$_1$/URXD$_2$/SCLK, P8$_2$/$\overline{CTS_2}$/SRXD, P8$_3$/$\overline{RTS_2}$/STXD, P8$_4$/UTXD$_1$, P8$_5$/URXD$_1$, P8$_6$/$\overline{CTS_1}$, P8$_7$/$\overline{RTS_1}$	Port P8		CMOS input level CMOS 3-state output	Serial I/O I/O pin UART2 I/O pin UART1 I/O pin	UART1, 2 control registers Serial I/O control register 1 Serial I/O control register 2 Port control register	(17) (18) (19) (20) (21) (22) (23) (24)

Notes 1: For details of the ports functions in modes other than single-chip mode, and how to use double-function ports as function I/O ports, refer to the applicable sections.

2: Make sure that the input level at each pin is either 0 V or V$_{CC}$ during execution of the STP instruction.
When an input level is at an intermediate potential, a rush current will flow from V$_{CC}$ to V$_{SS}$ through the input-stage gate.

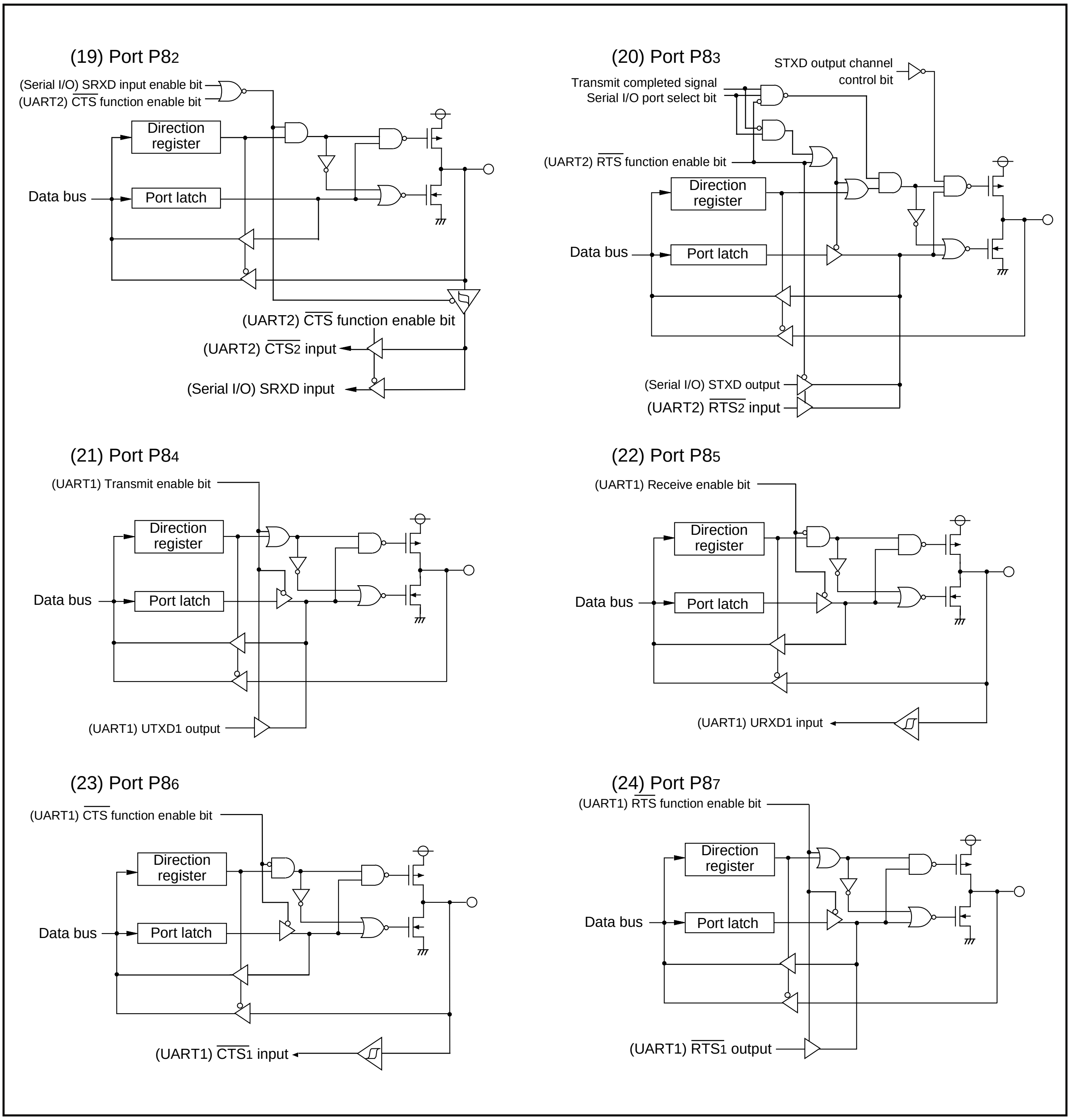
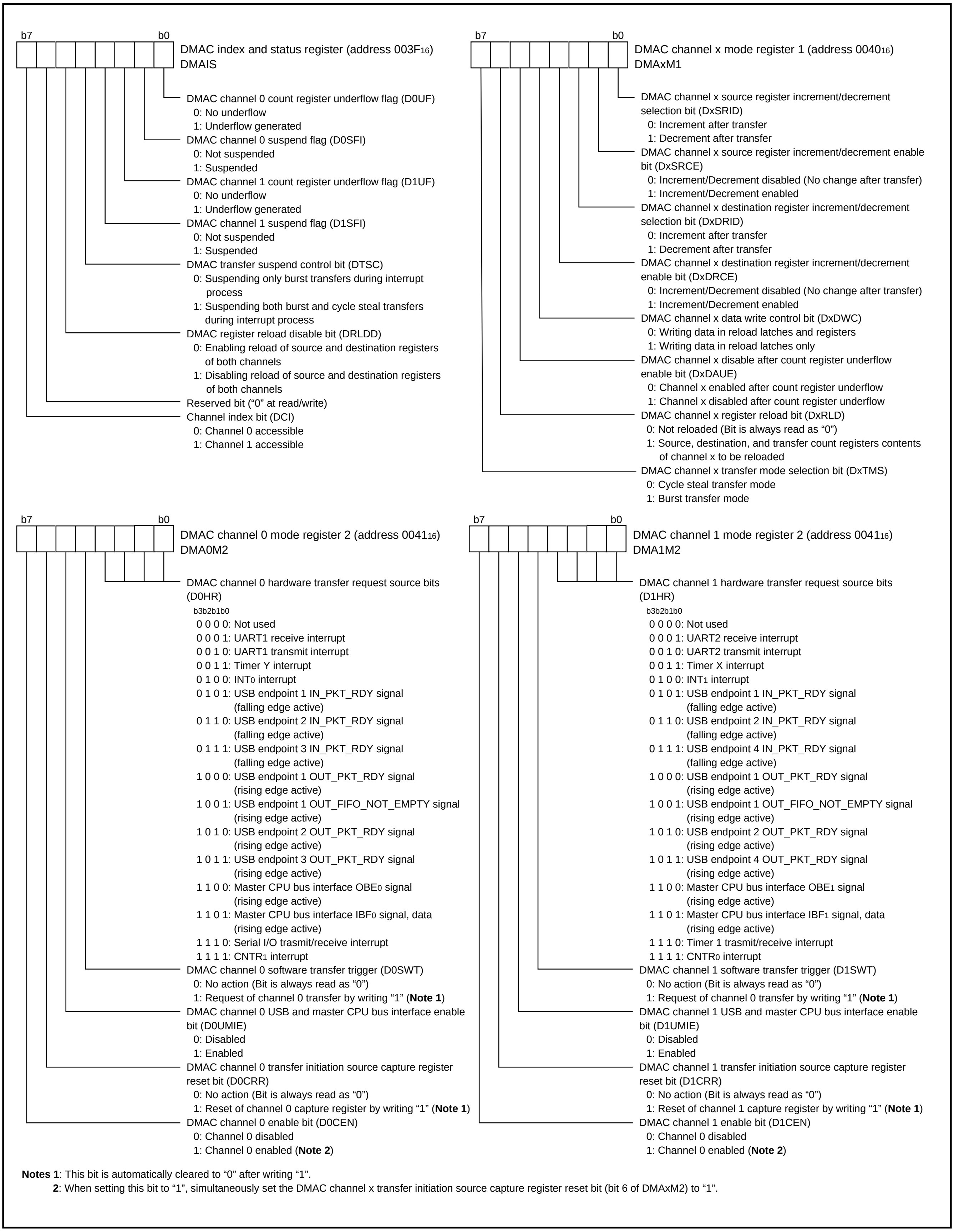

(19) Port P82
(Serial I/O) SRXD input enable bit
(UART2) $\overline{\text{CTS}}$ function enable bit
Direction register
Data bus
Port latch
(UART2) $\overline{\text{CTS}}$ function enable bit
(UART2) $\overline{\text{CTS2}}$ input
(Serial I/O) SRXD input
(20) Port P83
STXD output channel control bit
Transmit completed signal
Serial I/O port select bit
(UART2) $\overline{\text{RTS}}$ function enable bit
Direction register
Data bus
Port latch
(Serial I/O) STXD output
(UART2) $\overline{\text{RTS2}}$ input
(21) Port P84
(UART1) Transmit enable bit
Direction register
Data bus
Port latch
(UART1) UTXD1 output
(22) Port P85
(UART1) Receive enable bit
Direction register
Data bus
Port latch
(UART1) URXD1 input
(23) Port P86
(UART1) $\overline{\text{CTS}}$ function enable bit
Direction register
Data bus
Port latch
(UART1) $\overline{\text{CTS1}}$ input
(24) Port P87
(UART1) $\overline{\text{RTS}}$ function enable bit
Direction register
Data bus
Port latch
(UART1) $\overline{\text{RTS1}}$ output

Fig. 15 Port block diagram (4)



DMAC index and status register (address $003F_{16}$)
DMAIS

DMAC channel 0 count register underflow flag (D0UF)
0: No underflow
1: Underflow generated

DMAC channel 0 suspend flag (D0SFI)
0: Not suspended
1: Suspended

DMAC channel 1 count register underflow flag (D1UF)
0: No underflow
1: Underflow generated

DMAC channel 1 suspend flag (D1SFI)
0: Not suspended
1: Suspended

DMAC transfer suspend control bit (DTSC)
0: Suspending only burst transfers during interrupt process
1: Suspending both burst and cycle steal transfers during interrupt process

DMAC register reload disable bit (DRLDD)
0: Enabling reload of source and destination registers of both channels
1: Disabling reload of source and destination registers of both channels

Reserved bit (“0” at read/write)

Channel index bit (DCI)
0: Channel 0 accessible
1: Channel 1 accessible

DMAC channel x mode register 1 (address 0040_{16})
DMAxM1

DMAC channel x source register increment/decrement selection bit (DxSRID)
0: Increment after transfer
1: Decrement after transfer

DMAC channel x source register increment/decrement enable bit (DxSRCE)
0: Increment/Decrement disabled (No change after transfer)
1: Increment/Decrement enabled

DMAC channel x destination register increment/decrement selection bit (DxDRID)
0: Increment after transfer
1: Decrement after transfer

DMAC channel x destination register increment/decrement enable bit (DxDRCE)
0: Increment/Decrement disabled (No change after transfer)
1: Increment/Decrement enabled

DMAC channel x data write control bit (DxDWC)
0: Writing data in reload latches and registers
1: Writing data in reload latches only

DMAC channel x disable after count register underflow enable bit (DxDAUE)
0: Channel x enabled after count register underflow
1: Channel x disabled after count register underflow

DMAC channel x register reload bit (DxRLD)
0: Not reloaded (Bit is always read as “0”)
1: Source, destination, and transfer count registers contents of channel x to be reloaded

DMAC channel x transfer mode selection bit (DxTMS)
0: Cycle steal transfer mode
1: Burst transfer mode

DMAC channel 0 mode register 2 (address 0041_{16})
DMA0M2

DMAC channel 0 hardware transfer request source bits (D0HR)

b3b2b1b0
0 0 0 0: Not used
0 0 0 1: UART1 receive interrupt
0 0 1 0: UART1 transmit interrupt
0 0 1 1: Timer Y interrupt
0 1 0 0: INT$_0$ interrupt
0 1 0 1: USB endpoint 1 IN_PKT_RDY signal (falling edge active)
0 1 1 0: USB endpoint 2 IN_PKT_RDY signal (falling edge active)
0 1 1 1: USB endpoint 3 IN_PKT_RDY signal (falling edge active)
1 0 0 0: USB endpoint 1 OUT_PKT_RDY signal (rising edge active)
1 0 0 1: USB endpoint 1 OUT_FIFO_NOT_EMPTY signal (rising edge active)
1 0 1 0: USB endpoint 2 OUT_PKT_RDY signal (rising edge active)
1 0 1 1: USB endpoint 3 OUT_PKT_RDY signal (rising edge active)
1 1 0 0: Master CPU bus interface OBE$_0$ signal (rising edge active)
1 1 0 1: Master CPU bus interface IBF$_0$ signal, data (rising edge active)
1 1 1 0: Serial I/O trasmit/receive interrupt
1 1 1 1: CNTR$_1$ interrupt

DMAC channel 0 software transfer trigger (D0SWT)
0: No action (Bit is always read as “0”)
1: Request of channel 0 transfer by writing “1” (**Note 1**)

DMAC channel 0 USB and master CPU bus interface enable bit (D0UMIE)
0: Disabled
1: Enabled

DMAC channel 0 transfer initiation source capture register reset bit (D0CRR)
0: No action (Bit is always read as “0”)
1: Reset of channel 0 capture register by writing “1” (**Note 1**)

DMAC channel 0 enable bit (D0CEN)
0: Channel 0 disabled
1: Channel 0 enabled (**Note 2**)

DMAC channel 1 mode register 2 (address 0041_{16})
DMA1M2

DMAC channel 1 hardware transfer request source bits (D1HR)

b3b2b1b0
0 0 0 0: Not used
0 0 0 1: UART2 receive interrupt
0 0 1 0: UART2 transmit interrupt
0 0 1 1: Timer X interrupt
0 1 0 0: INT$_1$ interrupt
0 1 0 1: USB endpoint 1 IN_PKT_RDY signal (falling edge active)
0 1 1 0: USB endpoint 2 IN_PKT_RDY signal (falling edge active)
0 1 1 1: USB endpoint 4 IN_PKT_RDY signal (falling edge active)
1 0 0 0: USB endpoint 1 OUT_PKT_RDY signal (rising edge active)
1 0 0 1: USB endpoint 1 OUT_FIFO_NOT_EMPTY signal (rising edge active)
1 0 1 0: USB endpoint 2 OUT_PKT_RDY signal (rising edge active)
1 0 1 1: USB endpoint 4 OUT_PKT_RDY signal (rising edge active)
1 1 0 0: Master CPU bus interface OBE$_1$ signal (rising edge active)
1 1 0 1: Master CPU bus interface IBF$_1$ signal, data (rising edge active)
1 1 1 0: Timer 1 trasmit/receive interrupt
1 1 1 1: CNTR$_0$ interrupt

DMAC channel 1 software transfer trigger (D1SWT)
0: No action (Bit is always read as “0”)
1: Request of channel 0 transfer by writing “1” (**Note 1**)

DMAC channel 1 USB and master CPU bus interface enable bit (D1UMIE)
0: Disabled
1: Enabled

DMAC channel 1 transfer initiation source capture register reset bit (D1CRR)
0: No action (Bit is always read as “0”)
1: Reset of channel 1 capture register by writing “1” (**Note 1**)

DMAC channel 1 enable bit (D1CEN)
0: Channel 0 disabled
1: Channel 0 enabled (**Note 2**)

Notes 1: This bit is automatically cleared to “0” after writing “1”.
2: When setting this bit to “1”, simultaneously set the DMAC channel x transfer initiation source capture register reset bit (bit 6 of DMAxM2) to “1”.

Fig. 32 Structure of DMACx related register

(1) Cycle Steal Transfer Mode

When the DMAC Channel x (x = 0, 1) Transfer Mode Selection Bit (DxTMS) is set to “0”, the respective DMAC Channel x operates in the cycle steal transfer mode.

When a request of the specified transfer factor is generated, the selected channel transfers one byte of data from the address indicated by the Source Register into the address indicated by the Destination Register.

There are two kinds of DMA transfer triggers supported: hardware transfer factor and software trigger. Hardware transfer factors can be selected by the DMACx (x = 0, 1) Hardware Transfer Request Factor Bit (DxHR). To only use the Interrupt Request Bit, the interrupt can be disabled by setting its Interrupt Enable Bit of Interrupt Control Register to “0”.

The DMA transfer request as a software trigger can be generated by setting the DMA Channel x (x = 0, 1) Software Transfer Trigger Bit (DxSWT) to “1”.

The Source Registers and Transfer Destination Registers can be either decreased or increased by 1 after transfer completion by setting bits 0 to 3 in the DMAC Channel x (x = 0, 1) Mode Register. When the Transfer Count Register underflows, the Source Registers and Destination Registers are reloaded from their latches if the DMAC Register Reload Disable Bit (DRLDD) is “0”. The Transfer Count Register value is reloaded after an underflow regardless of DRLDD setting. At the same time, the DMAC Interrupt Request Bit and the DMA Channel x (x = 0, 1) Count Register Underflow Flag are set to “1”.

The DMAC Channel x Disable After Count Register Underflow Enable Bit (DxDAUE) is “1”, the DMAC Channel x Enable Bit (DxCEN) goes to “0” at an under flows of Transfer Count Register.

By setting the DMAC Channel x (x = 0, 1) Register Reload Bit (DxRLD) to “1”, the Source Registers, Destination Registers, and Transfer Count Registers can be updated to the values in their respective latches.

When one signal among USB endpoint signals is selected as the hardware transfer request factor, and DMAC Channel x (x = 0, 1) USB and Master CPU Bus Interface Enable Bit (DxUMIE) is “1”; transfer between the USB FIFO and the master CPU bus interface input/output buffer can be performed effectively. This transfer function is only valid in the cycle steal mode. To validate this function, the DMAC Channel x (x = 0, 1) USB and the Master CPU Bus Interface Enable Bit (bit 5 of DxTR) must be set to “1”. The following shows an example of a transfer using this function.

Packet Transfer from USB FIFO to Master CPU Bus Interface Buffer

When the USB OUT_PKT_RDY is selected as the hardware transfer request factor; if the USB OUT_PKT_RDY is “1” and the master CPU bus interface output buffer is empty, the transfer request is generated and the transfer is initiated. The OUT_PKT_RDY retains “1” and a transfer request is generated each time the output buffer empties until all the data in the corresponding endpoint FIFO has been transferred.

The transfer ends when the last byte in the USB receive packet is transferred and the OUT_PKT_RDY flag goes to “0” (in the case of AUTO_CLR bit = “1”).

Byte Transfer from USB FIFO to Master CPU Bus Interface Buffer

When the USB Endpoint 1 OUT_FIFO_NOT_EMPTY is selected as a hardware transfer request factor, if there is data in the USB Endpoint 1 FIFO and the master CPU bus interface output buffer is empty; a transfer request is generated and the transfer is initiated. The transfer is performed by unit of one byte.

Transfer from Master CPU Bus Interface Buffer to USB FIFO

When the USB Endpoint X (X = 1 to 4) IN_PKT_RDY (IN_PKT_RDY = “0”) is selected as a hardware transfer request factor, if there is data in the master CPU bus interface output buffer and the data in the USB FIFO is within the specified packet size, a transfer request is generated.

The DMA transfer is terminated when a command (A0 = “1”) is input to the master CPU bus interface input buffer.

The timing chart for a cycle steal transfer caused by a hardware-related transfer request and a software trigger are shown in Figure 33 and 34, respectively.

USB Reception

Endpoint 0 to Endpoint 4 have OUT (receive) FIFOs individually. Each endpoint's FIFO is configured in following way:

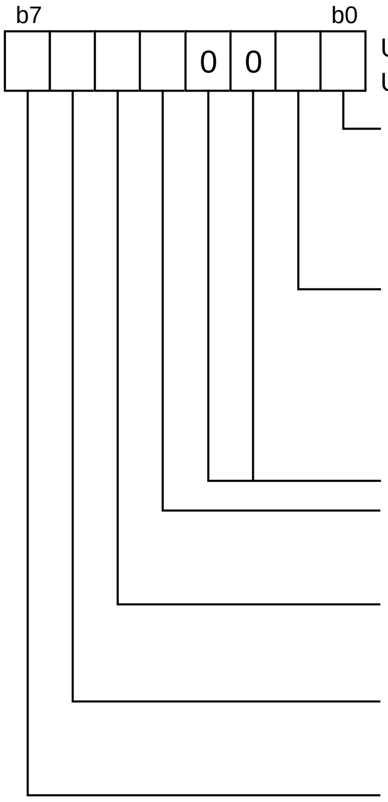
Endpoint 0:		16-byte
Endpoint 1:	Mode 0:	800-byte
	Mode 1:	1024-byte
	Mode 2:	2048-byte
	Mode 3:	0-byte
	Mode 4:	1280-byte
	Mode 5:	1168-byte
Endpoint 2:	Mode 0:	32-byte
	Mode 1:	128-byte
Endpoint 3:		16-byte
Endpoint 4:		16-byte

When Endpoint 1 or Endpoint 2 is used for data receive, the OUT FIFO size can be selected. Endpoint 1 and Endpoint 2 have programmable IN-FIFOs size; 6 modes for Endpoint 1, and 2 modes for Endpoint 2. Each mode can be selected by the USB endpoint FIFO mode selection register (address 005F$_{16}$).
Data transmitted from the host-PC is stored in Endpoint x FIFO (0060$_{16}$ to 0064$_{16}$). Every time the data is stored in the FIFO, the internal OUT FIFO write pointer is increased by 1. When one complete data packet is stored, the OUT_PKT_RDY flag is set to "1" and the number of received data packets is stored in USB Endpoint x OUT write count registers (Low and High). When the AUTO_CLR bit is "1" and the received data is read out from the OUT FIFO, the OUT_PKT_RDY flag is cleared to "0". When the AUTO_CLR bit is "1", the OUT_PKT_RDY flag will not be cleared automatically by the FIFO read; it must be cleared by software. (The AUTO-CLR bit function is not applicable in Endpoint 0.)

When MAXP size ≤ (a half of OUT FIFO size), the OUT_FIFO can receive 2 packets (double buffer). At this time, the OUT_ FIFO status can be checked by the OUT_PKT_RDY flag. When the FIFO holds two packets and one packet is read from the FIFO, the OUT_PKT_RDY flag is not cleared even if it is set to "0". (The flag returns from "0" to "1" in one ϕ cycle after the read-out). During double buffer mode, the USB Endpoint x OUT write count registers (Low and High) holds the number of previously received packets. This count register is updated after reading out one of packets in the OUT FIFO and clearing the OUT_PKT_RDY flag to "0".

TOGGLE Initialization

In order to initialize the data toggle sequence bit of the endpoint, in other words, resetting the next data packet to DATA0; set the ISO/TOGGLE_INT bit to "1" and then clear back to "0".



USB interrupt status register 2 (address 0053_{16})
USBIS2

USB endpoint 4 IN interrupt status flag (INTST8)
0: Except the following conditions
1: Set at which of the following conditions:
- A packet data of endpoint 4 is successfully sent
- UNDER_RUN bit of endpoint 4 is set to “1”.

USB endpoint 4 OUT interrupt status flag (INTST9)
0: Except the following conditions
1: Set at any one of the following conditions:
- A packet data of endpoint 4 is successfully received
- OVER_RUN bit of endpoint 4 is set to “1”
- FORCE_STALL bit of endpoint 4 is set to “1”.

Reserved bit (“0” at read/write)

USB overrun/underrun interrupt status flag (INTST12)
0: Except the following condition
1: Set at an occurrence of overrun/underrun (for isochronous data transfer)

USB reset interrupt status flag (INTST13)
0: Except the following condition
1: Set at receiving of USB reset signal

USB resume signal interrupt status flag (INTST14)
0: Except the following condition
1: Set at receiving of resume signal

USB suspend signal interrupt status flag (INTST15)
0: Except the following condition
1: Set at receiving of suspend signal

Fig. 41 Structure of USB interrupt status register 2

[USB Frame Number Registers Low and High]
USBSOFL, USBFOFH

These 11-bit registers contain the frame number of the SOF token received from the host computer. These are read-only registers.

[USB Endpoint Index Register] USBINDEX

This register specifies the accessible endpoint. It serves as an index to endpoint-specific USB Endpoint x IN Control Register, USB Endpoint x OUT Control Register, USB Endpoint x IN Max. Packet Size Register, USB Endpoint x OUT Max. Packet Size Register, USB Endpoint x OUT Write Count Register, and USB FIFO Mode Selection Register (x = 0 to 4).

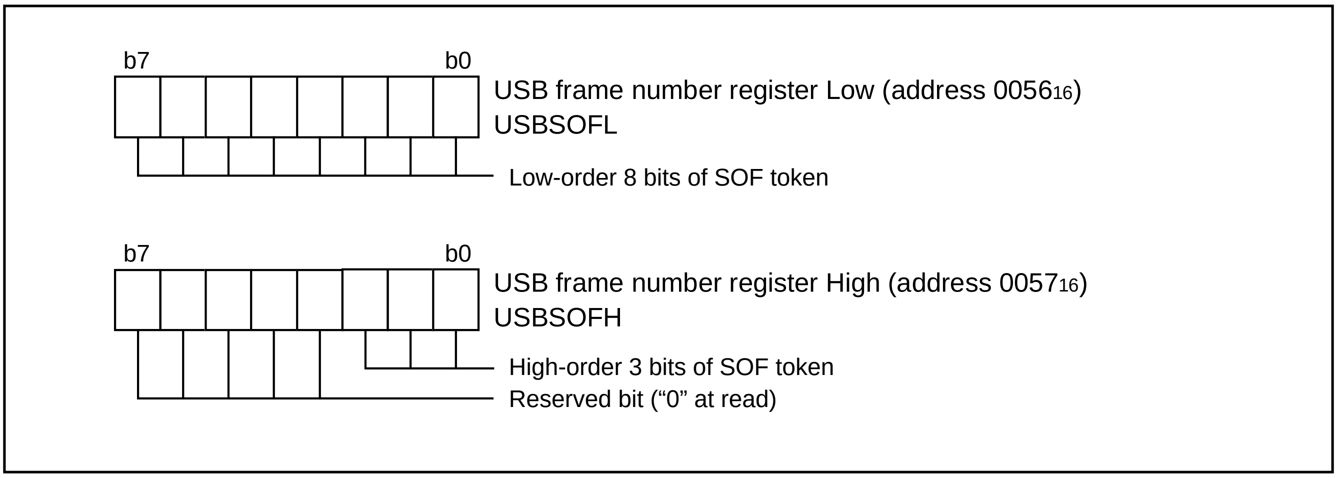

b7
b0
USB frame number register Low (address 0056$_{16}$)
USBSOFL
Low-order 8 bits of SOF token
b7
b0
USB frame number register High (address 0057$_{16}$)
USBSOFH
High-order 3 bits of SOF token
Reserved bit (“0” at read)

Fig. 44 Structure of USB frame number registers

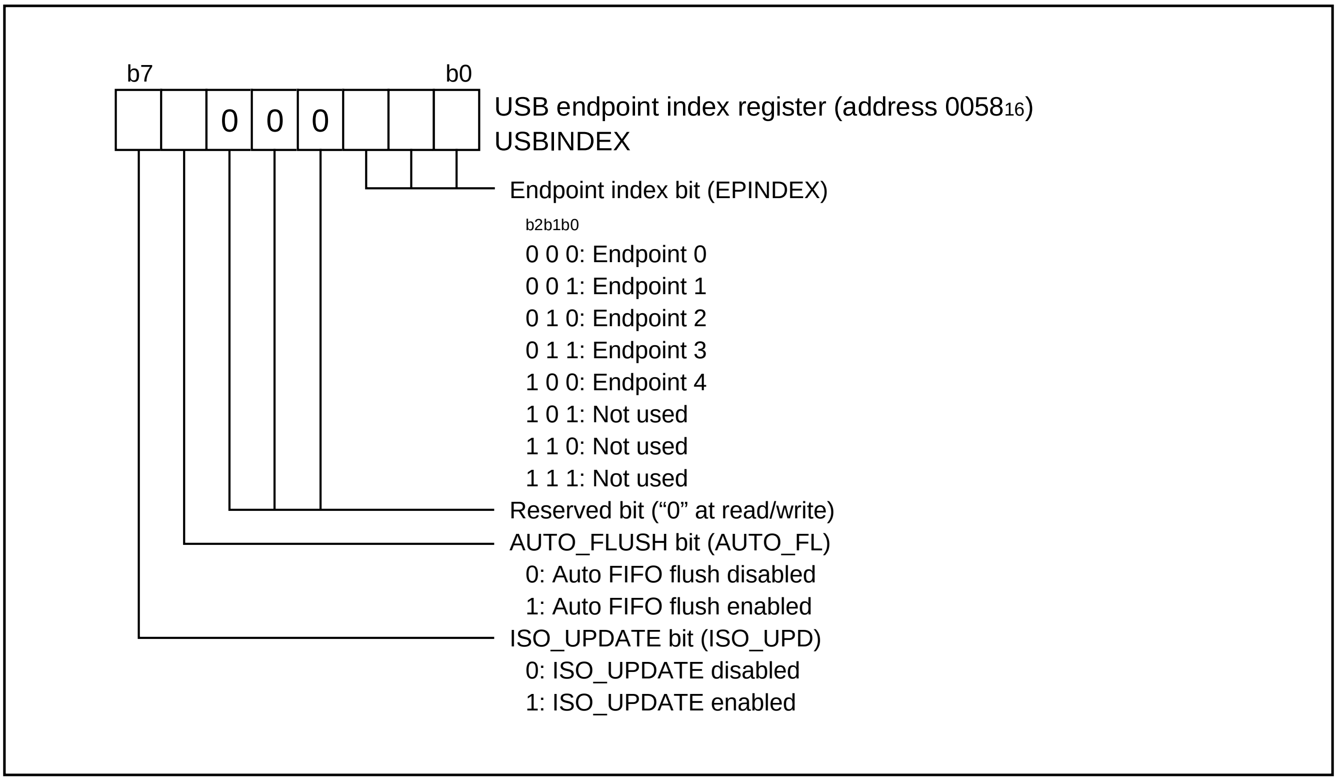


Fig. 45 Structure of USB frame number registers

MASTER CPU BUS INTERFACE

The 7641 group internally has a 2-byte bus interface which control signals from the host CPU side can operate (slave mode).
This bus interface allows the 7641 group to be directly connected with a R/W type of CPU bus or a $\overline{RD}$ and $\overline{WR}$ separated type of CPU bus. Figure 56 shows the block diagram of master CPU bus interface function.

The data bus buffer function I/O pins ($P5_2$ – $P5_7$, P6, $P7_2$–$P7_4$) also function as the normal I/O ports. When the Master CPU Bus Interface Enable bit of Data Bus Buffer Control Register (bit 6 of address $004A_{16}$) is “0”, these pins become the normal I/O ports. When it is “1”, these pins become the master CPU bus interface function pins.
Additionally, when using the master CPU bus interface function, set port P6 to input mode by setting “00_{16}” into its port direction register (address 0015_{16}).

The selection of either the single data bus buffer mode, which uses 1 byte: data bus buffer 0 only, or the double data bus buffer mode, which uses 2 bytes: data bus buffer 0 and data bus buffer 1, is performed by the Data Bus Buffer Function Select Bit of Data Bus Buffer Control Register 1 (bit 7 of address $004E_{16}$). Port $P7_2$ becomes $\overline{S_1}$ input pin in the double data bus buffer mode.
When data is written from the host CPU side, an input buffer full interrupt occurs. When data is read from the host CPU, an output buffer empty interrupt occurs. The 7641 group shares two input buffer full interrupt requests and two output buffer empty interrupt requests as shown in Figure 54, respectively.

The 7641 group can also operate the master CPU bus interface connecting with the Built-in DMAC. This could transfer a large amount of data fast.
An input signal level of data bus buffer function input pins can be selected between a CMOS level and a TTL level. Set it using the Master CPU Bus Input Level Select Bit of Port Control Register (address 0010_{16})
.

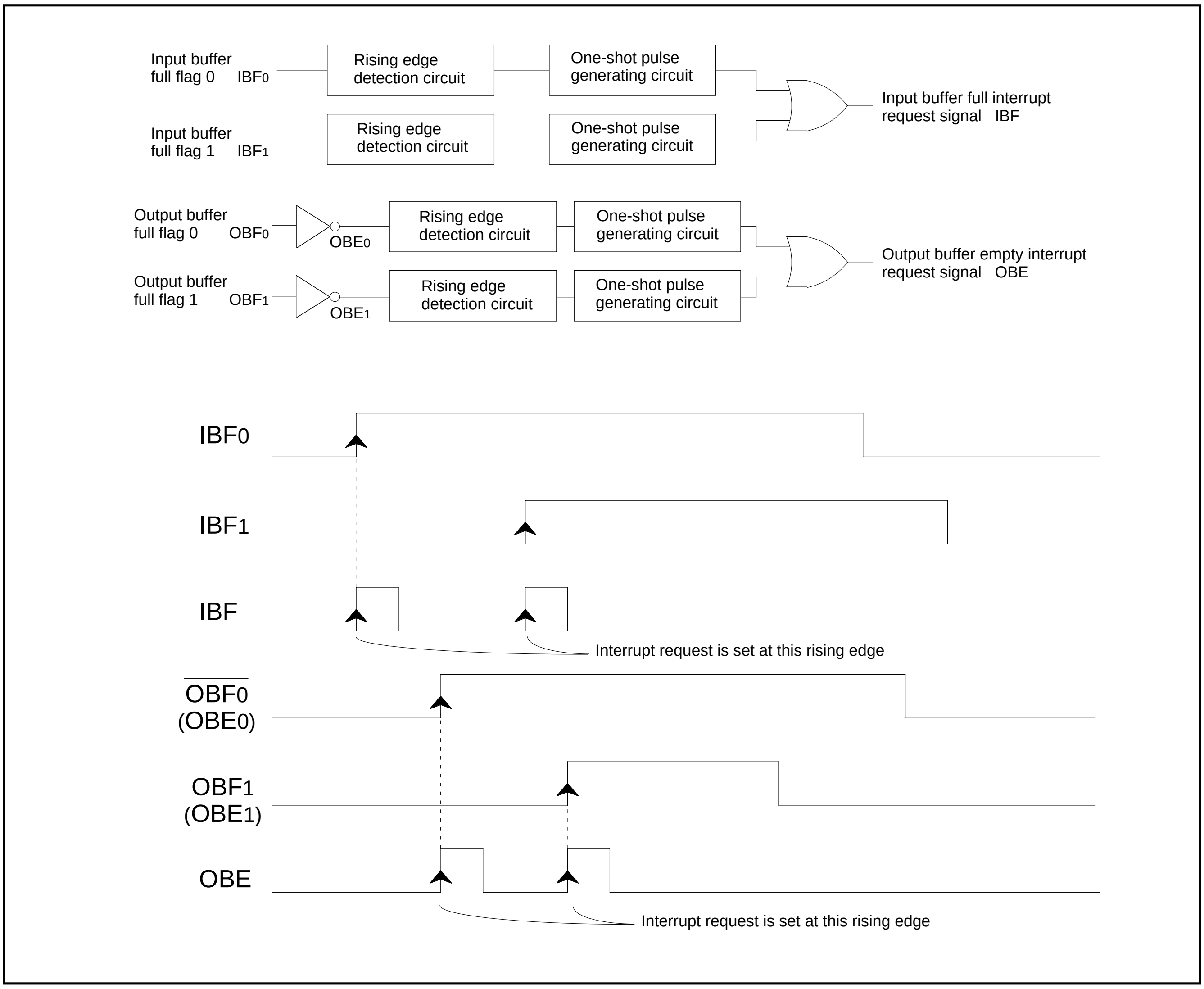


Fig. 54 Interrupt request circuit of data bus buffer

	Register	Address	Register contents
(1)	CPU mode register A (CPUA)	0000_{16}	0 0 0 0 1 1 0 0
(2)	CPU mode register B (CPUB)	0001_{16}	1 0 0 0 0 0 1 1
(3)	Interrupt request register A (IREQA)	0002_{16}	00_{16}
(4)	Interrupt request register B (IREQB)	0003_{16}	00_{16}
(5)	Interrupt request register C (IREQC)	0004_{16}	00_{16}
(6)	Interrupt control register A (ICONA)	0005_{16}	00_{16}
(7)	Interrupt control register B (ICONB)	0006_{16}	00_{16}
(8)	Interrupt control register C (ICONC)	0007_{16}	00_{16}
(9)	Port P0 (P0)	0008_{16}	00_{16}
(10)	Port P0 direction register (P0D)	0009_{16}	00_{16}
(11)	Port P1 (P1)	$000A_{16}$	00_{16}
(12)	Port P1 direction register (P1D)	$000B_{16}$	00_{16}
(13)	Port P2 (P2)	$000C_{16}$	00_{16}
(14)	Port P2 direction register (P2D)	$000D_{16}$	00_{16}
(15)	Port P3 (P3)	$000E_{16}$	00_{16}
(16)	Port P3 direction register (P3D)	$000F_{16}$	00_{16}
(17)	Port control register (PTC)	0010_{16}	00_{16}
(18)	Interrupt polarity select register (IPOL)	0011_{16}	00_{16}
(19)	Port P2 pull-up control register (PUP2)	0012_{16}	00_{16}
(20)	USB control register (USBC)	0013_{16}	00_{16}
(21)	Port P6 (P6)	0014_{16}	00_{16}
(22)	Port P6 direction register (P6D)	0015_{16}	00_{16}
(23)	Port P5 (P5)	0016_{16}	00_{16}
(24)	Port P5 direction register (P5D)	0017_{16}	00_{16}
(25)	Port P4 (P4)	0018_{16}	00_{16}
(26)	Port P4 direction register (P4D)	0019_{16}	00_{16}
(27)	Port P7 (P7)	$001A_{16}$	00_{16}
(28)	Port P7 direction register (P7D)	$001B_{16}$	00_{16}
(29)	Port P8 (P8)	$001C_{16}$	00_{16}
(30)	Port P8 direction register (P8D)	$001D_{16}$	00_{16}
(31)	Clock control register (CCR)	$001F_{16}$	00_{16}
(32)	Timer XL (TXL)	0020_{16}	FF_{16}
(33)	Timer XH (TXH)	0021_{16}	FF_{16}
(34)	Timer YL (TYL)	0022_{16}	FF_{16}
(35)	Timer YH (TYH)	0023_{16}	FF_{16}
(36)	Timer 1 (T1)	0024_{16}	FF_{16}
(37)	Timer 2 (T2)	0025_{16}	0 0 0 0 0 0 0 1
(38)	Timer 3 (T3)	0026_{16}	FF_{16}
(39)	Timer X mode register (TXM)	0027_{16}	00_{16}
(40)	Timer Y mode register (TYM)	0028_{16}	00_{16}
(41)	Timer 123 mode register (T123M)	0029_{16}	00_{16}
(42)	Serial I/O control register 1 (SIOCON1)	$002B_{16}$	0 1 0 0 0 0 0 0
(43)	Serial I/O control register 2 (SIOCON2)	$002C_{16}$	0 0 0 1 1 0 0 0
(44)	Special count source generator 1 (SCSG1)	$002D_{16}$	FF_{16}
(45)	Special count source generator 2 (SCSG2)	$002E_{16}$	FF_{16}
(46)	Special count source mode register (SCSGM)	$002F_{16}$	00_{16}
(47)	UART1 mode register (U1MOD)	0030_{16}	00_{16}

	Register	Address	Register contents
(48)	UART1 status register (U1STS)	0032_{16}	0 0 0 0 0 0 1 1
(49)	UART1 control register (U1CON)	0033_{16}	00_{16}
(50)	UART1 RTS control register (U1RTSC)	0036_{16}	1 0 0 0 0 0 0 0
(51)	UART2 mode register (U2MOD)	0038_{16}	00_{16}
(52)	UART2 status register (U2STS)	$003A_{16}$	0 0 0 0 0 0 1 1
(53)	UART2 control register (U2CON)	$003B_{16}$	00_{16}
(54)	UART2 RTS control register (U2RTSC)	$003E_{16}$	1 0 0 0 0 0 0 0
(55)	DMAC index and status register (DMAIS)	$003F_{16}$	00_{16}
(56)	DMAC channel x mode register 1 (DMAx1)	0040_{16}	00_{16}
(57)	DMAC channel x mode register 2 (DMAx2)	0041_{16}	00_{16}
(58)	DMAC channel x source register Low (DMAxSL)	0042_{16}	00_{16}
(59)	DMAC channel x source register High (DMAxSH)	0043_{16}	00_{16}
(60)	DMAC channel x destination register Low (DMAxDL)	0044_{16}	00_{16}
(61)	DMAC channel x destination register High (DMAxDH)	0045_{16}	00_{16}
(62)	DMAC channel x transfer count register Low (DMAxCL)	0046_{16}	00_{16}
(63)	DMAC channel x transfer count register High (DMAxCH)	0047_{16}	00_{16}
(64)	Data bus buffer register 0 (DBB0)	0048_{16}	00_{16}
(65)	Data bus buffer status register 0 (DBBS0)	0049_{16}	00_{16}
(66)	Data bus buffer control register 0 (DBBC0)	$004A_{16}$	00_{16}
(67)	Data bus buffer register 1 (DBB1)	$004C_{16}$	00_{16}
(68)	Data bus buffer status register 1 (DBBS1)	$004D_{16}$	00_{16}
(69)	Data bus buffer control register 1 (DBBC1)	$004E_{16}$	00_{16}
(70)	USB address register (USBA)	0050_{16}	00_{16}
(71)	USB power management register (USBPM)	0051_{16}	00_{16}
(72)	USB interrupt status register 1 (USBIS1)	0052_{16}	00_{16}
(73)	USB interrupt status register 2 (USBIS2)	0053_{16}	00_{16}
(74)	USB interrupt enable register 1 (USBIE1)	0054_{16}	FF_{16}
(75)	USB interrupt enable register 2 (USBIE2)	0055_{16}	0 0 1 1 0 0 1 1
(76)	USB frame number register Low (USBSOFL)	0056_{16}	00_{16}
(77)	USB frame number register High (USBSOFH)	0057_{16}	00_{16}
(78)	USB endpoint index register (USBINDEX)	0058_{16}	00_{16}
(79)	USB endpoint x IN control register (IN_CSR)	0059_{16}	00_{16}
(80)	USB endpoint x OUT control register (OUT_CSR)	$005A_{16}$	00_{16}
(81)	USB endpoint x IN max. packet size register (IN_MAXP) **(Note 1)**	$005B_{16}$	0 0 0 0 1 0 0 0
(82)	USB endpoint x OUT max. packet size register (OUT_MAXP) **(Note 1)**	$005C_{16}$	0 0 0 0 1 0 0 0
(83)	USB endpoint x OUT write count register Low (WRT_CNTL)	$005D_{16}$	00_{16}
(84)	USB endpoint x OUT write count register High (WRT_CNTH)	$005E_{16}$	00_{16}
(85)	USB endpoint FIFO mode register (USBFIFOMR)	$005F_{16}$	00_{16}
(86)	Flash memory control register (FMCR) **(Note 3)**	$006A_{16}$	0 0 0 0 0 0 0 1
(87)	Frequency synthesizer control register (FSC)	$006C_{16}$	0 1 1 0 0 0 0 0
(88)	Frequency synthesizer multiply register 1 (FSM1)	$006D_{16}$	FF_{16}
(89)	Frequency synthesizer multiply register 2 (FSM2)	$006E_{16}$	FF_{16}
(90)	Frequency synthesizer divide register (FSM2)	$006F_{16}$	FF_{16}
(91)	ROM code protect control register (ROMCP) **(Note 3)**	FFC916	FF_{16}
(92)	Processor status register	(PS)	× × × × × 1 × ×
(93)	Program counter	(PC_H)	$FFFB_{16}$ contents
		(PC_L)	$FFFA_{16}$ contents

X : Not fixed

Notes 1: When using the endpoint 1, this contents are “01_{16}”.

2: Since the initial values for other than above mentioned registers and RAM contents are indefinite at reset, they must be set.

3: The flash memory control register and the ROM code protect control register exists in the flash memory version only.

Fig. 63 Internal status at reset

Slow Memory Wait

The 7641 Group is equipped with the slow memory wait function (Software wait, RDY wait, and Extended RDY wait: software wait plus RDY input anytime wait) for easier interfacing with external devices that have long access times. The slow memory wait function can be enabled in the memory expansion mode and microprocessor mode. The appropriate wait mode is selected by setting bits 0 to 3 of CPU mode register B (address 0001_{16}). This function can extend the read cycle or write cycle only for access to an external memory. However, this wait function cannot be enabled for access to addresses 0008_{16} to $000F_{16}$.

(1) Software wait

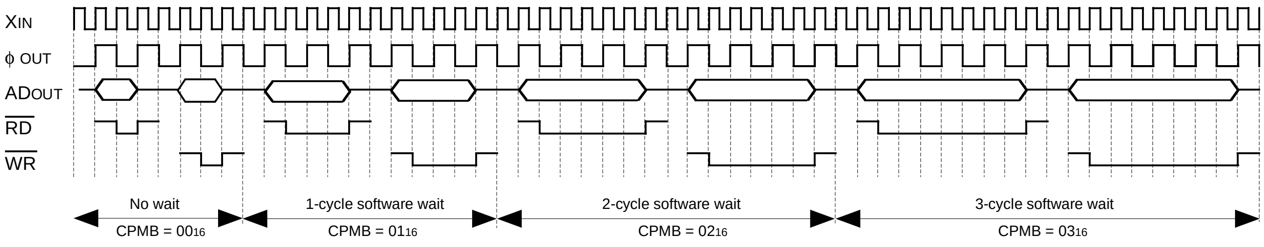
The software wait is selected by setting "00" to the Slow Memory Wait Mode Select Bits of CPU mode register B (address 0001_{16}). Read/write cycles ("L" width of $\overline{RD}$ pin/$\overline{WR}$ pin) can be extended by one to three φ cycles. The number of cycles to be extended can be selected with the Slow Memory Wait Select Bits. When the software wait function is selected, the RDY pin status becomes invalid.

(2) RDY wait

RDY Wait is selected by setting "10" to the Slow Memory Wait Mode Select Bits of CPU mode register B (address 0001_{16}). When a fixed time of "L" is input to the RDY pin at the beginning of a read/write cycle (before φ cycle falls), the MCU goes to the RDY state. The read/write cycle can then be extended by one to three φ cycles. The number of φ cycles to be added can be selected by the Slow Memory Wait Bits.

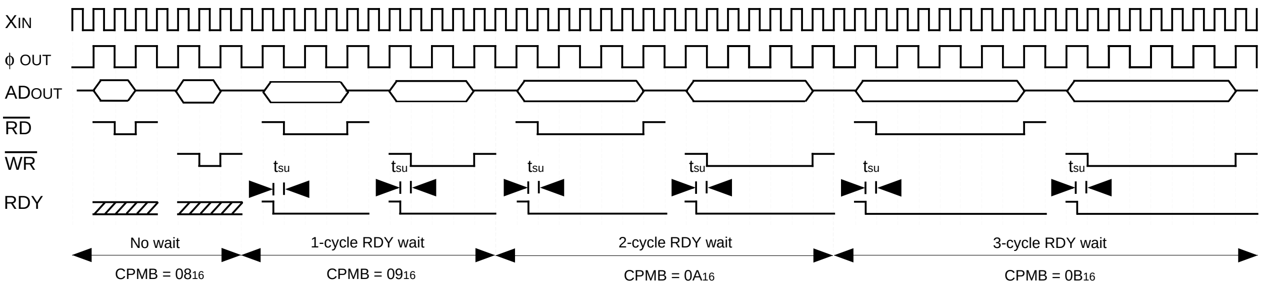
(3) Software wait + Extended RDY wait

Extended RDY Wait is selected by setting "11" to the Slow Memory Wait Mode Select Bits of CPU mode register B (address 0001_{16}). The read/write cycle can be extended when a fixed time of "L" is input to the RDY pin at the beginning of a read/write cycle (before φ cycle falls). The RDY pin state is checked continually at each fall of φ cycle until the RDY pin goes to "H". When "H" is input to the RDY pin, the wait is released within 1, 2, or 3 φ cycles (as selected with the Slow Memory Wait Bits).



Note: This diagram assumes φ = XIN/2.

Fig. 72 Software wait timing diagram



Note: This diagram assumes φ = XIN/2.

Fig. 73 RDY wait timing diagram

In Vcc = 5 V

Table 13 Electrical characteristics (2) (Vcc = 4.15 to 5.25 V, Vss = 0 V, Ta = –20 to 70°C, unless otherwise noted)

Symbol	Parameter	Test conditions	Limits			Unit
			Min.	Typ.	Max.	
ICC	Power source current (Output transistor is isolated.)	Normal mode (**Note 1**) f(XIN) = 24 MHz, ϕ = 12 MHz USB operating Frequency synthesizer ON		40	90	mA
		Wait mode (**Note 2**) f(XIN) = 24 MHz, ϕ = 12 MHz USB block enabled, USB clock stopped, Frequency synthesizer ON		5.0	11	mA
		Wait mode (**Note 3**) f(XCIN) = 32 kHz, ϕ = 16 kHz USB block disabled Frequency synthesizer OFF USB transceiver DC-DC converter OFF			10	µA
		Stop mode USB transceiver DC-DC converter ON Low current mode (USBC3 = “1”)		100	250	µA
		Stop mode USB transceiver DC-DC converter OFF Ta = 25 °C			1.0	µA
		Stop mode USB transceiver DC-DC converter OFF Ta = 70 °C			10	µA

<Test conditions>

Notes 1: Operating in single-chip mode
Clock input from XIN pin (XOUT oscillator stopped)
USB operating with USB transceiver DC-DC converter enabled
Operating functions: Frequency synthesizer, CPU, two UARTs, DMAC, Timers and Count source generator
Disabled functions: Master CPU bus interface and Serial I/O

2: Operating in single-chip mode with Wait mode
Clock input from XIN pin (XOUT oscillator stopped)
USB suspended due to USB clock stopped with USB transceiver DC-DC converter enabled
Operating functions: Frequency synthesizer, Timers and Count source generator
Disabled functions: CPU, two UARTs, DMAC, Master CPU bus interface and Serial I/O

3: Operating in single-chip mode with Wait mode
XIN - XOUT oscillator stopped
Clock input from XCIN pin (XCOUT oscillator stopped)
USB stopped, USB clock stopped and USB transceiver DC-DC converter disabled
Operating functions: Timers and Count source generator
Disabled functions: Frequency synthesizer, CPU, two UARTs, DMAC, Master CPU bus interface and Serial I/O

Notes 1: The total peak output current is the peak value of the peak currents flowing through all the applicable ports. The total average output current is the average value measured over 100 ms flowing through all the applicable ports.

2: The peak output current is the peak current flowing in each port.

3: The average output current is an average value measured over 100 ms.

4: The duty of oscillation frequency is 50 %.

5: Connect a ceramic resonator or a quartz-crystal oscillator between the X_{IN} and X_{OUT} pins. Its maximum oscillation frequency must be 24 MHz. However, make sure to set ϕ to 6 MHz or slower. More faster clocks are required as the f(X_{IN}) when using the frequency synthesizer as possible.

6: Connect a ceramic resonator or a quartz-crystal oscillator between the X_{CIN} and X_{COUT} pins. Its maximum oscillation frequency must be 50 kHz. Input an external clock having 5 MHz (max.) frequency from the X_{CIN} pin.

Timing Requirements
In Vcc = 3 V

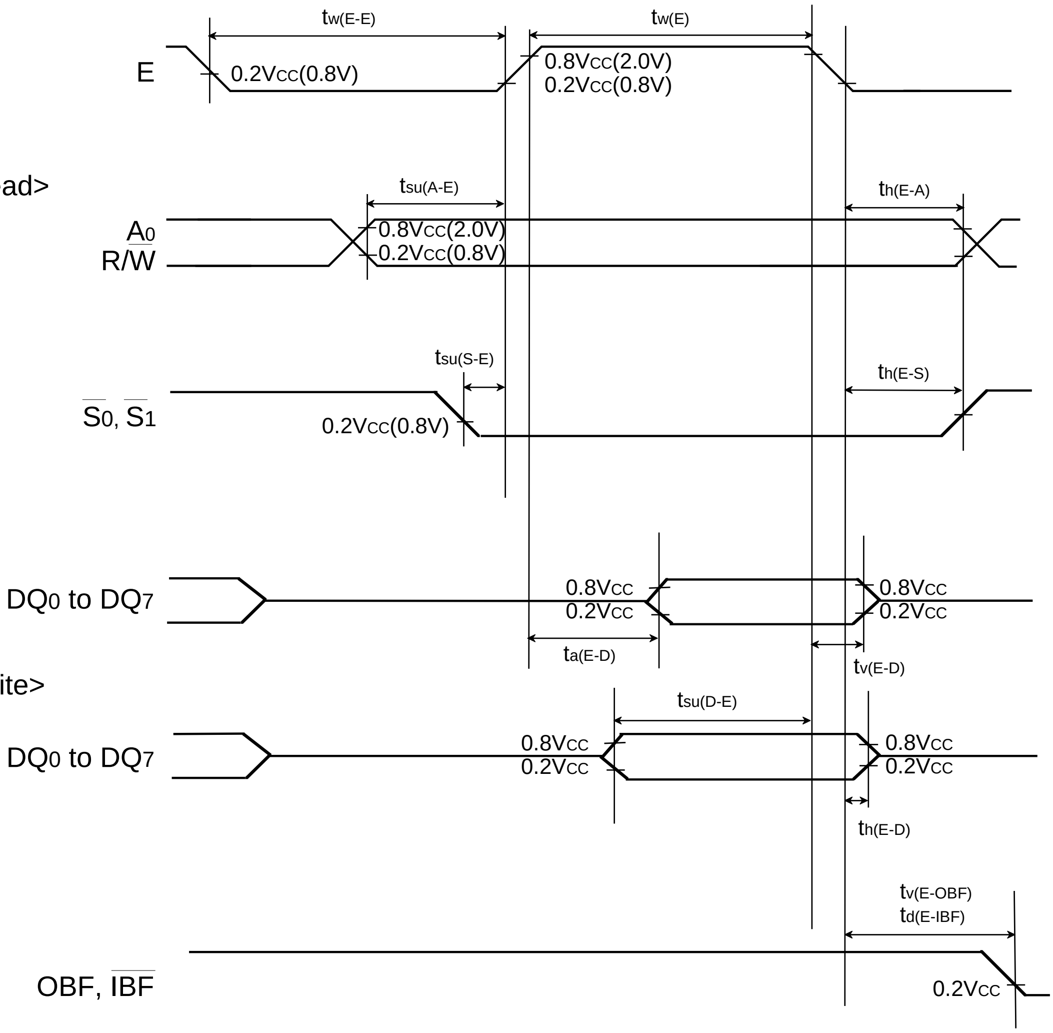
Table 21 Timing requirements (Vcc = 3.0 to 3.6 V, Vss = 0 V, Ta = –20 to 70°C, unless otherwise noted)

Symbol	Parameter	Limits			Unit
		Min.	Typ.	Max.	
$t_{w}(\overline{\text{RESET}})$	Reset input “L” pulse width	2			µs
$t_{C}(X_{IN})$	Main clock input cycle time (**Note**)	41.66			ns
$t_{WH}(X_{IN})$	Main clock input “H” pulse width	0.4•tc(X_{IN})			ns
$t_{WL}(X_{IN})$	Main clock input “L” pulse width	0.4•tc(X_{IN})			ns
$t_{C}(X_{CIN})$	Sub-clock input cycle time	200			ns
$t_{WH}(X_{CIN})$	Sub-clock input “H” pulse width	0.4•tc(X_{CIN})			ns
$t_{WL}(X_{CIN})$	Sub-clock input “L” pulse width	0.4•tc(X_{CIN})			ns
t_{C}(INT)	INT_0, INT_1 input cycle time	250			ns
t_{WH}(INT)	INT_0, INT_1 input “H” pulse width	110			ns
t_{WL}(INT)	INT_0, INT_1 input “L” pulse width	110			ns
t_{C}(CNTRI)	$CNTR_0$, $CNTR_1$ input cycle time	250			ns
t_{WH}(CNTRI)	$CNTR_0$, $CNTR_1$ input “H” pulse width	110			ns
t_{WL}(CNTRI)	$CNTR_0$, $CNTR_1$ input “L” pulse width	110			ns
t_{d}(ϕ -TOUT)	Timer T_{OUT} delay time			17	ns
t_{d}(ϕ -$CNTR_0$)	Timer $CNTR_0$ delay time (Pulse output mode)			16	ns
$t_{C}(CNTR_{E0})$	Timer $CNTR_0$ input cycle time (Event counter mode)	250			ns
$t_{WH}(CNTR_{E0})$	Timer $CNTR_0$ input “H” pulse width (Event counter mode)	0.4•tc($CNTR_{E0}$)			ns
$t_{WL}(CNTR_{E0})$	Timer $CNTR_0$ input “L” pulse width (Event counter mode)	0.4•tc($CNTR_{E0}$)			ns
t_{d}(ϕ -$CNTR_1$)	Timer $CNTR_1$ delay time (Pulse output mode)			15	ns
$t_{C}(CNTR_{E1})$	Timer $CNTR_1$ input cycle time (Event counter mode)	250			ns
$t_{WH}(CNTR_{E1})$	Timer $CNTR_1$ input “H” pulse width (Event counter mode)	0.4•tc($CNTR_{E1}$)			ns
$t_{WL}(CNTR_{E1})$	Timer $CNTR_1$ input “L” pulse width (Event counter mode)	0.4•tc($CNTR_{E1}$)			ns
$t_{C}(SCLK_{E})$	Serial I/O external clock input cycle time	450			ns
$t_{WH}(SCLK_{E})$	Serial I/O external clock input “H” pulse width	220			ns
$t_{WL}(SCLK_{E})$	Serial I/O external clock input “L” pulse width	190			ns
$t_{su}(SRXD\text{-}SCLK_{E})$	Serial I/O input setup time (external clock)	20			ns
$t_{h}(SCLK_{E}\text{-}SRXD)$	Serial I/O input hold time (external clock)	15			ns
$t_{d}(SCLK_{E}\text{-}STXD)$	Serial I/O output delay time (external clock)			34	ns
$t_{v}(SCLK_{E}\text{-}SRDY)$	Serial I/O $\overline{\text{SRDY}}$ valid time (external clock)			35	ns
$t_{c}(SCLK_{I})$	Serial I/O internal clock output cycle time	300			ns
$t_{WH}(SCLK_{I})$	Serial I/O internal clock output “H” pulse width	0.5•tc($SCLK_{I}$) – 5			ns
$t_{WL}(SCLK_{I})$	Serial I/O internal clock output “L” pulse width	0.5•tc($SCLK_{I}$) – 5			ns
$t_{su}(SRXD\text{-}SCLK_{I})$	Serial I/O input setup time (internal clock)	20			ns
$t_{h}(SCLK_{I}\text{-}SRXD)$	Serial I/O input hold time (internal clock)	5			ns
$t_{d}(SCLK_{I}\text{-}STXD)$	Serial I/O output delay time (internal clock)			5	ns

Note: Make sure not to exceed 6 MHz of ϕ, in other words, tc(ϕ) ≥ 166.66 ns).

● Timing diagram

[Master CPU bus interface: R/$\overline{W}$ mode]



Note: This timing applies in the case of the master bus input level select bit (PTC7) = “1” (TTL level input)

Fig. 85 Timing diagram (5)

Outline Performance (CPU Rewrite Mode)

CPU rewrite mode is usable in the single-chip, memory expansion or Boot mode. The only User ROM area can be rewritten in CPU rewrite mode.

In CPU rewrite mode, the CPU erases, programs and reads the internal flash memory by executing software commands. This rewrite control program must be transferred to a memory such as the internal RAM before it can be executed.

The MCU enters CPU rewrite mode by applying 4.50 V to 5.25 V to the CNVSS pin and setting “1” to the CPU Rewrite Mode Select Bit (bit 1 of address $006A_{16}$). Software commands are accepted once the mode is entered.

Use software commands to control program and erase operations. Whether a program or erase operation has terminated normally or in error can be verified by reading the status register.

Figure 89 shows the flash memory control register.

Bit 0 is the RY/$\overline{\text{BY}}$ status flag used exclusively to read the operating status of the flash memory. During programming and erase operations, it is “0” (busy). Otherwise, it is “1” (ready).

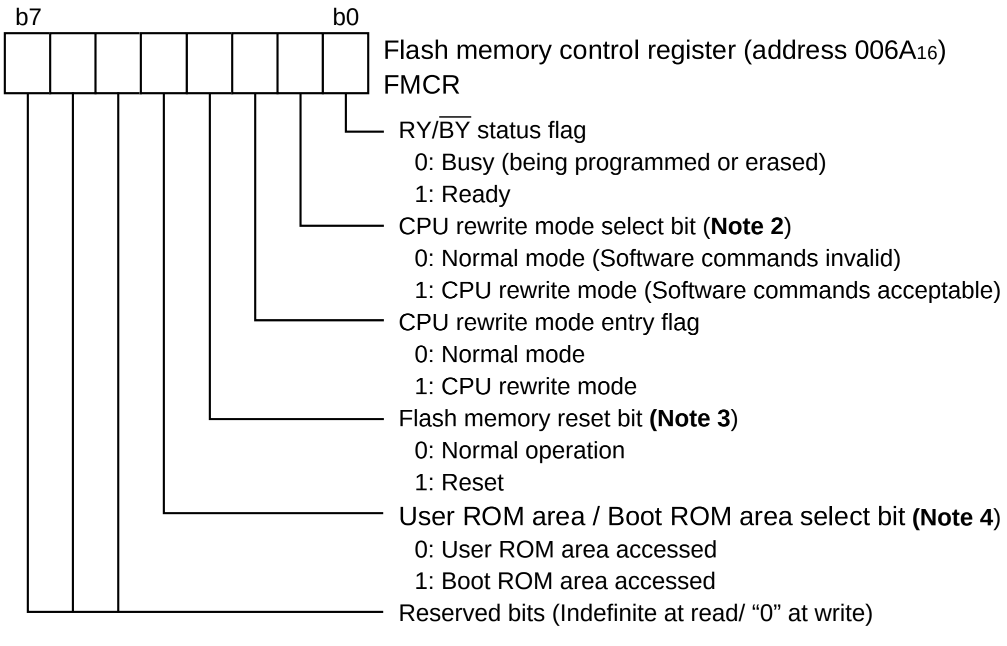
Bit 1 is the CPU Rewrite Mode Select Bit. When this bit is set to “1”, the MCU enters CPU rewrite mode. Software commands are accepted once the mode is entered. In CPU rewrite mode, the CPU becomes unable to access the internal flash memory directly. Therefore, use the control program in a memory other than internal flash memory for write to bit 1. To set this bit to “1”, it is necessary to write “0” and then write “1” in succession. The bit can be set to “0” by only writing “0”.

Bit 2 is the CPU Rewrite Mode Entry Flag. This flag indicates “1” in CPU rewrite mode, so that reading this flag can check whether CPU rewrite mode has been entered or not.

Bit 3 is the flash memory reset bit used to reset the control circuit of internal flash memory. This bit is used when exiting CPU rewrite mode and when flash memory access has failed. When the CPU Rewrite Mode Select Bit is “1”, setting “1” for this bit resets the control circuit. To set this bit to “1”, it is necessary to write “0” and then write “1” in succession. To release the reset, it is necessary to set this bit to “0”.

Bit 4 is the User Area/Boot Area Select Bit. When this bit is set to “1”, Boot ROM area is accessed, and CPU rewrite mode in Boot ROM area is available. In Boot mode, this bit is set to “1” automatically. Reprogramming of this bit must be in a memory other than internal flash memory.

Figure 90 shows a flowchart for setting/releasing CPU rewrite mode.


b7
b0
Flash memory control register (address 006A16)
FMCR
RY/BY status flag
0: Busy (being programmed or erased)
1: Ready
CPU rewrite mode select bit (Note 2)
0: Normal mode (Software commands invalid)
1: CPU rewrite mode (Software commands acceptable)
CPU rewrite mode entry flag
0: Normal mode
1: CPU rewrite mode
Flash memory reset bit (Note 3)
0: Normal operation
1: Reset
User ROM area / Boot ROM area select bit (Note 4)
0: User ROM area accessed
1: Boot ROM area accessed
Reserved bits (Indefinite at read/ “0” at write)

Notes **1**: The contents of flash memory control register are “XXX00001” just after reset release.

2: For this bit to be set to “1”, the user needs to write “0” and then “1” to it in succession. If it is not this procedure, this bit will not be set to ”1”. Additionally, it is required to ensure that no interrupt will be generated during that interval.
Use the control program in the area except the built-in flash memory for write to this bit.

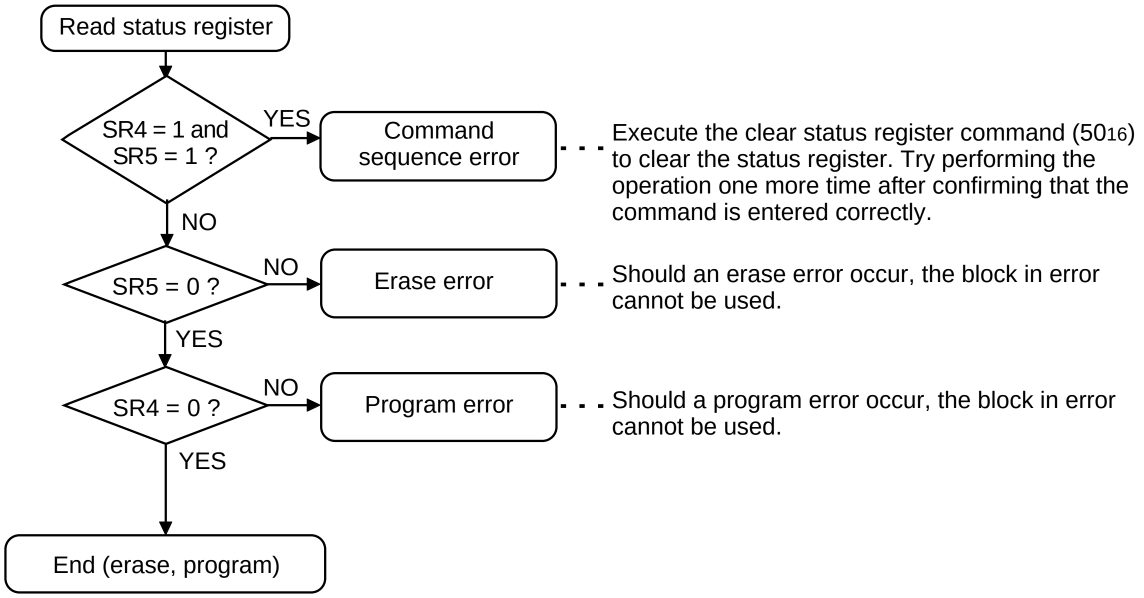
3: This bit is valid when the CPU rewrite mode select bit is “1”. Set this bit 3 to “0” subsequently after setting bit 3 to “1”.

4: Use the control program in the area except the built-in flash memory for write to this bit.

Fig. 89 Structure of flash memory control register

Full Status Check

By performing full status check, it is possible to know the execution results of erase and program operations. Figure 93 shows a full status check flowchart and the action to be taken when each error occurs.


Read status register
SR4 = 1 and SR5 = 1 ?
YES
NO
Command sequence error
Execute the clear status register command (50_{16}) to clear the status register. Try performing the operation one more time after confirming that the command is entered correctly.
SR5 = 0 ?
NO
YES
Erase error
Should an erase error occur, the block in error cannot be used.
SR4 = 0 ?
NO
YES
Program error
Should a program error occur, the block in error cannot be used.
End (erase, program)

Note: When one of SR5 and SR4 is set to “1”, none of the read aray, the program, erase all blocks, and block erase commands is accepted. Execute the clear status register command (50_{16}) before executing these commands.

Fig. 93 Full status check flowchart and remedial procedure for errors

(3) Standard serial I/O Mode

The standard serial I/O mode inputs and outputs the software commands, addresses and data needed to operate (read, program, erase, etc.) the internal flash memory. This I/O is clock synchronized serial. This mode requires the exclusive external equipment (flash programmer).

The standard serial I/O mode is different from the parallel I/O mode in that the CPU controls flash memory rewrite (uses the CPU rewrite mode), rewrite data input and so forth. The standard serial I/O mode is started by connecting “H” to the $P3_6$ ($\overline{\text{CE}}$) pin and “H” to the $P8_1$ (SCLK) pin and “H” to the CNV$_{SS}$ pin (apply 4.5 V to 5.25 V to Vpp from an external source), and releasing the reset operation. (In the ordinary microcomputer mode, set CNVss pin to “L” level.)

This control program is written in the Boot ROM area when the product is shipped from Mitsubishi. Accordingly, make note of the fact that the standard serial I/O mode cannot be used if the Boot ROM area is rewritten in parallel I/O mode. Figures 96 and 97 show the pin connections for the standard serial I/O mode.

In standard serial I/O mode, serial data I/O uses the four serial I/O pins SCLK, SRXD, STXD and $\overline{\text{SRDY}}$ (BUSY). The SCLK pin is the transfer clock input pin through which an external transfer clock is input. The STXD pin is for CMOS output. The $\overline{\text{SRDY}}$ (BUSY) pin outputs “L” level when ready for reception and “H” level when reception starts.

Serial data I/O is transferred serially in 8-bit units.

In standard serial I/O mode, only the User ROM area shown in Figure 88 can be rewritten. The Boot ROM area cannot.

In standard serial I/O mode, a 7-byte ID code is used. When there is data in the flash memory, commands sent from the peripheral unit (programmer) are not accepted unless the ID code matches.

Outline Performance (Standard Serial I/O Mode)

In standard serial I/O mode, software commands, addresses and data are input and output between the MCU and peripheral units (flash programer, etc.) using 4-wire clock-synchronized serial I/O.

In reception, software commands, addresses and program data are synchronized with the rise of the transfer clock that is input to the SCLK pin, and are then input to the MCU via the SRXD pin. In transmission, the read data and status are synchronized with the fall of the transfer clock, and output from the STXD pin.

The STXD pin is for CMOS output. Transfer is in 8-bit units with LSB first.

When busy, such as during transmission, reception, erasing or program execution, the $\overline{\text{SRDY}}$ (BUSY) pin is “H” level. Accordingly, always start the next transfer after the $\overline{\text{SRDY}}$ (BUSY) pin is “L” level.

Also, data and status registers in a memory can be read after inputting software commands. Status, such as the operating state of the flash memory or whether a program or erase operation ended successfully or not, can be checked by reading the status register. Here following explains software commands, status registers, etc.

●Page Read Command

This command reads the specified page (256 bytes) in the flash memory sequentially one byte at a time. Execute the page read command as explained here following.

(1) Transfer the “FF_{16}” command code with the 1st byte.

(2) Transfer addresses AB_8 to AB_{15} and AB_{16} to AB_{23} with the 2nd and 3rd bytes respectively.

(3) From the 4th byte onward, data (DB_0 to DB_7) for the page (256 bytes) specified with addresses AB_8 to AB_{23} will be output sequentially from the smallest address first synchronized with the fall of the clock.

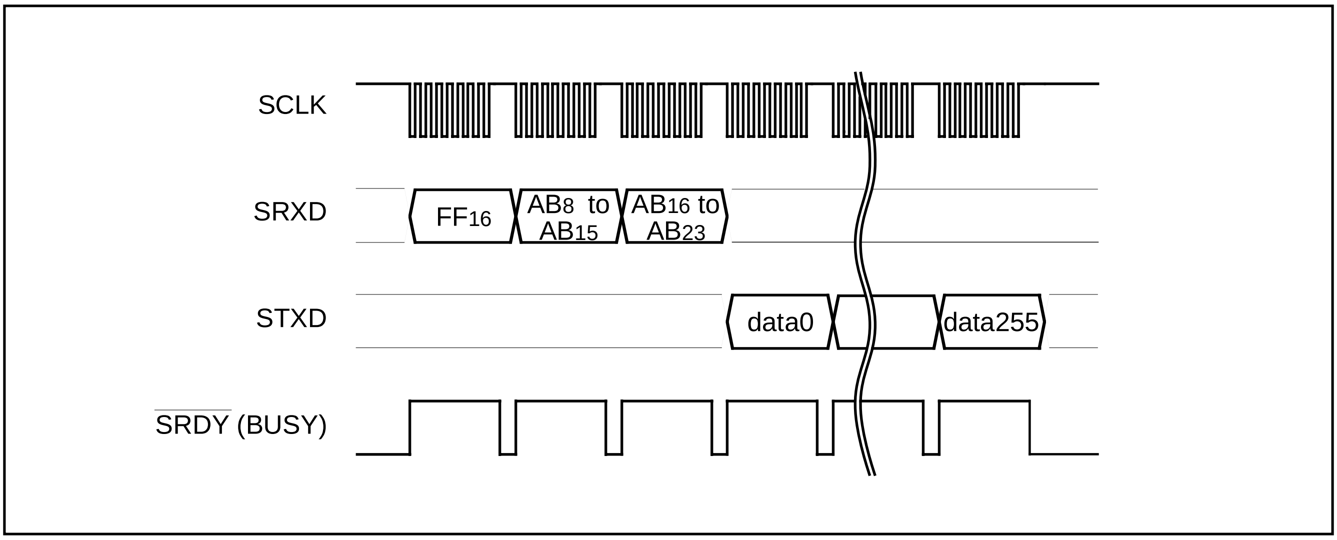


Fig. 98 Timing for page read

●Read Status Register Command

This command reads status information. When the “70_{16}” command code is transferred with the 1st byte, the contents of the status register (SRD) with the 2nd byte and the contents of status register 1 (SRD1) with the 3rd byte are read.

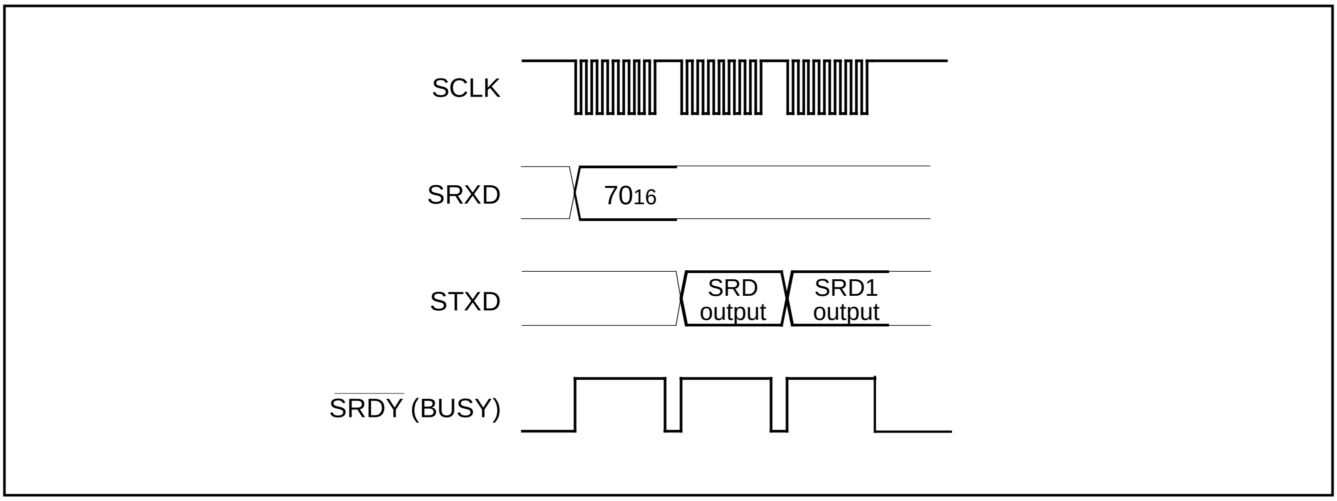


Fig. 99 Timing for reading status register