1.1 DESCRIPTION

The 7640 group, an enhanced family of CMOS 8-bit microcontrollers, offers high-speed operation, large internal-memory options, and a wide variety of standard peripherals. The series is code compatible with the 38000, 7200, 7400, and the 7500 series, and provides many performance enhancements to the instruction set.

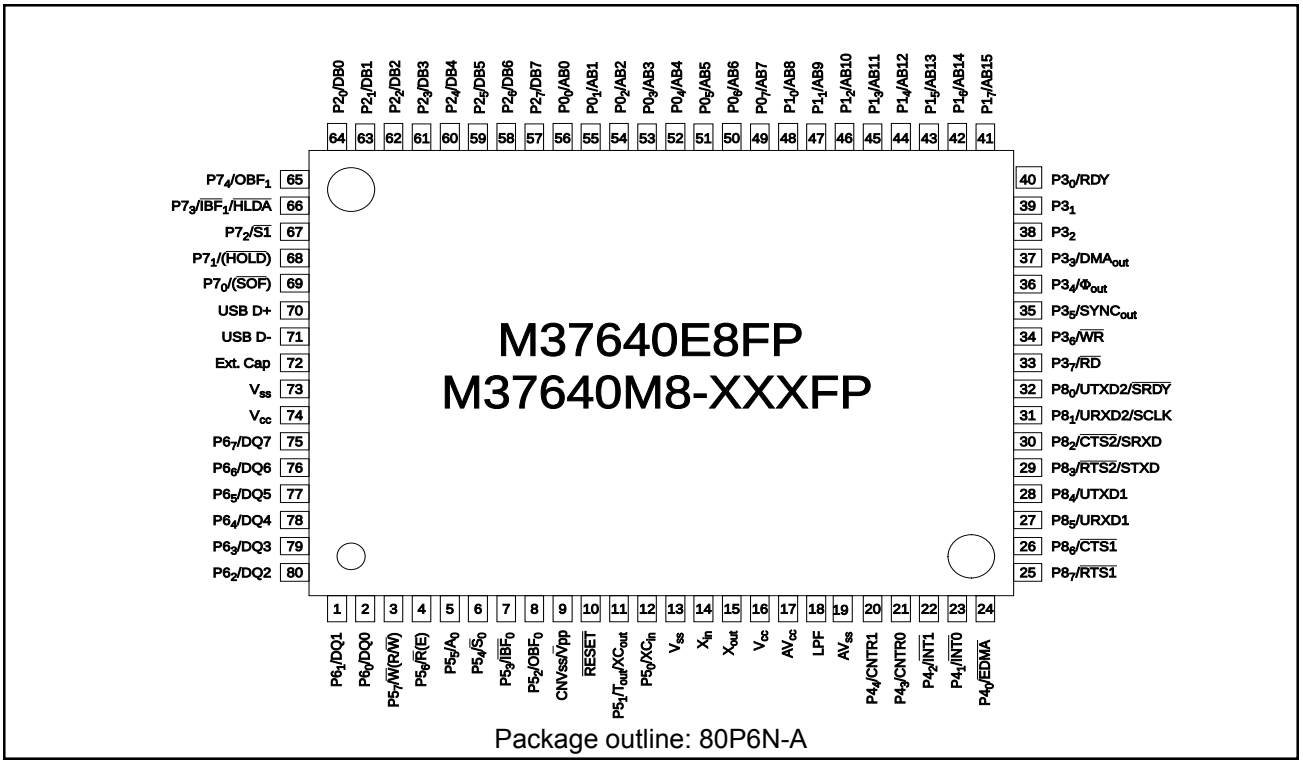
This device is a single chip PC peripheral microcontroller based on the Universal Serial Bus (USB) Version 1.1 specification. This device provides data exchange between a USB-equipped host computer and PC peripherals such as telephones, audio systems and digital cameras. See Figure 1.1 for a pin layout diagram. See Figure 1.2 for the functional block diagram.

1.2 MCU FEATURES

- Number of basic instructions 71
- Minimum instruction execution time 83ns
  (1-cycle instruction ..... Φ = 12 MHz)
- Clock frequency maximum ..... f(Xin) = 24 MHz
  ..... f(XCin) = 5 MHz
  ..... Φ = 12 MHz
- Memory size
  ROM ..... 32KB on chip
  RAM ..... 1 KB on chip
- Programmable I/O ports 66
  ..... 8 bit X 7, 5 bit X 2
- Master Bus Interface (MBI) ..... 17 signals
  ..... 8 data lines
- Serial I/O ..... 8 bit clock synchronous
- USB Function Control ..... 4 endpoints,1 control
- Interrupts ..... 4 external, 19 internal
  ..... 1 software,1 system
- DMAC ..... 2 channels, 16 address lines
  (Max. 6M byte/sec. transfer speed in burst mode)
- Timers ..... 8 bit X 3, 16 bit X 2
- Number of Full duplex UARTs available 2
- Supply voltage ..... V_{CC} = 4.15~5.25V
- Operating temperature range ..... -20 to 85°C
- Power-saving modes ..... WIT (Idle), STP (Clocks halt)

1.3 APPLICATIONS

Cameras, games, musical instruments, modems scanners, and PC peripherals.



Package outline: 80P6N-A

Fig. 1.1. Pin Layout

1.4 FUNCTIONAL BLOCK DIAGRAM

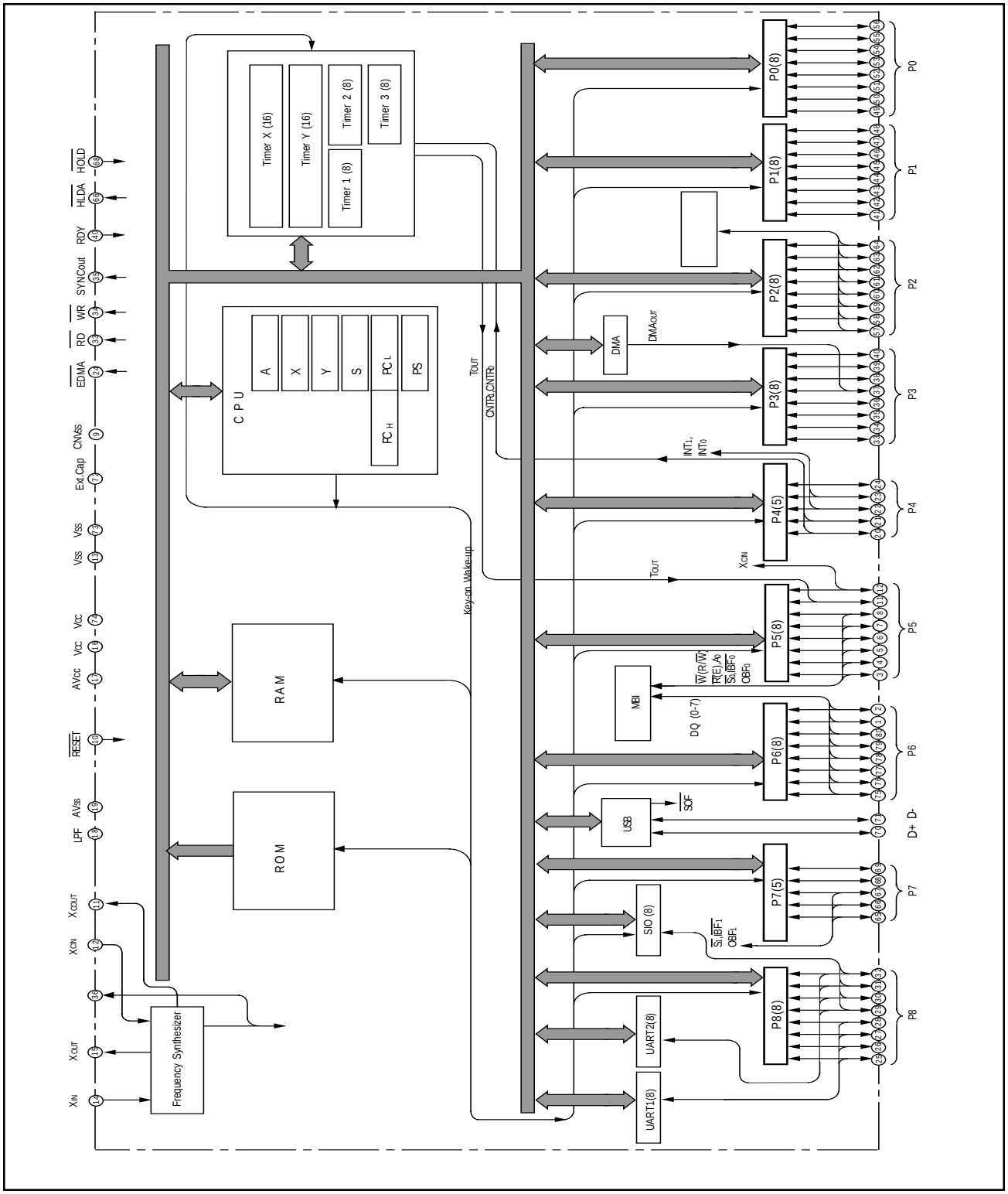


Fig. 1.2. Functional Block Diagram

1.5 PIN DESCRIPTION AND LAYOUT

Table 1.1. Pin Description and Layout

NAME	I/O	DESCRIPTION	PIN #
$P0_0$/AB0~ $P1_7$/AB15	I/O	CMOS I/O port (address bus). When the MCU is in memory expansion or microprocessor mode, these pins function as the address bus.	56-41
$P2_0$/DB0 ~ $P2_7$/DB7	I/O	CMOS I/O port (data bus). When the MCU is in memory expansion or microprocessor mode, these pins function as the data bus. These pins may also be used to implement the Key-on Wake up function.	64-57
$P3_0$/RDY	I/O	CMOS I/O port (Ready). When the MCU is in memory expansion or microprocessor mode, this pin functions as RDY (hardware wait cycle control).	40
$P3_1$	I/O	CMOS I/O port.	39
$P3_2$	I/O	CMOS I/O port.	38
$P3_3$/DMA_{out}	I/O	CMOS I/O port (DMA_{out}). When the MCU is in memory expansion or microprocessor mode, this pin is set to a "1" during a DMA transfer.	37
$P3_4$/Φ_{out}	I/O	CMOS I/O port. When the MCU is in memory expansion or microprocessor mode, this pin becomes Φ_{out} pin.	36
$P3_5$/$SYNC_{out}$	I/O	CMOS I/O port ($SYNC_{out}$). When the MCU is in memory expansion or microprocessor mode, this pin becomes the $SYNC_{out}$ pin.	35
$P3_6$/$\overline{WR}$	I/O	CMOS I/O port. ($\overline{WR}$ output). When the MCU is in memory expansion or microprocessor mode, this pin becomes $\overline{WR}$.	34
$P3_7$/$\overline{RD}$	I/O	CMOS I/O port. ($\overline{RD}$ output). When the MCU is in memory expansion or microprocessor mode, this pin becomes $\overline{RD}$.	33
$P4_0$/$\overline{EDMA}$	I/O	CMOS I/O port ($\overline{EDMA}$: Expanded Data Memory Access). When the MCU is in memory expansion or microprocessor mode, this pin can become the $\overline{EDMA}$ pin.	24
$P4_1$/$\overline{INT}$0~ $P4_2$/$\overline{INT}$1	I/O	CMOS I/O port or external interrupt ports $\overline{INT}$0 and $\overline{INT}$1. These external interrupts can be configured to be active high or low.	23-22
$P4_3$/CNTR0	I/O	CMOS I/O port or Timer X input pin for pulse width measurement mode and event counter mode or Timer X output pin for pulse output mode. This pin can also be used as an external interrupt when Timer X is not in output mode. The interrupt polarity is selected in the Timer X mode register.	21
$P4_4$/CNTR1	I/O	CMOS I/O port or Timer Y input pin for pulse period measurement mode, pulse H-L measurement mode and event counter mode or Timer Y output pin for pulse output mode. This pin can also be used as an external interrupt when Timer Y is not in output mode. The interrupt polarity is selected in the Timer Y mode register.	20
$P5_0$/XC_{in}	I/O	CMOS I/O port or XC_{in}.	12
$P5_1$/Tout/XC_{out}	I/O	CMOS I/O port or Timer half pulse output pin (can be configured initially high or initially low), or XCout.	11
$P5_2$/OBF_0	I/O	CMOS I/O port or OBF_0 output to master CPU for data bus buffer 0.	8
$P5_3$/$\overline{IBF}_0$	I/O	CMOS I/O port or $\overline{IBF}_0$ output to master CPU for data bus buffer 0.	7
$P5_4$/$\overline{S}_0$	I/O	CMOS I/O port or $\overline{S}_0$ input from master CPU for data bus buffer 0.	6
$P5_5$/A_0	I/O	CMOS I/O port or A_0 input from master CPU.	5
$P5_6$/$\overline{R}$(E)	I/O	CMOS I/O port or $\overline{R}$(E) input from master CPU.	4
$P5_7$/$\overline{W}$(R/$\overline{W}$)	I/O	CMOS I/O port or $\overline{W}$(R/$\overline{W}$) input from master CPU.	3
$P6_0$/DQ0~ $P6_7$/DQ7	I/O	CMOS I/O port or master CPU data bus.	2-1, 80-75
USB D-	I/O	USB D- voltage line interface, a series resistor of 33 Ω should be connected to this pin.	71
USB D+	I/O	USB D+ voltage line interface, a series resistor of 33 Ω should be connected to this pin.	70
$P7_0$/$\overline{SOF}$	I/O	CMOS I/O port or USB start of frame pulse output, an 80 ns pulse outputs on this pin for every USB frame.	69
$P7_1$/$\overline{HOLD}$	I/O	CMOS I/O port or $\overline{HOLD}$ pin.	68
$P7_2$/$\overline{S}_1$	I/O	CMOS I/O port or $\overline{S}_1$ input from master CPU for data bus buffer 1.	67

NAME	I/O	DESCRIPTION	PIN #
$P7_3/\overline{IBF_1}/\overline{HLDA}$	I/O	CMOS I/O port or $\overline{IBF_1}$ output to master CPU for data bus buffer 1, or $\overline{HLDA}$ pin. $\overline{IBF_1}$ and $\overline{HLDA}$ are mutually exclusive. $\overline{IBF_1}$ has priority over $\overline{HLDA}$	66
$P7_4/OBF_1$	I/O	CMOS I/O port or OBF_1 output to master CPU for data bus buffer 1.	65
$P8_0$/UTXD2/$\overline{SRDY}$	I/O	CMOS I/O port or UART2 pin UTXD2 or SIO pin $\overline{SRDY}$. UART2 and SIO are mutually exclusive, UART2 has priority over SIO.	32
$P8_1$/URXD2/SCLK	I/O	CMOS I/O port or UART2 pin URXD2 or SIO pin SCLK. UART2 and SIO are mutually exclusive, UART2 has priority over SIO.	31
$P8_2/\overline{CTS}2$/SRXD	I/O	CMOS I/O port or UART2 pin $\overline{CTS}2$ or SIO pin SRXD. UART2 and SIO are mutually exclusive, UART2 has priority over SIO.	30
$P8_3/\overline{RTS}2$/STXD	I/O	CMOS I/O port or UART2 pin $\overline{RTS}2$ or SIO pin STXD. UART2 and SIO are mutually exclusive, UART2 has priority over SIO.	29
$P8_4$/UTXD1	I/O	CMOS I/O port or UART1 pin UTXD1.	28
$P8_5$/URXD1	I/O	CMOS I/O port or UART1 pin URXD1.	27
$P8_6/\overline{CTS}1$	I/O	CMOS I/O port or UART1 pin $\overline{CTS}1$.	26
$P8_7/\overline{RTS}1$	I/O	CMOS I/O port or UART1 pin $\overline{RTS}1$.	25
AV_{CC}, AV_{SS}	I	Power supply inputs for analog circuitry AVcc = 4.15~ 5.25V, AVss = 0V	17,19
$CNV_{SS}/\overline{V}_{PP}$	I	Controls the processor mode of the chip. Normally connected to V_{SS} or V_{CC}. When the MCU is in EPROM program mode, this pin supplies the programming voltage to the EPROM.	9
V_{CC},V_{SS}	I	Power supply inputs: V_{CC} = 4.15~ 5.25V, V_{SS} = 0V	16/74 13/73
$\overline{RESET}$	I	To enter the reset state, this pin must be kept 'L' for more that 2μs (20 Φ cycles under normal V_{CC} conditions). If the crystal or ceramic resonator requires more time to stabilize, extend this 'L' level time appropriately.	10
XC_{in}	I	An external ceramic or quartz crystal oscillator can be connected between the XC_{in} and XC_{out} pins. If an external clock source is used, connect the clock source to the XC_{in} pin and leave the XC_{out} pin open.	12
XC_{out}	O		11
X_{in}	I	Input and output signals to and from the internal clock generation circuit. Connect a ceramic resonator or quartz crystal between X_{in} and X_{out} pins to set the oscillation frequency. If an external clock is used, connect the clock source to the X_{in} pin and leave the X_{out} pin open.	14
X_{out}	O		15
LPF	O	Loop filter for the frequency synthesizer.	18
Ext. Cap	I	An external capacitor (Ext. Cap) pin. When the USB transceiver voltage converter is used, a 2μf or larger capacitor should connect between this pin and Vss to ensure proper operation of the USB line driver. The voltage converter is enabled by setting bit 4 of the USB control register (0013_{16}) to a “1”.	72

1.6 PART NUMBERING

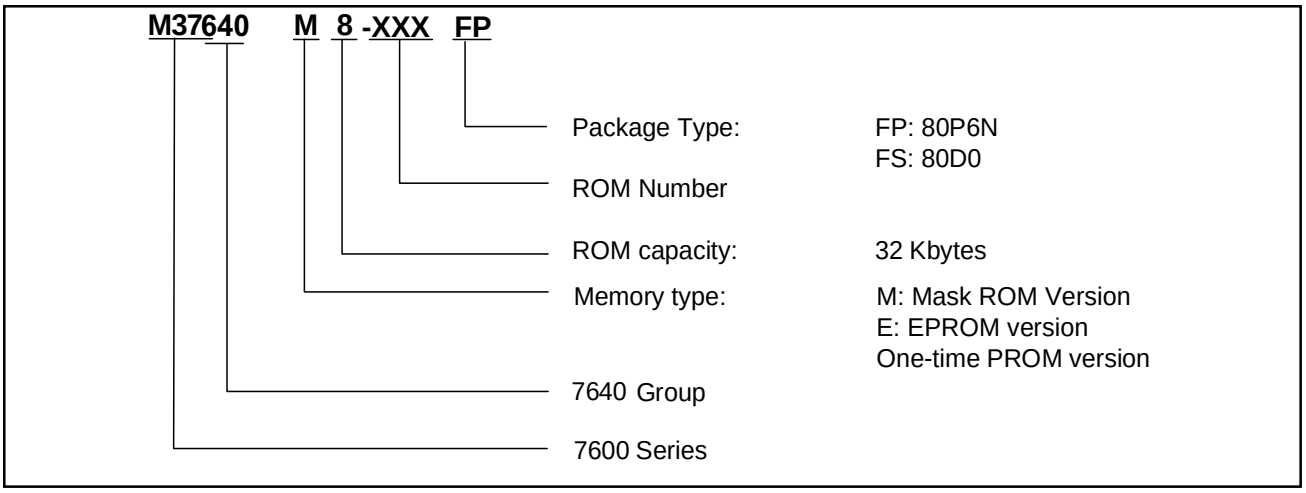


Fig. 1.3. Type no., memory size, and package

1.7 ROM EXPANSION

Table 1.2. ROM Expansion

ROM Size (Bytes)			
32K	M37640M8-XXXFP	M37640E8FP	M37640E8FS
	Mask ROM version	One-time PROM version	EPROM version

1.8 CURRENTLY SUPPORTED PRODUCTS

Table 1.3. Currently Supported Products

Type No.	ROM capacity	RAM capacity	Package type	Remarks
M37640M8-XXXFP	32K bytes	1 K bytes	80P6N-A	Mask ROM version
M37640E8FP	32K bytes	1 K bytes	80P6N-A	One-time PROM version
M37640E8FS	32K bytes	1 K bytes	80D0	EPROM version

1.9 CENTRAL PROCESSING UNIT

The central processing unit (CPU) has six registers:

- Accumulator (A)
- Index Register X (X)
- Index Register Y (Y)
- Stack Pointer (S)
- Processor Status Register (PS)
- Program Counter (PC)

1.9.1 Register Structure

Five of the CPU registers are 8-bit registers. These are the Accumulator (A), Index register X (X), Index Register Y (Y), Stack pointer (S), and the Processor Status register (PS) as shown in Figure 1.4.

The Program counter (PC) is a 16-bit register consisting of two 8-bit registers (PCH and PCL).

After a hardware reset, bit 2 (I Flag) of the PS is set high and the values at the address FFFA16 and FFFB16 are stored in the PC, but the values of the other bits of the PS and other registers are undefined. Initialization of the undefined registers may be necessary for some programs.

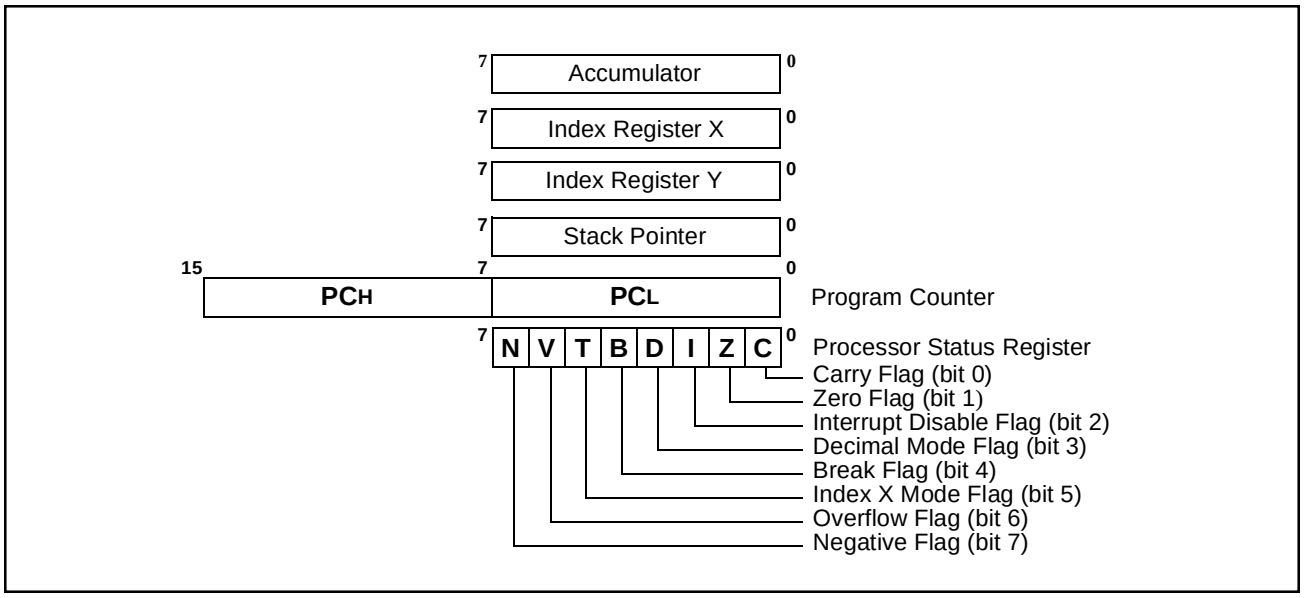


Fig. 1.4. Register Structure

1.10 CPU MODE REGISTERS

This device has two CPU mode registers:

- CPU Mode Register A (CPMA)
- CPU Mode Register B (CPMB)

These registers control the processor mode, clock, slow memory wait and other CPU functions. The bit representation of each register is described in Figure 1.5 and Figure 1.6.

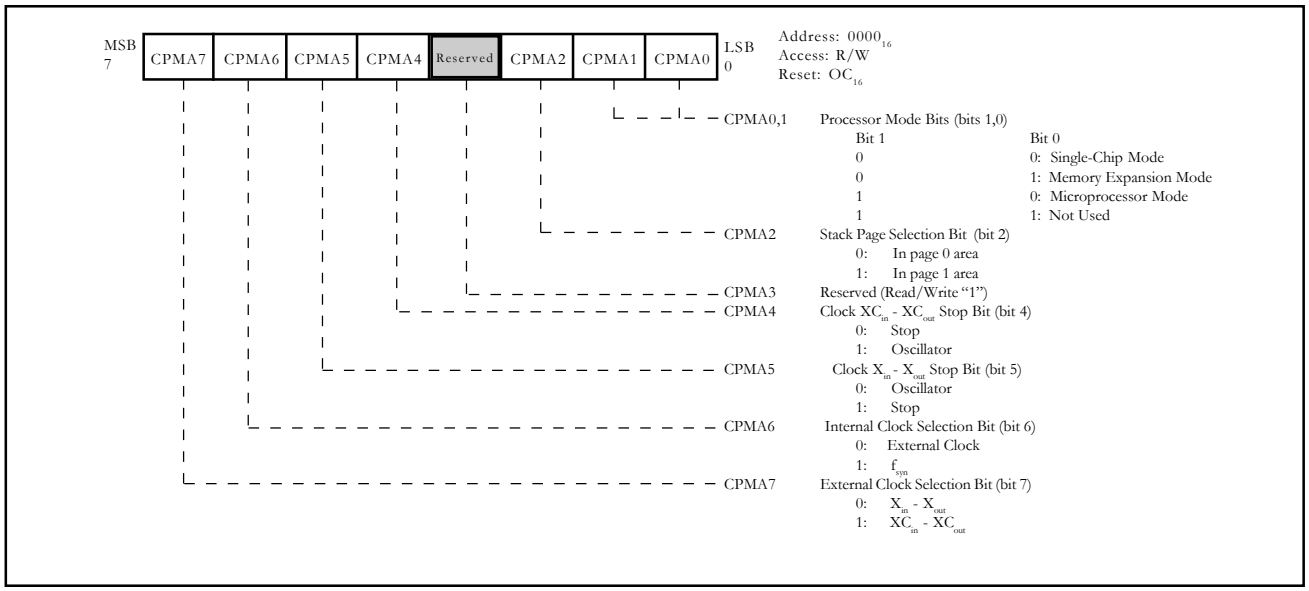


Fig. 1.5. CPU Mode Register A (CPMA)

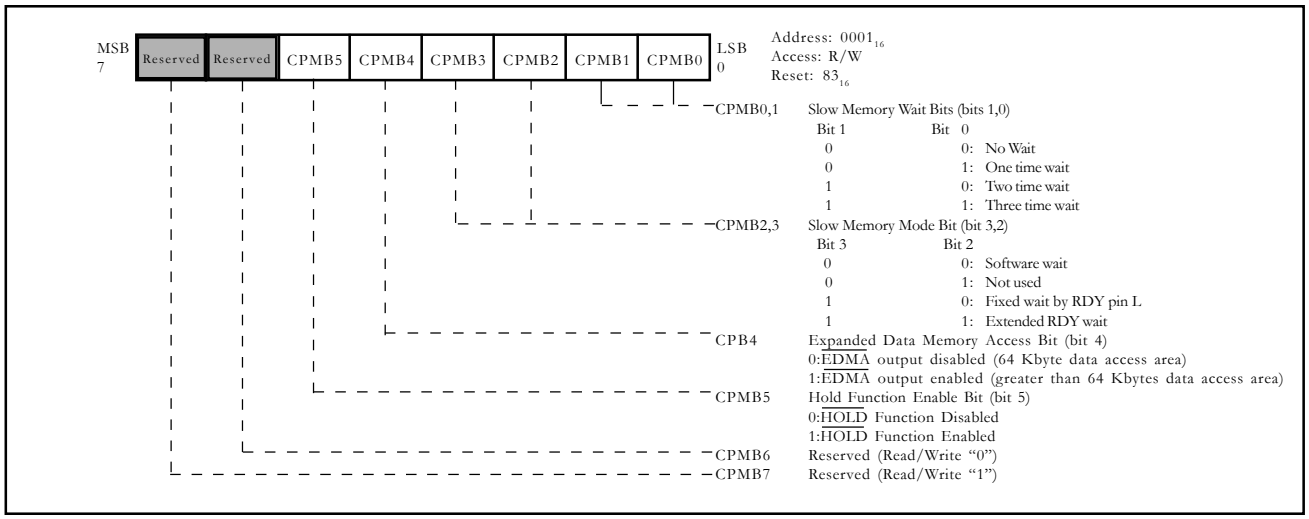


Fig.1.6. CPU Mode Register B (CPMB)

1.11 MEMORY MAP

The first 112 bytes of memory from 0000_{16} to $006F_{16}$ is the special function register (SFR) area and contains the CPU mode registers, interrupt registers, and other registers to control peripheral functions (see Figure 1.7).

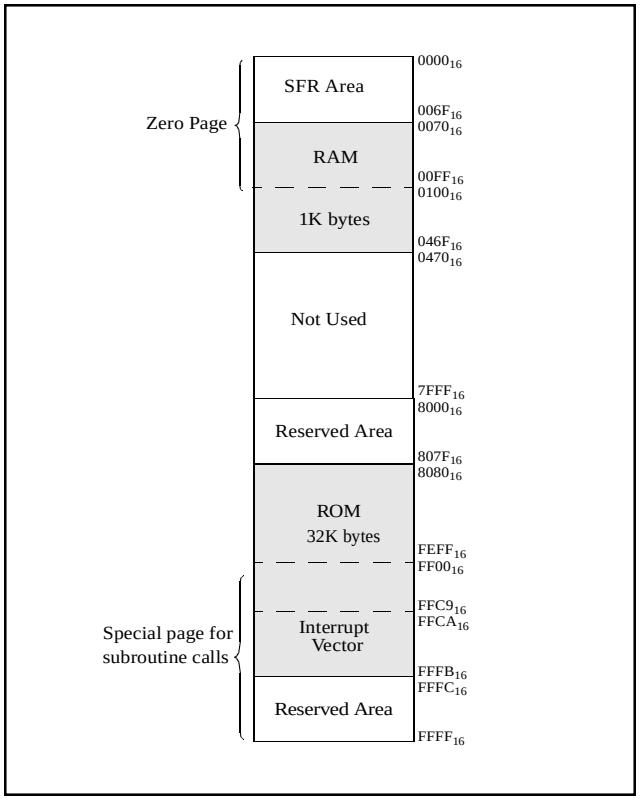


Fig. 1.7. Memory Map

The general purpose RAM resides from 0070_{16} to $046F_{16}$. When the MCU is in memory expansion or microprocessor mode and external memory is overlaid on the internal RAM, the CPU reads data from the internal RAM. However, the CPU writes data in both the internal and external memory. The area from 0470_{16} to $7FFF_{16}$ is not used in single-chip mode, but can be mapped for an external memory device when the MCU is in memory expansion or microprocessor mode.

The area from 8000_{16} to $807F_{16}$ and from $FFFC_{16}$ to $FFFF_{16}$ are factory reserved areas. Mitsubishi uses it for test and evaluation purposes. The user can not use this area in single-chip or memory expansion modes.

The user 32K byte ROM resides from 8080_{16} to $FFFB_{16}$. When the MCU is in microprocessor mode, the CPU accesses an external area rather than accessing the internal ROM.

1.11.1 Special page

The 256 bytes from address $FF00_{16}$ to $FFFF_{16}$ are called the special page area. In this area special page addressing can be used to specify memory addresses. This dedicated special page addressing mode enables access to this area with fewer instruction cycles. Frequently used subroutines are normally stored in this area.

1.12 PROCESSOR MODES

The operation modes are described below. The memory maps for the first three modes are shown in Figure 1.8. Single chip mode is normally entered after reset. However, if the MCU has a CNVss pin, holding this pin high will cause microprocessor mode to be entered after reset. After the reset sequence has completed, the mode can be changed with software by modifying the value of bits 0 and 1 of CPMA. However, while CNVss is high, bit 1 of CPMA is “1” and cannot be changed.

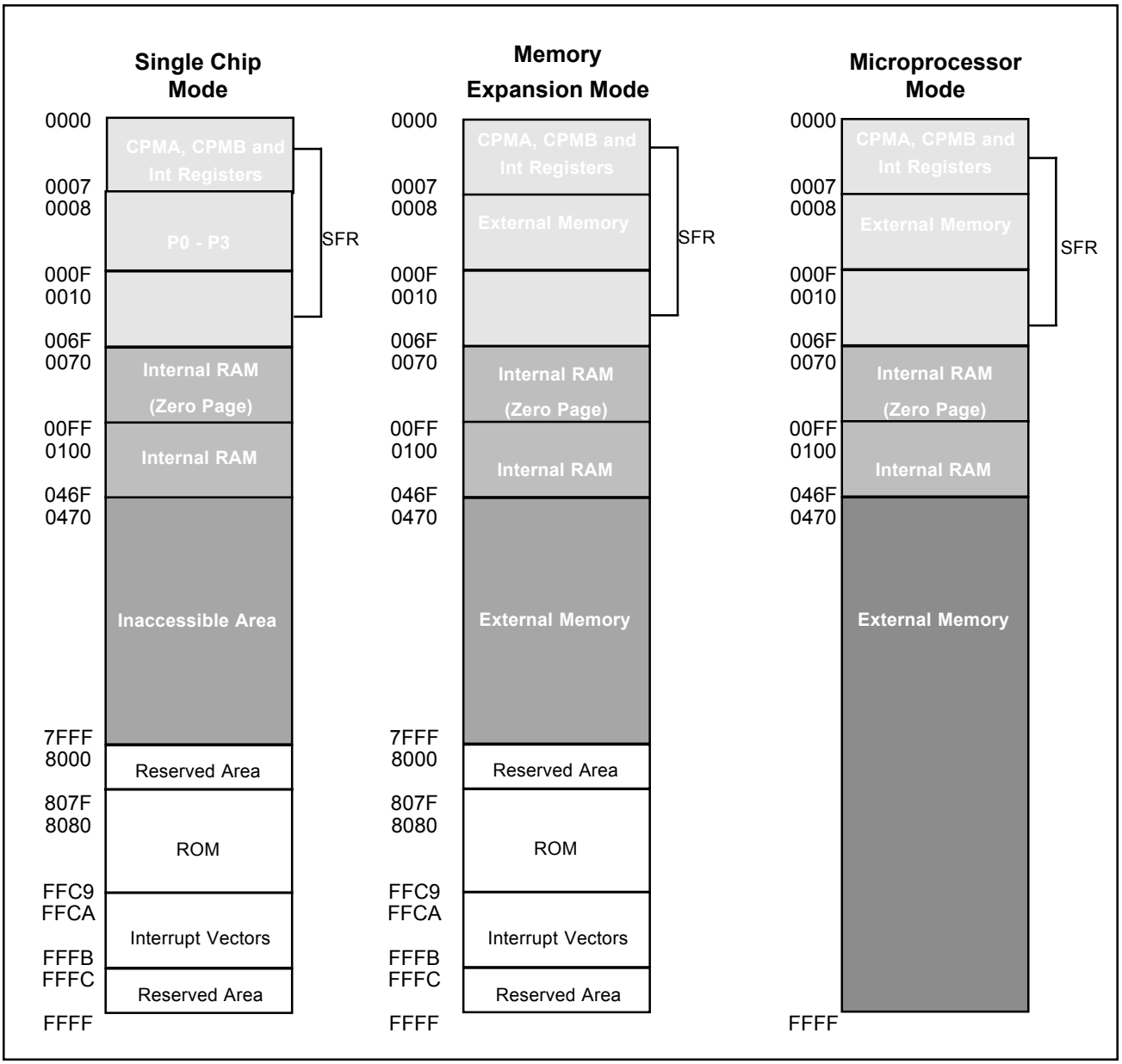


Fig. 1.8. Operation Modes Memory Maps

1.12.1 Single Chip

In this mode, all ports take on their primary function and all internal memory is accessible. Those areas that are not in internal memory are not accessible. Also, slow memory wait and EDMA are disabled in this mode.

1.12.2 Memory Expansion

In this mode, Ports 0 and 1 output the address bus (AB0-AB15), port 2 acts as the data bus input and output, and port 3 bits 7 to 3 output RD, WR, $SYNC_{out}$, Φ_{out}, and DMA_{out}, respectively. All memory areas that are not internal memory or SFR area are accessed externally. Because ports 0 to 3 lose their normal function in this mode, the address area for the ports and their direction registers are treated as external memory. In this mode, slow memory wait and EDMA can be enabled.

1.12.3 Microprocessor

This mode is primarily the same as memory expansion mode. The difference is that the internal ROM/EPROM area can not be accessed and is instead treated as external memory. Slow memory wait and EDMA can be enabled in this mode.

1.12.4 Slow Memory Wait

The wait function is used when interfacing with external memories that are too slow to operate at the normal read/write speed of the MCU. When this is the case, a wait can be used to extend the read/write cycle. Three different wait modes are supported; software wait, RDY wait, and extended RDY wait. The appropriate mode is chosen by the setting of bits 0 to 3 of CPMB. The wait function is disabled for internal memory and is valid only for memory expansion and microprocessor modes.

Software wait is used to extend the read/write cycle by one, two, or three cycles of Φ. The cycle number is determined by the value of bits 0 and 1 of CPMB. When software wait is selected, the value on the RDY pin is ignored. The timing for software wait is shown in Figure 1.9.

RDY wait is also used to extend the read/write cycle by one, two, or three cycles Φ. In this case, the read/write cycle is extended if the RDY pin is low when Φ_{out} goes low (taking into account setup and hold times) at the beginning of the read/write cycle. The extension time is fixed by the value of bits 0 and 1 of CPMB and does not depend on the state of the RDY pin once the read/write cycle has begun. If the RDY pin is high when Φ_{out} goes low at the beginning of the read/write cycle, the read/write cycle is not extended. The timing for RDY wait is shown in Figure 1.10.

The extended RDY wait mode is used to extend the read/write cycle by a variable number of cycles of Φ. The exact number is dependent on the state of the RDY pin and the value of bits 0 and 1 of CPMB. In this mode, the read/write cycle is extended if the RDY pin is low when Φ_{out} goes low at the beginning of the read/write cycle. The read/write cycle continues to be extended until the RDY pin is high when Φ_{out} goes low, at which point the read/write cycle completes in one, two, or three cycles of Φ (with respect to the previous low to high transition of Φ), dependent on the value in bits 0 and 1 of CPMB. If the RDY pin is high when Φ_{out} goes low at the beginning of the read/write cycle, the read/write cycle is not extended. The timing for this mode is shown in Figure 1.11.

The wait function can only be enabled for external memory access in microprocessor or memory expansion modes. However, the wait function can not be enabled for accesses to addresses 0008_{16} to $000F_{16}$ (Port 0 through Port 3 registers) in these modes, even though the locations are mapped as external memory.

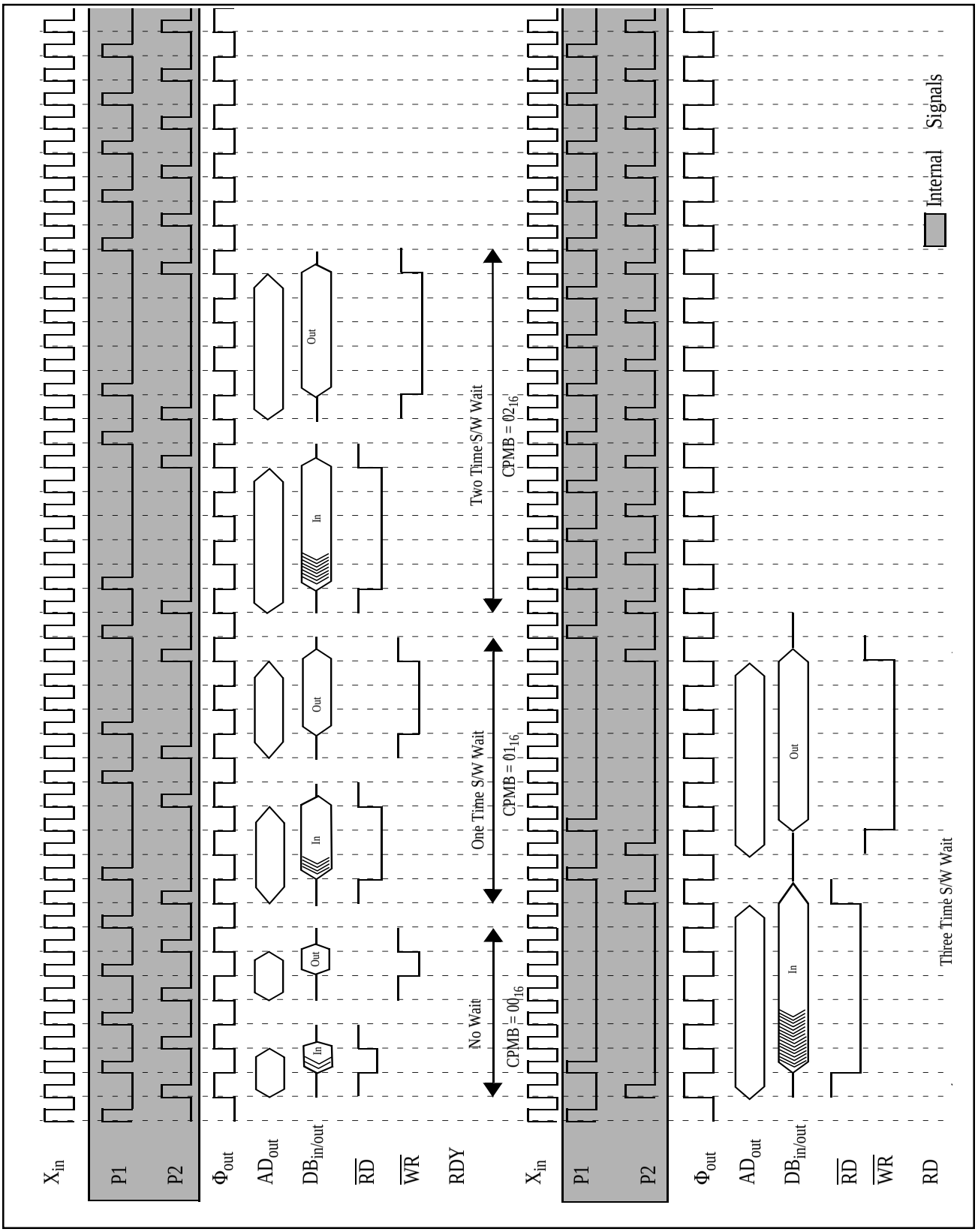


Fig. 1.9. Software Wait Timing Diagram

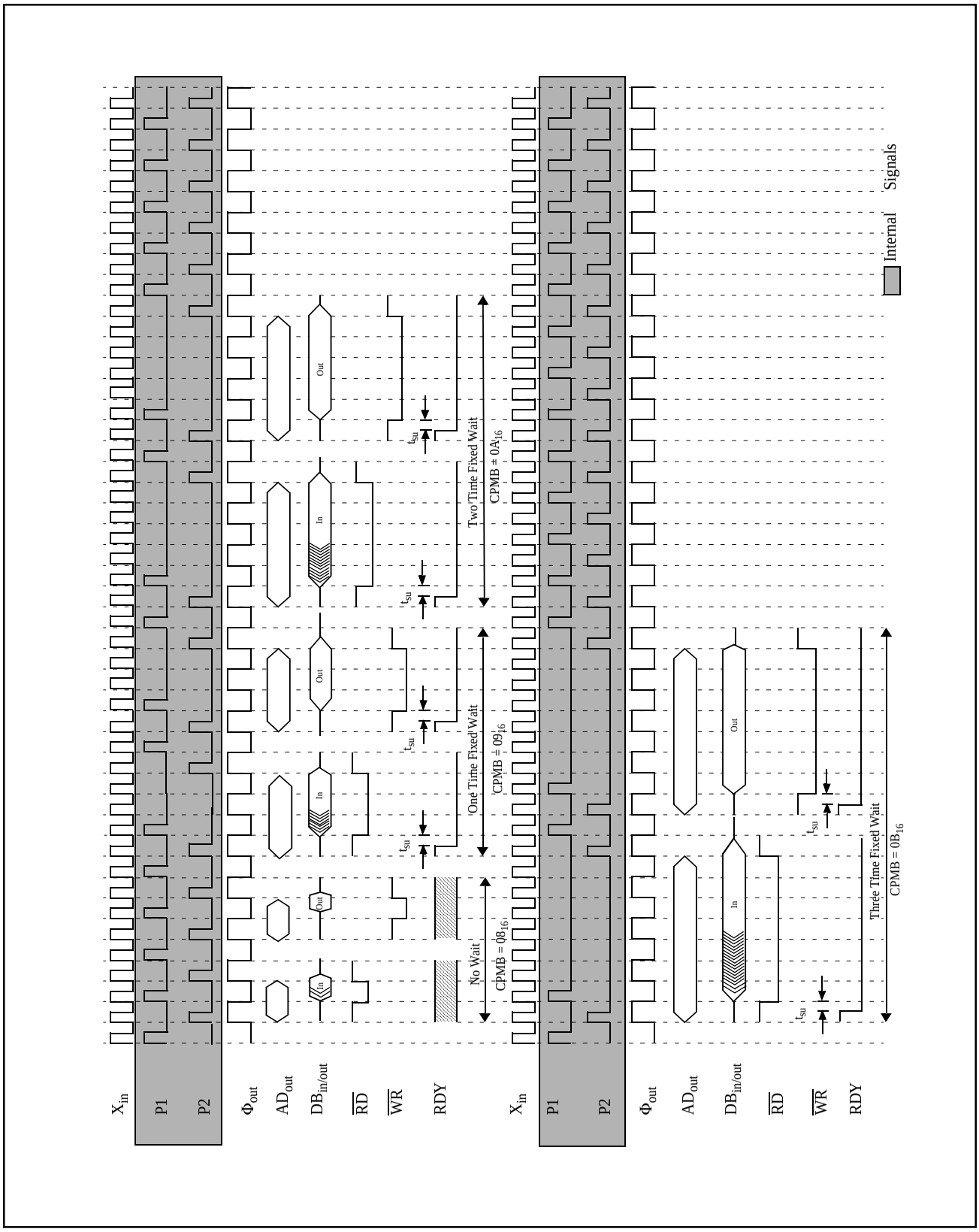


Fig. 1.10. RDY Wait Timing Diagram

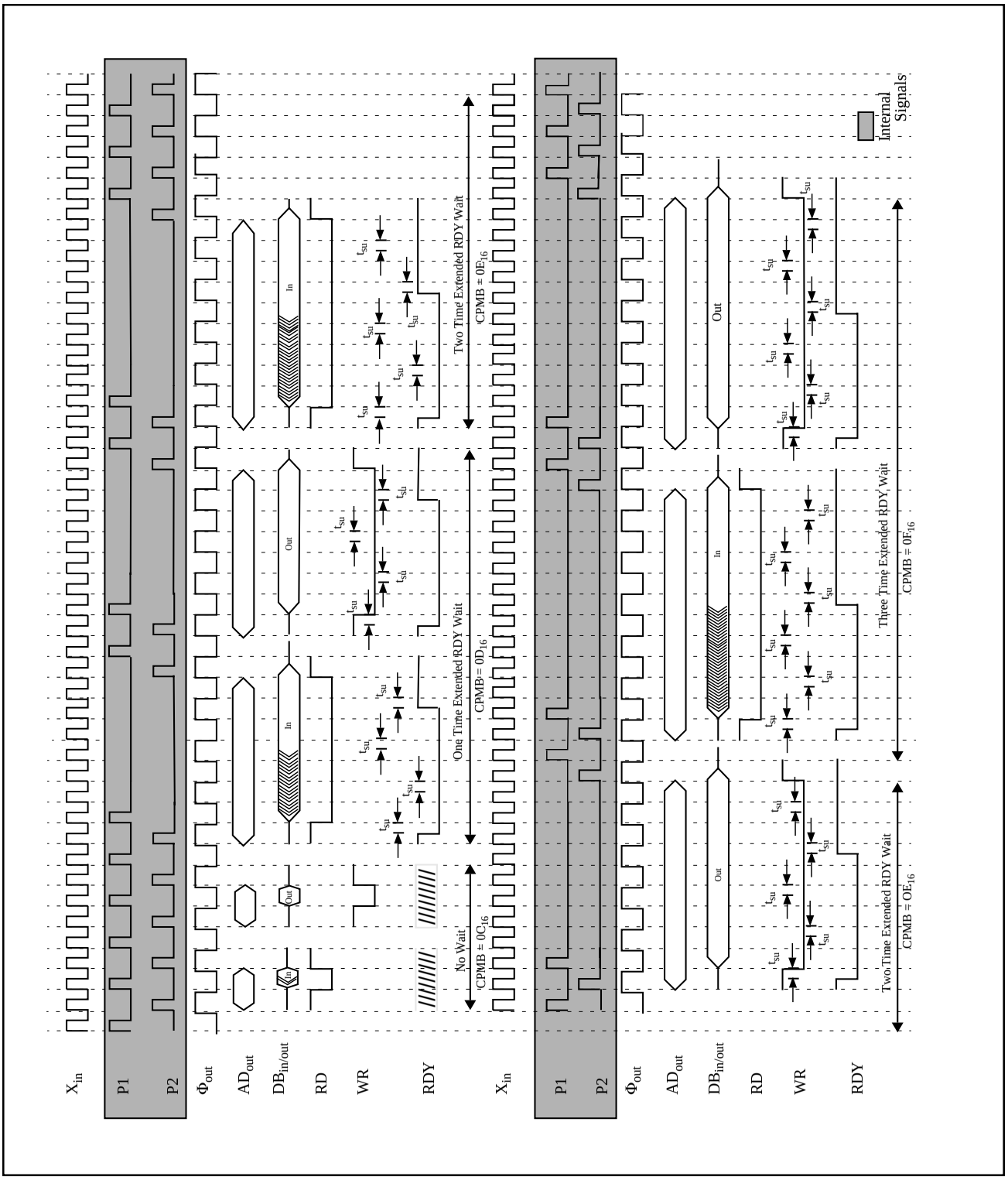


Fig. 1.11. Extended RDY Wait Timing Diagram

1.12.5 Hold Function

The hold function is used when the MCU is put in a system where more than one device will need control of the external address and data buses. Two signals are used to implement this function, $\overline{HOLD}$ ($P7_1$) and $\overline{HLDA}$ ($P7_3$). $\overline{HOLD}$ is an input to the MCU and is brought low when an external device wants the MCU to relinquish the address and data buses. $\overline{HLDA}$ is an output from the MCU that signals when the MCU has relinquished the buses. When this is the case, the MCU tri-state ports 0 and 1 (address bus) and port 2 (data bus), and holds port $P3_7$ ($\overline{RD}$) and port $P3_6$ ($\overline{WR}$) high. Ports $P3_7$ and $P3_6$ are held high to prevent any external device that is enabled by $\overline{RD}$ or $\overline{WR}$ from being falsely activated. The clocks to the CPU are stopped, but the peripheral clocks and port $P3_4$ (Φ_{out}) continue to oscillate. $\overline{HOLD}$ is brought high to allow the MCU to regain the address and data buses. When this occurs, $\overline{HLDA}$ will go high and ports $P1_1$, $P2_2$, $P3_7$ and $P3_6$ will begin to drive the external buses again. The timing for the hold function is shown in Figure 1.12. The hold function is only valid for memory expansion and microprocessor modes. Bit 5 of CPMB is used to enable the hold function. $\overline{HLDA}$ will loose its function when the $\overline{IBF_1}$ pin functionality is used.

1.12.6 Expanded Data Memory Access

The Expanded Data Memory Access ($\overline{EDMA}$) mode feature allows the user to access a greater than 64 Kbyte data area for instructions LDA (IndY) with T="0" and T="1", and STA (IndY). Bit 4 of CPMB is used to enable/disable the $\overline{EDMA}$ function. If bit 4 of CPMB equals "1", then during the data read/write cycle of instructions LDA (IndY) and STA (IndY) Port 4_0 (EDMA) is driven low. The EDMA signal output can be used by an external decoder to indicate when the read/write is to a different 64 Kbyte bank. The actual determination of which bank to access can be done by using a few bits of a port to represent the extended addresses above AB15. For example, if four banks are accessed, then two bits are needed to uniquely identify each bank. Two port bits can be used for this, one representing AB16 and the other AB17. The instruction sequences for STA (IndY) and LDA (IndY) are shown in Figure 1.13 and Figure 1.14.

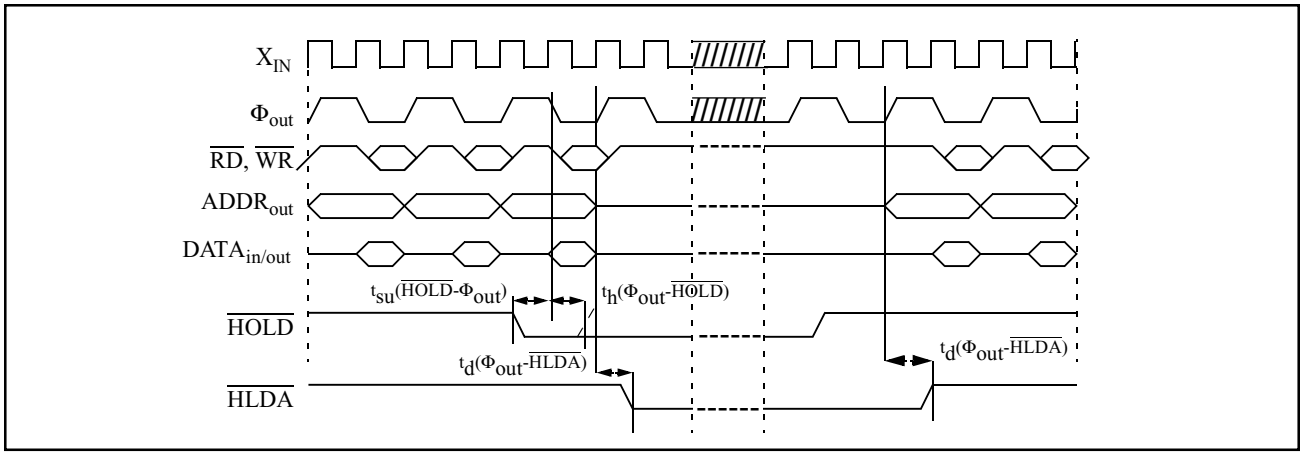


Fig. 1.12. Hold Function Timing Diagram

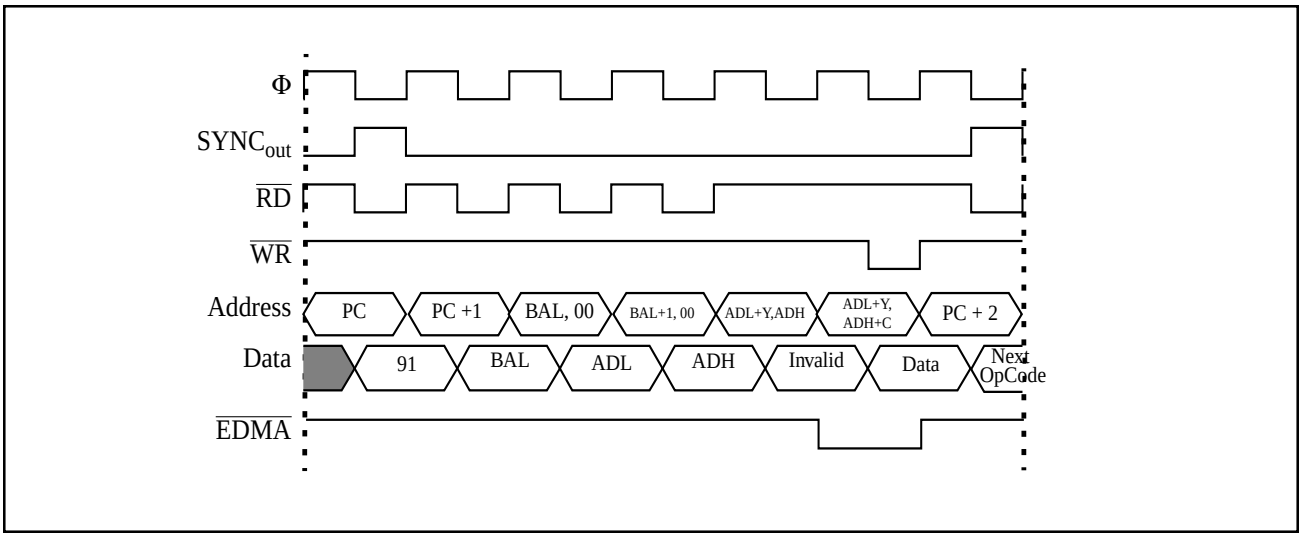


Fig. 1.13. Instruction sequences for STA ($zz) IndY with $\overline{\text{EDMA}}$ Enable

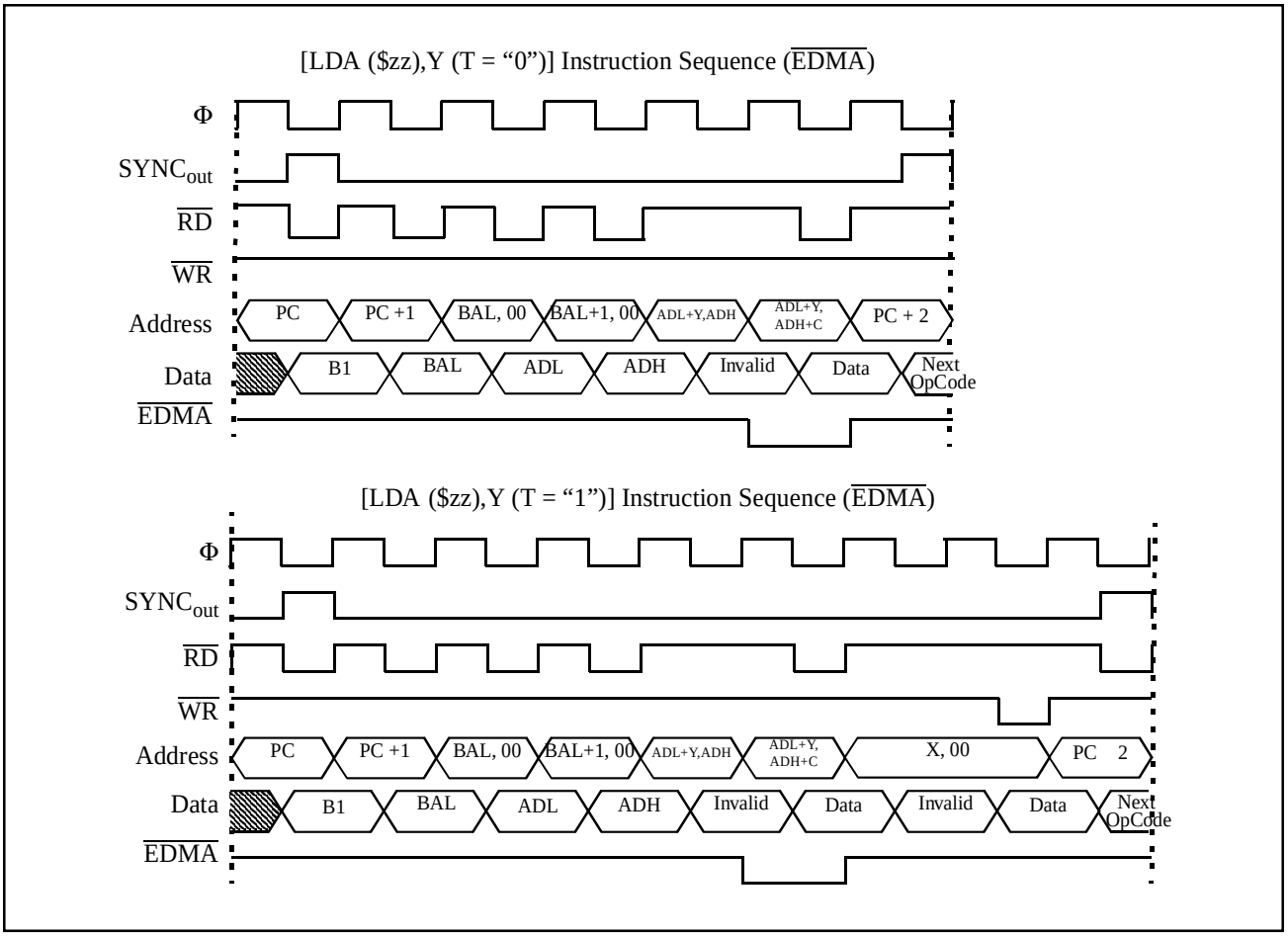


Fig. 1.14. Instruction sequences for LDA ($zz) IndY with $\overline{\text{EDMA}}$ Enable

1.13 SPECIAL FUNCTION REGISTERS

The special function registers (SFR) are used for controlling the functional blocks, such as I/O ports, Timers, UART, and so forth (see Table 1.4). The reserved addresses should not be read or written to.

Table 1.4. SFR Addresses

Addr	Description	Acronym and Value at Reset		Addr	Description	Acronym and Value at Reset
0000_{16}	CPU Mode Register A	CPMA=0C		0038_{16}	UART2 Mode Register	U2MOD=00
0001_{16}	CPU Mode Register B	CPMB=83		0039_{16}	UART2 Baud Rate Generator	U2BRG=XX
0002_{16}	Interrupt Request Register A	IREQA=00		$003A_{16}$	UART2 Status Register	U2STS=03
0003_{16}	Interrupt Request Register B	IREQB=00		$003B_{16}$	UART2 Control Register	U2CON=00
0004_{16}	Interrupt Request Register C	IREQC=00		$003C_{16}$	UART2 Transmit/Receiver Buffer 1	U2TRB1=XX
0005_{16}	Interrupt Control Register A	ICONA=00		$003D_{16}$	UART2 Transmit/Receiver Buffer 2	U2TRB2=XX
0006_{16}	Interrupt Control Register B	ICONB=00		$003E_{16}$	UART2 RTS Control Register	U2RTSC=80
0007_{16}	Interrupt Control Register C	ICONC=00		$003F_{16}$	DMAC Index and Status Register	DMAIS=00
0008_{16}	Port P0	P0=00		0040_{16}	DMAC Channel x Mode Register 1	DMAxM1=00
0009_{16}	Port P0 Direction Register	P0D=00		0041_{16}	DMAC Channel x Mode Register 2	DMAxM2=00
$000A_{16}$	Port P1	P1=00		0042_{16}	DMAC Channel x Source Register Low	DMAxSL=00
$000B_{16}$	Port P1 Direction Register	P1D=00		0043_{16}	DMAC Channel x Source Register High	DMAxSH=00
$000C_{16}$	Port P2	P2=00		0044_{16}	DMAC Channel x Destination Register Low	DMAxDL=00
$000D_{16}$	Port P2 Direction Register	P2D=00		0045_{16}	DMAC Channel x Destination Register High	DMAxDH=00
$000E_{16}$	Port P3	P3=00		0046_{16}	DMAC Channel x Count Register Low	DMAxCL=00
$000F_{16}$	Port P3 Direction Register	P3D=00		0047_{16}	DMAC Channel x Count Register High	DMAxCH=00
0010_{16}	Port Control Register	PTC=00		0048_{16}	Data Bus Buffer register 0	DBB0=00
0011_{16}	Interrupt Polarity Selection Register	IPOL=00		0049_{16}	Data Bus Buffer status register 0	DBBS0=00
0012_{16}	Port P2 pull-up Control Register	PUP2=00		$004A_{16}$	Data Bus Buffer Control Register 0	DBBC0=00
0013_{16}	USB Control Register	USBC=00		$004B_{16}$	Reserved	
0014_{16}	Port P6	P6=00		$004C_{16}$	Data Bus Buffer register 1	DBB1=00
0015_{16}	Port P6 Direction Register	P6D=00		$004D_{16}$	Data Bus Buffer Status Register 1	DBBS1=00
0016_{16}	Port P5	P5=00		$004E_{16}$	Data Bus Buffer Control Register 1	DBBC1=00
0017_{16}	Port P5 Direction Register	P5D=00		$004F_{16}$	Reserved	
0018_{16}	Port P4	P4=00		0050_{16}	USB Address Register	USBA=00
0019_{16}	Port P4 Direction Register	P4D=00		0051_{16}	USB Power Management Register	USBPM=00
$001A_{16}$	Port P7	P7=00		0052_{16}	USB Interrupt Status Register 1	USBIS1=00
$001B_{16}$	Port P7 Direction Register	P7D=00		0053_{16}	USB Interrupt Status Register 2	USBIS2=00
$001C_{16}$	Port P8	P8=00		0054_{16}	USB Interrupt Enable Register 1	USBIE1=FF
$001D_{16}$	Port P8 Direction Register	P8D=00		0055_{16}	USB Interrupt Enable Register 2	USBIE2=33
$001E_{16}$	Reserved			0056_{16}	USB Frame Number Register Low	USBSOFL=00
$001F_{16}$	Clock Control Register	CCR=00		0057_{16}	USB Frame Number Register High	USBSOFH=00
0020_{16}	Timer XL	TXL=FF		0058_{16}	USB Endpoint Index	USBINDEX=00
0021_{16}	Timer XH	TXH=FF		0059_{16}	USB Endpoint x IN CSR	IN_CSR=00
0022_{16}	Timer YL	TYL=FF		$005A_{16}$	USB Endpoint x OUT CSR	OUT_CSR=00
0023_{16}	Timer YH	TYH=FF		$005B_{16}$	USB Endpoint x IN MAXP	IN_MAXP (endpoint dependent)
0024_{16}	Timer 1	T1=FF		$005C_{16}$	USB Endpoint x OUT MAXP	OUT_MAXP (endpoint dependent)
0025_{16}	Timer 2	T2=01		$005D_{16}$	USB Endpoint x OUT WRT_CNT Low	WRT_CNTL=00
0026_{16}	Timer 3	T3=FF		$005E_{16}$	USB Endpoint x OUT WRT_CNT High	WRT_CNTH=00
0027_{16}	Timer X Mode Register	TXM=00		$005F_{16}$	Reserved	
0028_{16}	Timer Y Mode Register	TYM=00		0060_{16}	USB Endpoint 0 FIFO	USBFIFO0=N/A
0029_{16}	Timer 123 Mode Register	T123M=00		0061_{16}	USB Endpoint 1 FIFO	USBFIFO1=N/A
$002A_{16}$	SIO Shift Register	SIOSHT=XX		0062_{16}	USB Endpoint 2 FIFO	USBFIFO2=N/A
$002B_{16}$	SIO Control Register 1	SIOCON1=40		0063_{16}	USB Endpoint 3 FIFO	USBFIFO3=N/A
$002C_{16}$	SIO Control Register 2	SIOCON2=18		0064_{16}	USB Endpoint 4 FIFO	USBFIFO4=N/A
$002D_{16}$	Special Count Source Generator1	SCSG1=FF		0065_{16}	Reserved	
$002E_{16}$	Special Count Source Generator2	SCSG2=FF		0066_{16}	Reserved	
$002F_{16}$	Special Count Source Mode Register	SCSM=00		0067_{16}	Reserved	
0030_{16}	UART1 Mode Register	U1MOD=00		0068_{16}	Reserved	
0031_{16}	UART1 Baud Rate Generator	U1BRG=XX		0069_{16}	Reserved	
0032_{16}	UART1 Status Register	U1STS=03		$006A_{16}$	Reserved	
0033_{16}	UART1 Control Register	U1CON=00		$006B_{16}$	Reserved	
0034_{16}	UART1 Transmit/Receiver Buffer 1	U1TRB1=XX		$006C_{16}$	Freq Synthesizer Control	FSC=60
0035_{16}	UART1 Transmit/Receiver Buffer 2	U1TRB2=XX		$006D_{16}$	Freq Synthesizer Multiply Register 1	FSM1=FF
0036_{16}	UART1 RTS Control Register	U1RTSC=80		$006E_{16}$	Freq Synthesizer Multiply Register 2	FSM2=FF
0037_{16}	Reserved			$006F_{16}$	Freq Synthesizer Divide Register	FSD=FF

1.14 INPUT AND OUTPUT PORTS

Table 1.5. Input and Output Ports

Pin	Name	Input/ Output	I/O Format	Non-Port Function	Related SFR's	Ref. Number
$P0_0$ – $P0_7$	Port 0	Input/output, individual bits	CMOS I/O port	Address Bus	CPU Mode Register	Fig. 1.15
$P1_0$ – $P1_7$	Port 1	Input/output, individual bits	CMOS I/O port	Address Bus	CPU Mode Register	Fig. 1.15
$P2_0$ – $P2_7$	Port 2	Input/output, individual bits	CMOS I/O port	Data Bus	CPU Mode Register	Fig. 1.15
$P3_0$ – $P3_7$	Port 3	Input/output, individual bits	CMOS I/O port	Control signal I/O	CPU Mode Register	Fig. 1.15
$P4_0$	Port 4	Input/output, individual bits	CMOS I/O port	Expanded Data Memory Access	CPU Mode Register	Fig. 1.16
$P4_1$				External Interrupt Input ($\overline{INT0}$)	Interrupt Edge Selection Register	Fig. 1.16
$P4_2$				External Interrupt Input ($\overline{INT1}$)	Interrupt Edge Selection Register	Fig. 1.16
$P4_3$				External Interrupt Input (CNTR0)	Timer X Mode Register	Fig. 1.16
$P4_4$				External Interrupt Input (CNTR1)	Timer Y Mode Register	Fig. 1.16
$P5_0$	Port 5	Input/output, individual bits	CMOS I/O port	X_{cin}	CPU Mode Register	Fig. 1.17
$P5_1$				X_{cout} or T_{out}	CPU Mode Register or T123 Mode Register	Fig. 1.17
$P5_2$				OBF_0 Output	MBI Interface	Fig. 1.17
$P5_3$				$\overline{IBF_0}$ Output	MBI Interface	Fig. 1.17
$P5_4$				$\overline{S_0}$ Input	MBI Interface	Fig. 1.17
$P5_5$				A_0 Input	MBI Interface	Fig. 1.17
$P5_6$				$\overline{R}$(E) Input	MBI Interface	Fig. 1.17
$P5_7$				$\overline{W}$(R/$\overline{W}$) Input	MBI Interface	Fig. 1.17
$P6_0$– $P6_7$	Port 6	Input/output, individual bits	CMOS I/O port	Master CPU data bus	MBI Interface	Fig. 1.18
$P7_0$	Port 7	Input/output, individual bits	CMOS I/O port	$\overline{SOF}$ Output	USB Interface	Fig. 1.19
$P7_1$				$\overline{HOLD}$	CPU Mode Register	Fig. 1.19
$P7_2$				$\overline{S_1}$ Input	MBI Interface	Fig. 1.19
$P7_3$				$\overline{IBF_1}$ Output	MBI Interface	Fig. 1.19
$P7_4$				OBF_1 Output	MBI Interface	Fig. 1.19
$P8_0$	Port 8	Input/output, individual bits	CMOS I/O port	UTXD2/$\overline{SRDY}$	UART Mode Register/SIO Mode Register	Fig. 1.20
$P8_1$				URXD2/SCLK	UART Mode Register/SIO Mode Register	Fig. 1.20
$P8_2$				$\overline{CTS2}$/STXD	UART Mode Register/SIO Mode Register	Fig. 1.21
$P8_3$				$\overline{RTS2}$/SRXD	UART Mode Register/SIO Mode Register	Fig. 1.21
$P8_4$				UTXD1	UART Mode Register	Fig. 1.21
$P8_5$				URXD1	UART Mode Register	Fig. 1.22
$P8_6$				$\overline{CTS1}$	UART Mode Register	Fig. 1.22
$P8_7$				$\overline{RTS1}$	UART Mode Register	Fig. 1.22

1.14.1 I/O Ports

This device has 66 programmable I/O pins arranged as ports P00 to P87. Each port bit can be configured as input or output. To set the I/O port bit direction, write a “1” to the corresponding direction register bit to select output mode, or write a “0” to the direction register bit to select input mode.

At reset, all of the direction registers are initialized to 00_{16}, setting all of the I/O ports to input mode.

If data is written to a pin and then read from that pin while it is in output mode, the data read is the value of the port latch rather than the value of the pin itself. Therefore, if an external load changes the value of an output pin, the intended output value will still be read correctly. Pins set to input mode are floating (provided that the pull up resistors are not being used) to ensure that the value input to such a pin can be read accurately. In the case when data is written to a pin configured as an input, the data is written only to the port latch; the pin itself remains floating.

Most of the I/O Ports are multiplexed with secondary functions. When a GPI/O is multiplexed with a second function, the control signal from the peripheral overrides the direction register. The multiplexing is briefly described below. The second function signals to and from the I/O ports are described in detail in their respective block’s description.

1.14.1.1 Ports P0, P1, and P3

Ports P0 and P1 act as the address bus (AB_0-AB_{15}) in Microprocessor and Memory Expansion modes. Bits 0 and 3-7 of Port P3 acts as control signals in Microprocessor and Memory Expansion modes.

1.14.1.2 Port P2

Port P2 is an 8-bit general purpose I/O port when in single chip mode. In this mode, the port has key-on wake up circuitry which can be used to restart the chip externally from a WIT or STP low power mode. This port also acts as the data bus during microprocessor and memory expansion modes. Port P2 input level can be set to reduced VIHL level or CMOS level by bit 6 of the port control register (PTC).

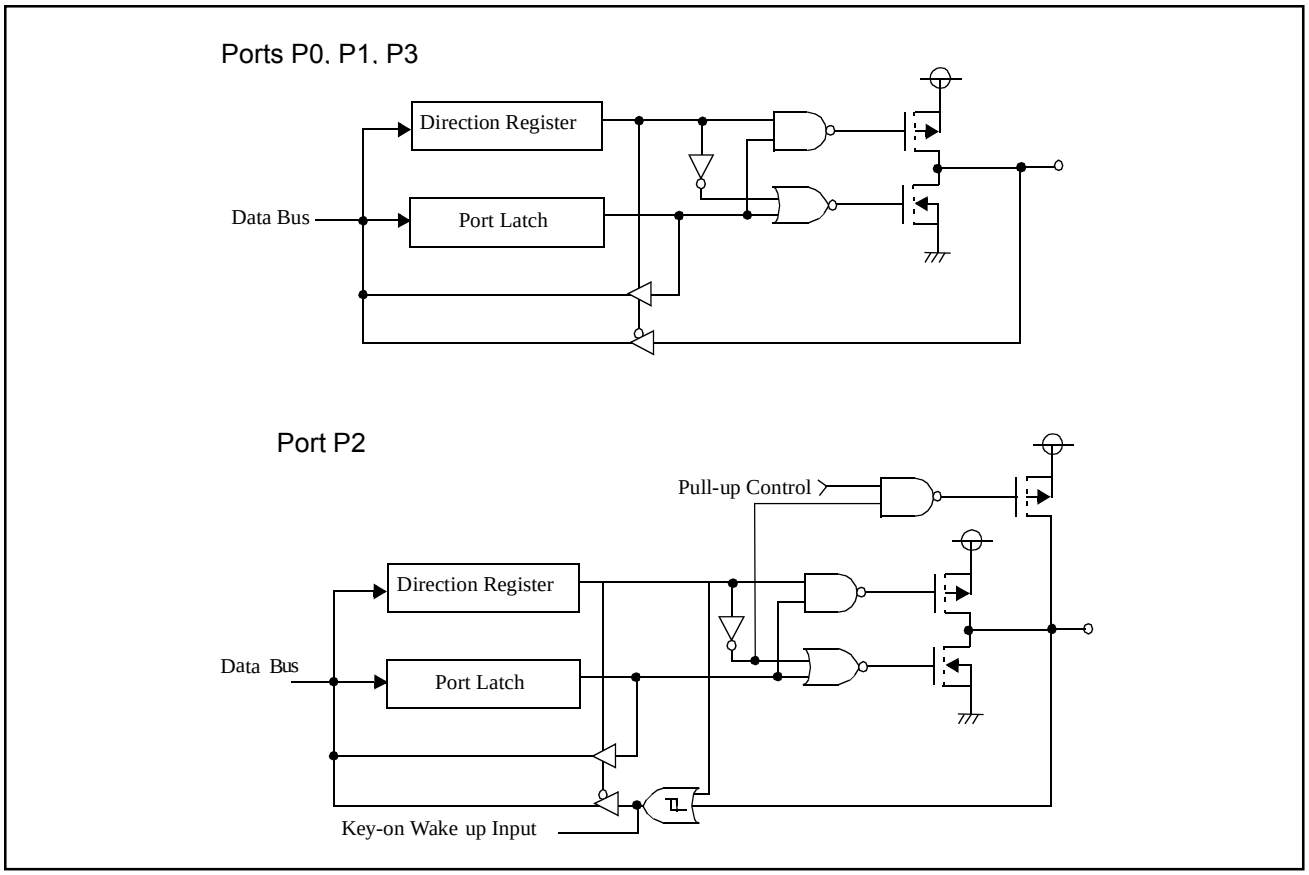


Fig. 1.15. Ports P0, P1, P2, P3 Block Diagram

1.14.1.3 Port P4

Port 4 is a 5-bit general purpose I/O port that can be configured to access special second functions. The port can be set up in any configuration in all three processor modes.

Port $P4_0$

This pin is multiplexed with the $\overline{EDMA}$ (Extended Data Memory Access) function. When the MCU is in memory expansion or microprocessor mode and CPMB4 is set to "1", this pin operates as the $\overline{EDMA}$ output as described in section 1.12.6.

Port $P4_1$- $P4_2$

These pins are multiplexed with external interrupts 0 and 1 ($\overline{INT}0$ and $\overline{INT}1$). The external interrupt function is enabled by setting the bits to "1" in the interrupt control register that correspond to $\overline{INT}0$ and $\overline{INT}1$. The interrupt polarity register can be configured to define $\overline{INT}0$ and $\overline{INT}1$ as active high or low interrupts. See section 1.15.1 for more information on configuring interrupts.

Port $P4_3$- $P4_4$

These pins are multiplexed with Timer X and Y functions for $P4_3$ and $P4_4$ respectively. The timer functions of the pins are independently defined by configuring the timer peripheral. $P4_3$ acts as Timer X input pin for pulse width measurement mode and event counter mode or as Timer X output pin for pulse output mode. $P4_3$ can also be used as an external interrupt (CNTR0) when Timer X in not in output mode. The polarity is selected in the Timer X mode register. The external interrupt function is enabled by setting the bit to "1" in the interrupt control register that corresponds to CNTR0. See section 1.15.1 for more information on configuring interrupts.

$P4_4$ acts as Timer Y input pin for pulse period measurement mode, pulse H-L measurement mode, and event counter mode or as Timer Y output pin for pulse output mode. $P4_3$ can also be used as an external interrupt (CNTR1) when Timer Y in not in output mode. The polarity is selected in the Timer Y mode register. The external interrupt function is enabled by setting the bit to "1" in the interrupt control register that corresponds to CNTR1. See section 1.15.1 for more information on configuring interrupts.

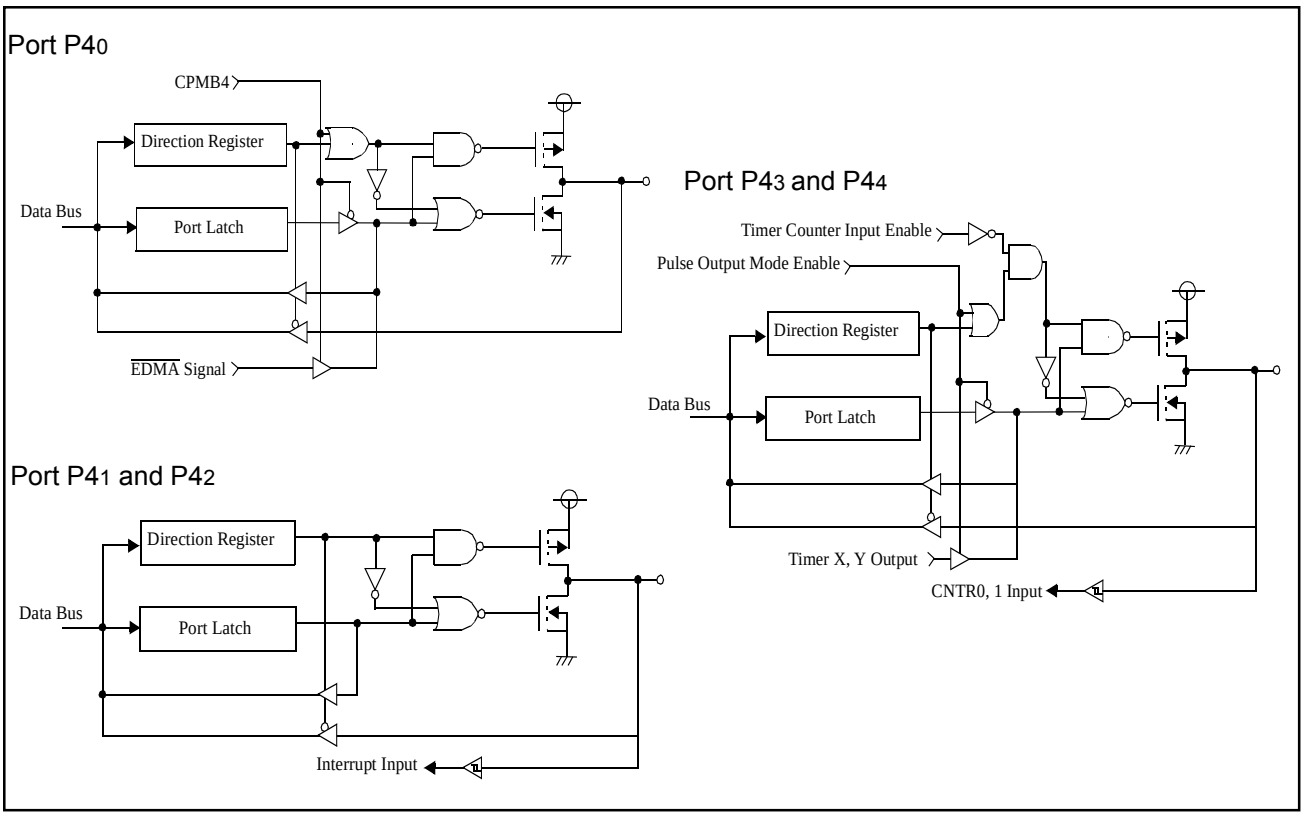


Fig. 1.16. Port P4 Block Diagram

1.14.1.4 Port P5

Port 5 is an 8-bit general purpose I/O port that can be configured to access special second functions. The port can be set up in any configuration in all three processor modes.

Port $P5_0$

This pin is multiplexed with the XC_{in} clock input. When the XC_{in} clock is activated, the pin's I/O is disabled.

Port $P5_1$

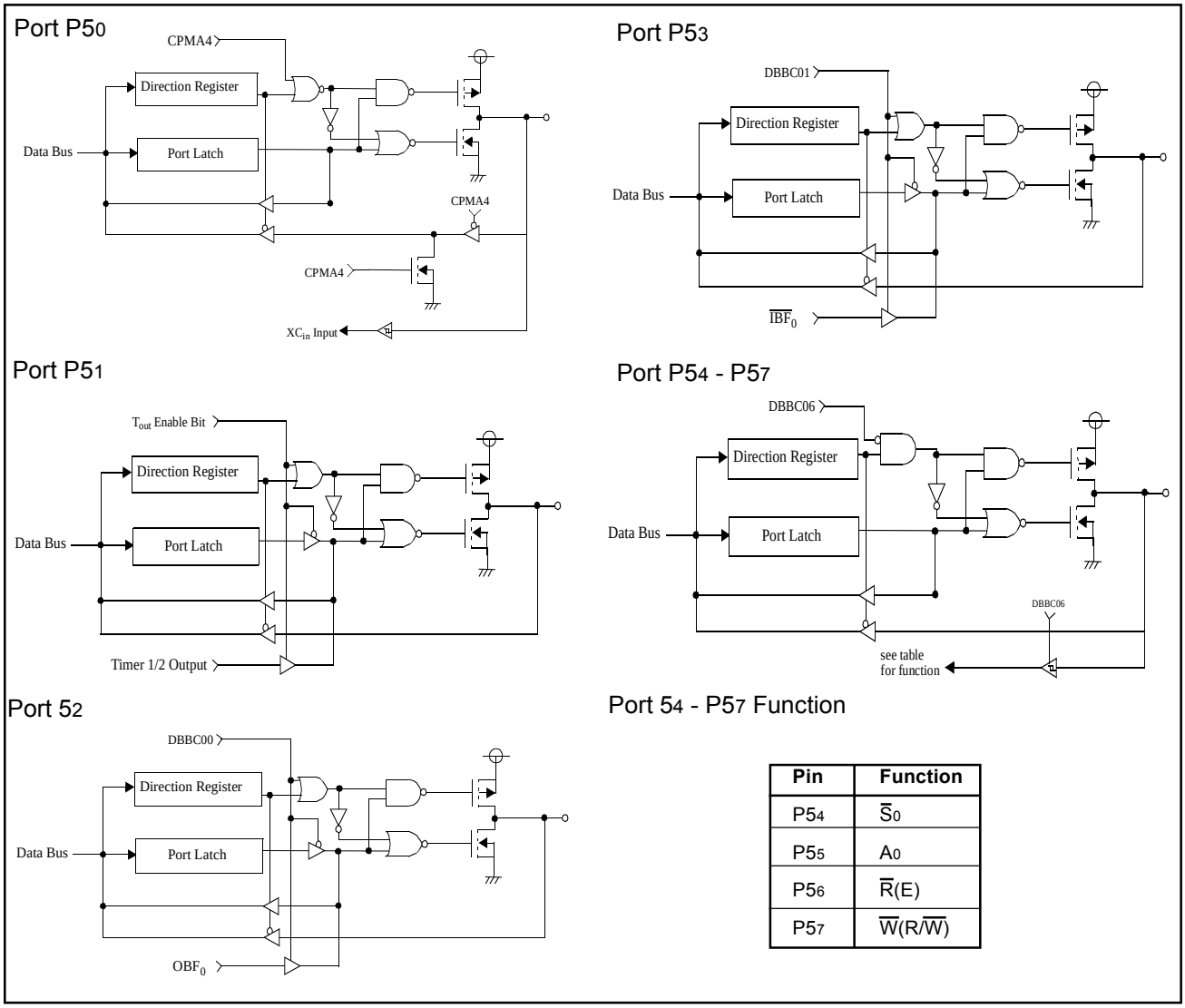
This pin is multiplexed with the XC_{out} clock output and the Timer 1/2 pulse output. When the XC_{in} clock is activated, the pin's I/O is disabled. If XC_{in} is not being used as a system clock or XC_{out} oscillation is disabled, the pin can be configured as the Timer 1/2 pulse output pin. This feature is configured in the Timer123 mode register as described in section 1.17.

Port $P5_2$- $P5_7$

These pins are multiplexed with control pins for the bus interface control block. $P5_2$ acts as OBF_0 output to a Master CPU when DBBC00 is "1".

$P5_3$ acts as $\overline{IBF_0}$ output to a Master CPU when DBBC01 is "1".

$P5_4$-$P5_7$ act as input control signals from a Master CPU when DBBC06 is "1". The table featured in Figure 1.17 shows the bus interface control signal that corresponds to each pin.



Port $P5_4$ - $P5_7$ Function

Pin	Function
$P5_4$	$\overline{S}_0$
$P5_5$	A_0
$P5_6$	$\overline{R}$(E)
$P5_7$	$\overline{W}$(R/$\overline{W}$)

Fig. 1.17. Port P5 Block Diagram

1.14.1.5 Port P6

Port P6 is an 8-bit general purpose I/O port that can be configured to access special second functions. The port acts as the data bus interface for the bus interface control block when DBBC06 is “1”. The port can be set up in any configuration in all three processor modes.

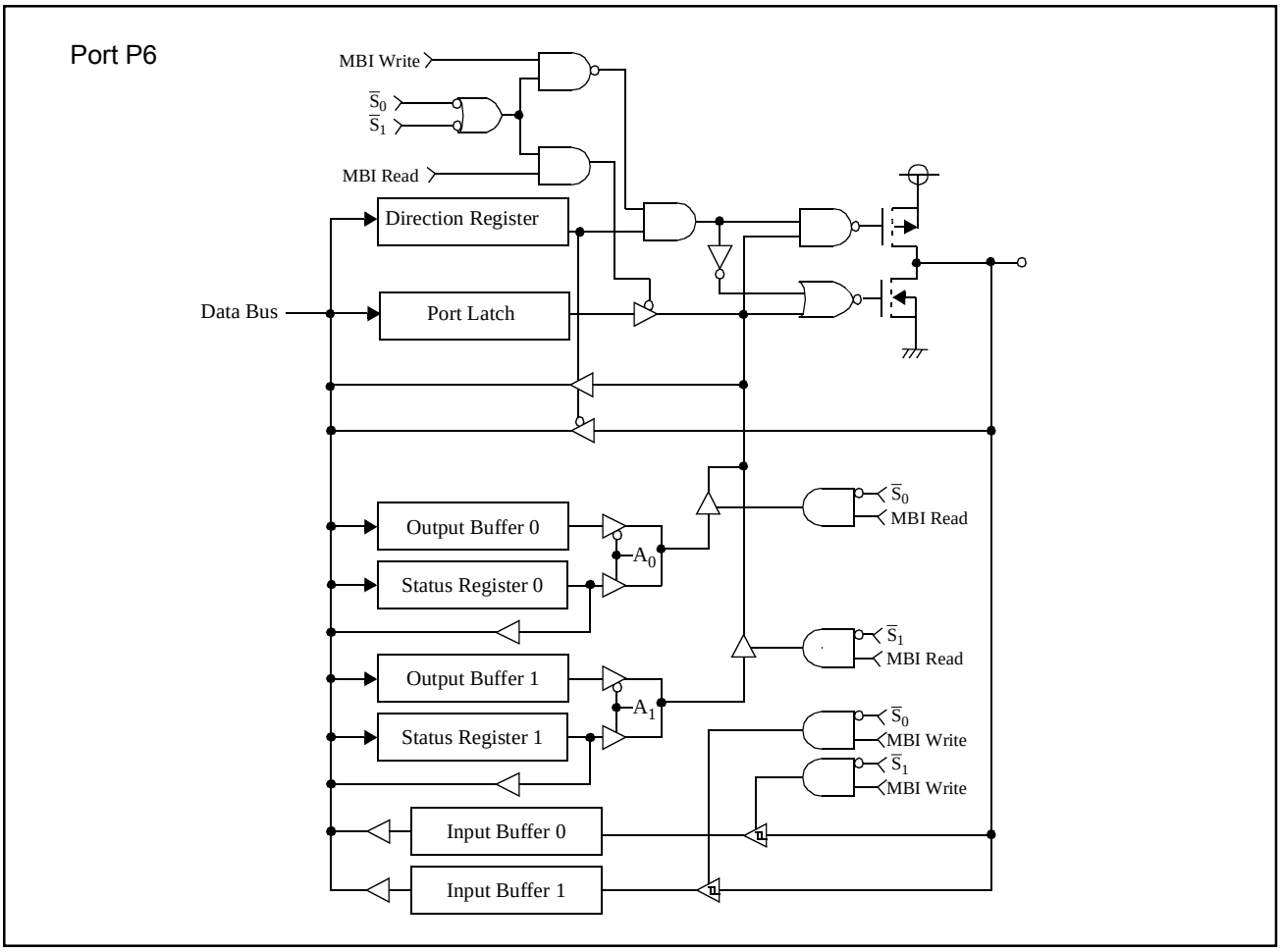


Fig. 1.18. Port P6 Block Diagram

1.14.1.6 Port P7

Port P7 is a 5-bit general purpose I/O port that can be configured to access special second functions.

Port P7$_0$

This pin is multiplexed with the USB start of frame pulse ($\overline{SOF}$) output. When USBC6 is a "1", this pin outputs the USB $\overline{SOF}$.

Port P7$_1$

This pin is multiplexed with the $\overline{HOLD}$ function. When the MCU is in memory expansion or microprocessor mode and CPMB5 is set to "1".

Port P7$_2$

This pin is multiplexed with the $\overline{S}_1$ input control signal from a Master CPU. When DBBC17 is "1", the pin takes on the function of the $\overline{S}_1$ input control signal.

Port P7$_3$

This pin is multiplexed with the $\overline{IBF}_1$ output control signal for a Master CPU and the $\overline{HLDA}$ function. When DBBC11 and DBBC17 are "1", the pin takes on the function of the $\overline{IBF}_1$ output control signal. When the MCU is in memory expansion or microprocessor mode, CPMB5 is set to "1", and the $\overline{IBF}_1$ function is not enabled.

Port P7$_4$

This pin is multiplexed with the OBF_1 control pin for the bus interface control block. P7$_4$ acts as OBF_1 output to a Master CPU when DBBC10 and DBBC17 are "1".

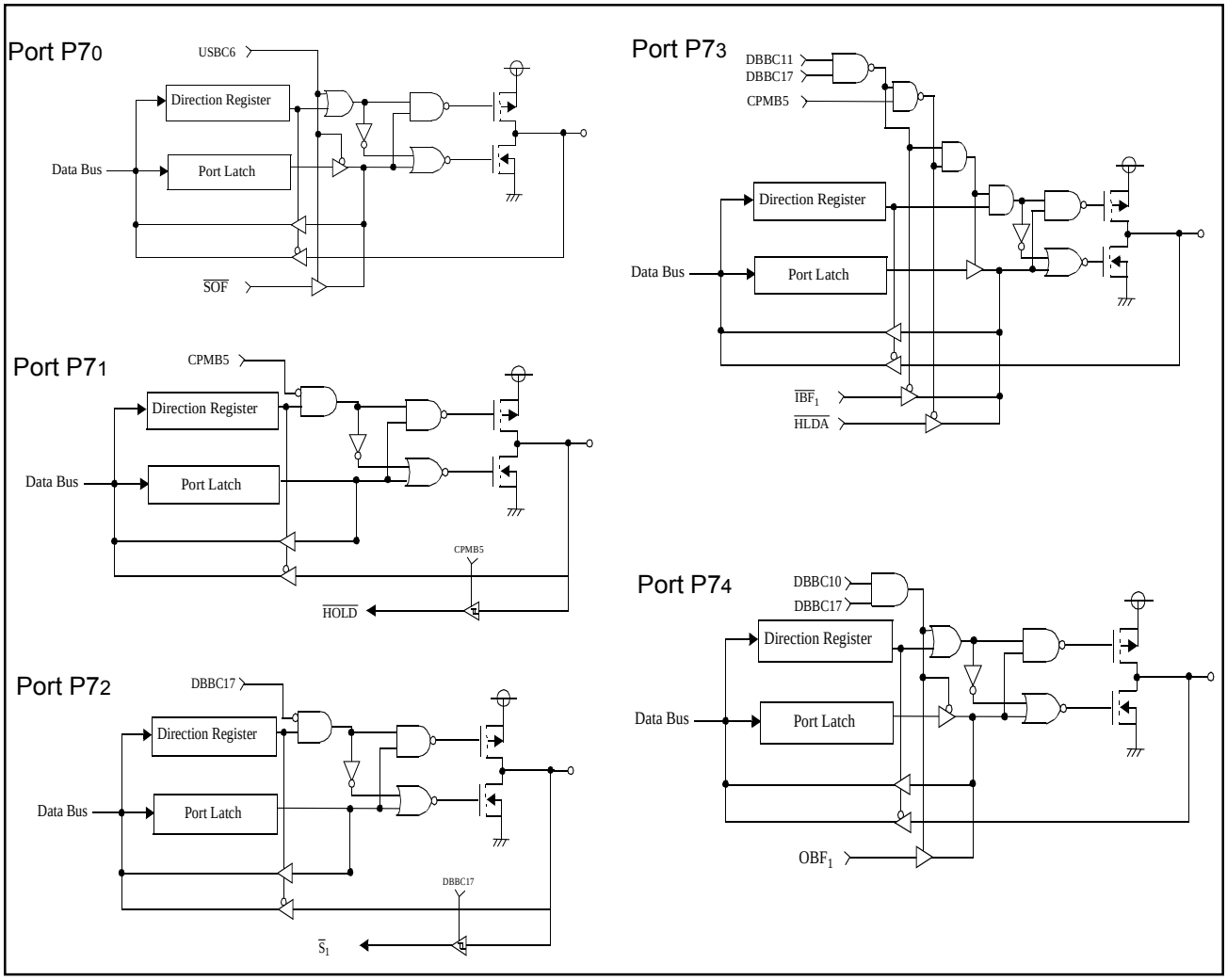


Fig. 1.19. Port P7 Block Diagram

1.14.5.7 Port 8

Port 8 is an 8-bit general purpose I/O port that can be configured to access special second functions. The port can be set up in any configuration in all three processor modes.

Port P80

This pin is multiplexed with the SIO $\overline{\text{SRDY}}$ signal and the UART2 TxD signal. When UART2 is in transmit mode, the pin acts as the TxD output signal. When the pin is not being used as the UART2 TxD output and bit 4 of the SIO control register 1 (SIOCON1) is a "1", the port acts as the SIO $\overline{\text{SRDY}}$ output signal. If during this function, the SIO is configured in slave mode, this pin acts as a slave input from a master. See section 1.18 for more SIO information.

Port P81

This pin is multiplexed with the SIO SCLK signal and the UART2 RxD signal. When UART2 is in receive mode, the pin acts as the RxD input signal. When the pin is not being used as the UART2 RxD input and bit 2 of the SIO control register 1 (SIOCON1) is a "1", the port acts as the SIO SCLK signal. In this mode a "1" in bit 6 of SIOCON1 configures the pin to output SCLK whereas a "0" configures the pin to input SCLK.

Port P82

This pin is multiplexed with the SIO SRxD signal and the UART2 $\overline{\text{CTS}}$ signal. When bit 5 of the UART2 control register (U2CON) is a "1", the port acts as the $\overline{\text{CTS}}$ input signal. When the pin is not being used as the UART2 $\overline{\text{CTS}}$ input and bit 2 of the SIO control register 2 (SIOCON2) is a "1", the port acts as the SIO SRxD input signal.

Port P83

This pin is multiplexed with the SIO STxD signal and the UART2 $\overline{\text{RTS}}$ signal. When bit 6 of the UART2 control register (U2CON) is a "1", the port acts as the $\overline{\text{RTS}}$ output signal. When the pin is not being used as the UART2 $\overline{\text{RTS}}$ output and bit 3 of the SIO control register 1 (SIOCON1) is a "1", the port acts as the SIO STxD output signal.

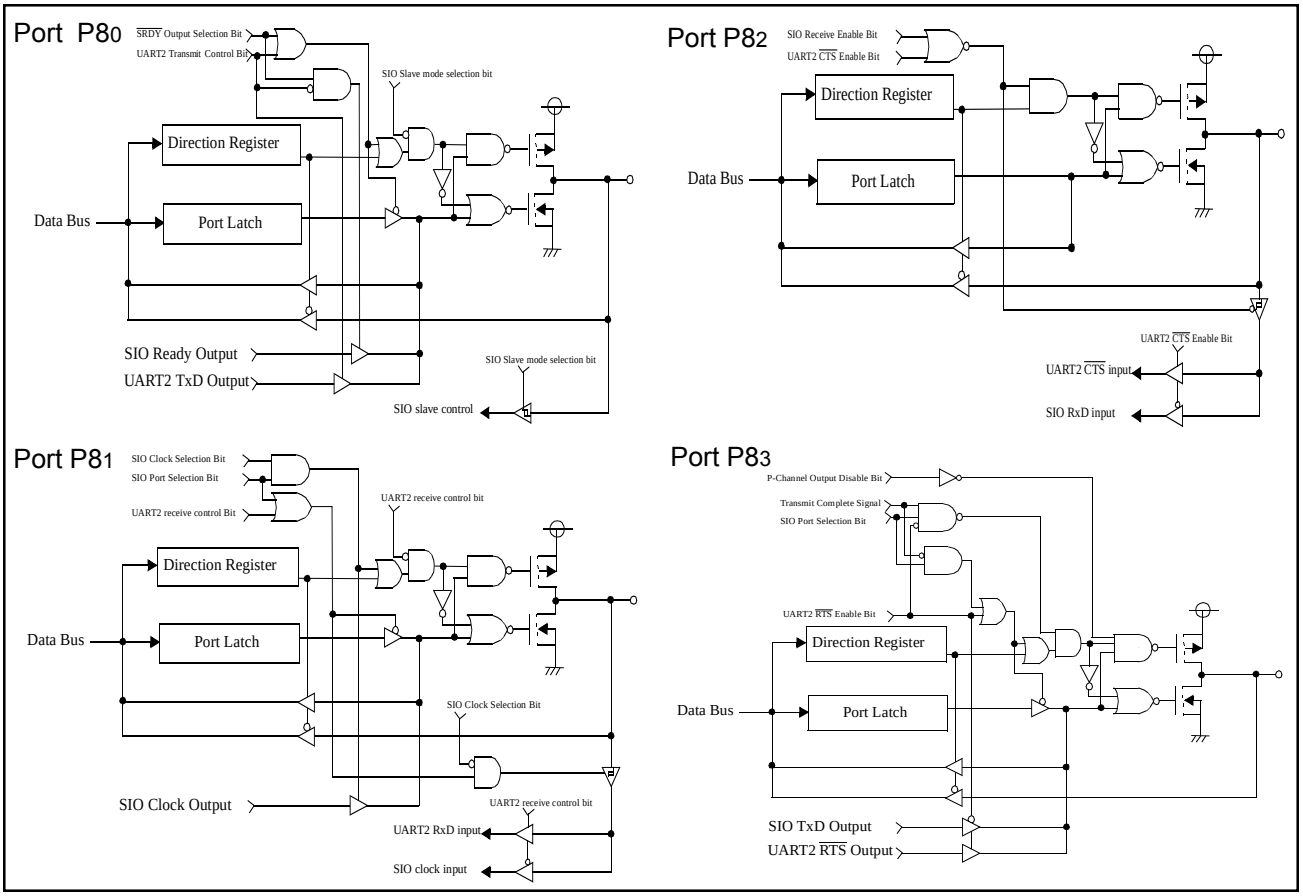


Fig. 1.20. Port P80, P81, P82, P83 Block Diagram

Port P84

This pin is multiplexed with the UART1 TxD signal. When UART1 is in transmit mode, the pin acts as the TxD output signal.

Port P85

This pin is multiplexed with the UART1 RxD signal. When UART1 is in receive mode, the pin acts as the RxD input signal.

Port P86

This pin is multiplexed with the UART1 $\overline{\text{CTS}}$ signal. When bit 5 of the UART1 control register (U1CON) is a "1", the port acts as the $\overline{\text{CTS}}$ input signal.

Port P87

This pin is multiplexed with the UART1 $\overline{\text{RTS}}$ signal. When bit 6 of the UART1 control register (U1CON) is a "1", the port acts as the $\overline{\text{RTS}}$ output signal.

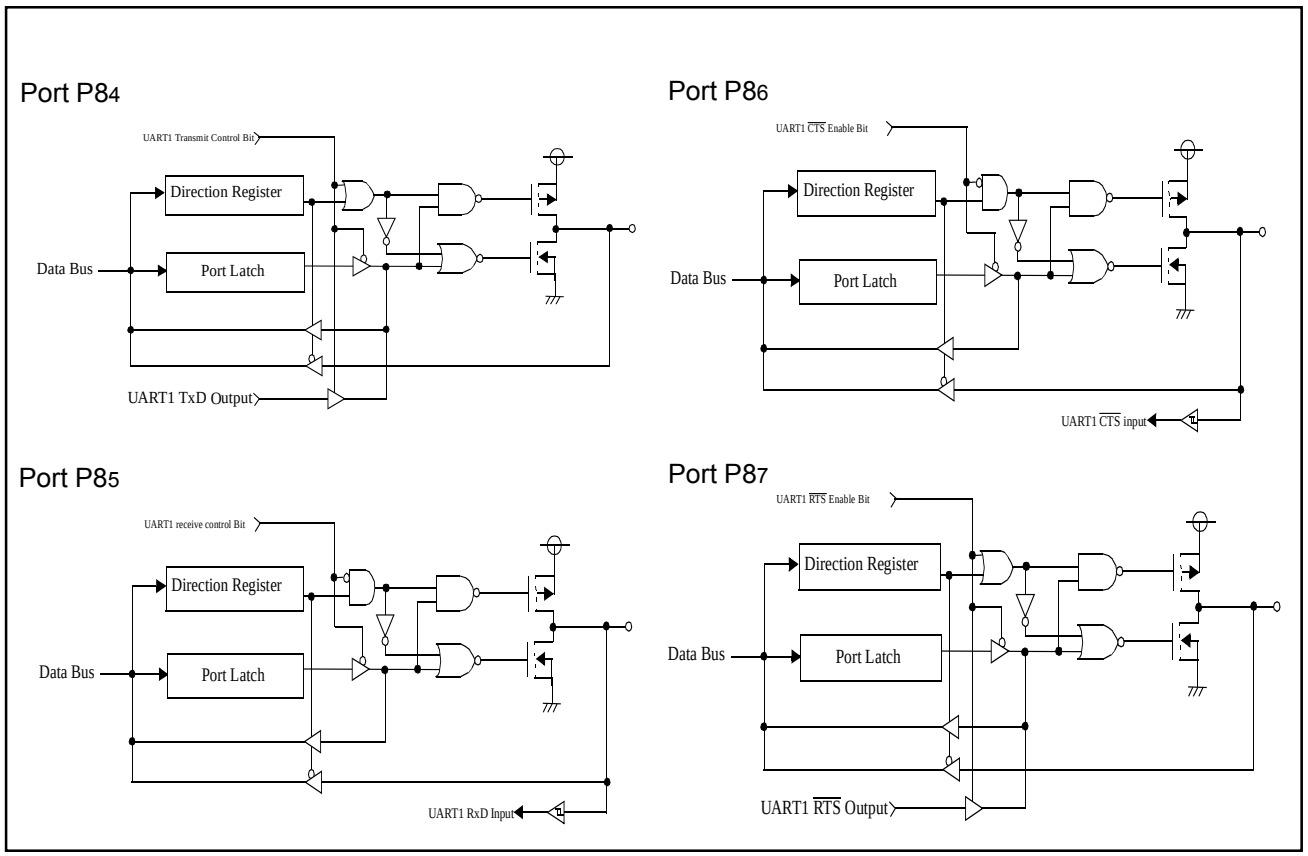


Fig. 1.21. Port P84, P85, P86 and P87 Block Diagram

1.14.2 Port Control Register (PTC)

This device is equipped with a port control register to turn on and off the slew rate control and to control the input levels for Port P2 and the MBI pins (see Figure 1.22).

1.14.3 Port P2 Pull-up Control Register (PUP2)

This device is equipped with internal pull ups on Port P2 that can be enabled by software. Each bit of the pull-up control register controls a corresponding pin of Port P2. The pull-up control register pulls up the port when the port is in input mode. The value of the pull-up control regsiter has no effect when the port is in output mode (see Figure 1.23).

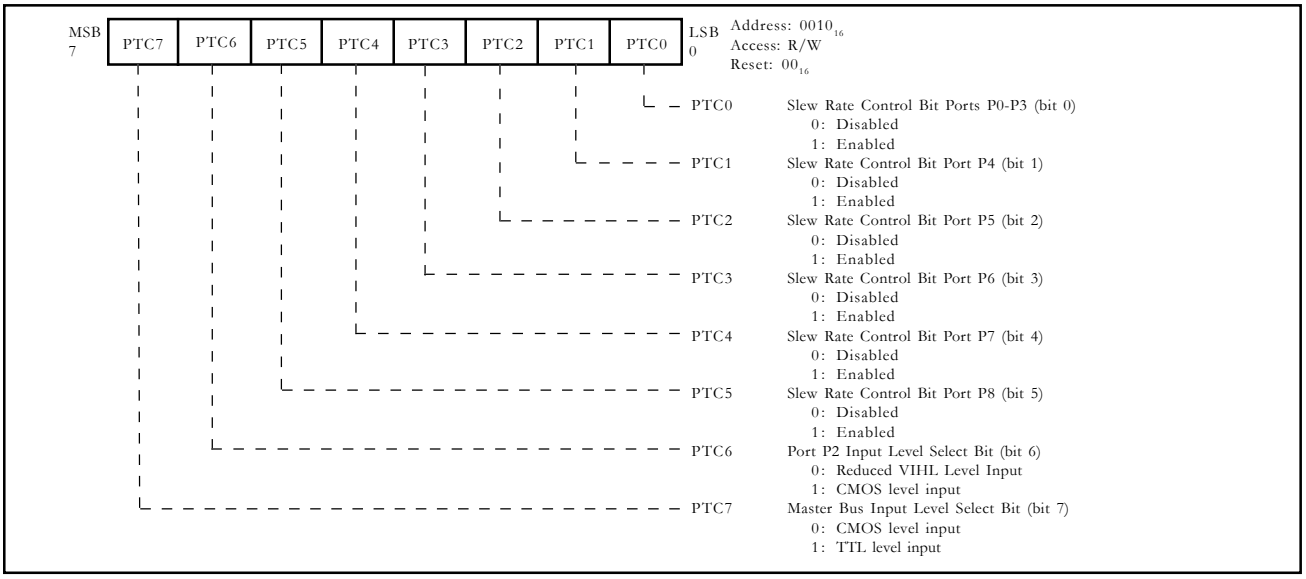


Fig. 1.22. Port Control Register (PTC)

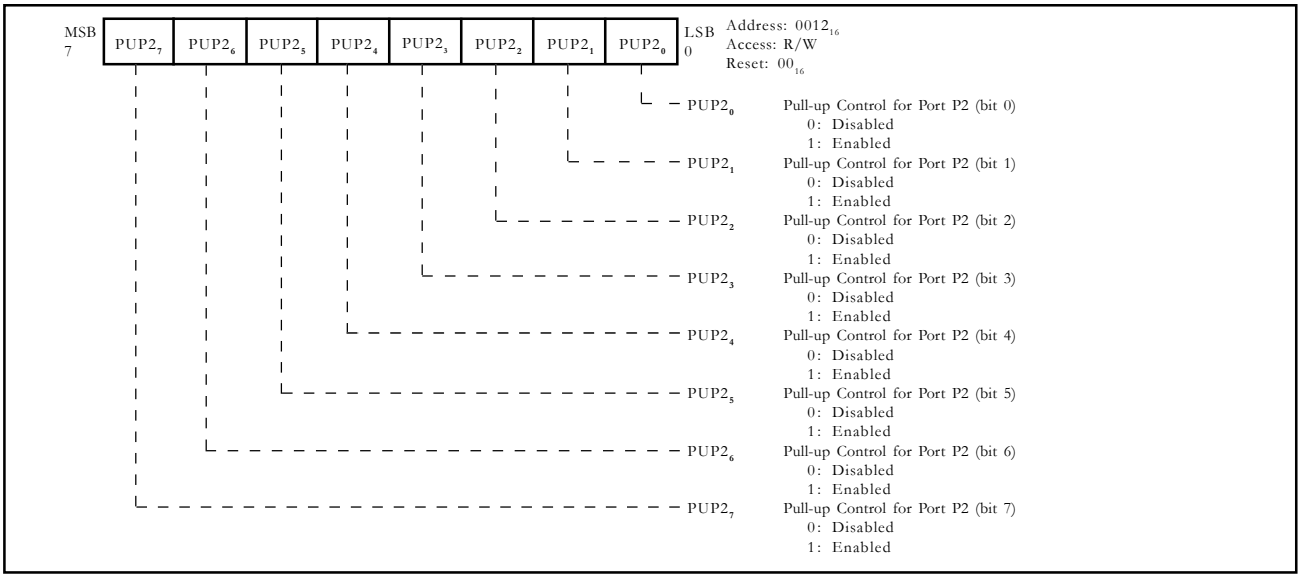


Fig. 1.23. Pull-up Control Register (PUP2)

1.15 INTERRUPT CONTROL UNIT

This section details a specialized peripheral, the interrupt control unit (ICU).

This series supports a maximum of 23 maskable interrupts, one software interrupts, and one reset vector that is treated as a non-maskable interrupt.

Table 1.6 describes the interrupt registers. See Table 1.7 for the interrupt sources, jump destination addresses, interrupt priorities, and section references for the interrupt request sources.

Table 1.6. Interrupt Registers

Address	Description	Acronym and Value at Reset
0002_{16}	Interrupt request register A	IREQA=00
0003_{16}	Interrupt request register B	IREQB=00
0004_{16}	Interrupt request register C	IREQC=00
0005_{16}	Interrupt control register A	ICONA=00
0006_{16}	Interrupt control register B	ICONB=00
0007_{16}	Interrupt control register C	ICONC=00
0011_{16}	Interrupt polarity selection register	IPOL=00

Table 1.7. Interrupt Vector

Priority	Interrupt	Jump Destination Storage Address (Vector Address)		Remarks				
		High-order Byte	Low-order Byte					Reference
1	RSRV1	FFFF	FFFE	Reserved for factory use				
2	RSRV2	FFFD	FFFC	Reserved for factory use				
3	$\overline{\text{RESET}}$	FFFB	FFFA	User $\overline{\text{RESET}}$ (Non-Maskable)				
4	USB	FFF9	FFF8	USB Function Interrupt	0	LSB	IREQA & ICONA	Section 1.21.2.1
5	$\overline{\text{SOF}}$	FFF7	FFF6	USB $\overline{\text{SOF}}$ Interrupt	1			Section 1.21.2.2
6	$\overline{\text{INT0}}$	FFF5	FFF4	External Interrupt 0	2			Section 1.15.1
7	$\overline{\text{INT1}}$	FFF3	FFF2	External Interrupt 1	3			Section 1.15.1
8	DMA1	FFF1	FFF0	DMAC Channel 0 Interrupt	4			Section 1.23
9	DMA2	FFEF	FFEE	DMAC Channel 1 Interrupt	5			Section 1.23
10	U1RBF	FFED	FFEC	UART1 Receiver Buffer Full	6			Section 1.19.4.2
11	U1TX	FFEB	FFEA	UART1 Transmit Interrupt	7	MSB		Section 1.19.4.1
12	U1ES	FFE9	FFE8	UART1 Error Sum Interrupt	0	LSB	IREQB & ICONB	Section 1.19.4.2
13	U2RBF	FFE7	FFE6	UART2 Receiver Buffer Full	1			Section 1.19.4.2
14	U2TX	FFE5	FFE4	UART2 Transmit Interrupt	2			Section 1.19.4.1
15	U2ES	FFE3	FFE2	UART2 Error Sum Interrupt	3			Section 1.19.4.2
16	TX	FFE1	FFE0	Timer X Interrupt	4			Section 1.17
17	TY	FFDF	FFDE	Timer Y Interrupt	5			Section 1.17
18	T1	FFDD	FFDC	Timer 1 Interrupt	6			Section 1.17
19	T2	FFDB	FFDA	Timer 2 Interrupt	7	MSB		Section 1.17
20	T3	FFD9	FFD8	Timer 3 Interrupt	0	LSB	IREQC & ICONC	Section 1.17
21	CNTR0	FFD7	FFD6	External CNTR0 Interrupt	1			Section 1.17.1.2
22	CNTR1	FFD5	FFD4	External CNTR1 Interrupt	2			Section 1.17.2
23	SIO	FFD3	FFD2	SIO Interrupt	3			Section 1.18
24	$\overline{\text{IBF}}$	FFD1	FFD0	Input Buffer Full Interrupt	4			Section 1.22
25	OBE	FFCF	FFCE	Output Buffer Empty Interrupt	5			Section 1.22
26	KEY	FFCD	FFCC	Key-on Wake Up	6	MSB		Section 1.16
27	BRK	FFCB	FFCA	BRK Instruction (Non-Maskable)				

Corresponding Register Assignment

1.15.1 Interrupt Control

Each maskable interrupt has associated with it an interrupt request bit and an interrupt enable bit. These bits, along with the I flag, determine whether interrupt events can cause an interrupt service request to be generated. An interrupt request bit is set to "1" when its corresponding interrupt event is activated. The bit is cleared to a "0" when the interrupt is serviced or when a "0" is written to the bit. The bit can not be set high by writing "1" to it. Each interrupt enable bit determines whether the interrupt request bit it is paired with is seen when the interrupts are polled. When the interrupt enable bit is a "0", the interrupt request bit is not seen; and when the enable bit is a "1", the interrupt request is seen.

The interrupt request registers (IREQ) for the 23 maskable interrupts are shown in Figure 1.24, Figure 1.25, and Figure 1.26. The interrupt control registers (ICON) for the 23 maskable interrupts are shown in Figure 1.27, Figure 1.28, and Figure 1.29.

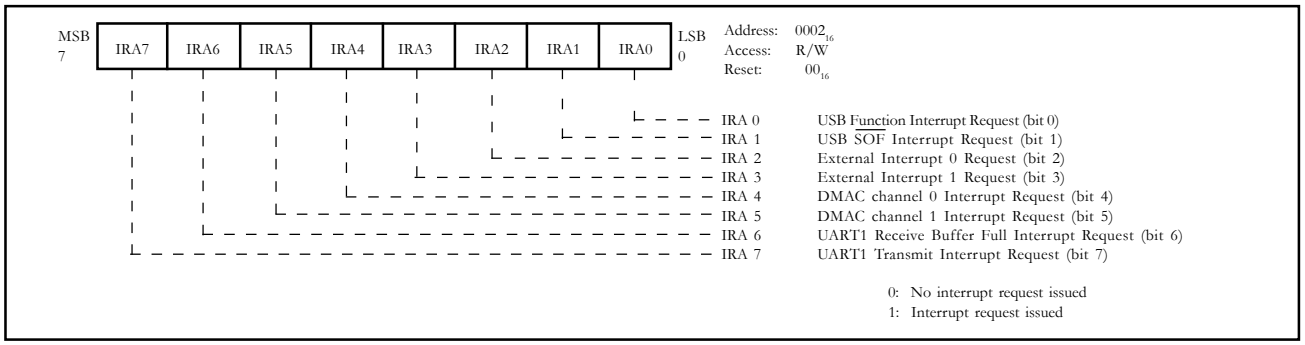


Fig. 1.24. Interrupt Request Register A (IREQA)

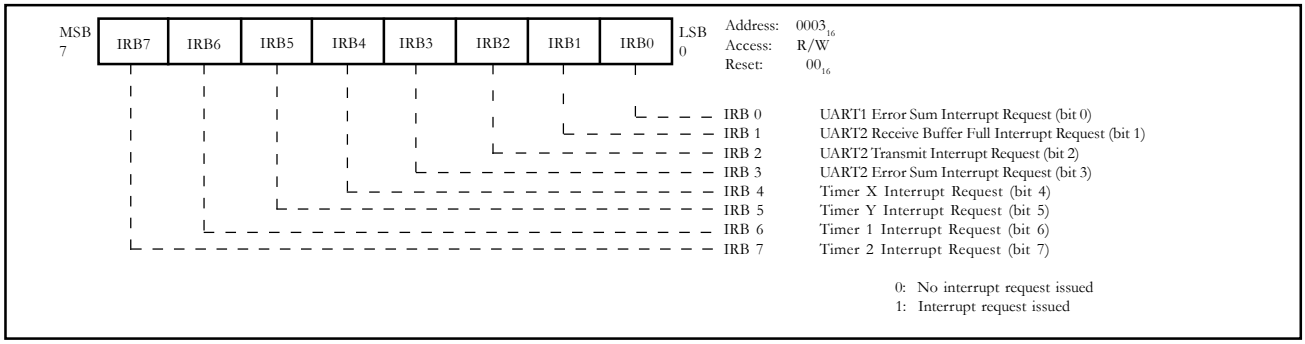


Fig. 1.25. Interrupt Request Register B (IREQB)

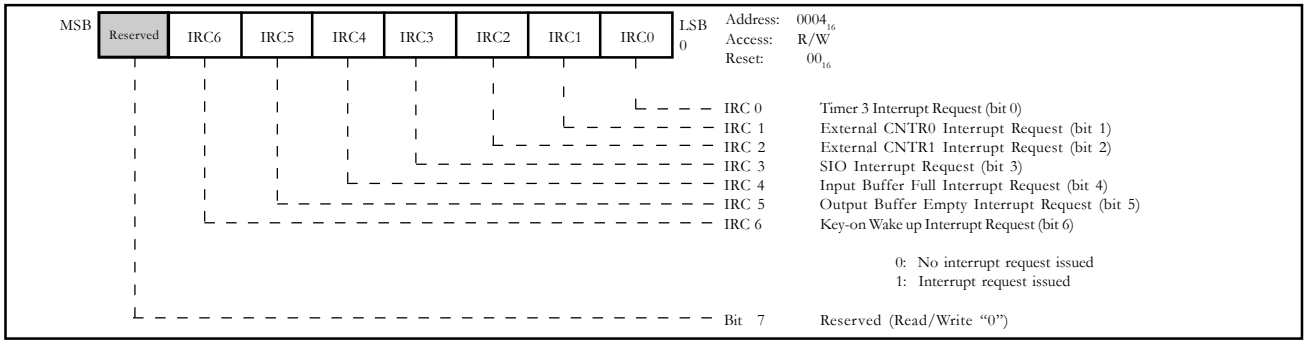


Fig. 1.26. Interrupt Request Register C (IREQC)

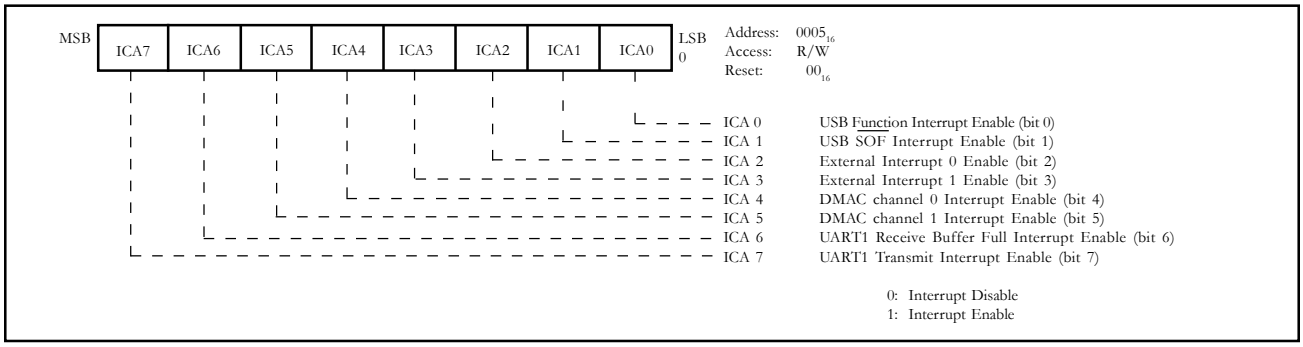


Fig. 1.27 Interrupt Control Register A (ICONA)

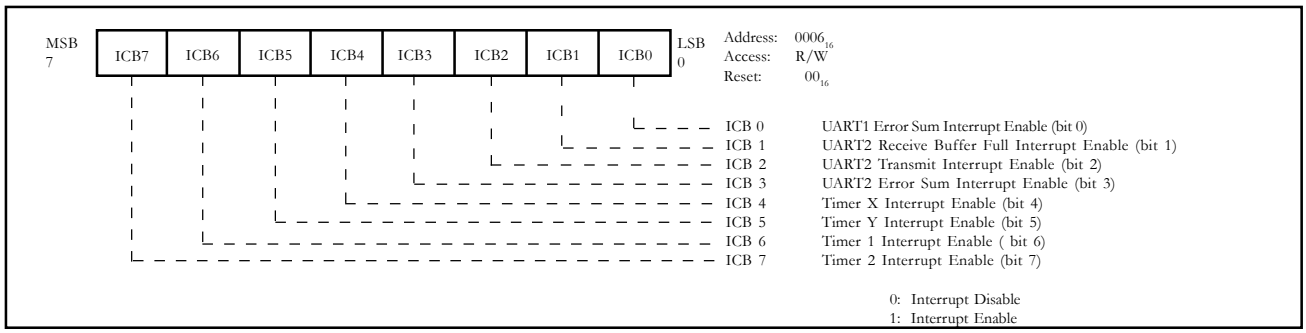


Fig. 1.28. Interrupt Control Register B (ICONB)

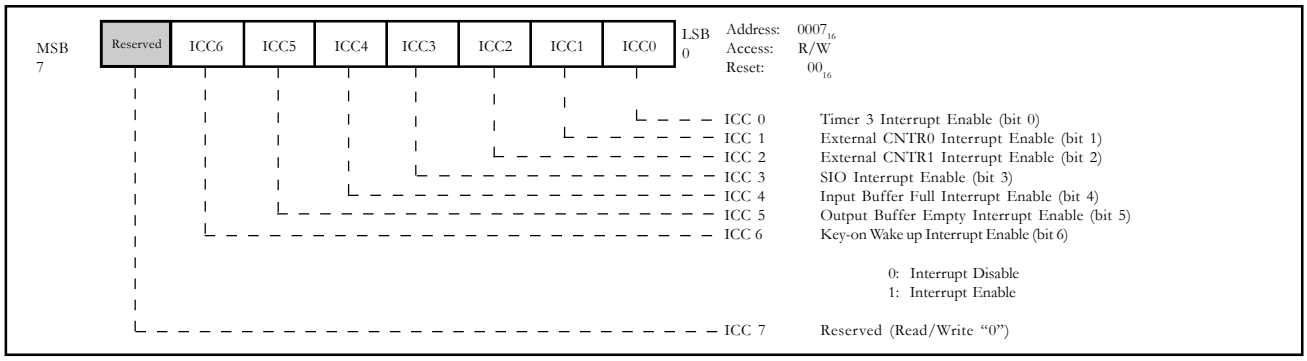


Fig. 1.29. Interrupt Control Register C (ICONC)

The interrupt polarity register allows the user to select the edge that will trigger an external interrupt request. The polarity register (IPOL) for the external interrupts is shown in Figure 1.30.

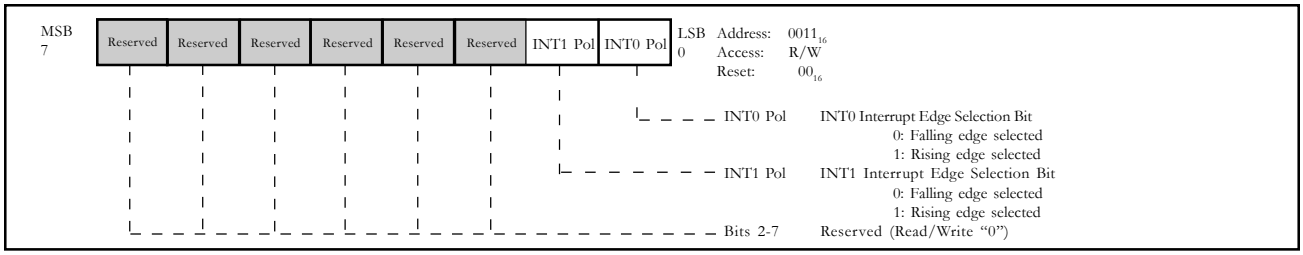


Fig. 1.30. Interrupt Polarity Register (IPOL)

1.16 KEY-ON WAKE UP

This device contains a key-on wake up interrupt function. The key-on wake up interrupt function is one way of returning from a power-down state caused by the STP or WIT instructions. This interrupt is generated by applying low level to any pin of Port P2. If a key matrix is connected as shown in Figure 1.31, the microcomputer can be returned to a normal state by pressing any one of the keys. Key-on wake up is enabled in single-chip mode only.

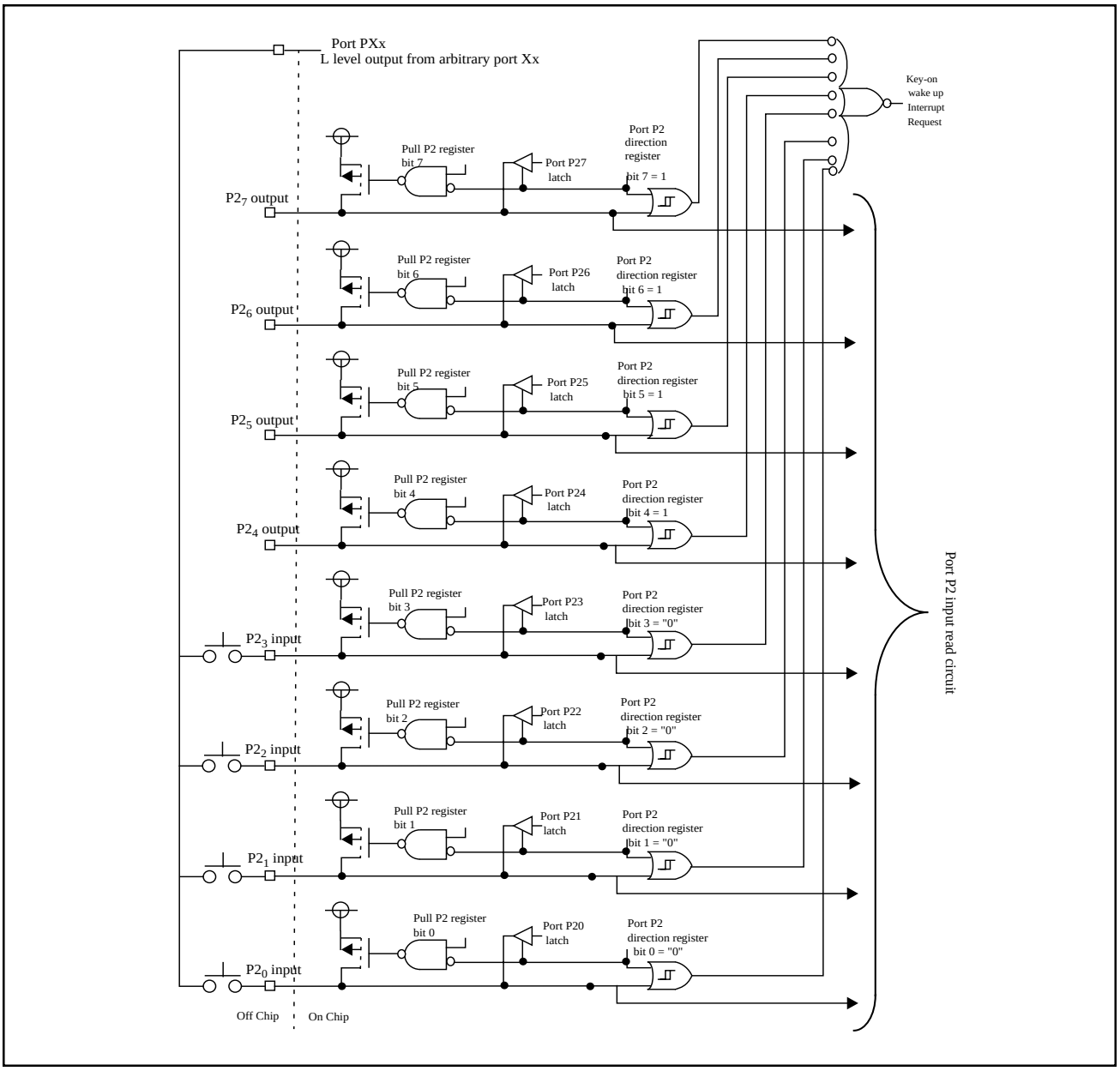


Fig. 1.31. Port P2 with Key-on Wake up function

1.17 TIMERS

This device has five built-in timers: Timer X, Timer Y, Timer 1, Timer 2, and Timer 3.

The contents of the timer latch, corresponding to each timer, determine the divide ratio. The timers can be read or written at any time. However, the read and write operations on the high and low-order bytes of the 16-bit timers (Timer X and Y) must be performed in a specific order.

The timers are all down count timers; when the count of a timer reaches 00_{16} (0000_{16} for Timer X and Y), an underflow occurs at the next count pulse and the contents of the corresponding timer reload latch are reloaded into the timer. When a timer underflows, the interrupt request bit corresponding to that timer is set to a "1".

The divide ratio of a timer is given by 1/(n + 1), where n is the value written to the timer. When the STP instruction is executed or $\overline{\text{RESET}}$ is asserted, 01_{16} is loaded into Timer 2 and the Timer 2 reload latch, and FF_{16} is loaded into Timer 1 and the Timer 1 reload latch.

Figure 1.32 is a block diagram of the five timers.

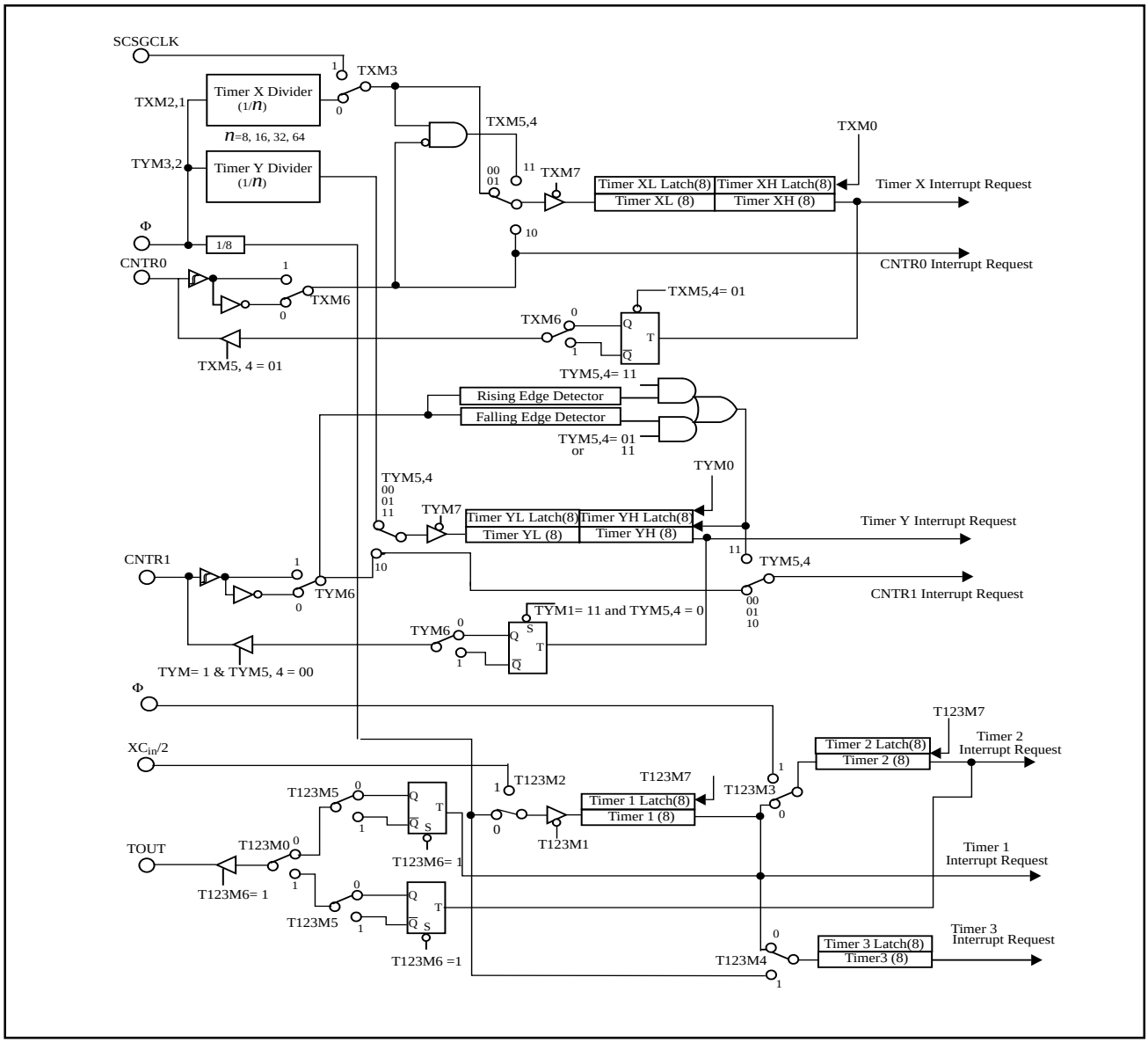


Fig. 1.32. Block diagram of Timers X, Y, 1, 2, and 3

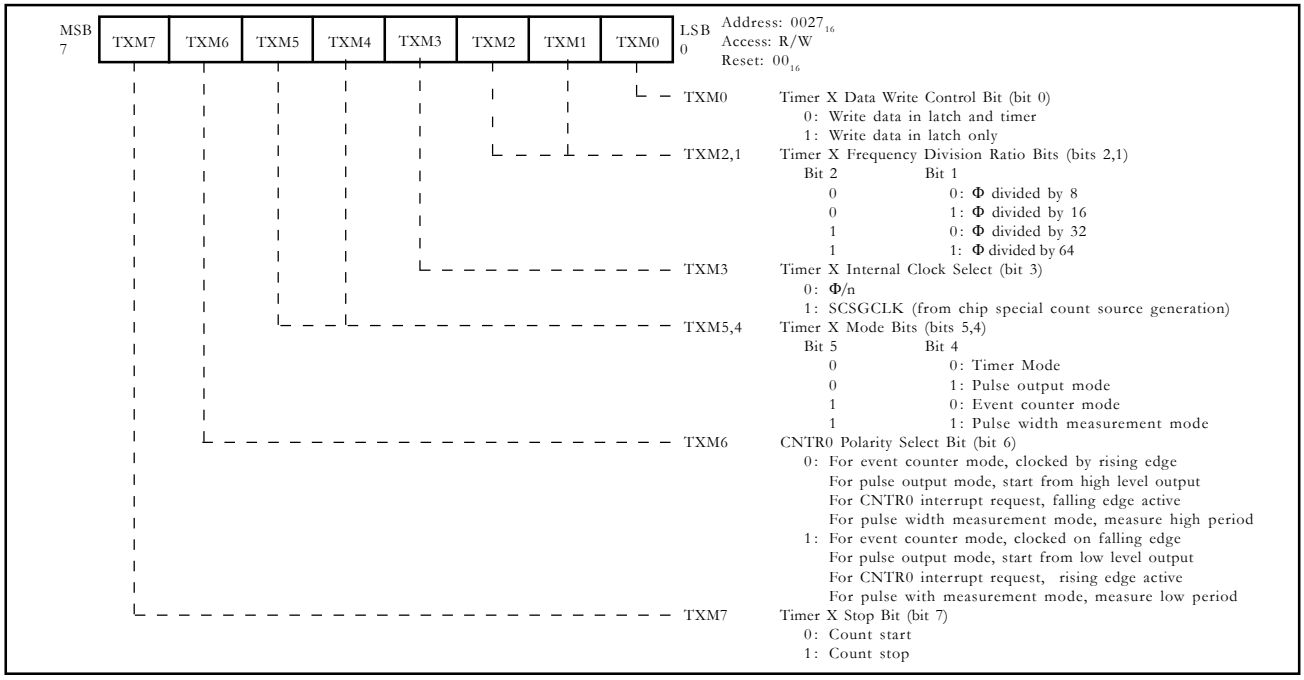


Fig. 1.33. Timer X Mode Register (TXM)

1.17.1 Timer X

Timer X is a 16-bit timer that has a 16-bit reload latch, and can be placed in one of four modes by setting bits TXM4 and TXM5 (bits 4 and 5 of the Mode Register, TXM). The bit assignment of the TXM is shown in Figure 1.33.

1.17.1.1 Read and Write Method

Read and write operations on the high and low-order bytes of Timer X must be performed in a specific order.

•Write Method

When writing to the timer, the lower order byte is written first. This data is placed in a temporary register that is assigned the same address as Timer XL. Next, the higher order byte is written. When this is done, the data is placed in the Timer XH reload latch and the low-order byte is transferred from its temporary register to the Timer XL reload latch. At this point, if the Timer X Data Write Control Bit (TXM0) (bit 0) is “0”, the value in the Timer X reload latch is also loaded in Timer X. If TXM0 is “0”, the data in the Timer X reload latch is loaded in Timer X after Timer X underflows.

•Read Method

When reading Timer X, the high-order byte is real first. Reading the high-order byte causes the values of Timer XH and Timer XL to be placed in temporary registers assigned the same addresses as Timer XH and Timer XL. The low-order byte of Timer X is then read from its temporary register. This operation assures the correct reading of Timer X while it is counting.

1.17.1.2 Count Stop Control

If the Timer X Count Stop Bit (TXM7) (bit 7 of the TXM) is set to a “1”, Timer X stops counting in all four modes.

•Timer Mode

Count Source: Φ/n (where n is 8, 16, 32, or 64) or SCSGCLK

In this mode, each time the timer underflows, the corresponding timer interrupt request bit is set to a “1”, the contents of the timer latch are loaded into the timer, and the count down sequence begins again.

•Pulse Output Mode

Count Source:Φ/n (where n is 8, 16, 32, or 64) or SCSGCLK

Each time the timer X underflows, the output of the CNTR0 pin is inverted, and the corresponding Timer X interrupt request bit is set to a "1". The repeated inversion of the CNTR0 pin output produces a rectangular waveform with a duty ratio of 50 percent. The initial level of the output is determined by the CNTR0 polarity select bit (bit 6). When this bit is low, the output starts from a high level. When this bit is high, the output starts from a low level.

•Event Counter Mode

Count Source: CNTR0

Timer countdown is triggered by inputs to the CNTR0 pin. Each time a timer underflows, the corresponding timer interrupt request bit is set to a "1", the contents of the timer reload latch are loaded into the timer, and the countdown sequence begins again.

The edge used to clock Timer X is determined by the CNTR0 polarity select bit (bit 6).

•Pulse Width Measurement Mode

Count Source: Φ/n (where n is 8, 16, 32, or 64) or SCSGCLK

This mode measures either the high or low-pulse width of the signal on the CNTR0 pin. The pulse width measured is determined by the CNTR0 polarity select bit (bit 6). When this bit is "0", the high pulse is measured. When this bit is "1", the low pulse is measured.

The timer counts down while the level on the CNTR0 pin is the polarity selected by the CNTR0 polarity select bit. When the timer underflows, the Timer X interrupt request bit is set to a "1", the contents of the timer reload latch are reloaded into the timer, and the timer continues counting down. Each time the signal polarity switches to the inactive state, a CNTR0 interrupt occurs indicating that the pulse width has been measured. The width of the measured pulse can be found by reading Timer X during the CNTR0 interrupt service routine.

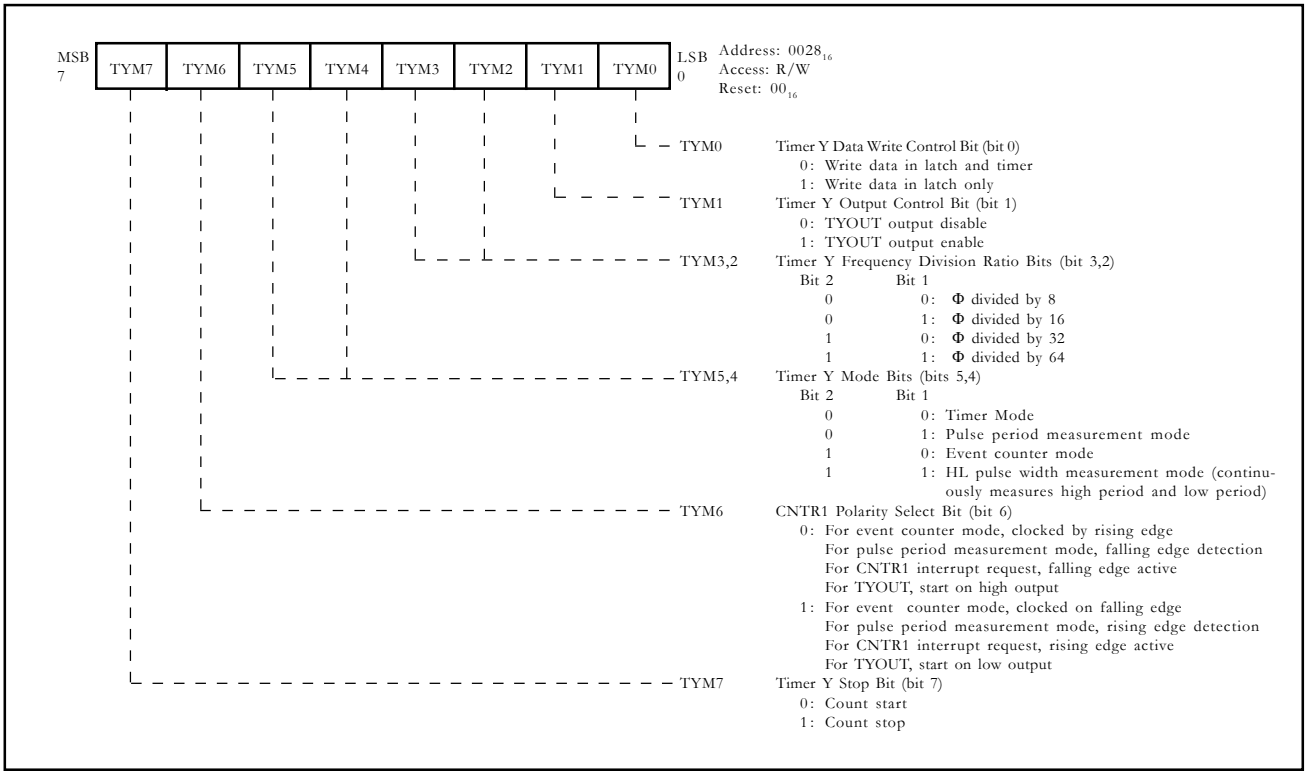


Fig. 1.34. Timer Y Mode Register (TYM)

1.17.2 Timer Y

Timer Y is a 16-bit timer that has a 16-bit reload latch, and can be placed in any of four modes by setting TYM4 and TYM5 (bits 4 and 5) (see Figure 1.34). The desired mode is selected by modifying the values of TYM4 and TYM5.

1.17.2.1 Read and Write Method

Read and write operations on the high and low-order bytes of Timer Y must be performed in a specific order.

•Write Method

When writing to the timer, the lower order byte is written first. This data is placed in a temporary register that is assigned the same address as Timer YL. Next, the high-order byte is written. Then, the data is placed in the Timer YH reload latch and the low-order byte is transferred from its temporary register to the Timer YL reload latch. At this point, if the Timer Y Data Write Control Bit (TYM0) (bit 0) is low, the value in the Timer Y reload latch is also loaded in Timer Y. If TYM0 is "1", the data in the Timer Y reload latch is loaded in Timer Y after Timer Y underflows.

•Read Method

When reading Timer Y, the high-order byte is read first. Reading the high-order byte causes the values of Timer YH and Timer YL to be placed in temporary registers that are assigned the same addresses as Timer YH and Timer YL. The low-order byte of Timer Y is then read from its temporary register. This operation assures the correct reading of Timer Y while it is counting.

1.17.2.2 Count Stop Control

If the Timer Y Count Stop Bit (TYM7) (bit 7) is set to a "1", Timer Y stops counting in all four modes.

•Timer Mode

Count Source:Φ/n (where n is 8, 16, 32, or 64)

In this mode, each time the timer underflows, the corresponding timer interrupt request bit is set to a "1", the contents of the timer latch are loaded into the timer, and the count down sequence begins again.

In Timer mode, the signal TYOUT can also be brought out on the CNTR1 pin. This is controlled by TYM1 (bit1).

Each time the Timer Y underflows, the output of the CNTR1 pin is inverted, and the corresponding Timer Y interrupt request bit is set to a "1". The repeated inversion of the CNTR1 pin output produces a rectangular waveform with a duty ratio of 50 percent. The initial level of the output is determined by the CNTR1 polarity select bit (bit 6). When this bit is low, the output starts from a high level. When this bit is high, the output starts from a low level.

•Pulse Period Measurement Mode

Count Source:Φ/n (where n is 8, 16, 32, or 64).

This mode measures the period of the event waveform input to the CNTR1 pin.

•CNTR1 Polarity Select Bit (TYM6) = "0"

When the falling edge of an event waveform is detected on the CNTR1 pin, the contents of Timer Y are stored in the temporary register that is assigned the same address as Timer Y. Simultaneously, the value in the Timer Y reload latch is transferred to Timer Y, and Timer Y continues counting down. The falling edge of an event waveform also causes the CNTR1 interrupt request; therefore, the period of the event waveform from falling edge to falling edge is found by reading Timer Y in the CNTR1 interrupt routine. The data read from Timer Y is the data previously stored in its temporary register.

•CNTR1 Polarity Select Bit (TYM6) = "1"

When the rising edge of an event waveform is detected on the CNTR1 pin, the contents of Timer Y are stored in the temporary register that is assigned the same address as Timer Y. Simultaneously, the value in the Timer Y reload latch is transferred to Timer Y, and Timer Y continues counting down.

The rising edge of an event waveform also causes the CNTR1 interrupt request; therefore, the period of the event waveform from rising edge to rising edge is found by reading Timer Y in the CNTR1 interrupt routine. The data read from Timer Y is the data previously stored in its temporary register.

Each time the timer underflows, the Timer Y interrupt request bit is set to a "1", the contents of the timer reload latch are loaded into the timer, and the countdown sequence begins again.

•Event Counter Mode

Count Source: CNTR1

Timer countdown is triggered by input to the CNTR1 pin. Each time a timer underflows, the corresponding timer interrupt request bit is set to a "1", the contents of the timer reload latch are loaded into the timer, and the countdown sequence begins again.

The edge used to clock Timer Y is determined by the CNTR1 polarity select bit (bit 6). When these bits are "0"s, the timers are clocked on the rising edge. When these bits are "1"s, the timers are clocked on the falling edge

•HL Pulse-width Measurement Mode

Count Source:Φ/n (where n is 8, 16, 32, or 64).

This mode continuously measures both the logical high pulse width and the logical low pulse width of an event waveform input to the CNTR1 pin. When the falling (or rising) edge of the event waveform is detected on the CNTR1 pin, the contents of Timer Y are stored in the temporary register that is assigned the same address as Timer Y, regardless of the setting of the CNTR1 polarity select bit. Simultaneously, the value in the Timer Y reload latch is transferred to Timer Y, which continues counting down. The falling or rising edge of an event waveform causes the CNTR1 interrupt request; therefore, the width of the event waveform from the falling or rising edge to rising or falling edge is found by reading Timer Y in the CNTR1 interrupt routine. The data read from Timer Y is the data previously stored in its temporary register.

Each time the timer underflows, the Timer Y interrupt request bit is set to a "1", the contents of the timer reload latch are loaded into the timer, and the countdown sequence begins again.

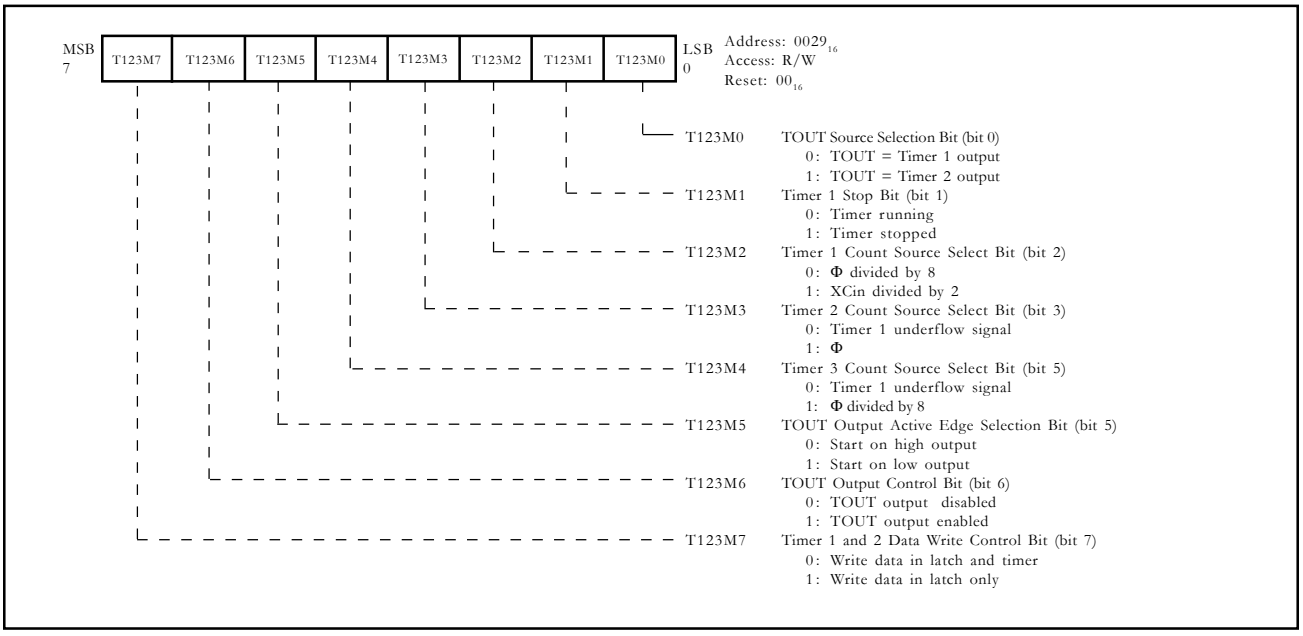


Fig. 1.35. Timer 1, 2, 3 Mode Register (T123M)

1.17.3 Timer 1

Timer 1 is an 8-bit timer with an 8-bit reload latch and has a pulse output option (see Figure 1.35).

T123M7 of Timer123 mode register (T123M) is the Timer 1 and 2 Data Write Control Bit. If T123M7 is “1”, data written to Timer 1 is placed only in the Timer 1 reload latch. The latch value is loaded into Timer 1 after Timer 1 underflows. If T123M7 is “0”, the value written to Timer 1 is placed in Timer 1 and the Timer 1 reload latch. At reset, T123M7 is set to a “0”.

The output signal TOUT is controlled by T123M5 and T123M6. T123M5 controls the polarity of TOUT. Setting the bit T123M5 to “1” causes TOUT to start at a low level, and clearing this bit to “0” causes TOUT to start at a high level. Setting T123M6 to “1” enables TOUT, and clearing T123M6 to “0” disables TOUT.

•Timer Mode

Count Source: Φ/8 or XCin/2

In Timer mode, each time the timer underflows, the corresponding timer interrupt request bit is set to a “1”, the contents of the timer latch are loaded into the timer, and the count down sequence begins again.

•Pulse Output Mode

Count Source: Φ/8 or XCin/2

Timer 1 Pulse Output mode is enabled by setting T123M6 to “1” and T123M0 to a “0”. Each time the Timer 1 underflows, the output of the TOUT pin is inverted, and the corresponding Timer 1 interrupt request bit is set to a “1”. The repeated inversion of the TOUT pin output produces a rectangular waveform with a duty ratio of 50 percent. The initial level of the output is determined by the TOUT polarity select bit (T123M5). When this bit is “0”, the output starts from a high level. When this bit is “1”, the output starts from a low level.

1.17.4 Timer 2

Timer 2 is an 8-bit timer with an 8-bit reload latch (see Figure 1.35).

T123M7 (bit 7 of T123M) is the Timer 1 and 2 Data Write Control Bit. If T123M7 is "1", data written to Timer 2 is placed only in the Timer 2 reload latch (see Figure 1.32). The latch value is loaded into Timer 2 after Timer 2 underflows. If the T123M7 is "0", the value written to Timer 2 is placed in Timer 2 and the Timer 2 reload latch. At reset, T123M2 is set to a "0".

The Timer 2 reload latch value is not affected by a change of the count source. However, because changing the count source may cause an inadvertent countdown of the timer, the timer should be rewritten when the count source is changed.

•Timer Mode

Count Source:

•If T123M3 is "0", the Timer 2 count source is the Timer 1 underflow output.

•If T123M3 is "1", the Timer 2 count source is Φ.

In Timer mode, each time the timer underflows, the corresponding timer interrupt request bit is set to a "1", the contents of the timer latch are loaded into the timer, and the count down sequence begins again.

•Pulse Output Mode

Count Source:

•If T123M3 is "0", the Timer 2 count source is the Timer 1 underflow output.

•If T123M3 is "1", the Timer 2 count source is Φ.

Timer 2 Pulse Output mode is enabled by setting T123M6 to a "1" and T123M0 to a "1". Each time the Timer 2 underflows, the output of the TOUT pin is inverted, and the corresponding Timer 2 interrupt request bit is set to a "1". The repeated inversion of the TOUT pin output produces a rectangular waveform with a duty ratio of 50 percent. The initial level of the output is determined by the TOUT polarity select bit (T123M5). When this bit is "0", the output starts from a high level. When this bit is "1", the output starts from a low level.

1.17.5 Timer 3

Timer 3 is an 8-bit timer with an 8-bit reload latch (see Figure 1.35). The Timer 3 reload latch value is not affected by a change of the count source. Because changing the count source may cause an inadvertent countdown of the timer, the timer should be rewritten whenever the count source is changed.

•Timer Mode

Count Source:

•If T123M4 is "0", the Timer 3 count source is the Timer 1 underflow output.

•If T123M4 is "1", the count source is Φ/8

In Timer mode, each time the timer underflows, the corresponding timer interrupt request bit is set to a "1", the contents of the timer latch are loaded into the timer, and the count down sequence begins again.

Data written to Timer 3 is always placed in Timer 3 and the Timer 3 reload latch.

1.18 SERIAL I/O

The Serial I/O has the following main features:

- Synchronous transmission or reception
- Handshaking via $\overline{\text{SRDY}}$ output signal
- 8-bit character length
- Interrupt after transmission or reception
- Internal Clock

  (When serial I/O synchronous clock select bit is "1", internal clock source divided by 2, 4, 8, 16, 32, 64, 128, 256 can be selected). If bit 1 of SIO Control Reg ister 2 is "0", internal clock source = Φ; if bit 1 of SIO Control Register 2 is "1", internal clock source = SCSGCLK.)

- External Clock

  (When SIO synchronous clock select bit is "1", an external clock input from the SCLK pin is selected).

•An SPI compatible mode in which the TxD and RxD pins function as MOSI and MISO pins, respectively.

•Four (SPI compatible) clock phase and polarity options.

A block diagram of the clock synchronous SIO is shown in Figure 1.36.

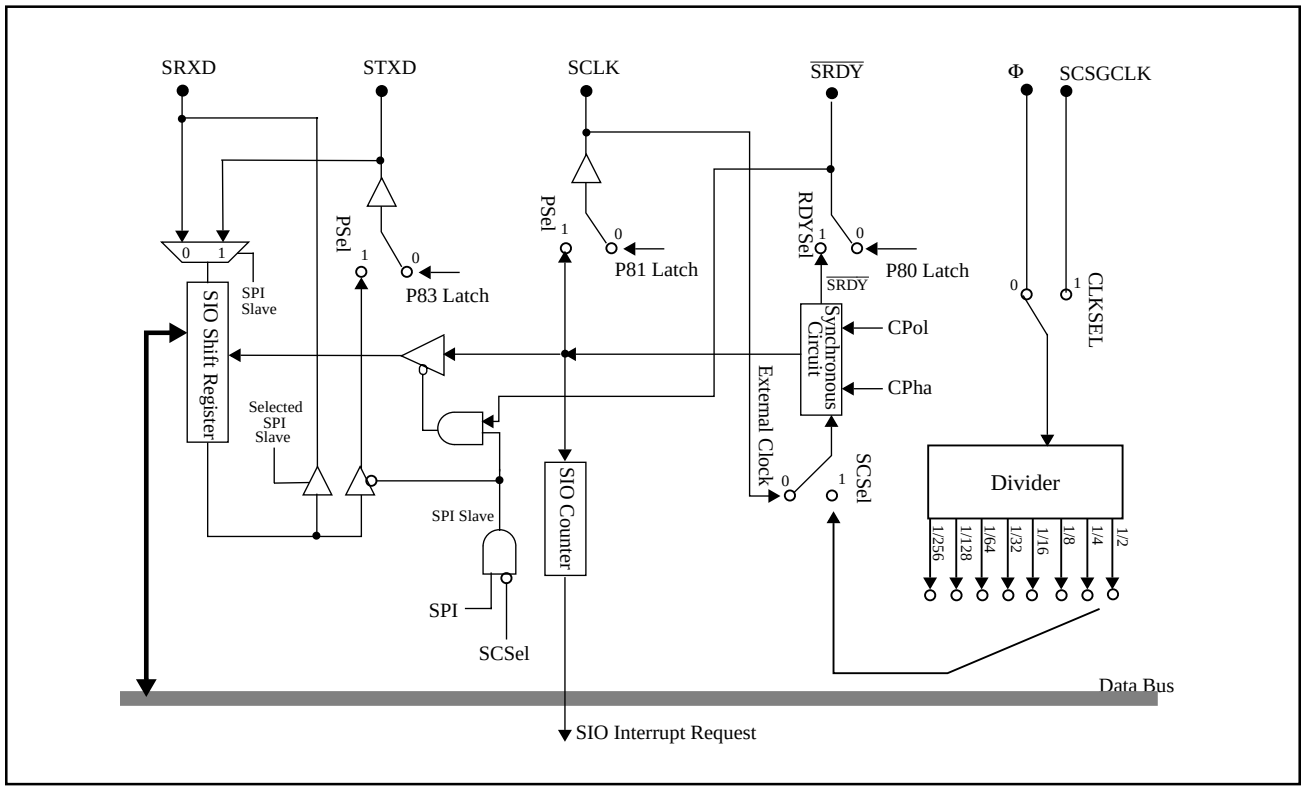


Fig. 1.36. Clock Synchronous SIO Block Diagram

1.18.1 SIO Control Registers (SIOCON)

The Serial I/O Control Register 1 controls various SIO functions such as transfer direction and transfer clock divisor (see Figure 1.37). All of this register's bits can be read from and written to by software. At reset, this register is cleared to 00_{16}.

SIO Control Register 2 determines the transfer clock phase and polarity, and also whether the SIO is to function in SPI compatible mode (see Figure 1.38). All of this register's bits can be read from and written to by software. At reset, this register is set to 18_{16}.

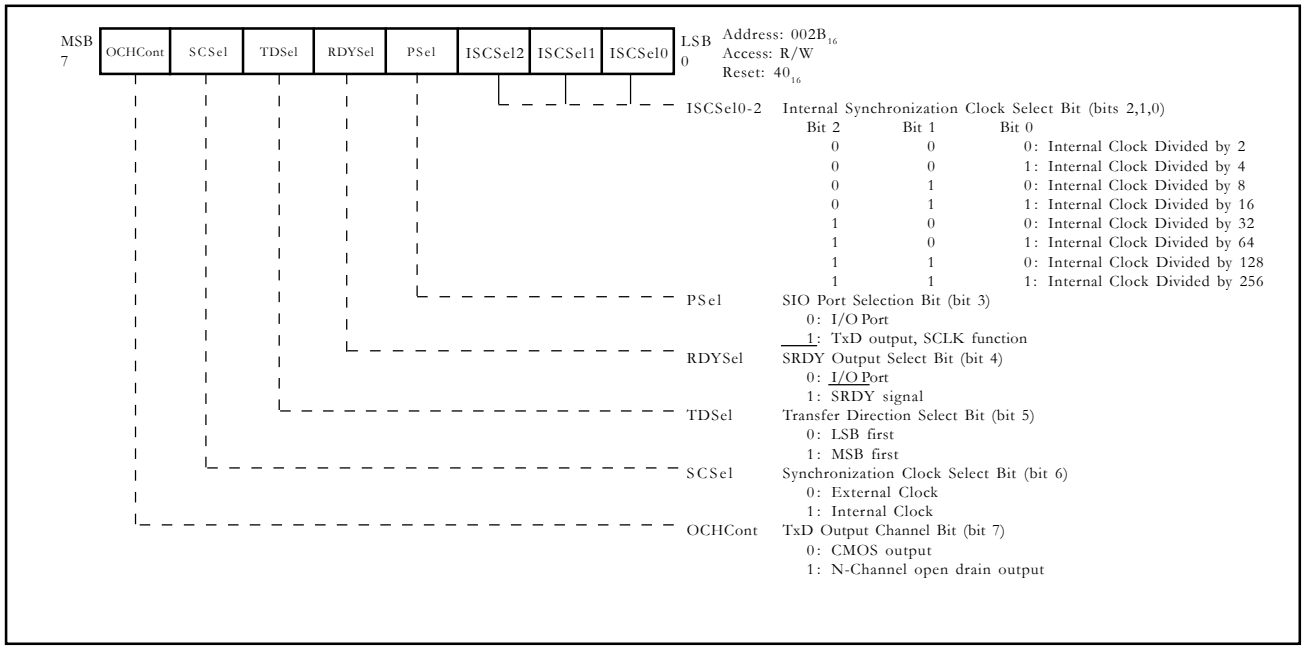


Fig. 1.37. SIO Control Register 1 (SIOCON1)

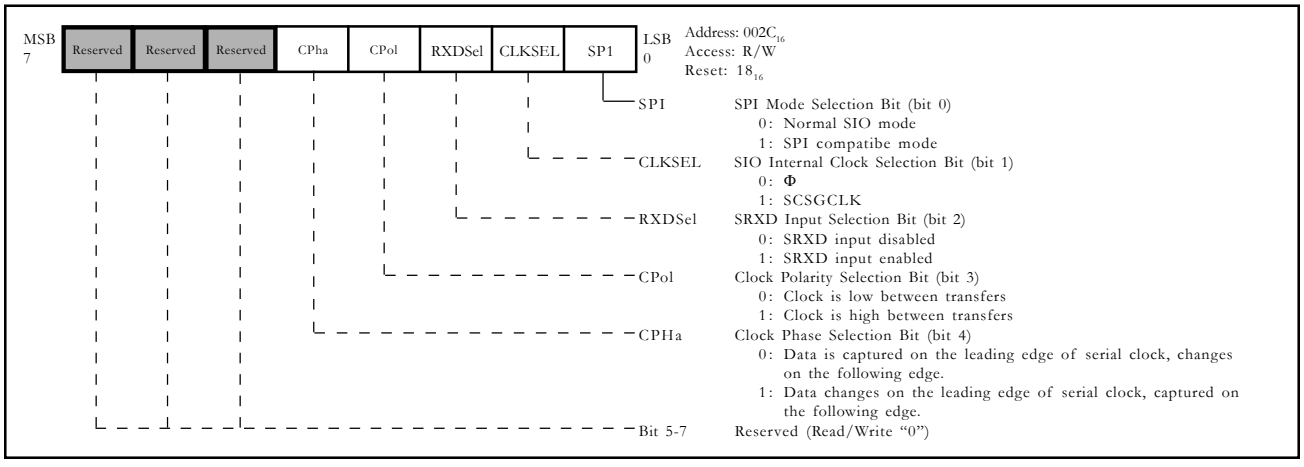
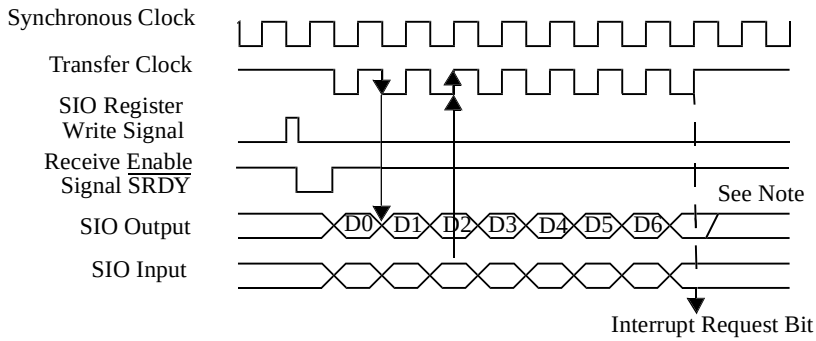


Fig. 1.38. SIO Control Register 2 (SIOCON2)

1.18.2 SIO Normal Operation

An internal clock or an external clock can be selected as the synchronous clock. When the internal clock is chosen, dividers are built in to provide eight different clock selections. The start of a transfer is initiated by a write signal to the SIO shift register (address 002A16). The $\overline{SRDY}$ signal then drops active low. On the negative edge of the transfer clock $\overline{SRDY}$ returns high and the data is transmitted out the STXD pin. Data is latched in from the SRXD pin on the rising edge of the transfer clock. If an internal clock is selected, the STXD pin enters a high-impedance state after an 8-bit transfer is completed. If an external clock is selected, the contents of the serial I/O register continue to be shifted while the send/receive clock is being input. Therefore, the clock needs to be controlled by the external source. Also there is no STXD high-impedance function after data is transferred.

Regardless of whether an internal or external clock is selected, after an 8-bit transfer, the interrupt request bit is set. Figure 1.39 shows the timing for the serial I/O with the LSB-first option selected.



Note: When the internal clock is selected, the TxD pin goes into high-impedance after the data is transferred.

Fig. 1.39. Normal Mode SIO Function Timing (with LSB-First selected)

1.18.3 SPI Compatible Operation

Setting the SPI bit (bit 0 in SIOCON2) puts the SIO in an SPI compatible mode. The internal/external clock select bit (bit 6 in SIOCON1) determines whether the SIO is an SPI master or slave. If internal clock is selected the SIO is a master, and if external clock is selected the SIO is in slave mode.

Entering SPI mode has the following effects on operation:

1. The RxD pin functions as a MISO (Master In/Slave Out) pin. This means that when the SPI is in slave mode transmit data will be output on this pin. In mas ter mode receive data is input on this pin.
2. The TxD pin functions as a MOSI (Master Out/Slave In) pin. When the SPI is in slave mode receive data is input on this pin. In master mode this pin drives the transmit data.
3. The $\overline{\text{SRDY}}$ pin functions as a slave/chip select. If the SPI is in slave mode, this pin functions as a slave se lect input. When configured as an SPI master, the $\overline{\text{SRDY}}$ pin functions as a chip select output.

Figure 1.40 shows the four possible SPI clock-to-data relationships.

The CPol and CPha bits (bits 3 and 4 in SIOCON2) are used to select the format.

1.18.3.1 SPI Slave Mode

When configured as an SPI slave the SIO does not initiate any serial transfers. All transfers are initiated by an external SPI bus master. When the CPha (bit 4 in SIOCON2) is “0” serial transfers begin with the falling edge of the $\overline{\text{SRDY}}$ input. For CPha = “1” serial transfers begin when the SCLK leaves its idle state (the clock idle state is defined by CPol, bit 3 in SIOCON2).

If $\overline{\text{SRDY}}$ is held high, the shift clock is inhibited, SRXD (MISO) is tri-stated, and the shift count is reset. If $\overline{\text{SRDY}}$ is held low, then the shift operation is performed. The $\overline{\text{SRDY}}$ input must be deasserted (brought high) between transfers; this resets the SIO’s internal bit counter.

When the SIO is in SPI slave mode, all transfers are initiated by an external SPI bus master, not by the MCU. Therefore, an application must implement some form of handshaking or synchronization to avoid writing to the SIO shift register during a serial transfer. Writing to the SIO shift register during a transfer will corrupt the transfer in progress.

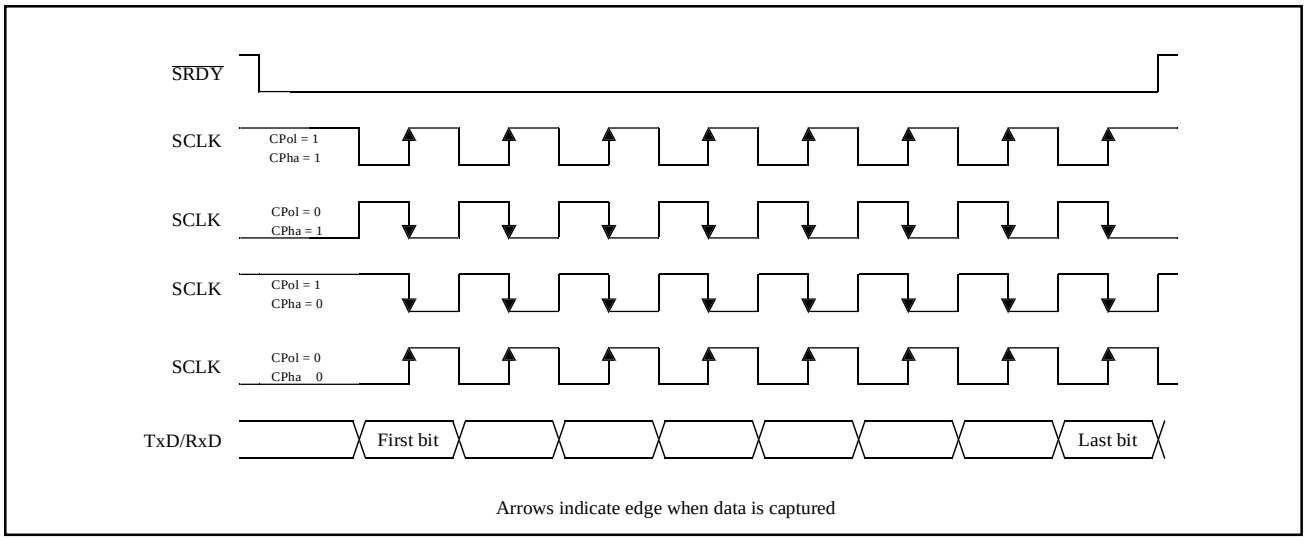


Fig. 1.40. SPI Compatible Transmission Formats

1.19 UART

This chip contains two identical UARTs. Each UART has the following main features:

- Clock selection ..... Φ or SCSGCLK
- Prescaler selection ..... x1/x8/x32/x256 divisions ..... (both Φ and SCSGCLK)
- Baud rate ..... (at Φ = 12MHz) ..... 11.4 bits/second-750 Kbytes/second
- Error detection ..... parity/framing/ ..... overrun/error sum
- Parity ..... odd/even/none
- Stop bits ..... 1 or 2
- Character length ..... 7, 8, or 9 bits
- Transmit/receive buffer ..... 2 stages ..... (double buffering)
- Handshaking ..... Clear-to-Send (CTS) ..... Request-to-Send (RTS)
- Interrupt generation conditions ..... Transmit Buffer ..... Empty or Transmit Complete ..... Receive Buffer ..... Receive Error Sum
- Address ..... mode for multi-receiver environment

The following descriptions apply to both UARTs.

The UART receives parallel data from the core or DMAC, converts it into serial data, and transmits the results to the send data output terminal UTXDx. The UART receives serial data from an external source through the receive data input URXDx, converts it into parallel data, and makes it available to the core or DMAC. The UART can detect parity, overrun, and framing errors in the input stream and report the appropriate status information. A double buffering configuration is used for the UART's transmit and receive operations. This double buffering is accomplished by the use of a transmit buffer and transmit shift register on the transmit side and the receive buffer and receive shift register on the receive side.

The UART supports an address mode for use in a multi-receiver environment where an address is sent before each message to designate which UART or UARTs are to wake-up and receive the message. Figure 1.41 is a block diagram of the UART. It is valid for both UART1 and UART2.

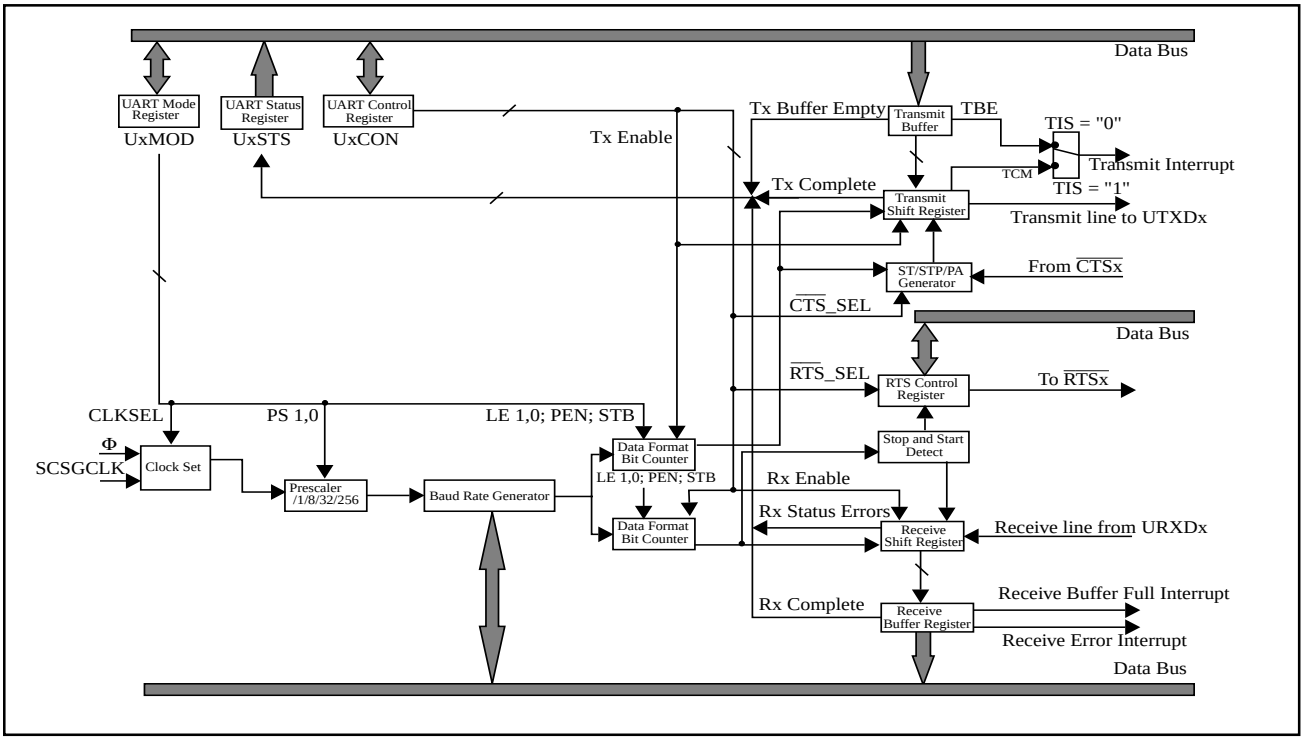


Fig. 1.41. UART Block Diagram

1.19.1 UART Mode Register (UxMOD)

UxMOD defines data formats and selects the clock to be used (see Figure 1.42).

1.19.2 UART Control Register (UxCON)

The UxCON specifies the initialization and enabling of a transmit/receive process (see Figure 1.43). Data can be read from and written to the Control Register.

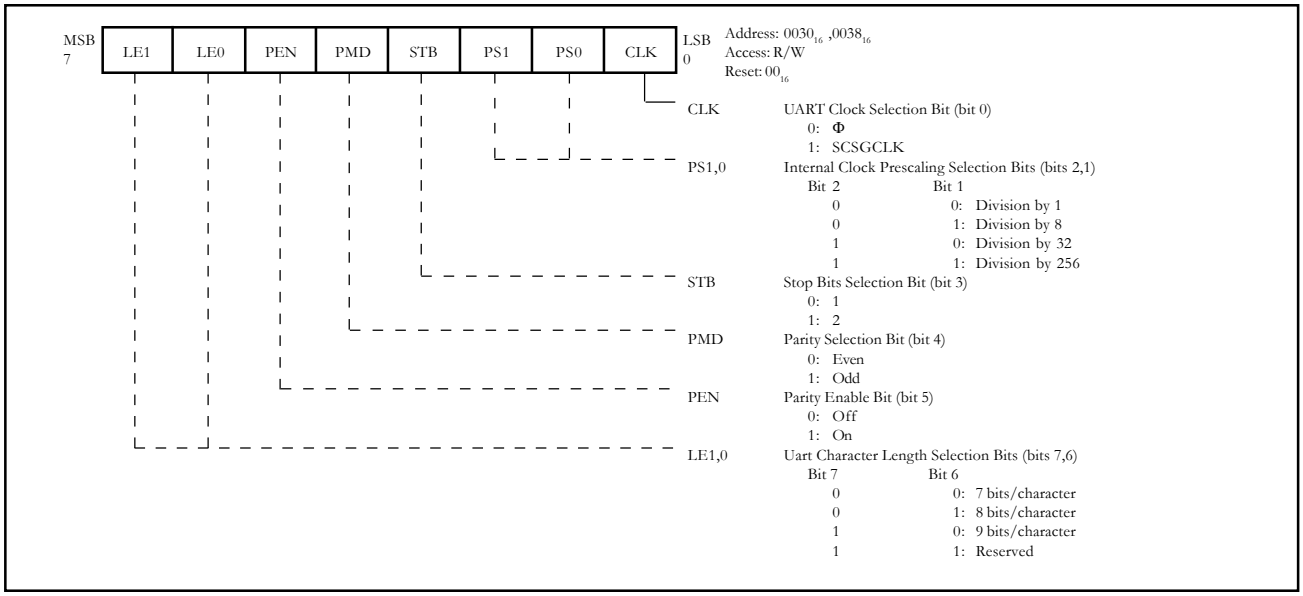


Fig. 1.42. UART Mode Register (U1MOD, U2MOD)

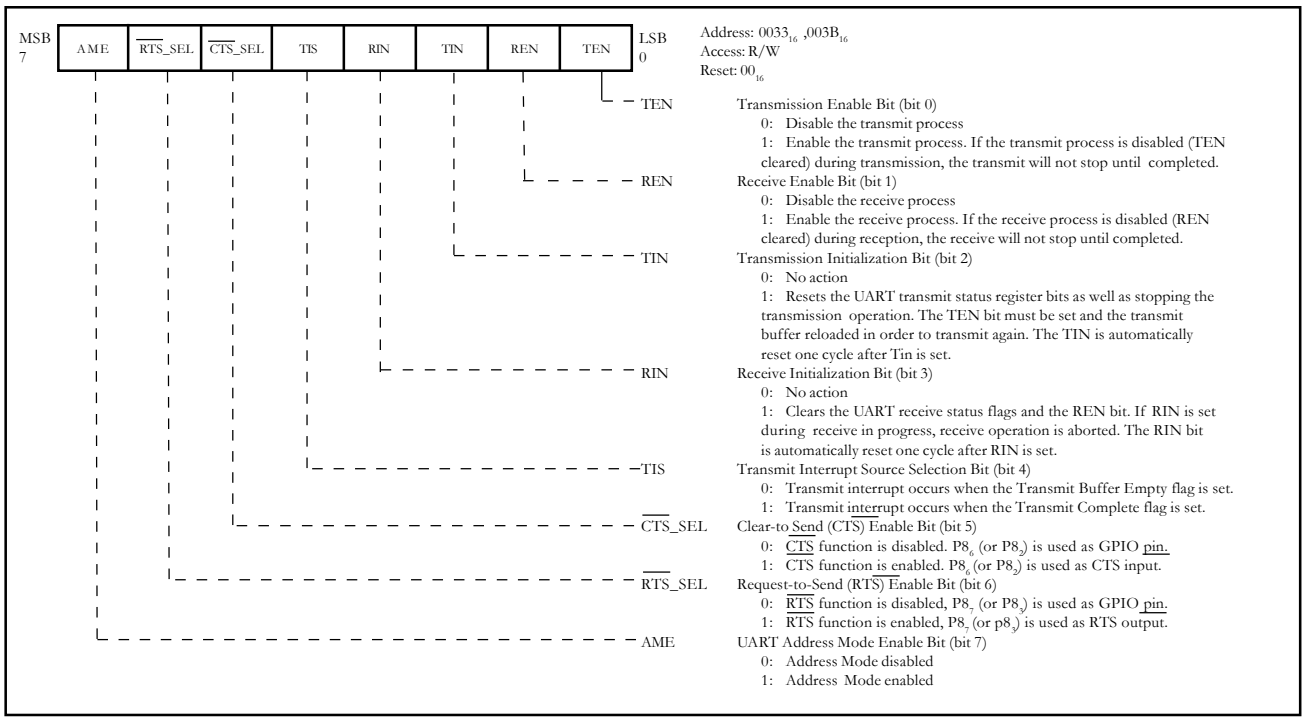


Fig. 1.43. UART Control Register (U1CON, U2CON)

1.19.3 UART Status Register (UxSTS)

The UART Status Register (UxSTS) reflects both the transmit and receive status (see Figure 1.44). The status register is read only. The MSB is always "0" during a read operation. Writing to this register has no effect. Status flags are set and reset under the conditions indicated below. The setting and resetting of the transmit and receive status are not affected by transmit and receive enable flags. The setting and resetting of the receive error flags and receive buffer full flag differs when UART address mode is enabled. These differences are described in section "1.19.7 UART Address Mode".

1.19.3.1 Receive Error Sum Flag

The Receive Error Sum Flag (SER) is set when an overrun, framing, or parity error occurs after completion of a receive operation.

It is reset when the status register is read, the hardware reset is asserted, or the receiver is initialized by setting the Receive Initialization Bit (RIN). If the receive operation completes while the status register is being read, the status information is updated upon completion of the status register read.

1.19.3.2 Receive Overrun Flag

The Receive Overrun Flag (OER) is set if the previous data in the low-order byte of the receive buffer (UxTRB1) is not read before the current receive operation is completed. It is also set if a receive error occurred for the previous data and the status register is not read before the current receive operation is completed. This flag is reset when the status register is read. This flag is also reset when the hardware reset is asserted or the receiver is initialized by RIN. If the receive operation completes while the status register is being read, the status information is updated upon completion of the status register read.

1.19.3.3 Receive Framing Error Flag

The Receive Framing Error Flag (FER) is set when the stop bit of the received data is "0". If the Stop Bit Selection Bit (STB, bit 3 of UxMOD) is set, the flag is set if either of the two stop bits is a "0". This flag is reset when the status register is read, the hardware reset is asserted, or the receiver is initialized by RIN. If the receive operation completes while the status register is being read, the status information is updated upon completion of the status register read.

1.19.3.4 Receive Parity Error Flag

The Receive Parity Error Flag (PER) is set when the parity of received data and the Parity Selection Bit (PMD, bit 4 of UxMOD) are different. It is enabled only if the Parity Enable Bit (PEN, bit 5 of UxMOD) is set.

This flag is reset when the status register is read, the hardware reset is asserted, or the receiver is initialized by RIN. If the receive operation completes while the status register is being read, the status information is updated upon completion of the status register read.

1.19.3.5 Receive Buffer Full Flag

The Receive Buffer Full Flag (RBF) is set when the last stop bit of the data is received. It is not set when a receive error occurs. This flag is reset when the low-order byte of the receive buffer (UxTRB1) is read, the hardware reset is asserted, or the receive process is initialized by RIN. If the receive operation completes while the status register is being read, the status information is updated upon completion of the status register read.

1.19.3.6 Transmission Complete Flag

In the case where no data is contained in the transmit buffer, the Transmission Complete Flag (TCM) is set when the last bit in the transmit shift register is transmitted. In the case where the transmit buffer does contain data, the TCM flag is set when the last bit in the transmit shift register is transmitted if TBE is a "0" or $\overline{CTS}$ handshaking is enabled and $\overline{CTS}$x is "1". The TCM flag is also set when the hardware reset is asserted or when the transmitter is initialized by setting the Transmit Initialization Bit (TIN, bit 2 of UxCON). It is reset when a transmission operation begins.

1.19.3.7 Transmission Buffer Empty Flag

The Transmission Buffer Empty Flag (TBE) is set when the contents of the transmit buffer are loaded into the transmit shift register. The TBE flag is also set when the hardware reset is asserted or when the transmitter is initialized by TIN. It is reset when a write operation is performed to the low-order byte of the transmit buffer (UxTRB1).

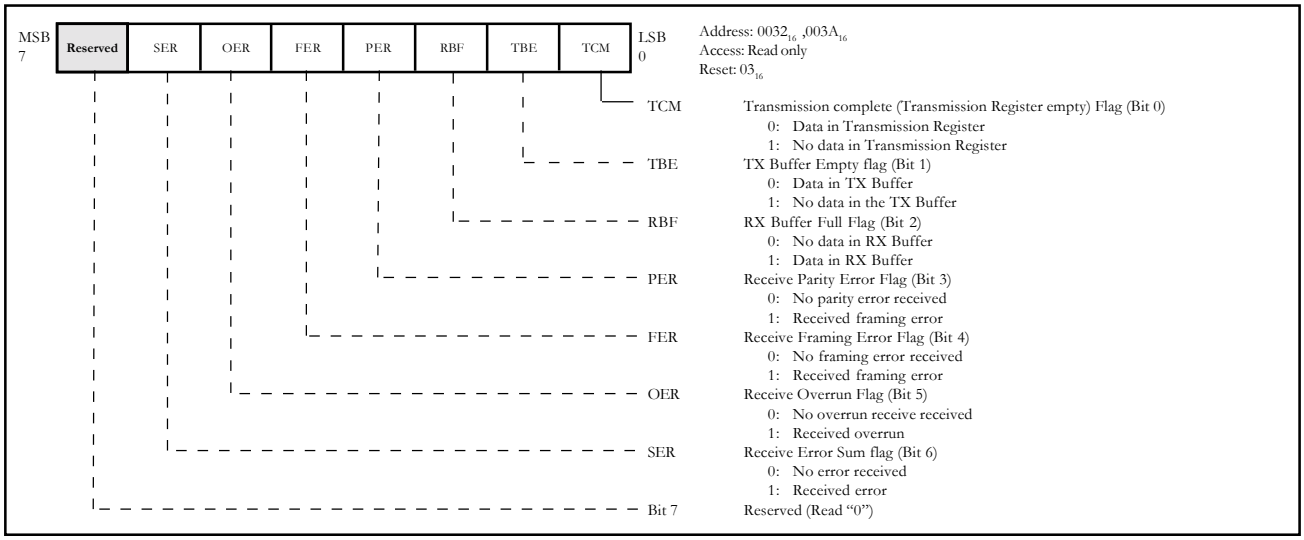


Fig. 1.44. UART Status Register (U1STS, U2STS)

1.19.4 Transmit/Receive Methods

1.19.4.1 Transmit Method

Setup

•Define the baud rate by writing a value from 0-255 into the UxBRG (see Figure 1.45).

•Set the Transmission Initialization Bit (TIN, bit 2 of UxCON), to "1". This will reset the transmit status to a value of 03₁₆.

•Select the interrupt source to be either TBE or TCM by clearing or setting the Transmit Interrupt Source Selection Bit (TIS, bit 4 of UxCON).

• Configure the data format and clock selection by writing the appropriate value to UxMOD.

• Set the Clear-To-Send Enable Bit ($\overline{\text{CTS}}$_SEL, bit 5 of UxCON), if $\overline{\text{CTS}}$ handshaking will be used.

•Set the Transmit Enable Bit (TEN, bit 0 of UxCON), to "1".

Operation

•When data is written to the low-order byte of the transmit buffer (UxTRB1), TBE is cleared to "0". If 9-bit character length has been selected, the high-order byte of the transmit buffer (UxTRB2) should be written before the low-order byte (UxTRB1).

•If no data is being shifted out of the transmit shift register and $\overline{\text{CTS}}$ handshaking is disabled, the data written to the transmit buffer is transferred to the transmit shift register and the TCM flag in UxSTS is cleared to a "0". In addition, the TBE flag is set to a "1", signaling that the next byte of data can be written to the transmit buffer. If $\overline{\text{CTS}}$ handshaking is enabled, the operation described above does not take place until $\overline{\text{CTS}}$x is brought low.

•Data from the transmit shift register is transmitted one bit at a time beginning with the start bit and ending with the stop bit. Note that the LSB is transmitted first.

•If the TEN bit is cleared to a "0" while data is still being transmitted, the transmitter will continue until the last bit is sent. This is also the case when $\overline{\text{CTS}}$ handshaking is enabled and $\overline{\text{CTS}}$x is brought back high during transmission.

•When the last bit is transmitted, the TCM flag is set to a "1" if the transmit buffer is empty, TEN is a "0", or $\overline{\text{CTS}}$ handshaking is enabled and $\overline{\text{CTS}}$x is "1". If the transmit buffer is not empty, TEN is a "1", and $\overline{\text{CTS}}$ handshaking is disabled or $\overline{\text{CTS}}$ handshaking is enabled and $\overline{\text{CTS}}$x is low, the TCM flag is not set because transfer of the contents of the transmit buffer to the transmit shift register occurs immediately.

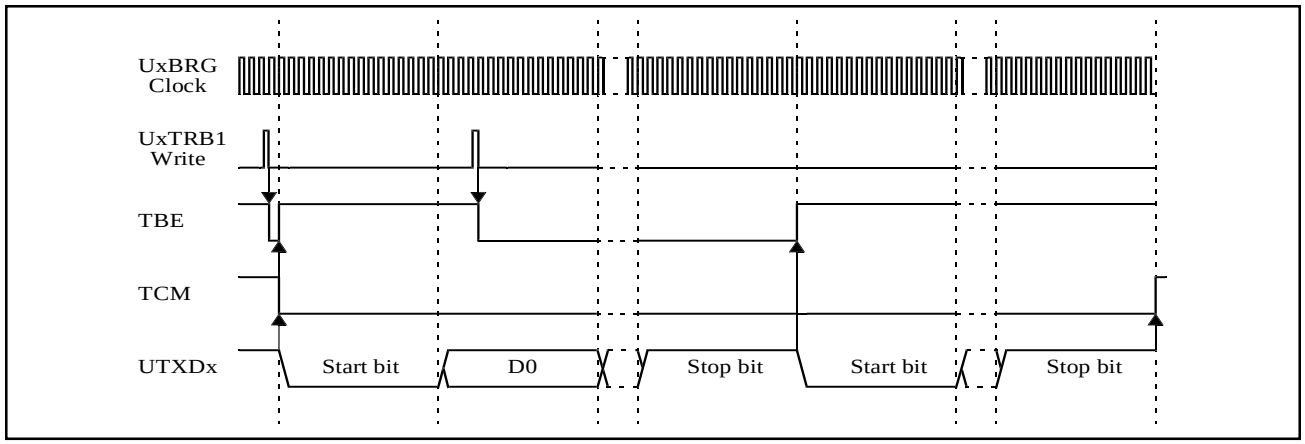


Fig. 1.45. UART Transmit Operation Waveform

1.19.4.2 Receive Method

Set up

•Define the baud rate by writing a value from 0-255 into UxBRG.

•Set the Receive Initialization Bit (RIN, bit 3 in the UxCON), to "1".

•Configure the data format and the clock selection by writing the appropriate value to UxMOD.

•Set the Request-To-Send Enable Bit ($\overline{\text{RTS}}$_SEL, bit 6 of UxCON), if $\overline{\text{RTS}}$ handshaking will be used.

•Set the Receive Enable Bit (REN, bit 1 in the UxCON), to "1".

Operation

•When a falling edge is detected on the URXDx pin, the value on the pin is sampled at the basic clock rate, which is 16 times faster than the baud rate. If the pin is low for at least two cycles of the basic clock, the start bit is detected. Sampling is again performed three times in the approximate middle of the start bit. If two or more of the samples are low, the start bit is deemed valid. If two or more of the samples are not low, the start bit is invalidated and the UART again begins waiting for a falling edge on the URXDx pin.

•Once a valid start bit has been detected, input data received through the URXDx pin is read one bit at a time, LSB first, into the receive shift register. As is the case with the start bit, three samples are taken in the approximate middle of each data bit, the parity bit, and the stop bit(s). If two or more of the samples are low, a "0" is latched, and if two or more of the samples are high, a "0" is latched.

•When the number of bits specified by the data format has been received and the last stop bit is detected, the contents of the receive shift register are transferred to the receive buffer and the Receive Buffer Full Flag in the UxSTS is set to a "1", if a receive error has not occurred (see Figure 1.46). The RBF interrupt request is also generated at this time if a receive error has not occurred. However, if a receive error did occur, the appropriate error flags are set and the Receive Error Sum (SER) interrupt request is generated at this time.

•When the low-order byte of the receive buffer (UxTRB1) is read, the Receive Buffer Full Flag is cleared, and the receive buffer is now ready for the next byte. If 9-bit character length has been selected, the high-order byte of the receive buffer (UxTRB2) should be read before reading the low-order byte (UxTRB1).

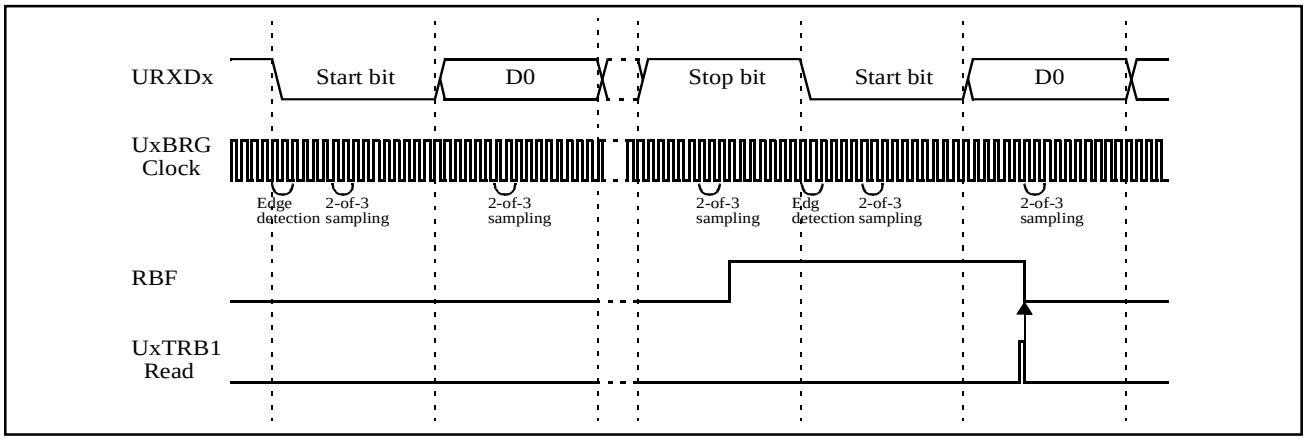


Fig. 1.46. UART Receive Operation Waveforms

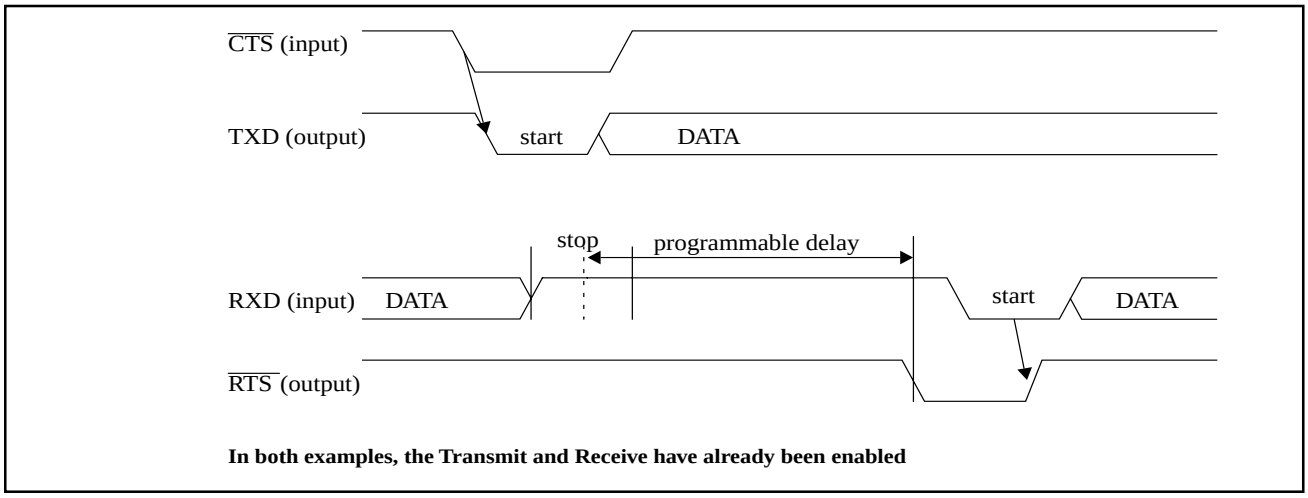


Fig. 1.47. $\overline{\text{CTS}}$x and $\overline{\text{RTS}}$x Timing Examples

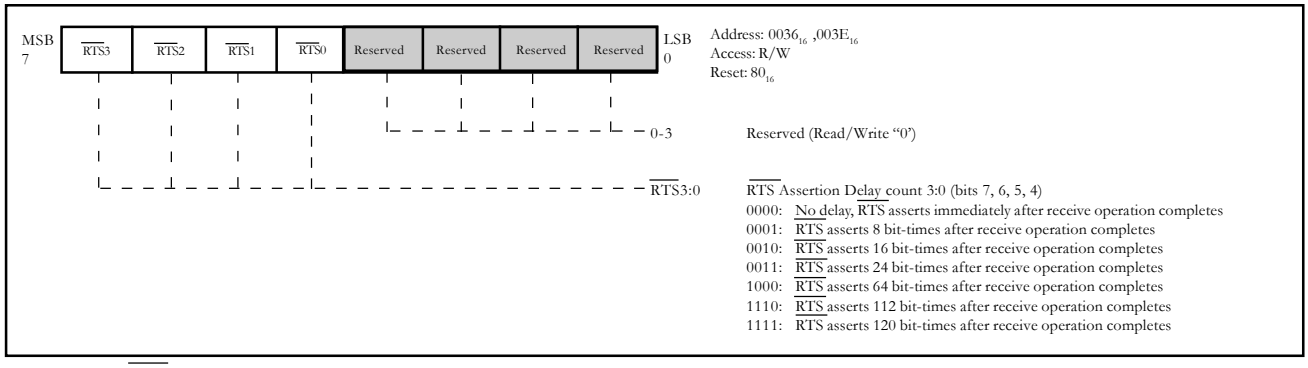


Fig. 1.48. Ux$\overline{\text{RTS}}$C Register

1.19.5 Interrupts

The transmit and receive interrupts are generated under the conditions described below. The generation of the receive interrupts differs when UART Address mode is enabled.

1.19.5.1 Transmit interrupts

The UART generates a Transmit interrupt to the CPU core. The source of the Transmit interrupt is selectable by setting TIS.

- If TIS = "0", the Transmit interrupt is generated when the transmit buffer register becomes empty (that is, when TBE flag set).
- If TIS = "1", the Transmit interrupt is generated after the last bit is sent out of the transmit shift register and no data has been written to the transmit buffer or $\overline{CTS}$ handshaking is enabled and $\overline{CTS}$x is high (that is, when TCM flag set).

1.19.5.2 Receive Interrupts

The UART generates the Receive Buffer Full (RBF) and Receive Error Sum (SER) interrupts to the CPU core when receiving.

- The RBF interrupt is generated when a receive operation completes and a receive error is not generated.
- The SER interrupt is generated when an overrun, framing or parity error occurs.

1.19.6 Clear-to Send ($\overline{CTS}$x) and Request-to-Send ($\overline{RTS}$x) Signals

The UART, as a transmitter, can be configured to recognize the Clear-to-Send ($\overline{CTS}$x) input as a handshaking signal. As a receiver, the UART can be configured to generate the Request-to-Send ($\overline{RTS}$x) handshaking signal.

1.19.6.1 Clear-to-Send ($\overline{CTS}$x) Input

$\overline{CTS}$ handshaking is enabled by setting the Clear-to-Send Enable Bit ($\overline{CTS}$_SEL, bit 5 of UxCON) to a "1". If $\overline{CTS}$ handshaking is enabled, when TEN is a "1" and the low-order byte of the transmit buffer (UxTRB1) is loaded, the UART begins the transmission process when the $\overline{CTS}$x pin is asserted (low input). After beginning a send operation, the UART does not stop sending until the transmission is completed, even if $\overline{CTS}$x is deasserted (high input). If TEN is cleared to "0", the UART will not stop transmitting and the port pins will remain under the control of the UART until the end of the transmission. If $\overline{CTS}$ handshaking is disabled and TEN is a "1", the UART begins the transmission process as soon as data is available in the low-order byte of the transmit buffer (UxTRB1). Figure 1.47 shows a timing example for $\overline{CTS}$x.

1.19.6.2 Request-to-Send ($\overline{RTS}$x) Output

$\overline{RTS}$ handshaking is enabled by setting the Request-to-Send Enable Bit ($\overline{RTS}$_SEL, bit 6 of UxCON) to a "1". When $\overline{RTS}$ handshaking is enabled, the UART drives the $\overline{RTS}$x output low or high based on the following conditions:

- Assertion conditions (driven low):
  - The Receive Enable Bit (REN) is set to a "1".
  - Receive operation has completed with the reception of the last stop bit, REN is still a "1", and the programmable assertion delay has expired.
- De-assertion conditions (driven high):
  - A valid start bit is detected and REN is a "1".
  - REN is cleared to a "0" before a receive operation is in progress.
  - Receive operation has completed and REN is a "0".
  - UART Receiver is initialized (RIN is set to a "1").

The delay time from the reception of the last stop bit to the re-assertion of $\overline{RTS}$x is programmable. The amount of delay is selected by setting the $\overline{RTS}$ Assertion Delay Count Bits ($\overline{RTS}$ 3~0, bits 3 to 0 of UxRTSC) (see Figure 1.48). The time can be from no delay to 120 bit-times, with the delay beginning from the middle of the last stop bit. If a start bit is detected before the assertion delay has expired, the delay countdown is stopped and the $\overline{RTS}$x pin remains high. A full assertion delay countdown will begin again once the last stop bit of the incoming data has been received. See Figure 1.47 for a timing example for $\overline{RTS}$x.

1.19.7 UART Address Mode

The UART address mode is intended for use in a multi-receiver environment where an address is sent before each message to designate which UART or UARTs are to wake-up and receive the message. An address is identified by the MSB of the incoming data byte being a “1”. The bit is “0” for non-address data. UART address mode can be used in either 8-bit or 9-bit character length mode. The character length is chosen by writing the appropriate values to the UART Character Length Selection Bits (LE1,0).

UART address mode is enabled by setting the UART Address Mode Enable Bit (AME) to “1”. When UART address mode is enabled, the MSB of a newly received byte of data (that is either 8 or 9 bits in length) is examined if a valid stop bit is detected and a parity error has not occurred (if parity is enabled). If the MSB is “1”, then the receive buffer full interrupt and flag are set and AME is automatically cleared, disabling UART address mode. If the MSB is “0”, then the receive buffer full interrupt is not set. However, the RBF flag is still set for this case. If a valid stop bit is not detected or a parity error has occurred, neither the receive buffer full flag nor interrupt is set and the MSB of the data is not examined. Instead, either the framing error or parity error flag is set, the error sum flag is set, and the error sum interrupt is set.

While in UART address mode, the generation of overrun errors is disabled after the first byte of data is received. Therefore, when non-address data is received without errors while in the UART address mode, it is not necessary to read the UART receive buffer prior to the reception of the next byte of data. Also, if a framing or parity error occurs while in UART address mode, it is not necessary to read the UxSTS prior to the reception of the next byte of data. However, an overrun error will occur if an address byte is received and the UART receive buffer is not read before a new byte of data is received. This is the case because the UART address mode was automatically disabled when the address byte was received. Also, an overrun error will occur for the first byte received after UART address mode is enabled if the preceding byte received did not generate an error and the UART receive buffer was not read, or the preceding byte did generate an error and UxSTS was not read.

1.20 SPECIAL COUNT SOURCE GENERATOR

This device has a built-in special count source generator. It cons ists of two 8-bit timers: SCSG1, and SCSG2 (see Figure 1.49). The contents of the timer latch, corresponding to each timer, determine the divide ratio. The timers can be written to at any time. The output of the special count source generator can be a clock source for Timer X, SIO and the two UARTs.

1.20.1 SCSG Operation

The SCSG1 and SCSG2 are both down count timers. When the count of a timer reaches 00_{16}, an underflow occurs at the next count pulse and the contents of the corresponding timer reload latch are loaded into the timer. For the count operation for SCSG1 with the Data Write Mode set to write to the latch only (see Figure 1.50).

A memory map and the initial values after reset of the timers and timer reload latches are detailed above. The divide ratio of each timer is given by 1/(n + 1), where n is the value written to the timer. The output of the first timer (SCSG1) is effectively ANDed with the original clock (Φ) to provide a count source for the second timer (SCSG2). This results in a count source of n/(n + 1) being fed to SCSG2.

The output of the SCSG is a clock, SCSGCLK. The frequency is calculated as follows:

$$SCSGCLK = \phi \bullet \frac{SCSG1}{SCSC1+1} \bullet \frac{1}{SCSG2+1}$$

where SCSG1 is the value written to SCSG1 and SCSG2 is the value written to SCSG2.

See Figure 1.51 for the Special Count Source Mode Register.

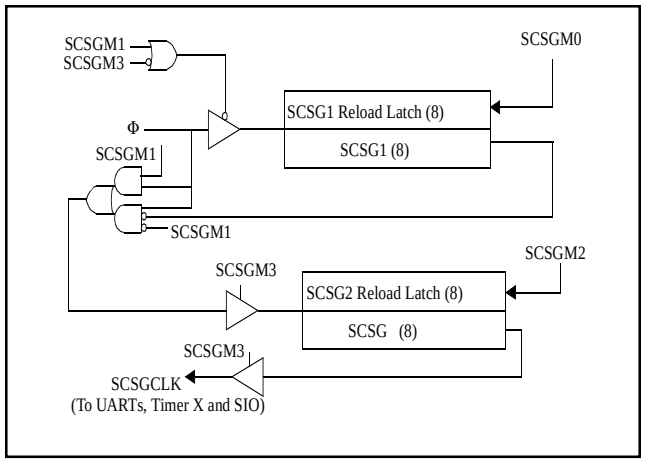


Fig. 1.49. SCSG Block Diagram

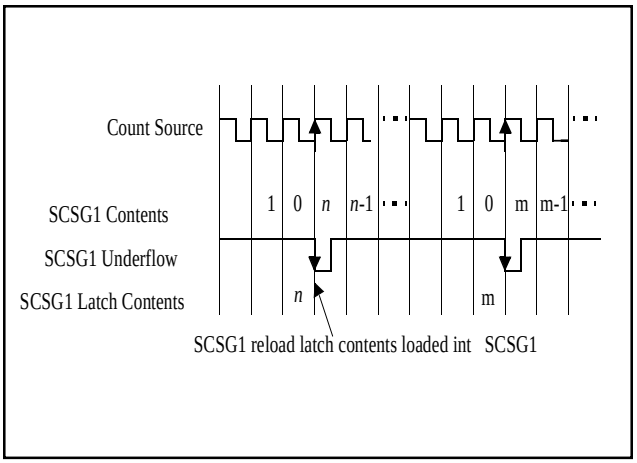


Fig. 1.50. Timer Count Operation for SCSG1

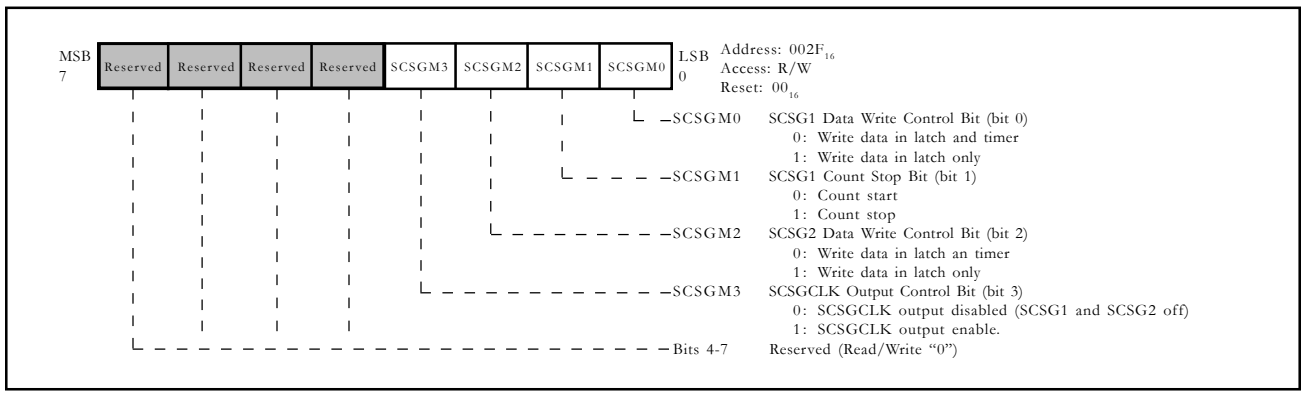


Fig. 1.51. Special Count Source Mode Register (SCSM)

1.21 UNIVERSAL SERIAL BUS

The Universal Serial Bus (USB) has the following features:

•Complete USB Specification (version 1.1) Compatibility
•Error Handling capabilities
•FIFOs:
  •Endpoint 0: IN 16-byte OUT 16-byte
  •Endpoint 1: IN 512-byte OUT 800-byte
  •Endpoint 2: IN 32-byte OUT 32-byte
  •Endpoint 3: IN 16-byte OUT 16-byte
  •Endpoint 4: IN 16-byte OUT 16-byte
•Five independent IN endpoints
•Five independent OUT endpoints
•Complete Device Configuration
•Supports All Device Commands
•Supports Full-Speed Functions
•Support of All USB Transfer Types:
  •Isochronous
  •Bulk
  •Control
  •Interrupt
•Suspend/Resume Operation
•On-chip USB Transceiver with voltage converter
•Start-of-frame interrupt and output pin

1.21.1 USB Function Control Unit (USBFCU)

The implementation of the USB by this device is accomplished chiefly through the device's USB Function Control Unit. The Function Control Unit's overall purpose is to handle the USB packet protocol layer. The Function Control Unit notifies the MCU that a valid token has been received. When this occurs, the data portion of the token is routed to the appropriate FIFO. The MCU transfers the data to, or from, the host by interacting with that endpoint's FIFO and CSR register (see Figure 1.51).

The USB Function Control Unit is composed of five sections:

•Serial Interface Engine (SIE)
•Generic Function Interface (GFI)
•Serial Engine Interface Unit (SIU)
•Microcontroller Interface (MCI)
•USB Transceiver

1.21.1.1 Serial Interface Engine

The SIE interfaces to the USB serial data and handles deserialization/serialization of data, NRZI encoding/decoding, clock extraction, CRC generation and checking, bit stuffing, and other specifications pertaining to the USB protocol such as handling inter-packet time-outs and PID decoding 1.

1.21.1.2 Generic Function Interface

The GFI handles all USB standard requests from the host through the control endpoint (endpoint 0), and handles Bulk, Isochronous and Interrupt transfers through Endpoints 1-4. The GFI handles read pointer reversal for re-transmission of the current data set; write pointer reversal for re-reception of the last data set, and data toggle synchronization.

1.21.1.3 Serial Engine Interface Unit

The SIU block decodes the Address and Endpoint fields from the USB host.

1.21.1.4 Microcontroller Interface Unit

The MCI block handles the microcontroller interface and performs address decoding and synchronization of control signals.

1.21.1.5 USB Transceiver

The USB transceiver, designed to interface with the physical layer of the USB, is compliant with the USB Specification (version 1.1) for high speed devices. It consists of two 6-ohm drivers, a receiver, and Schmitt triggers for single-ended receive signals.

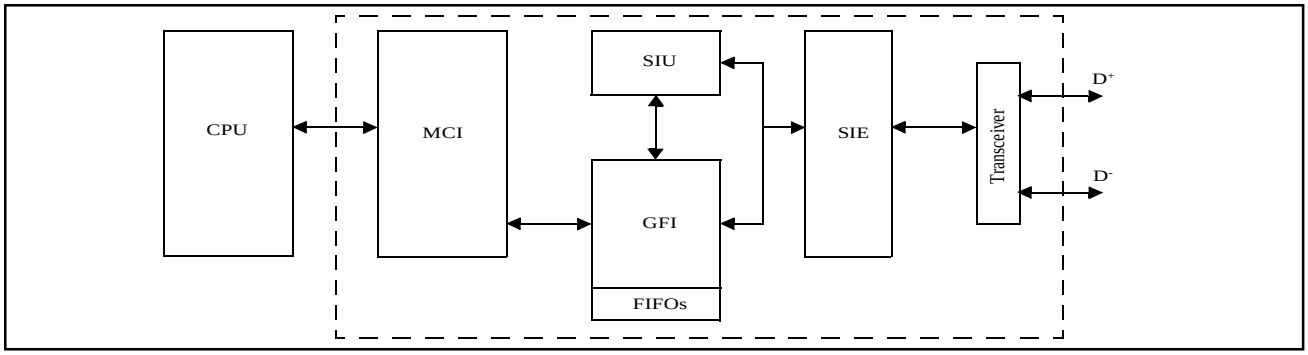


Fig. 1.51. USB Function Control Unit Block Diagram

1.21.2 USB Interrupts

There are two types of USB interrupts in this device.

USB function (including overrun/underrun, reset, suspend and resume) interrupt, which is used to control the flow of data.

The second type is start-of-frame ($\overline{SOF}$) interrupt, which is used to monitor the transfer of isochronous (ISO) data.

1.21.2.1 USB Function Interrupts

Endpoints 1-4 each have two interrupt status flags associated with them to control data transfer or to report a STALL/UNDER_RUN/OVER_RUN condition. The USB Endpoint x Out Interrupt Status Flag is set when the USB FCU successfully receives a packet of data, or the USB FCU sets the FORCE_STALL bit or the OVER_RUN bit of the Endpoint x OUT CSR. The USB Endpoint x In Interrupt Status Flag is set when the USB FCU successfully sends a packet of data or sets the UNDER_RUN bit of the Endpoint x IN CSR. Endpoint 0 (the control endpoint) has one interrupt status bit associated with it to control data transfer or report a STALL condition. The USB Endpoint 0 Interrupt Status Flag is set when the USB FCU successfully receives/sends a packet of data, sets the SETUP_END bit or the FORCE_STALL bit, or clears the DATA_END bit in the Endpoint 0 IN CSR. Each endpoint interrupt is enabled by setting the corresponding bit in the USB Interrupt Enable Register 1 and 2 (see Figure 1.57 and Figure 1.58). The USB Interrupt Status Register 1 and 2, shown in Figure 1.55 and Figure 1.56, are used to indicate pending interrupts for a given endpoint. The USB FCU sets the interrupt status bits. The CPU writes a “1” to clear the corresponding status bit. By writing back the same value it read, the CPU will clear all the existing interrupts. The CPU must read then write both status registers, writing status register 1 first and status register 2 second to guarantee proper operation.

The Suspend Signaling Interrupt Status Flag is set if the USB FCU does not detect any bus activity on D+/D- for at least 3ms. The Resume Signaling Interrupt Status Flag is set when a USB FCU is in the suspend state and detects non-idle signaling on D+/D-. There is an interrupt enable bit for the suspend interrupt (bit 7 of Interrupt Enable Register 2), but not one for the resume interrupt. The resume interrupt is always enabled.

The USB Reset Interrupt Status Flag is set if the USB FCU sees a SE0 present on D+/D- for at least 2.5ms. When this bit is set, all USB internal registers (except for this bit) are reset to their default values. This bit is cleared by the CPU writing a “1” to it. When the CPU detects a USB reset interrupt, it needs to re-initialize the USB FCU in order for it to accept packets from the host. The USB reset interrupt is always enabled.

The Overrun/Underrun Interrupt Status Flag is set (applicable to endpoints used for isochronous data transfer) when an overrun condition occurs in an endpoint (CPU is too slow to unload the data from the FIFO), or when an underrun condition occurs in an endpoint (CPU is too slow to load the data to the FIFO).

The USB Function Interrupt (sum of all individual function interrupts) is enabled by setting bit 0 of Interrupt Control Register A (ICONA) to a “1”.

1.21.2.2 USB $\overline{SOF}$ Interrupt

The USB $\overline{SOF}$ (Start-Of-Frame) interrupt is used to control the transfer of isochronous data. The USB FCU generates a start-of-frame interrupt when a start-of-frame packet is received. The USB $\overline{SOF}$ interrupt is enabled by setting bit 1 of ICONA to a “1”.

1.21.3 USB Endpoint FIFOs

The USB FCU has an IN (transmit) FIFO and an OUT (receive) FIFO for each endpoint. Each endpoint (except endpoint 0) can be configured to support both single packet mode (only a single data packet is allowed to reside in the endpoint’s FIFO) or dual packet mode (up to two data packets are allowed to reside in the endpoint’s FIFO), which provides support for back-to-back transmission or back-to-back reception. The mode configuration is automatically set by the MAXP value. When MAXP > 1/2 of the endpoint’s FIFO size, single packet mode is set. When MAXP <= 1/2 of the endpoint’s FIFO size, dual packet mode is set.

Throughout this specification, the terms “IN FIFO” and “OUT FIFO” refer to the FIFOs associated with the current endpoint as specified by the Endpoint Index Register.

In the event of a bad transmission/reception, the USB FCU handles all the read/write pointer reversal and data set management tasks when it is applicable.

1.21.3.1 IN (Transmit) FIFOs

The CPU/DMA writes data to the endpoint's IN FIFO location specified by the FIFO write pointer, which automatically increments by "1" after a write. The CPU/DMA should only write data to the IN FIFO if the IN_PKT_RDY bit of the IN CSR is a "0".

•Endpoint 0 IN FIFO Operation:

The CPU writes a "1" to the IN_PKT_RDY bit after it finishes writing a packet of data to the IN FIFO. The USB FCU clears the IN_PKT_RDY bit after the packet has been successfully transmitted to the host (ACK is received from the host) or the SETUP_END bit of the IN CSR is set to a "1".

•Endpoint 1-4 IN FIFO Operation when AUTO_SET (bit 7 of IN CSR) = "0":

MAXP > 1/2 of the IN FIFO size: The CPU writes a "1" to the IN_PKT_RDY bit after the CPU/DMAC finishes writing a packet of data to the IN FIFO. The USB FCU clears the IN_PKT_RDY bit after the packet has been successfully transmitted to the host (ACK is received from the host).

MAXP <= 1/2 of the IN FIFO size: The CPU writes a "1" to the IN_PKT_RDY bit after the CPU/DMAC finishes writing a packet of data to the IN FIFO. The USB FCU clears the IN_PKT_RDY bit as soon as the IN FIFO is ready to accept another data packet. (The FIFO can hold up to two data packets at the same time in this configuration for back-to-back transmission). Since the set and the clear operations could be as fast as 83ns (one 12MHz clock period) apart from each other, the set may be transparent to the user.

•Endpoint 1-4 IN FIFO Operation when AUTO_SET (bit 7 of IN CSR) = "1":

MAXP > 1/2 of the IN FIFO size: When the number of bytes of data equal to the MAXP (maximum packet size) is written to the IN FIFO by the CPU/DMAC, the USB FCU sets the IN_PKT_RDY bit to a "1" automatically. The USB FCU clears the IN_PKT_RDY bit after the packet has been successfully transmitted to the host (ACK is received from the host).

MAXP <= 1/2 of the IN FIFO size: When the number of bytes of data equal to the MAXP (maximum packet size) is written to the IN FIFO by the CPU/DMAC, the USB FCU sets the IN_PKT_RDY bit to a "1" automatically. The USB FCU clears the IN_PKT_RDY bit as soon as the IN FIFO is ready to accept another data packet. (The FIFO can hold up to two data packets at the same time in this configuration for back-to-back transmission). Since the set and the clear operations could be as fast as 83ns (one 12MHz clock period) apart from each other, the set may be transparent to the user.

A software or a hardware flush causes the USB FCU to act as if a packet has been successfully transmitted out to the host. If there is one packet in the IN FIFO, a flush will cause the IN FIFO to be empty. If there are two packets in the IN FIFO, a flush will cause the older packet to be flushed out from the IN FIFO. A Flush will also update the IN FIFO status bits IN_PKT_RDY and TX_NOT_EMPTY.

The status of endpoint 1-4 IN FIFOs for both of the above cases can be obtained from the IN CSR of the corresponding IN FIFO as shown in Table 1.8.

Table 1.8. Endpoint 1_4 IN FIFO Status

IN_PKT_RDY	TX_NOT_EMPTY	TX FIFO Status
0	0	No Data packet in TX FIFO
0	1	One data packet in TX FIFO if MAXP <= 1/2 of the FIFO size Invalid if MAXP > 1/2 of the FIFO size/
1	0	Invalid
1	1	Two data packets in TX FIFO if MAXP <= 1/2 of the FIFO size OR One data packet in TX FIFO if MAXP > 1/2 of the FIFO size

•Interrupt Endpoints:

Any endpoint can be used for interrupt transfers. For normal interrupt transfers, the interrupt transactions behave the same as bulk transactions, i.e.; no special setting is required. The IN endpoints may also be used to communicate rate feedback information for certain types of isochronous functions. This is done by setting the INTPT bit in the IN CSR register of the corresponding endpoint. When the INTPT bit is set, the data toggle bits will be changed after each packet is sent to the host without regard to the presence or type of handshake packet.

The following outlines the operation sequence for an IN endpoint used to communicate rate feedback information:

1. Set MAXP > 1/2 of the endpoint's FIFO size
2. Set the INTPT bit of the IN CSR
3. Flush the old data in the FIFO
4. Load interrupt status information and set the IN_PKT_RDY bit in the IN CSR
5. Repeat steps 3 and 4 for all subsequent interrupt status updates.

In real applications, if an interrupt endpoint is used for rate feedback, the function always has data to send back to the host, even if that data conveys that everything is 'fine'. Therefore the device never NAKs an IN token from the host. The device always sends out the data in the FIFO in response to an IN token irrespective of the IN_PKT_RDY bit.

1.21.3.2 OUT (Receive) FIFOs

The USB FCU writes data to the endpoint's OUT FIFO location specified by the FIFO write pointer, which automatically increments by one after a write. When the USB FCU has successfully received a data packet, it sets the OUT_PKT_RDY bit to a "1" in the OUT CSR. The CPU/DMAC should only read data from the OUT FIFO if the OUT_PKT_RDY bit of the OUT CSR is a "1", with the exception of endpoint 1 (see detailed description below).

•Endpoint 0 OUT FIFO Operation:

The USB FCU sets the OUT_PKT_RDY bit to a "1" after it has successfully received a packet of data from the host. The CPU sets bit SERVICED_OUT_PKT_RDY to a "1" to clear the OUT_PKT_RDY bit after the packet of data has been unloaded from the OUT FIFO by the CPU.

•Endpoint 1-4 OUT FIFO Operation when AUTO_CLR (bit 7 of OUT CSR) = "0":

MAXP > 1/2 of the OUT FIFO size: The USB FCU sets the OUT_PKT_RDY bit to a "1" after it has successfully received a packet of data from the host. The CPU writes a "0" to the OUT_PKT_RDY bit after the packet of data has been unloaded from the OUT FIFO by the CPU/DMAC.

MAXP <= 1/2 of the OUT FIFO size: The USB FCU sets the OUT_PKT_RDY bit to a "1" after it has successfully received a packet of data from the host. The CPU writes a "0" to the OUT_PKT_RDY bit after the packet of data has been unloaded from the OUT FIFO by the CPU/DMAC. In this configuration, the FIFO can hold up to two data packets at the same time for back-to-back reception. Therefore, the OUT_PKT_RDY bit will remain set after the CPU writes a "0" to it if there is another packet in the OUT FIFO.

•Endpoint 1-4 OUT FIFO Operation when AUTO_CLR (bit 7 of OUT CSR) = "1":

MAXP > 1/2 of the OUT FIFO size: The USB FCU sets the OUT_PKT_RDY bit to a "1" after it has successfully received a packet of data from the host. The USB FCU clears the OUT_PKT_RDY bit to a "0" automatically when the number of bytes of data equal to the MAXP (maximum packet size) have been unloaded from the OUT FIFO by the CPU/DMAC.

MAXP <= 1/2 of the OUT FIFO size: The USB FCU sets the OUT_PKT_RDY bit to a "1" after it has successfully received a packet of data from the host. The USB FCU clears the OUT_PKT_RDY bit to a "0" automatically when the number of bytes of data equal to the MAXP (maximum packet size) have been unloaded from the OUT FIFO by the CPU/DMAC. In this configuration, the FIFO can hold up to two data packets at the same time for back-to-back reception. Therefore, the OUT_PKT_RDY bit will remain set after one packet (size equal to MAXP) of data has been unloaded if there is another packet in the OUT FIFO.

A software flush causes the USB FCU to act as if a packet has been unloaded from the OUT FIFO. If there is one packet in the OUT FIFO, a flush will cause the OUT FIFO to be empty. If there are two packets in the OUT FIFO, a flush will cause the older packet to be flushed out from the OUT FIFO.

Special case for OUT Endpoint 1:

In addition to the OUT FIFO operations described above, the DMAC can also start unloading the OUT FIFO as soon as there is data in it (byte-by-byte transfer). This feature should only be used with ISO transfers. See DMAC section for details.

1.21.4 USB Special Function Registers

The MCU controls USB operation through the use of special function registers (SFR). This section describes in detail each USB related SFR. Certain USB SFRs are endpoint-indexed: the Control & Status Registers (IN CSR and OUT CSR), the Maximum Packet Size Registers (IN MAXP and OUT MAXP), and the Write Count Registers (OUT WRT CNT). To access each endpoint-indexed SFR, the target endpoint number should be written to the Endpoint Index Register first. The lower 3 bits (EPINDX2:0) of the Endpoint Index Register are used for endpoint selection.

Note: Each endpoint's FIFO Register is NOT endpoint-indexed.

Some USB special function registers have a mix of read/write, read only, and write-only register bits. Additionally, the bits may be configured to allow the user to write only a "0" or a "1" to individual bits. When accessing these registers, writing a "0" to a register that can only be set to a "1" by the CPU will have no affect on that register bit. Each figure and description of the special function registers will detail this operation.

The **USB Control Register (USBC)**, shown in Figure 1.52, is used to control the USB FCU. A USB reset signaling does not reset this register. After the USB is enabled (USBC7 set to "1"), a minimum delay of 250 ns (three 12Mhz clock periods) is needed before performing any other USB register read/write operations.

The **USB Function Address Register (USBA),** shown in Figure 1.53, maintains the 7-bit USB address assigned by the host. The USB FCU uses this register value to decode USB token packet addresses. At reset, when the device is not yet configured, the value is 00_{16}.

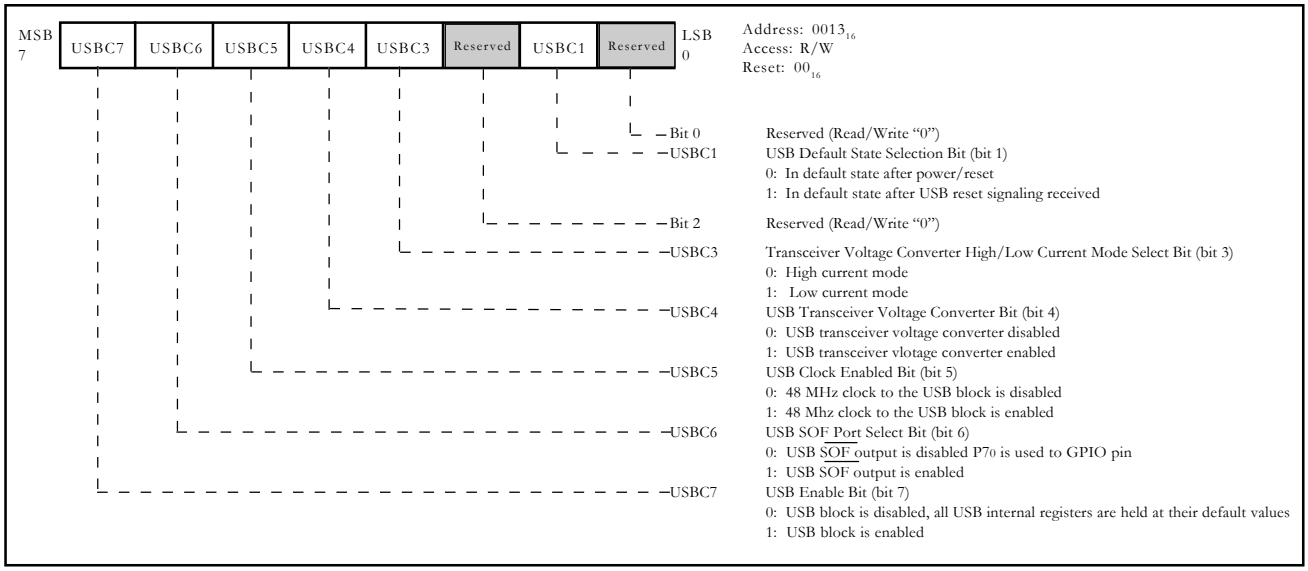


Fig. 1.52. USB Control Register (USBC)

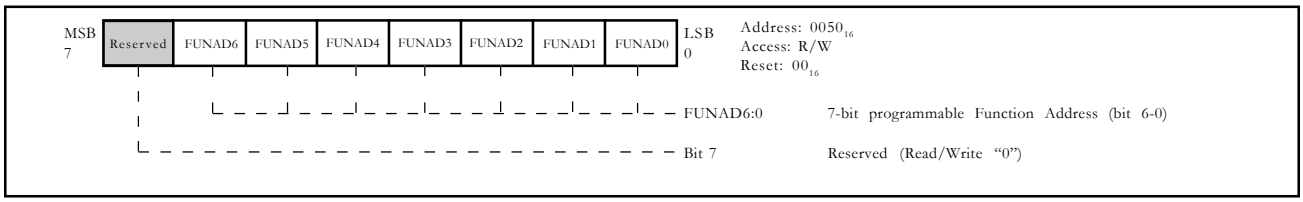


Fig. 1.53. Function Address Register (USBA)

The **USB Power Management Register**, shown in Figure 1.54, is used for power management in the USB FCU.

•USB Suspend Detection Flag (SUSPEND)

When the USB FCU does not detect any bus activity on D+/D- for at least 3ms, it sets the Suspend Detection Flag (SUSPEND) and generates an interrupt. This bit is cleared when signaling (from the host) is detected on D+/D- [which sets the Resume Detection Flag (RESUME) and generates an interrupt] or the Remote Wake-up Bit (WAKEUP) is set and then cleared by the CPU. If the USB clock was disabled during the suspend state, the SUSPEND bit is not cleared until after the USB clock is re-enabled.

•USB Resume Detection Flag (RESUME)

When the USB FCU is in the suspend state and detects signaling on D+/D- (from the host), it sets the Resume Detection Flag (RESUME) and generates an interrupt. The CPU writes a “1” to INST14 (bit 6 of USB Interrupt Status Register 2) to clear this flag.

•USB Remote Wake-up Bit (WAKEUP)

The CPU writes a “1” to the WAKEUP bit for remote wake-up. While this bit is set, and the USB FCU is in suspend mode, it will generate resume signaling to the host. The CPU must keep this bit set for a minimum of 10ms and a maximum of 15ms before writing a “0” to this bit.

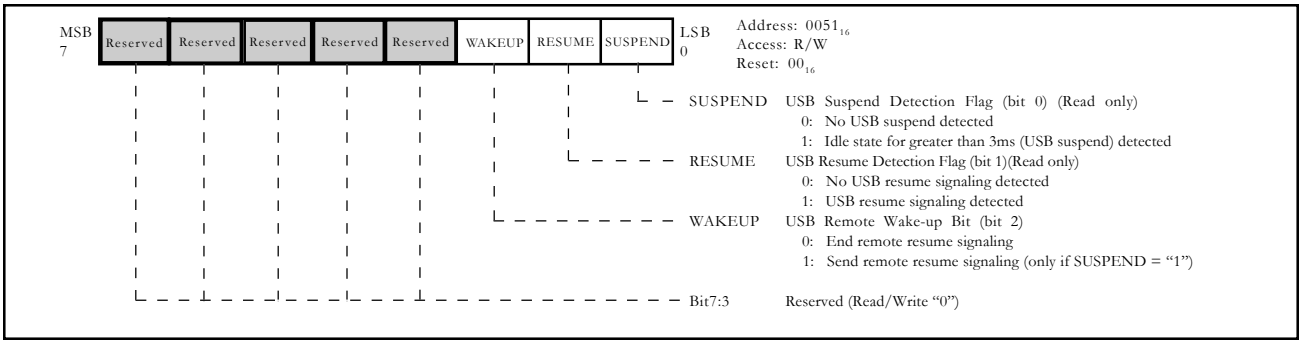


Fig. 1.54. USB Power Management Register (USBPM)

The USB FCU is able to generate a USB function interrupt as discussed in section 1.21.2.1. The **USB Interrupt Status Registers (USBIS1, USBIS2)**, shown in Figure 1.55 and Figure 1.56, are used to indicate the condition that caused a USB function interrupt to be generated. A "1" indicates that the corresponding condition caused a USB function interrupt. The USB Interrupt Status Register can be cleared by writing back to the register the same value that was read. To ensure proper operation, the CPU should read both USB interrupt status registers, then write back the same values it read to these two registers for clearing the status bits. The CPU must write to USB Interrupt Status Register 1 first and then to USB Interrupt Status Register 2. The registers cannot be cleared by writing a "0" to the bits that are a "1".

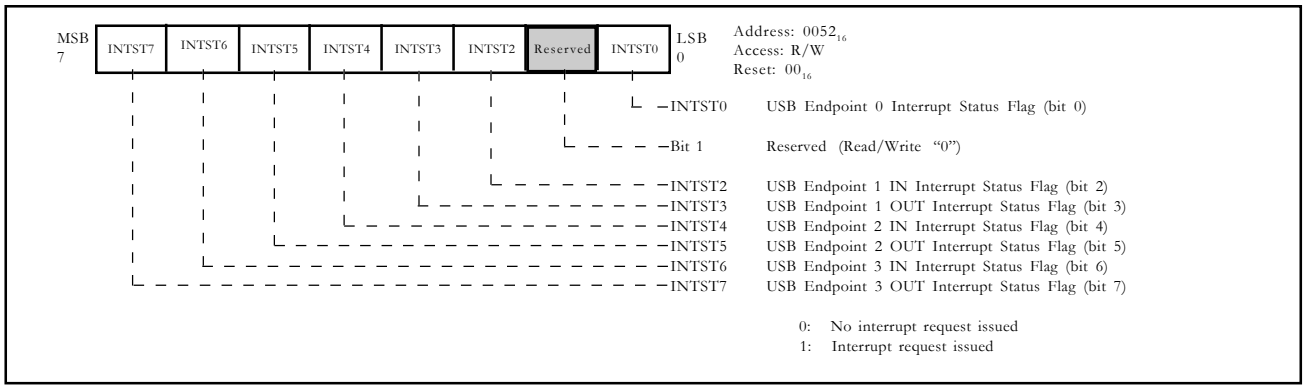


Fig. 1.55. USB Interrupt Status Register 1 (USBIS1)

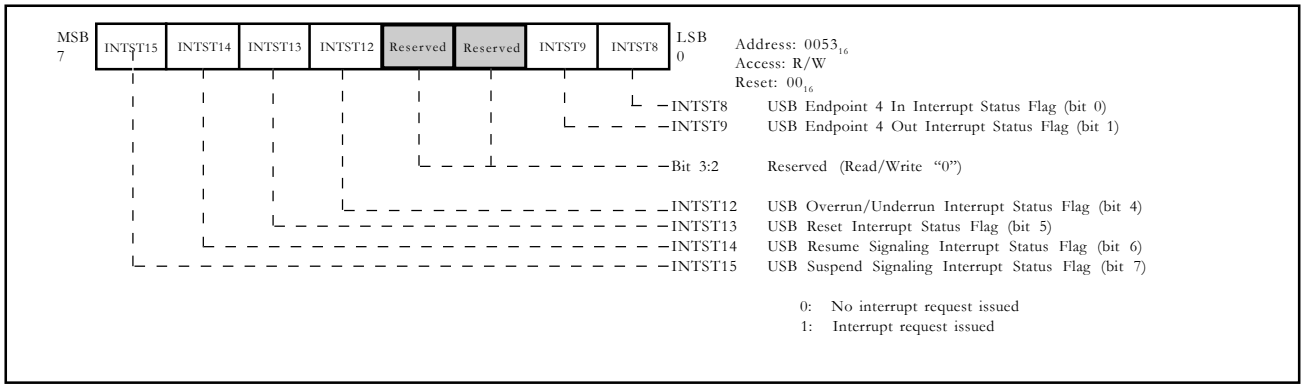


Fig. 1.56. USB Interrupt Status Register 2 (USBIS2)

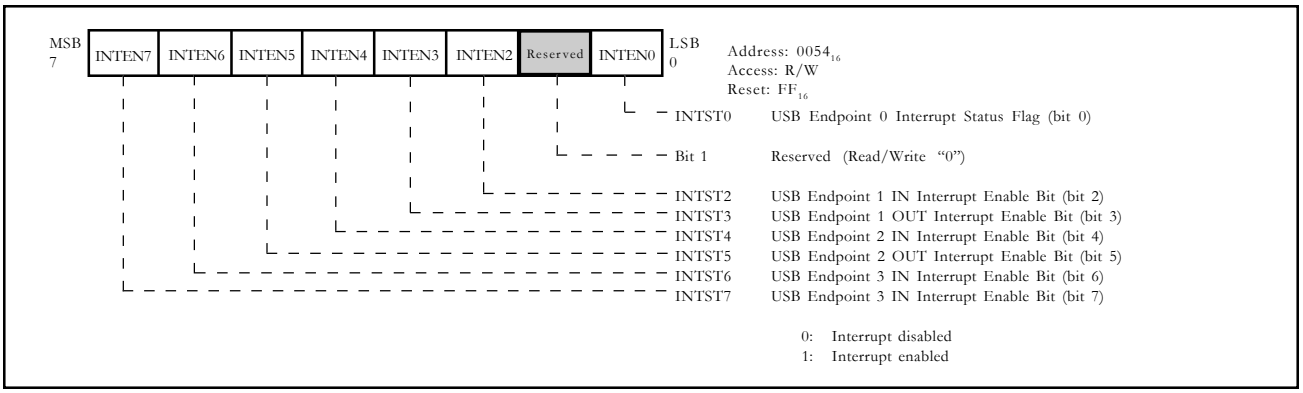


Fig. 1.57. USB Interrupt Enable Register 1 (USBIE1)

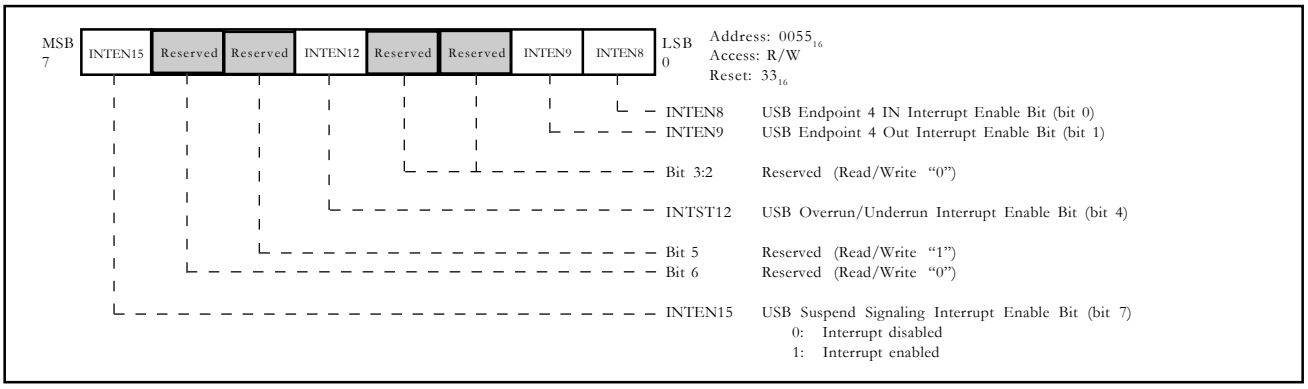


Fig. 1.58. USB Interrupt Enable Register 2 (USBIE2)

The **USB Interrupt Enable Registers (USBIE1, USBIE2)** shown in Figure 1.57 and Figure 1.58, are used to enable the corresponding interrupt status conditions that can generate a USB function interrupt. If the bit to a corresponding interrupt condition is "0", that condition will not generate a USB function interrupt. If the bit is a "1", that condition can generate a USB function interrupt. Upon reset, all USB interrupt status conditions are enabled except the USB Suspend Signaling Interrupt (bit 7 of USB Interrupt Enable Register 2), which is disabled. The USB Reset Interrupt and USB Resume Signaling Interrupt are always enabled.

INTST0 is set to a "1" by the USB FCU if (in Endpoint 0 IN CSR):

•A packet of data is successfully received

•A packet of data is successfully sent

•IN0CSR3 (DATA_END) bit is cleared (by USB FCU)

•IN0CSR4 (FORCE_STALL) bit is set (by the USB FCU)

•IN0CSR5 (SETUP_END) bit is set (by the USB FCU)

INTST2, INTST4, INTST6 or INTST8 is set to a "1" by USB FCU if (in Endpoint x IN CSR):

•A packet of data is successfully sent

•INXCSR1 (UNDER_RUN) bit is set (by USB FCU)

INTST3, INTST5, INTST7 or INTST9 is set to a "1" by USB FCU if (in Endpoint xOUT CSR):

•A packet of data is successfully received

•OUTXCSR1 (OVER_RUN) bit is set (by USB FCU)

•OUTXCSR4 (FORCE_STALL) bit is set (by USB FCU)

INTST12 is set to a "1" by the USB FCU if an overrun or underrun condition occurs in any of the endpoints.

INTST13 is set to a "1" by the USB FCU if USB reset signaling from the host is received. All USB internal registers other than this bit are reset to their default values when the USB reset is received.

INTST14 is set to a "1" by the USB FCU when the USB FCU is in the suspend state and non-idle signaling on D+/D- is received.

INTST15 is set to a "1" by the USB FCU when D+/D- are in the idle state for more than 3ms.

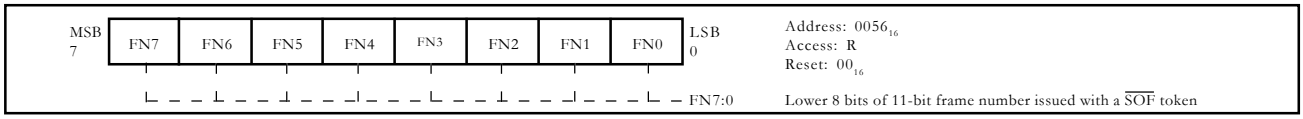


Fig. 1.59. USB Frame Number Low Register (USBSOFL)

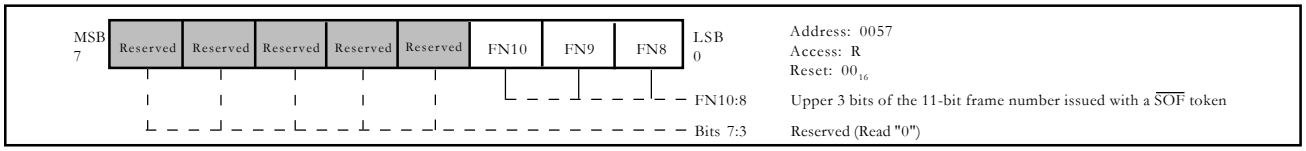


Fig. 1.60. USB Frame Number High Register (USBSOFH)

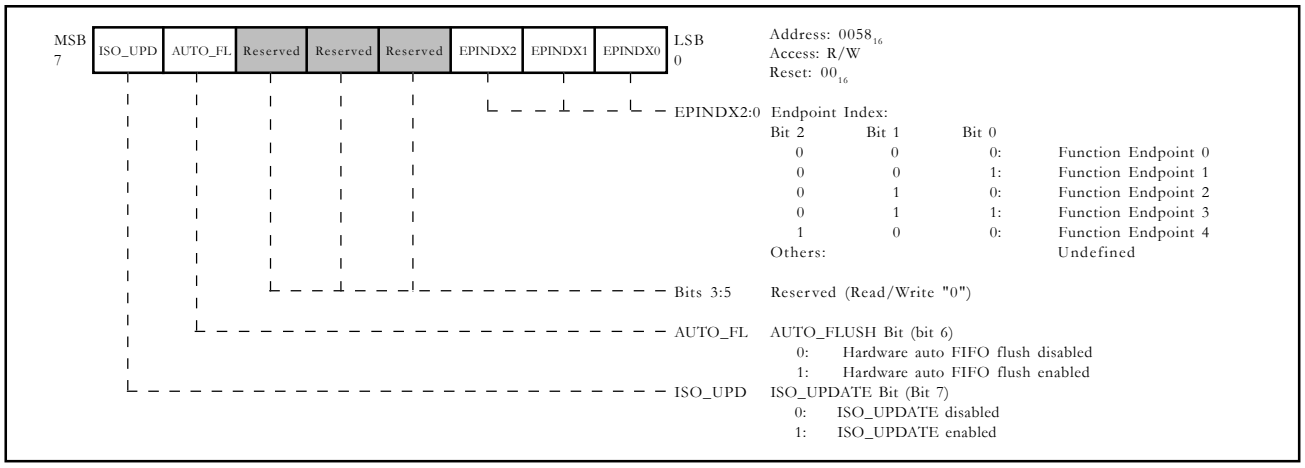


Fig. 1.61. USB Endpoint Index Register (USBINDEX)

The **USB Frame Number Low Register (USBSOFL)**, shown in Figure 1.59, contains the lower 8 bits of the 11-bit frame number received from the host.

The **USB Frame Number High Register (USBSOFH)**, shown in Figure 1.60, contains the upper 3 bits of the 11-bit frame number received from the host.

The **USB Endpoint Index Register (USBINDEX)**, shown in Figure 1.61, identifies the endpoint pair. It serves as an index to endpoint-specific IN CSR, OUT CSR, IN MAXP, OUT MAXP and OUT WRT CNT registers.

This register also contains two global bits, ISO_UPD and AUTO_FL, which affect isochronous data transfers for endpoints 1-4.

If ISO_UPD = “0”, a data packet in an endpoint’s IN FIFO is always ‘ready to transmit’ upon receiving the next IN_TOKEN from the host (with matched address and endpoint number). If ISO_UPD = “1” and the ISO bit of the corresponding endpoint’s IN CSR is set, then the internal ‘ready to transmit’ signal to the transmit control logic is delayed until the next SOF. In this way the data loaded in frame n will be transmitted out in frame n+1. The ISO_UPD bit is a global bit for endpoints 1 to 4, and works with isochronous pipes only.

If AUTO_FL = “1”, ISO_UPD = “1”, and a particular IN endpoint’s ISO bit is set, then when the USB FCU detects an SOF packet, if the corresponding IN endpoint’s IN_PKT_RDY = “1”, the USB FCU automatically flushes the oldest packet from the IN FIFO. In this case, IN_PKT_RDY = “1” indicates that two data packet are in the IN FIFO. Since, for ISO transfer, double buffering is a requirement; MAXP must set to be less than or equal to 1/2 of the FIFO size.

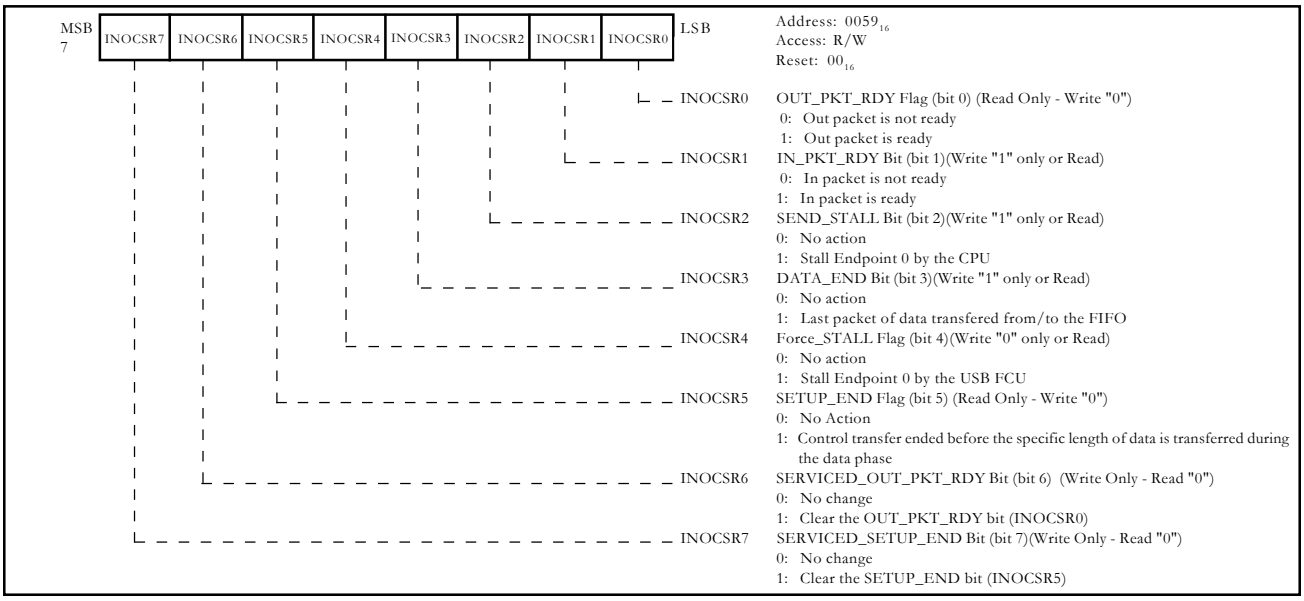


Fig. 1.62. USB Endpoint 0 IN CSR (IN_CSR)

The **USB Endpoint 0 IN CSR Control & Status Register**, shown in Figure 1.62, contains the control and status information of Endpoint 0.

IN0CSR0 (OUT_PKT_RDY): The USB FCU sets this bit to a “1” upon receiving a valid SETUP/OUT token from the host. The CPU clears this bit after unloading the FIFO, by way of writing a “1” to IN0CSR6. The CPU should not clear the OUT_PKT_RDY bit before it finishes decoding the host request. If IN0CSR2 (SEND_STALL) needs to be set (because the CPU decodes an invalid or unsupported request), the setting of IN0CSR6 = “1” and IN0CSR2 = “1” should be done in the same CPU write.

IN0CSR1 (IN_PKT_RDY): The CPU writes a “1” to this bit after it finishes writing a packet of data to the endpoint 0 FIFO. The USB FCU clears this bit after the packet has been successfully transmitted to the host, or the IN0CSR5 (SETUP_END) bit is set.

IN0CSR2 (SEND_STALL): The CPU writes a “1” to this bit if it decodes an invalid or unsupported request from the host. If the OUT_PKT_RDY bit is a “1” at the time the CPU wants to set the SEND_STALL bit is to a “1”, the CPU must also set SERVICED_OUT_PKT_RDY to a “1” to clear the OUT_PKT_RDY. The USB FCU returns a STALL handshake for all subsequent IN/OUT transactions (during control transfer data or status stages) while this bit is set. The CPU writes a “0” to this bit to clear it.

IN0CSR3 (DATA_END): For control transfers, the CPU writes a “1” to this bit after it writes (IN data phase) or reads (OUT data phase) the last packet of data from/to the FIFO. This bit indicates to the USB FCU that the specific amount of data in the setup phase has been transferred. The USB FCU will advance to the status phase once this bit is set. When the status phase completes, the USB FCU clears this bit. When this bit is set to a “1” and the host again requests or sends more data to the device, the USB FCU returns a STALL handshake and terminates the current control transfer.

IN0CSR4 (FORCE_STALL): The USB FCU sets this bit to a “1” to report an error status if the one of the following occurs:

- Host sends an IN or OUT token in the absence of a SETUP stage
- Host sends a bad data toggle in the STATUS stage, i.e. DATA0 is used
- Host sends a bad data toggle in the SETUP stage, i.e. DATA1 is used
- Host requests more data than specified in the SETUP stage, i.e. IN token comes after DATA_END bit is set
- Host sends more data than specified in the SETUP stage, i.e. OUT token comes after DATA_END bit is set
- Host sends larger data packet than MAXP size of the corresponding endpoint.

All of the conditions stated on the preceeding page (except the bad data toggle in the SETUP state case) cause the device to send a STALL handshake for the IN/OUT token in question. In the bad data toggle in the SETUP stage case, the device sends ACK for the SETUP stage and then sends STALL for the next IN/OUT token. A STALL handshake caused by the above conditions lasts for only one transaction and terminates the ongoing control transfer. Any packet after the STALL handshake will be seen as the beginning of a new control transfer.

The CPU writes a “0” to clear this FORCE_STALL status bit.

IN0CSR5 (SETUP_END): The USB FCU sets this bit to a “1” if a control transfer has ended before the specific length of data is transferred during the data phase. The CPU clears this bit by way of writing a “1” to IN0CSR7. Once the CPU sees the SETUP_END bit set, it should stop accessing the FIFO to service the previous setup transaction. If OUT_PKT_RDY is set at the same time that SETUP_END is set, it indicates that the previous setup transaction ended and a new SETUP token is in the FIFO.

IN0CSR6 and IN0CSR7: These bits are used to clear IN0CSR0 and IN0CSR5 respectively. Writing a “1” to these bits will clear the corresponding register bit.

The **USB Endpoint x IN CSR**, shown in Figure 1.63, contains control and status information of the respective IN endpoint 1-4. The USB Endpoint Index Register selects the specific endpoint.

INXCSR0 (IN_PKT_RDY) and INXCSR5 (TX_FIFO_NOT_EMPTY): These two bits are read together to determine IN FIFO status. A “1” can be written to the INXCSR0 bit by the CPU to indicate a packet of data is written to the FIFO (See Chapter. for detail).

INXCSR1 (UNDER_RUN): This bit is used in ISO mode only to indicate to the CPU that a FIFO underrun has occurred. The USB FCU sets this bit to a “1” at the beginning of an IN token if no data packet is in the FIFO. Setting this bit will cause the INST12 bit of the Interrupt Status Register 2 to set. The CPU writes a “0” to clear this bit.

INXCSR2 (SEND_STALL): The CPU writes a “1” to this bit when the endpoint is stalled (transmitter halt). The USB FCU returns a STALL handshake while this bit is set. The CPU writes a “0” to clear this bit.

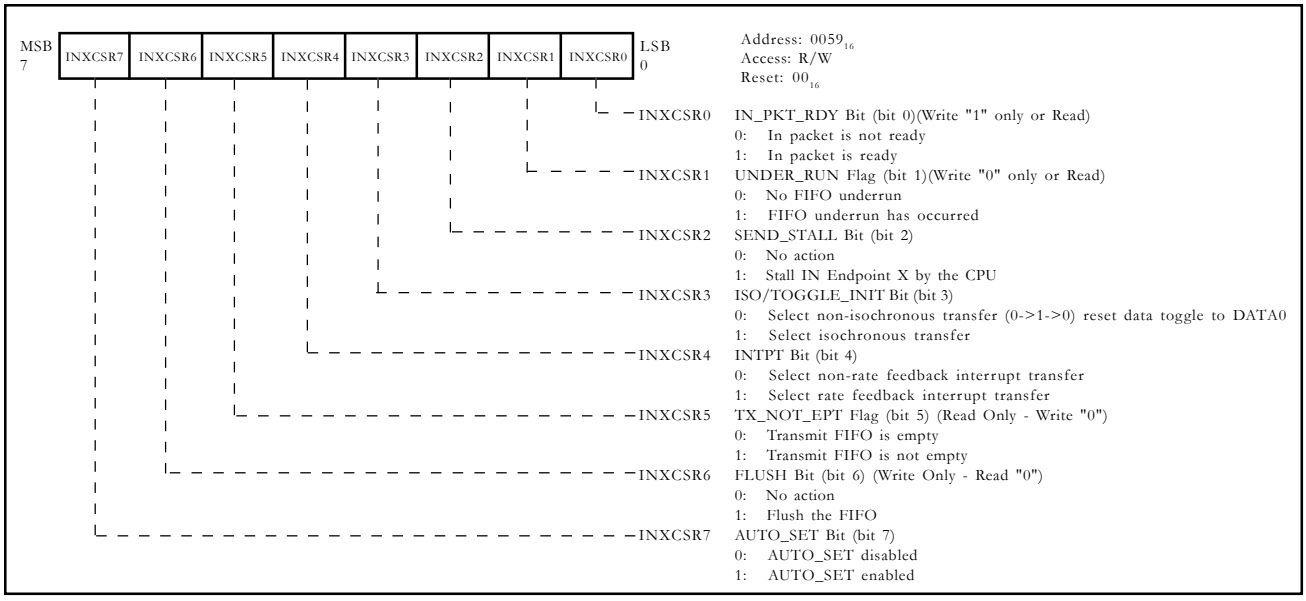


Fig. 1.63. USB Endpoint x IN CSR (IN_CSR)

INXCSR3 (ISO/TOGGLE_INIT): When the endpoint is used for isochronous data transfer, the CPU sets this bit to a “1” for the entire duration of the isochronous transfer. With the ISO bit set to a “1”, the device uses DATA0 as the PID for all packets sent back to the host.

When the endpoint is required to initialize the data toggle sequence bit (reset to DATA0 for the next data packet), the CPU sets this bit to a “1” and then resets it to a “0” to initialize the respective endpoint’s data toggle.

As with any other method to initialize the data toggle, this set/reset of the TOGGLE_INIT bit method assumes that there is no active IN transaction to the respective endpoint on the bus at the time the initialization process is ongoing. Set/reset of the TOGGLE_INIT bit is performed only when an endpoint experiences a configuration event.

INXCSR4 (INTPT): The CPU writes a “1” to this bit to initialize this endpoint as a status change endpoint for IN transactions. This bit is set only if the corresponding endpoint is to be used to communicate rate feedback information (see section 1.21.3.1 for details).

INXCSR5 (TX_FIFO_NOT_EMPTY): The USB FCU sets this bit to a “1” when there is data in the IN FIFO. This bit in conjunction with IN_PKT_RDY bit will provide the transmit FIFO status information (see section 1.21.3.1 for details).

INXCSR6 (FLUSH): The CPU writes a “1” to this bit to flush the IN FIFO. If there is one packet in the IN FIFO, a flush will cause the IN FIFO to be empty. If there are two packets in the IN FIFO, a flush will cause the older packet to be flushed out from the IN FIFO. Setting the INXCSR6 (FLUSH) bit during transmission could produce unpredictable results.

INXCSR7 (AUTO_SET): If the CPU sets this bit to a “1”, the IN_PKT_RDY bit is set automatically by the USB FCU after the number of bytes of data equal to the maximum packet size (MAXP) are written into the IN FIFO (see section 1.21.3.1 for details).

All bits in **USB Endpoint 0 OUT CSR (Control & Status Register)**, shown in Figure 1.64, are reserved (all control and status information is in Endpoint 0 IN CSR)

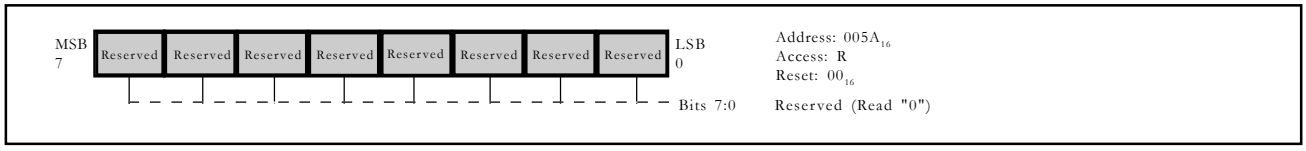


Fig. 1.64. USB Endpoint 0 OUT CSR (OUT_CSR)

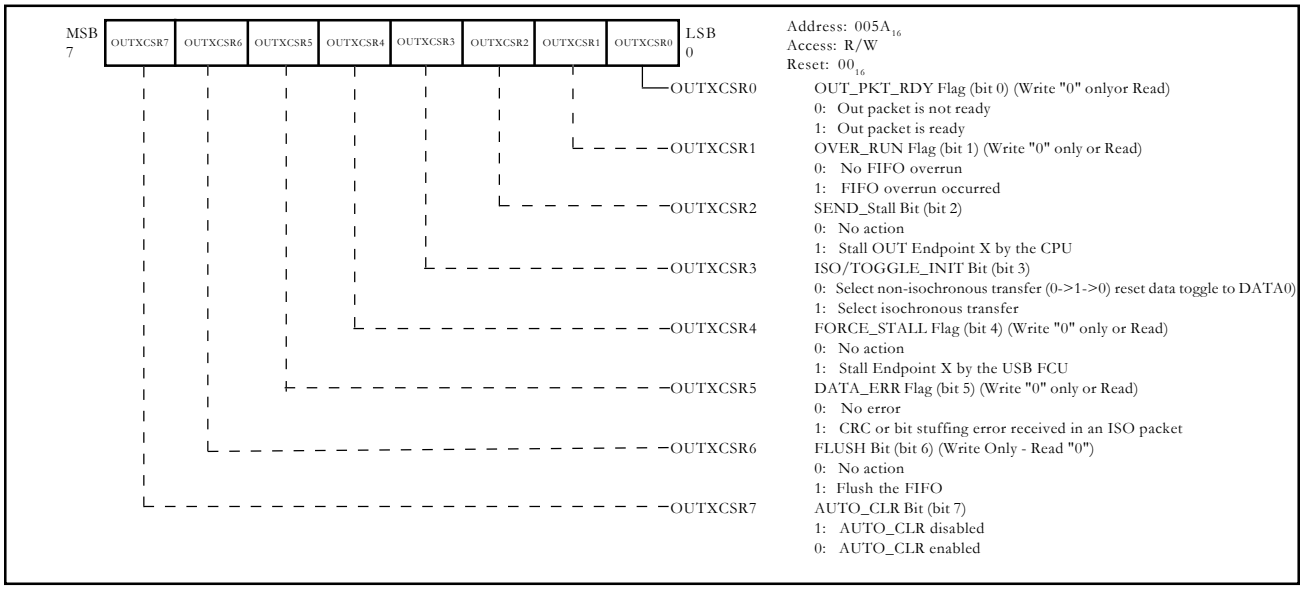


Fig. 1.65. USB Endpoint x OUT CSR (OUT_CSR)

The **USB Endpoint x OUT CSR (Control & Status Register)**, shown in Figure 1.65, contains control and status information of the respective OUT endpoint 1-4. The USB Endpoint Index Register selects the specific endpoint.

OUTXCSR0 (OUT_PKT_RDY): The USB FCU sets this bit to a “1” after it successfully receives a packet of data from the host. This bit is cleared by the CPU or by the USB FCU after a packet of data has been unloaded from the FIFO (See section 1.21.3.2 for details).

OUTXCSR1 (OVER_RUN): This bit is used in ISO mode only to indicate to the CPU that a FIFO overrun has occurred. The USB FCU sets this bit to a “1” at the beginning of an OUT token if the OUTXCSR0 (OUT_PKT_RDY) bit is not cleared. Setting this bit will cause the INST12 bit of the Interrupt Status Register 2 to set. The CPU writes a “0” to clear this bit.

OUTXCSR2 (SEND_STALL): The CPU writes a “1” to this bit when the endpoint is stalled (receiver halt). The USB FCU returns a STALL handshake while this bit is set. The CPU writes a “0” to clear this bit.

OUTXCSR3 (ISO/TOGGLE_INIT): When the endpoint is used for isochronous data transfer, the CPU sets this bit to a “1” for the entire duration of the isochronous transfer. With the ISO bit set to a “1”, the device accepts either DATA0 or DATA1 for the PID sent by the host.

When the endpoint is required to initialize the data toggle sequence bit (reset to DATA0 for the next data packet), the CPU sets this bit to a “1” and then resets it to a “0” to initialize the respective endpoint’s data toggle.

As with any other method to initialize the data toggle, this set/reset of the TOGGLE_INIT bit method assumes that there is no active OUT transaction to the respective endpoint on the bus at the time the initialization process is ongoing. Set/reset of the TOGGLE_INIT bit is performed only when an endpoint experiences a configuration event.

OUTXCSR4 (FORCE_STALL): The USB FCU sets this bit to a “1” if the host sends out a larger data packet than the MAXP size. The USB FCU returns a STALL handshake while this bit is set. The CPU writes a “0” to clear this bit.

OUTXCSR5 (DATA_ERR): The USB FCU sets this bit to a “1” to indicate the reception of a CRC error or a bit stuffing error in an ISO packet. The CPU writes a “0” to clear this bit.

OUTXCSR6 (FLUSH): The CPU writes a “1” to flush the OUT FIFO. If there is one packet in the OUT FIFO, a flush will cause the OUT FIFO to be empty. If there are two packets in the OUT FIFO, a flush will cause the older packet to be flushed out from the OUT FIFO. Setting the OUTXCSR6 (FLUSH) bit during reception could produce unpredictable results.

OUTXCSR7 (AUTO_CLR): If the CPU sets this bit to a “1”, the OUT_PKT_RDY bit is cleared automatically by the USB FCU after the number of bytes of data equal to the maximum packet size (MAXP) is unloaded from the OUT FIFO (see section 1.21.3.2 for details).

The **USB Endpoint x IN MAXP**, shown in Figure 1.66, indicates the maximum packet size (MAXP) of an Endpoint x IN packet. The default value for Endpoint 0 and 2-4 is 8. The default value for Endpoint 1 is 1. The CPU can change this value as negotiated with the host controller through the SET_DESCRIPTOR command. The setting of this register also affects the configuration of single/dual packet operation. When MAXP > 1/2 of the FIFO size, single packet mode is set. When MAXP <= 1/ 2 of the FIFO size, dual packet mode is set.

The **USB Endpoint x OUT MAXP**, shown in Figure 1.67, indicates the maximum packet size (MAXP) of an Endpoint x OUT packet. The default value for Endpoint 0 and 2-4 is 8. The default value for endpoint 1 is 1. For endpoint 0, the IN_MAXP and OUT_MAXP registers shadow each other. Changing one register's value effectively changes the other register's value. The CPU can change this value as negotiated with the host controller through the SET_DESCRIPTOR command. The setting of this register also affects the configuration of single/dual packet operation. When MAXP > 1/2 of the FIFO size, single packet mode is set. When MAXP <= 1/2 of the FIFO size, dual packet mode is set.

The **USB Endpoint x OUT WRT CNT Low** and the **USB Endpoint x OUT WRT CNT High** registers, shown in Figure 1.68 and Figure 1.69, contain the number of bytes in the Endpoint x OUT FIFO. The USB FCU sets the values in these two Write Count Registers after having successfully received a packet of data from the host. The CPU reads these two registers to determine the number of bytes to be read from the FIFO. The CPU should read WRT CNT Low first and then WRT CNT High.

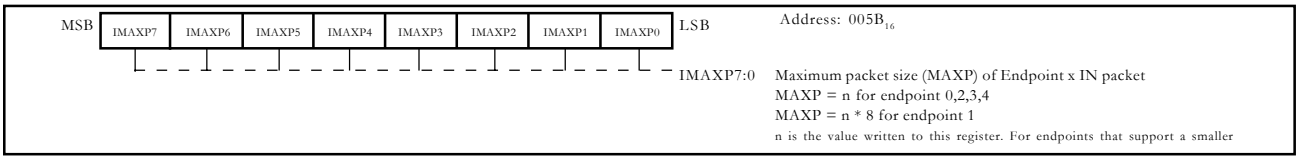


Fig. 1.66. USB Endpoint x IN MAXP Register (IN_MAXP)

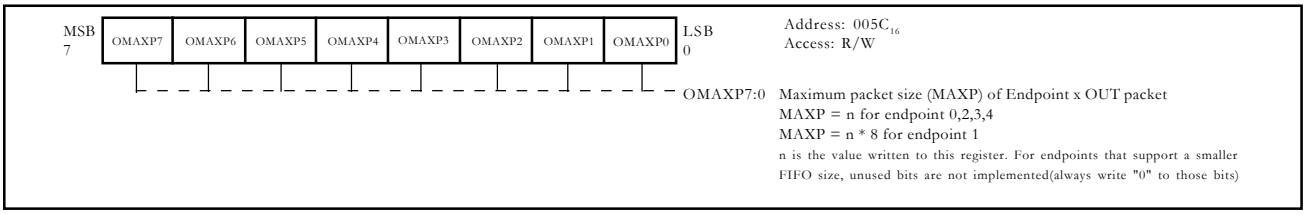


Fig. 1.67. USB Endpoint x OUT MAXP Register (OUT_MAXP)

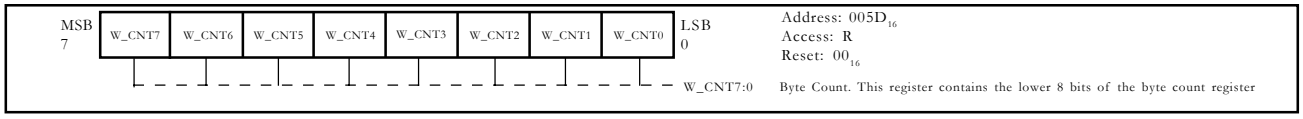


Fig. 1.68. USB Endpoint x OUT WRT CNT Low Register (WRT_CNTL)

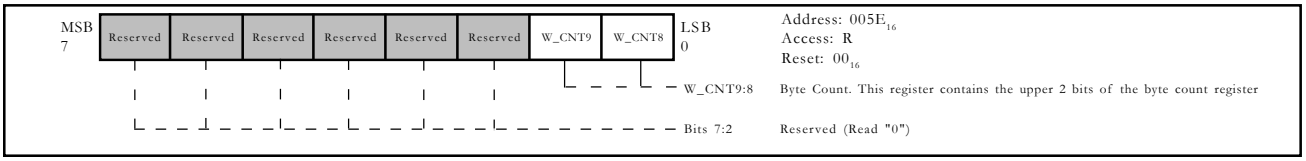


Fig. 1.69. USB Endpoint x OUT WRT CNT High Register (WRT_CNTH)

The **USB Endpoint x FIFO Registers**, shown in Figure 1.70 through Figure 1.74, are the USB IN (transmit) and OUT (receive) FIFO data registers. The CPU writes data to these registers for the corresponding Endpoint IN FIFO and reads data from these registers for the corresponding Endpoint OUT FIFO.

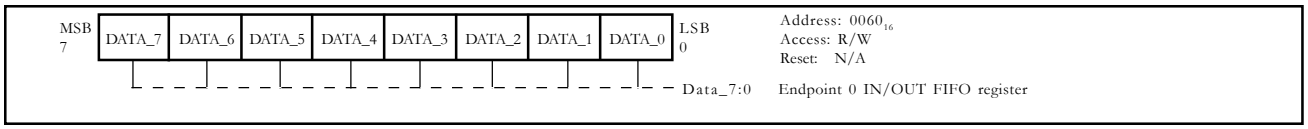


Fig. 1.70. USB Endpoint 0 FIFO Register (USBFIFO0)

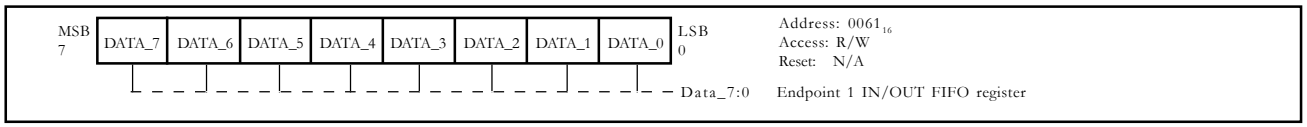


Fig. 1.71. USB Endpoint 1 FIFO Register (USBFIFO1)

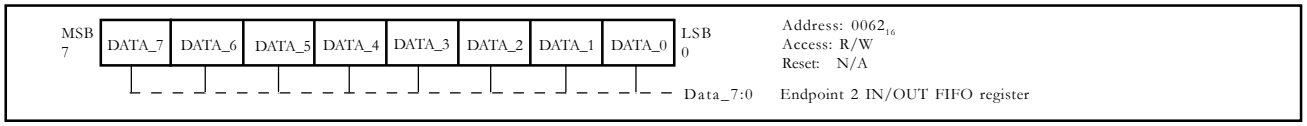


Fig. 1.72. USB Endpoint 2 FIFO Register (USBFIFO)

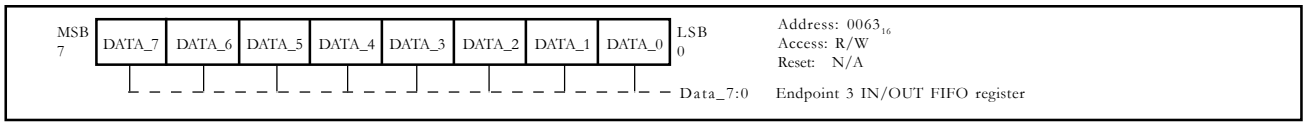


Fig. 1.73. USB Endpoint 3 FIFO Register (USBFIFO3)

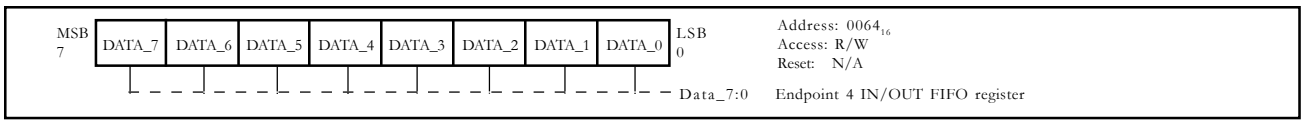


Fig. 1.74. USB Endpoint 4 FIFO Register (USBFIFO4)

1.22 MASTER CPU BUS INTERFACE

This device has a bus interface function with 2 I/O buffers that can be operated in slave mode by control signals from the master CPU (see Figure 1.75). Bus Interface Circuit). The bus interface can be connected directly to either a R/$\overline{W}$ type of CPU or a CPU with $\overline{RD}$ and $\overline{WR}$ separate signals. Slave mode is selected with the bit 7 of the data buffer control register 0. The single data bus buffer mode and the double data bus buffer mode are selected with bit 7 of the Data Bus Buffer Control register 1. When selecting the double data bus buffer mode, Port P72 becomes $\overline{S}_1$ input. Prior to enabling the MBI, Port 6 must be placed in input mode by writing 00_{16} to P6D (0015_{16}).

When data is written to the MCU from the master CPU, an input buffer full interrupt request occurs. Similarly, when data is read from the master CPU, an output buffer empty interrupt request occurs.

When the bus interface is operating, DQ_0-DQ_7 become a 3-state data bus that sends and receives data, command, and status to and from the master CPU. At the same time, $\overline{W}$, $\overline{R}$, $\overline{S}_0$, $\overline{S}_1$, and A_0 become host CPU control signal input pins.

The two input buffer full interrupt requests and two output buffer full requests are multiplexed as shown in Figure 1.76.

The bus interface can be operated under normal MCU control or under on-chip DMA control for fast data transfer. If a master CPU has a large amount of data to be transferred, use of the on-chip DMA controller is highly recommended.

The bus interface signal input level can be programmed as CMOS level (default) or as TTL level. Bit7 of the Port Control Register (PTC_7) is used for the input level selection.

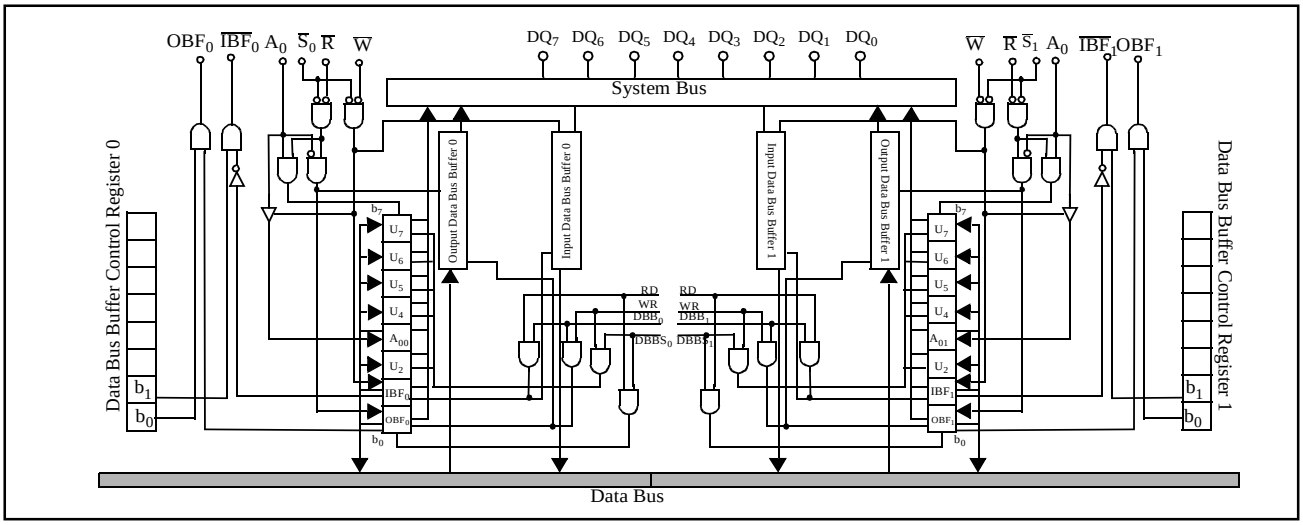


Fig. 1.75. Bus Interface Circuit

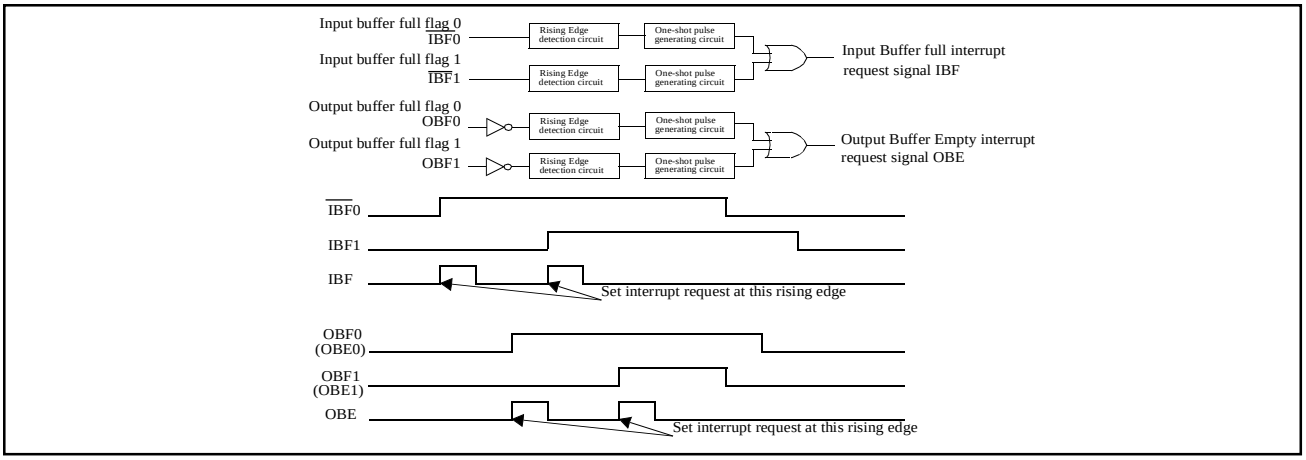


Fig. 1.76. Data Bus Buffer Interrupt Request Circuit

1.22.1 Data Bus Buffer Status Registers (DBBS0, DBBS1)

The data bus buffer status register is an 8-bit register that indicates the data bus status, with bits 0, 1, and 3 being dedicated read-only bits. Bits 2, 4, 5, 6, and 7 are user definable flags set by software, and can be read and write. When the A0 pin is high, the master CPU can read the contents of this register. See Figures 1.77 to 1.80.

Output Buffer Full Flag (OBF0, OBF1)

The OBF0 and the OBF1 flags are set high when data is written to the output data bus buffer by the slave CPU and is cleared to "0" when data is read by the master CPU.

Input Buffer Full Flag ($\overline{IBF}_0$, $\overline{IBF}_1$)

The $\overline{IBF}_0$ and the $\overline{IBF}_1$ flags are set high when data is written to the input data bus buffer by the master CPU and is cleared to "0" when data is read by the slave CPU.

A0 Flag (A00, A01)

The level of the A0 pin is latched when data has been written from the host CPU to the input data bus buffer.

1.22.2 Input Data Bus Buffer Registers (DBBIN0, DBBIN1)

The data on the data bus is latched into DBBIN0 or DBBIN1 by a write request from the master CPU. The data in DBBIN0 or DBBIN1 can be read from the data bus buffer register in the SFR area.

1.22.3 Output Data Bus Buffer Registers (DBBOUT0, DBBOUT1)

Data is set in DBBOUT0 or DBBOUT1 by writing to the data bus buffer register in the SFR area. When the A0 pin is low, the data of this register is output by a read request from the host CPU.

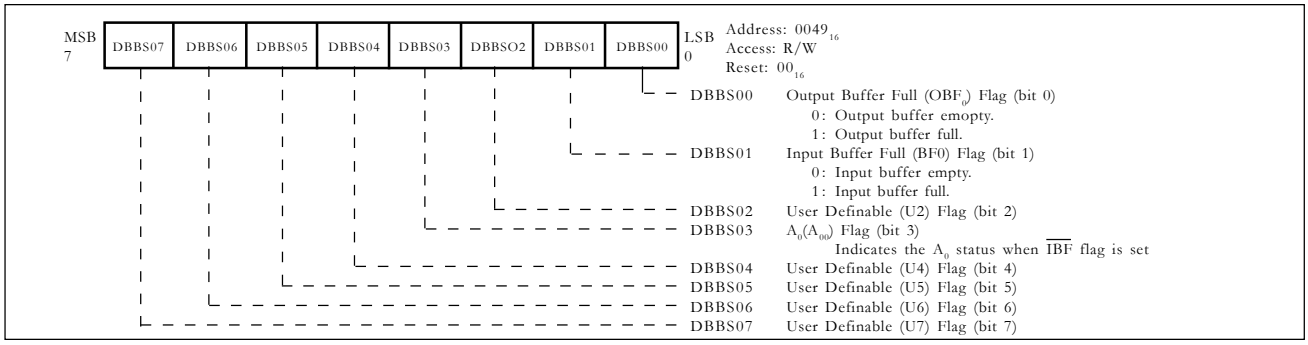


Fig. 1.77. Data Bus Buffer Status Register 0 (DBBS0)

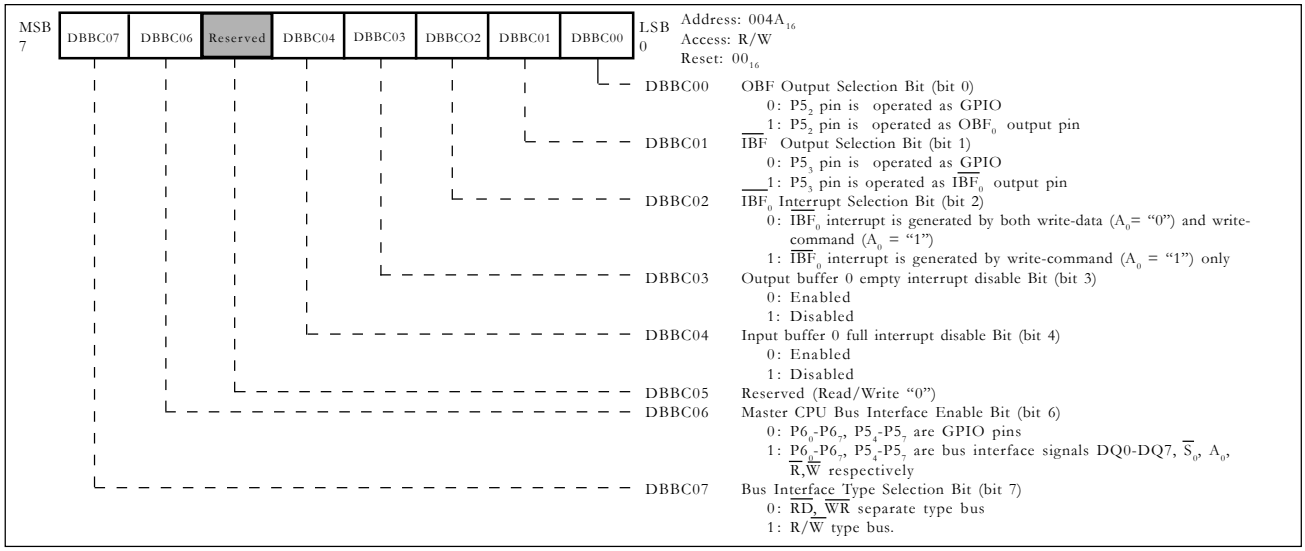


Fig. 1.78. Data Bus Buffer Control Register 0 (DBBC0)

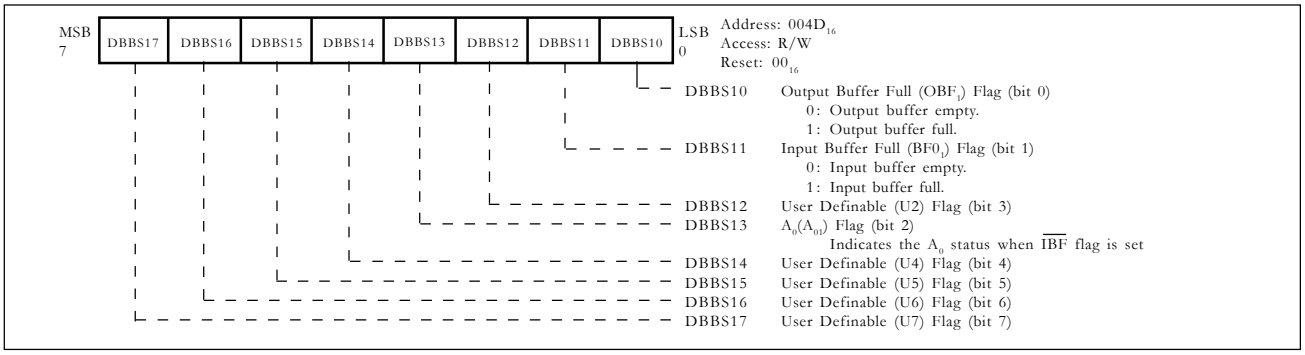


Fig. 1.79. Data Bus Buffer Status Register 1 (DBBS1)

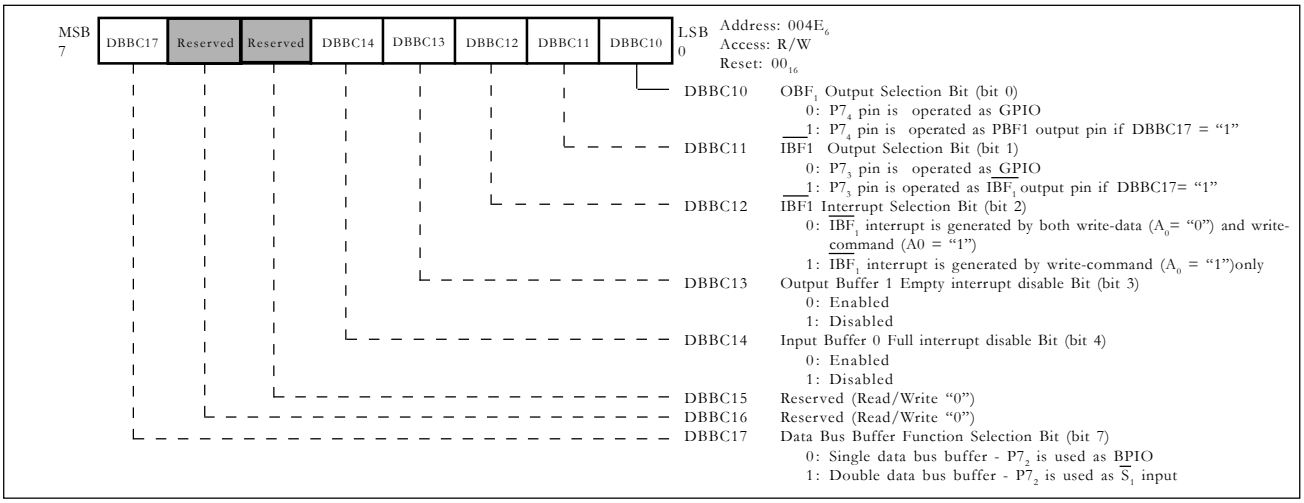


Fig. 1.80. Data Bus Buffer Control Register 1 (DBBC1)

1.23 DIRECT MEMORY ACCESS CONTROLLER

This device contains a two-channel Direct Memory Access Controller (DMAC). Each channel performs fast data transfers between any two locations in the memory map initiated by specific peripheral events or software triggers.

The main features of the DMAC are as follows:

- Two independent channels
- Single-byte and burst transfer modes
- 16-bit source and destination address registers (for a 64K byte address space)
- 16-bit transfer count registers (for up to 64K bytes transferred before underflow)
- Source/Destination register automatic increment/decrement and no-change options
- Source/Destination/Transfer count register reload on write or after transfer count register underflow options
- Transfer requests from USB (9), MBI (4), external interrupts (4), UART1 (2), UART2 (2), SIO (1), TimerX (1), TimerY (1), Timer1 (1), and software triggers
- Closely coupled with USB and MBI for efficient data transfers
- Interrupt generated for each channel when their respective transfer count register underflows
- Fixed channel priority (channel 0 > channel 1)
- Two cycles of Φ required per byte transferred

Each channel of the DMAC is made up of the following:

- 16-bit source and destination registers
- A 16-bit transfer count register
- Two mode registers
- Status flags contained in a status register shared by the two channels
- Control and timing logic

The 16-bit source and destination registers allow accesses to any two locations in the 64K byte memory area. The 16-bit transfer count register decrements by one for each transfer performed and causes an interrupt and flag to be set when it underflows. The mode registers control the configuration and operation of the DMAC channel associated with the registers. A block diagram of the DMAC is shown in Figure 1.81.

The SFR addresses for the two mode, source, destination, and transfer count registers of a channel are the same for each channel. The accessible channel registers are is determined by the value of the DMAC Channel Index Bit (DCI) (bit 7 of the DMAC Index and Status Register (DMAIS). When this bit is a “0”, channel 0 registers are accessible, and when this bit is a “1”, channel 1 registers are accessible.

The configuration of DMAIS and the mode registers are shown in Figures 1.82, 1.83, 1.84, and 1.85.

Sample timing diagrams are shown in Figures 1.86, 1.87, and 1.88, for a single-byte transfer initiated by a hardware source, a single-byte transfer initiated by the software trigger, and a burst transfer initiated by a hardware source, respectively.

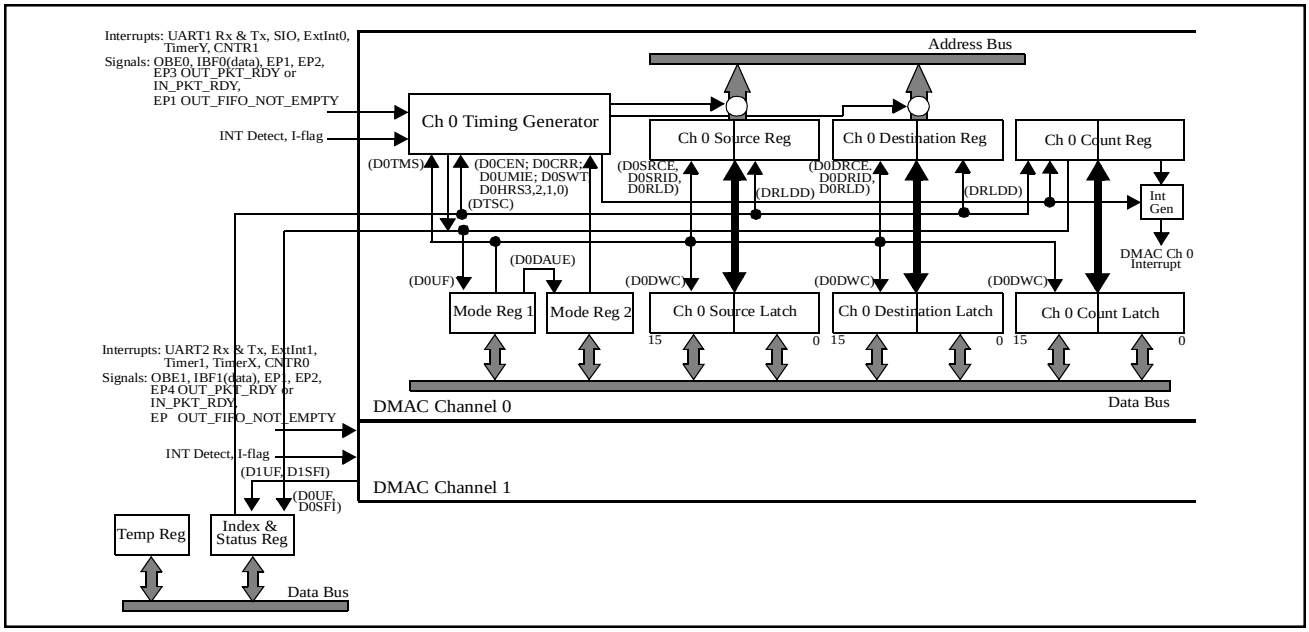


Fig. 1.81. DMAC Block Diagram

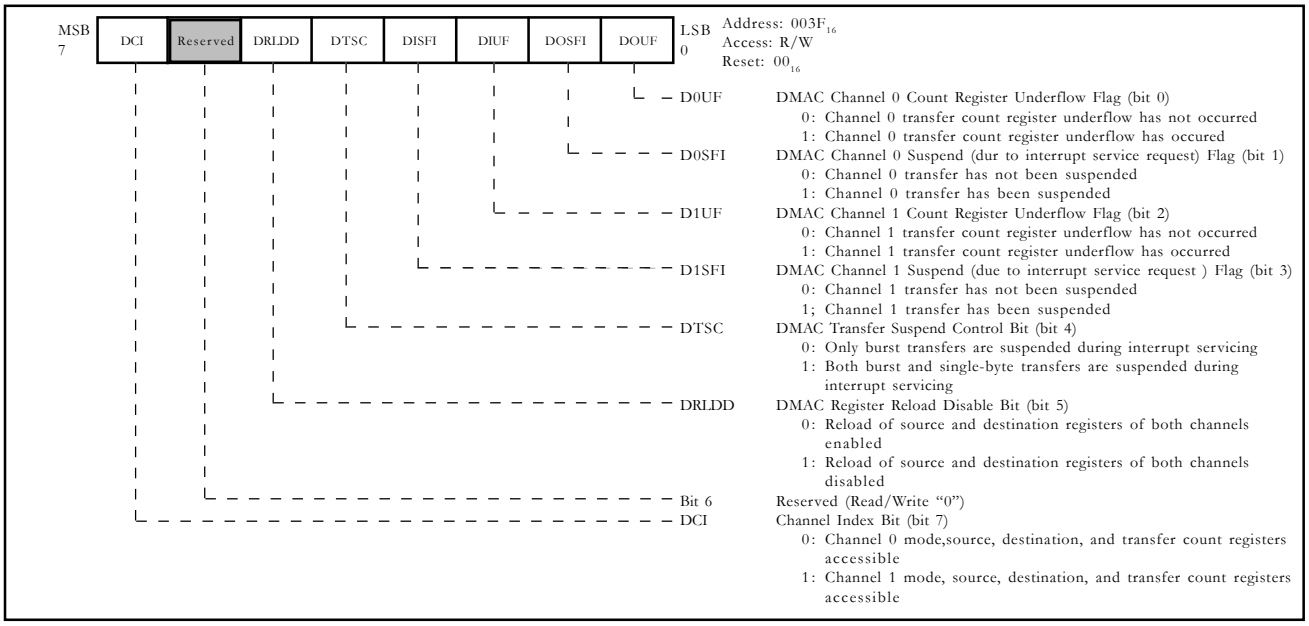


Fig. 1.82. DMAC Index and Status Mode Register (DMAIS)

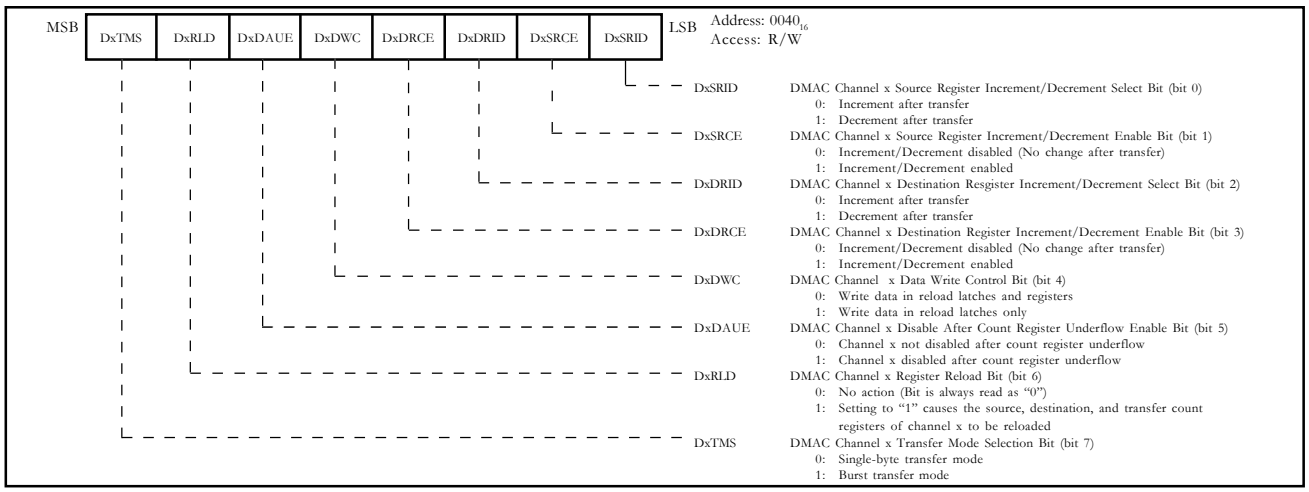


Fig. 1.83. DMAC Channel x Mode Register 1 (DMAxM1)

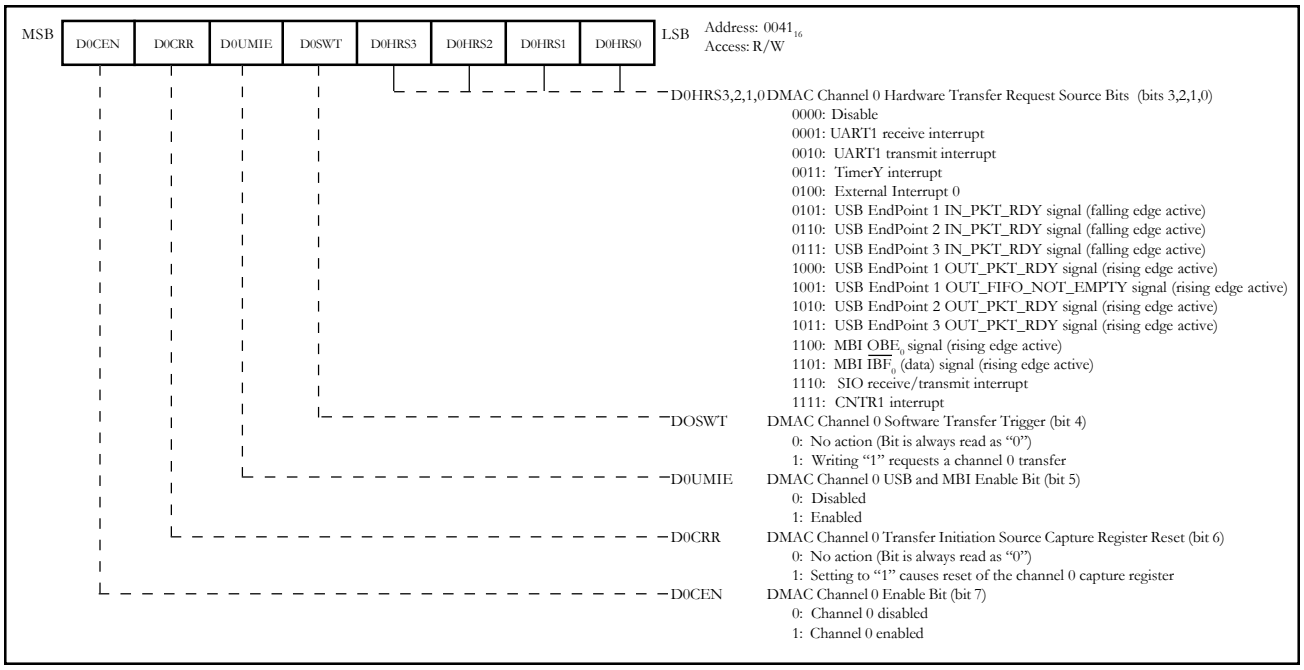


Fig. 1.84. DMAC Channel 0 Mode Register 2 (DMA0M2)

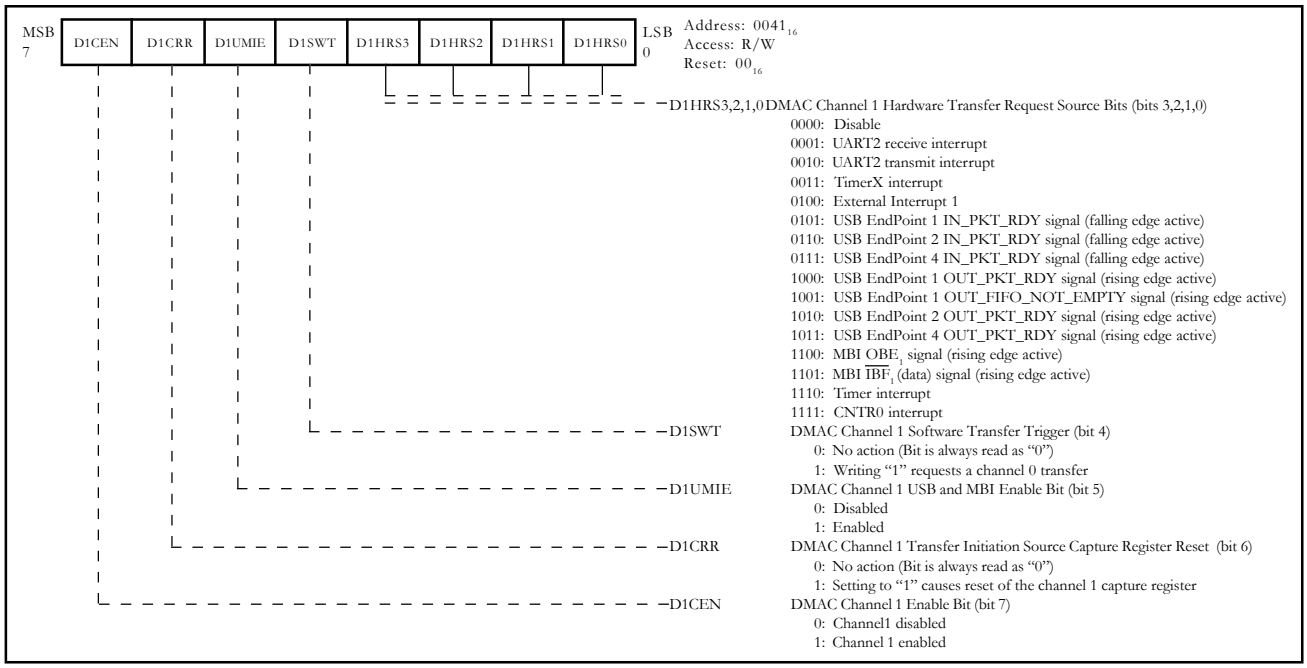


Fig. 1.85. DMAC Channel 1 Mode Register 2 (DMA1M2)

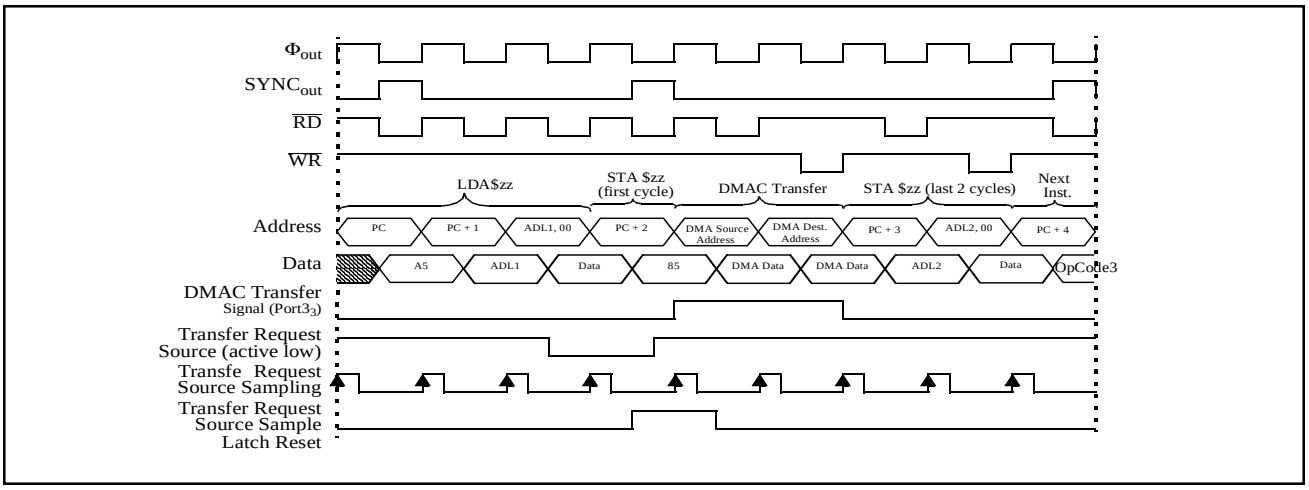


Fig. 1.86. DMAC Transfer-Hardware Source Initiated

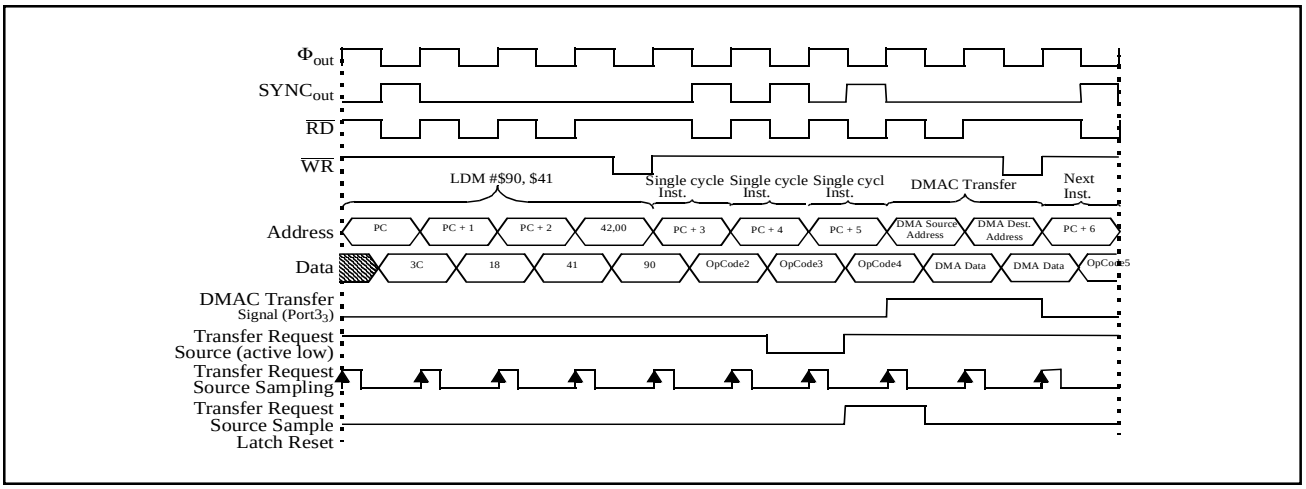


Fig. 1.87. DMAC Transfer-Software Trigger Initiated

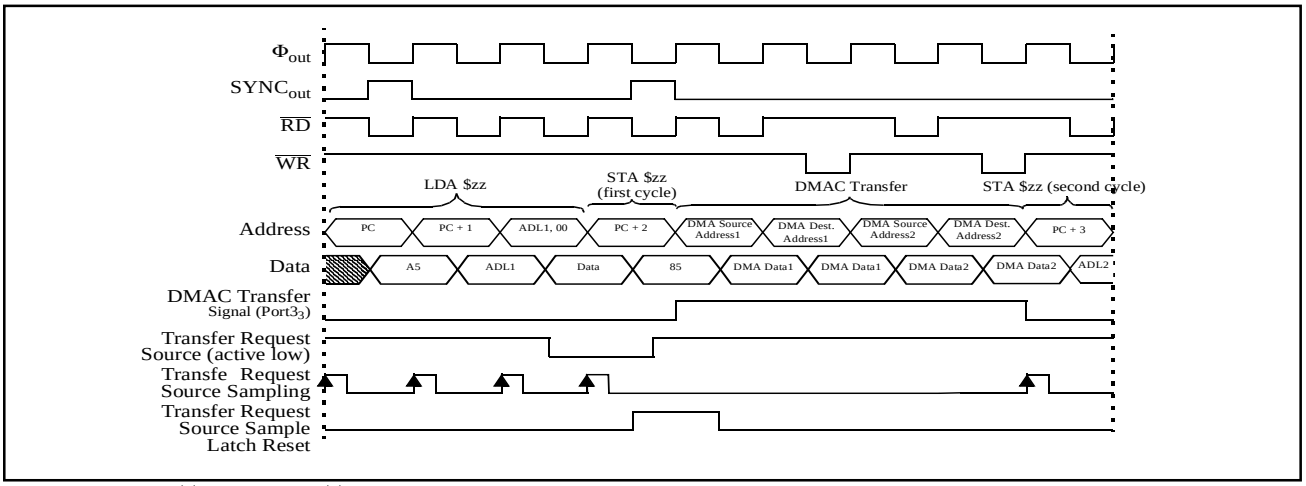


Fig. 1.88. DMAC Transfer-Burst Transfer Initiated

1.24 OSCILLATOR CIRCUIT

An on-chip oscillator provides the system and peripheral clocks as well as the USB clock necessary for operation. This oscillator circuit is comprised of amplifiers that provide the gain necessary for oscillation, oscillation control logic, a frequency synthesizer, and buffering of the clock signals.

A Clock Control register (CCR) is shown in Figure 1.89 and a flow diagram for the oscillator circuit is shown in Figure 1.90.

The following external clock inputs are supported:

- A quartz crystal oscillator of up to 24 MHz, connected to the Xin and Xout pins.
- A ceramic resonator or quartz crystal oscillator of 32.768 kHz, connected to the XCin and XCout pins.
- An external clock signal of up to 5.00 MHz, connected to the XCin pin.

The frequency synthesizer can be used to generate a 48MHz clock signal (fUSB) needed by the USB block and clock fSYN, which can be chosen as the source for the system and peripheral clocks. Both fUSB and fSYN are phase-locked frequency multiples of the frequency synthesizer input. The inputs to the frequency synthesizer can be either Xin or XCin.

The two-phase non-overlapping system clock (CPU and peripherals) is derived from the source to the clock circuit and is half the frequency of the source. (i.e. Source = 24 MHz, system clock = 12 MHz) Any one of four clock signals can be chosen as the source for the system and peripheral clocks; f(Xin)/2, f(Xin), f(XCin), or fSYN. The selection is based on the values of bits CPMA6, CPMA7 and CCR7. The default source after reset is fXin/2.

The default source for the system and peripheral clocks is f(Xin)/2. If f(Xin)= 24MHz, then the CPU will be running at Φ = 6MHz (low frequency mode. For the CPU to run in high frequency mode, i.e., source of clock = f(Xin), write a “1” to bit 7 of the clock control register. (If an external clock signal is input to Xin or XCin, the inverting amplifiers can be disabled by means of the CCR6 and CCR7 bits, respectively, in order to reduce power consumption).

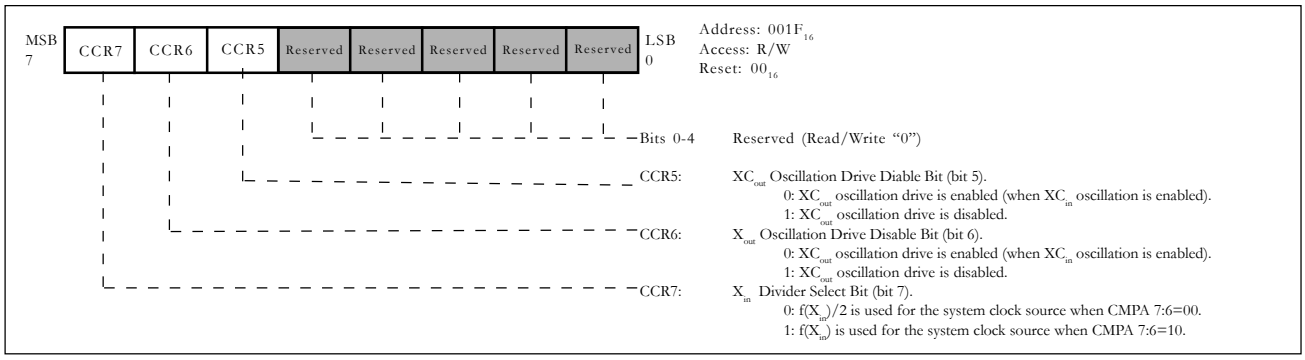
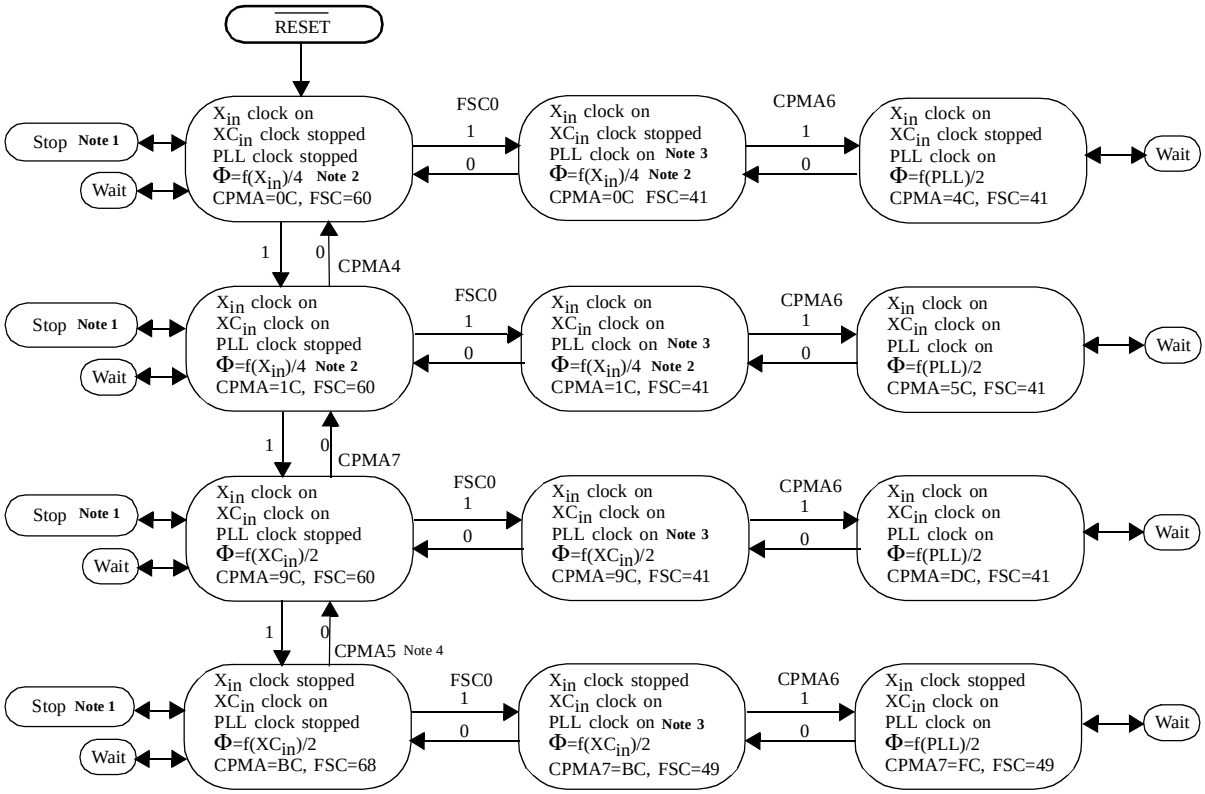


Fig. 1.89. Clock Control Register (CCR)



Note 1: Stop mode stops the oscillators that are also the inputs to the frequency synthesizer. However, the frequency synthesizer is not disabled and so its output is unstable. So, always set the system clock to an external oscillator and disable the frequency synthesizer before entering stop mode.

Note 2: $\Phi = f(X_{in})/4$ can be interchanged with $\Phi = f(X_{in})/2$ by setting CCR7 to "1". The same flow chart applies to both cases.

Note 3: The input to the frequency synthesizer is independent of the system clock. It can be either X_{in} or XC_{in} depend ing on bit 3 of FSC. In the above flow, the input has been chosen to be the same as the system clock only for simplicity. The oscillator selected to be the input to the frequency synthesizer must be enable before the frequency synthesizer is enabled.

Note 4: The input clock for the frequency synthesizer must be set to XC_{in} by setting FIN (bit 3 of FSC) to a "1" before X_{in} oscillation can be disabled.

Note: CPMA values shown assume single-chip mode with stack in one page.

Fig. 1.90. Clock Flow Diagram

1.24.1 Frequency Synthesizer Circuit

The Frequency Synthesizer Circuit generates a 48MHz clock needed by the USB block and a clock f_{SYN} that are both a multiple of the external input reference clock f_{IN}. A block diagram of the circuit is shown in Figure 1.91.

The frequency synthesizer consists of a prescaler, frequency multiplier macro, a frequency divider macro, and four registers, namely FSM1, FSM2, FSC and FSD. Two multiply registers (FSM1, FSM2) control the frequency multiply amount. Clock f_{IN} is prescaled using FSM2 to generate f_{PIN}. f_{PIN} is multiplied using FSM1 to generate an f_{VCO} clock which is then divided using FSD to produce the clock f_{SYN}. The f_{VCO} clock is optimized for 48 MHz operation and is buffered and sent out of the frequency synthesizer block as signal f_{USB}. This signal is used by the USB block. The clock block diagram is shown in Figure 1.92.

Clock f_{PIN} is a divided down version of clock f_{IN}, which can be either f(X_{in}) or f(XC_{in}). The default clock after reset is fXin. The relationship between f_{PIN} and the clock input to the prescaler (f_{IN}) is as follows:

- $f_{PIN} = f_{IN}/2(n+1)$ where n is a decimal number between 0 and 254. (See Figure 1.95).
  Setting FSM2 to 255 disables the prescaler and $f_{PIN} = f_{IN}$.

The relationship between f_{PIN}, f_{VCO}, f_{SYN}, and f_{USB} is as follows:

- $f_{VCO} = f_{PIN} \times 2(n+1)$ where n is the decimal equivalent of the value loaded in FSM1. (See Figure 1.94).
  n must be chosen such that f_{VCO} equals 48 MHz.
- $f_{SYN} = f_{VCO} / 2(m+1)$ where m is the decimal equivalent of the value loaded in FSD. (See Figure 1.96).
  Setting m=255 disables the divider and disables f_{SYN}.
- f_{USB} is a buffered version of f_{VCO}, i.e., FSD has no effect on f_{USB}.
  Setting USB control register bit 5 to "0" disables f_{USB} by tri-stating the buffer.

The FSC0 bit in the FSC Register (FSC) enables the frequency synthesizer block. When disabled (FSC0 = "0"), f_{VCO} is held at either a high or low state. When the frequency synthesizer control bit is active (FSC0 = "1"), a lock status (LS = "1") indicates that f_{SYN} and fVCO are the correct frequency. The LS and FSCO control bits in the FSC register are shown in Figure 1.93.

When using the frequency synthesizer, a low-pass filter must be connected to the LPF pin.

Once the frequency synthesizer is enabled, a delay of 2-5ms is recommended before the output of the frequency synthesizer is used. This is done to allow the output to stabilize. It is also recommended that none of the registers be modified once the frequency synthesizer is enabled as it will cause the output to be temporarily (2-5ms) unstable.

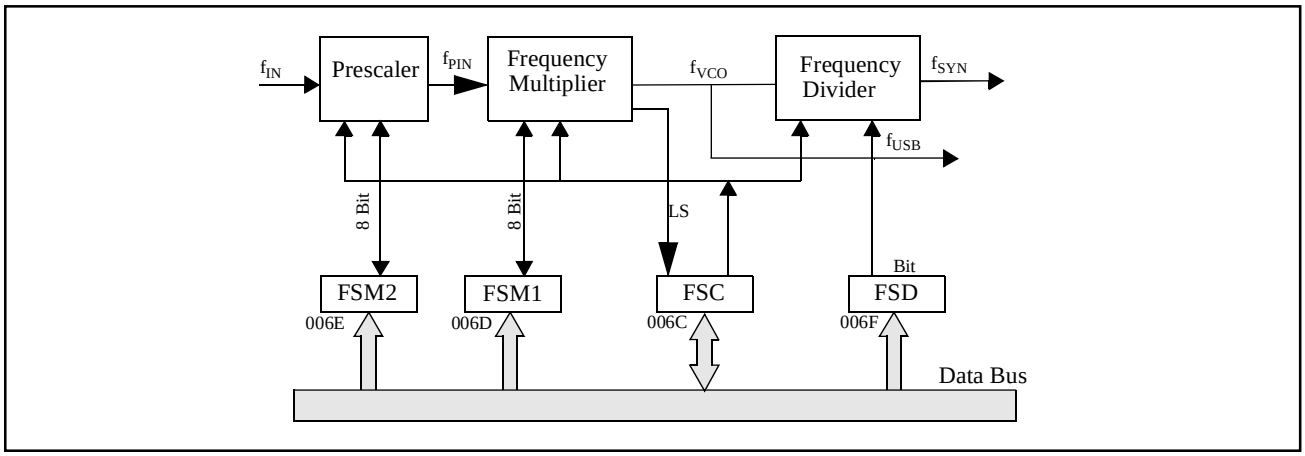


Fig. 1.91. Frequency Synthesizer Circuit

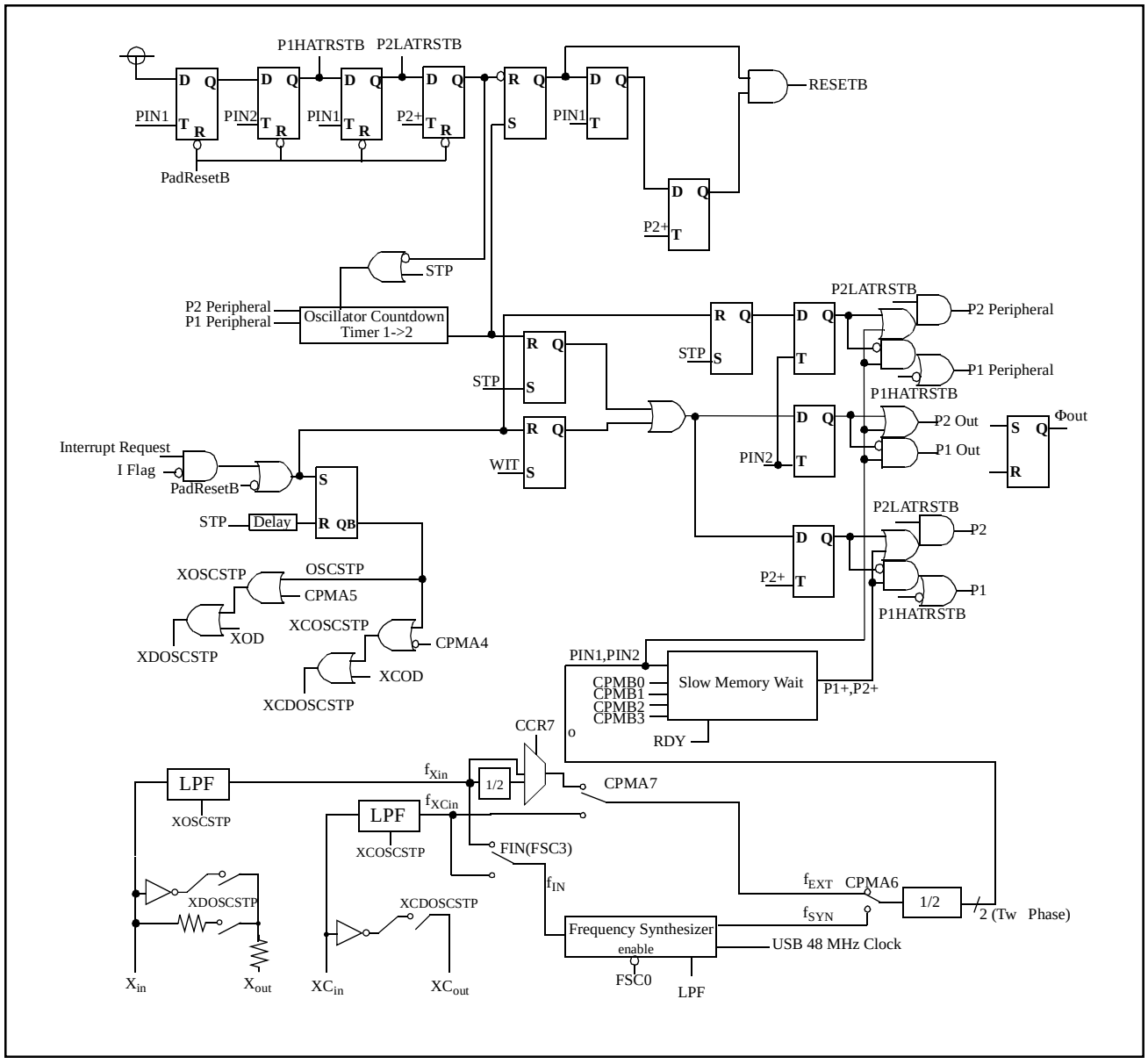


Fig. 1.92. Clock Block Diagram

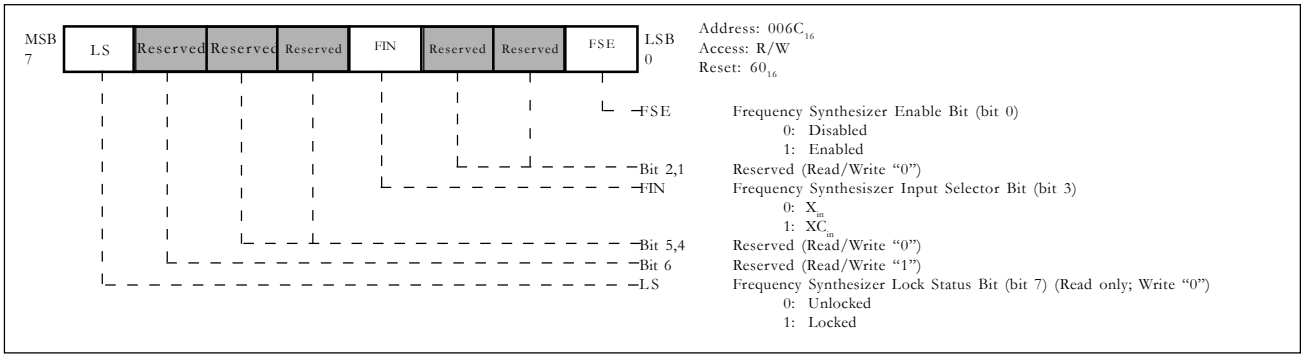


Fig. 1.93. Frequency Synthesizer Control Register (FSC)

Address: $006D_{16}$
Access: R/W
Reset: FF_{16}

Frequency synthesizer multiply value, n, that is used to multiply the prescaler output frequency, f_{PIN}, up to 48.00 MHz

f_{PIN}	FSM1		f_{VCO}
	n_{10}	n_{16}	
320.00 KHz	74	4A	48.00 MHz
2.00 MHz	11	0B	48.00 MHz
4.00 MHz	5	05	48.00 MHz
6.00 MHz	3	03	48.00 MHz
12.00 MHz	1	01	48.00 MHz
24.00 MHz	0	00	48.00 MHz

$f_{VCO} = 2 \times f_{PIN} \times (n+1)$

Fig. 1.94. Frequency Synthesizer Multiply Control Register (FSM1)

Address: $006E_{16}$
Access: R/W
Reset: FF_{16}

Frequency synthesizer prescaler divide value, n. The input clock, f_{IN}, is divided by this value to produce the intermediate frequency, f_{PIN}, which is then multiplied up to 48.00 MHz. A value of 255 (FF_{16}) turns the prescaler off (no division of f_{IN}).

f_{IN}	FSM2		f_{PIN}
	n_{10}	n_{16}	
24.00 MHz	255	FF	24.00 MHz
24.00 MHz	11	0B	1.00 MHz
24.00 MHz	5	05	2.00 MHz
24.00 MHz	3	03	3.00 MHz
24.00 MHz	1	01	6.00 MHz
24.00 MHz	0	00	12.00 MHz

$f_{PIN} = f_{IN} / (2 \times (n+1))$

Fig. 1.95. Frequency Synthesizer Multiply Control Register (FSM2)

Address: $006F_{16}$
Access: R/W
Reset: FF_{16}

Frequency synthesizer divide value, m, by which the 48.00 MHz VCO frequency is divided to produce the F_{SYN} system clock frequency.

f_{VCO}	FSD		f_{SYN}
	m_{10}	m_{16}	
48.00 MHz	0	00	24.00 MHz
48.00 MHz	2	02	8.00 MHz
48.00 MHz	5	05	4.00 KHz
48.00 MHz	127	7F	187.50 KHz
48.00 MHz	128	80	93.75 KHz
48.00 MHz	255	FF	0.00 KHz

$f_{SYN} = f_{VCO} / (2 \times (m+1))$

Fig. 1.96. Frequency Synthesizer divided Ratio Register (FSD)

1.25 LOW POWER MODES

This device has two low-power dissipation modes:

- Stop
- Wait

1.25.1 Stop Mode

Use of the stop mode allows the MCU to be placed in a state where no internal excitation of the circuitry is taking place, thus resulting in extremely low power dissipation. The MCU enters the stop mode when the STP instruction is executed. The internal state of the mcu after execution of the STP instruction is as follows:

- Timer 1 and Timer 2 are loaded with FF_{16} and 01_{16} respectively.
- All T123M mode register bits are reset to their default value except bit 4.
- The count source for Timer 1 is set to Φ/8 and the count source for Timer 2 is set to Timer 1 underflow.

Oscillation is restarted when a reset or an external interrupt is received. The interrupt control bit of the interrupt used to release the stop mode must be set to a "1" and the I flag set to a "0" prior to the execution of the STP instruction. To allow the oscillation source time to stabilize, the oscillation source is connected as the clock source for the wake-up timer (Timer 1 and Timer 2 cascaded). When Timer 2 underflows, the MCU services the interrupt that caused the return from the stop state. Afterwards, it services any other enabled interrupts that occurred, in the order of their respective priorities, and returns to its state prior to the execution of the STP instruction. The timing for the STP instruction is shown in Figure 1.97.

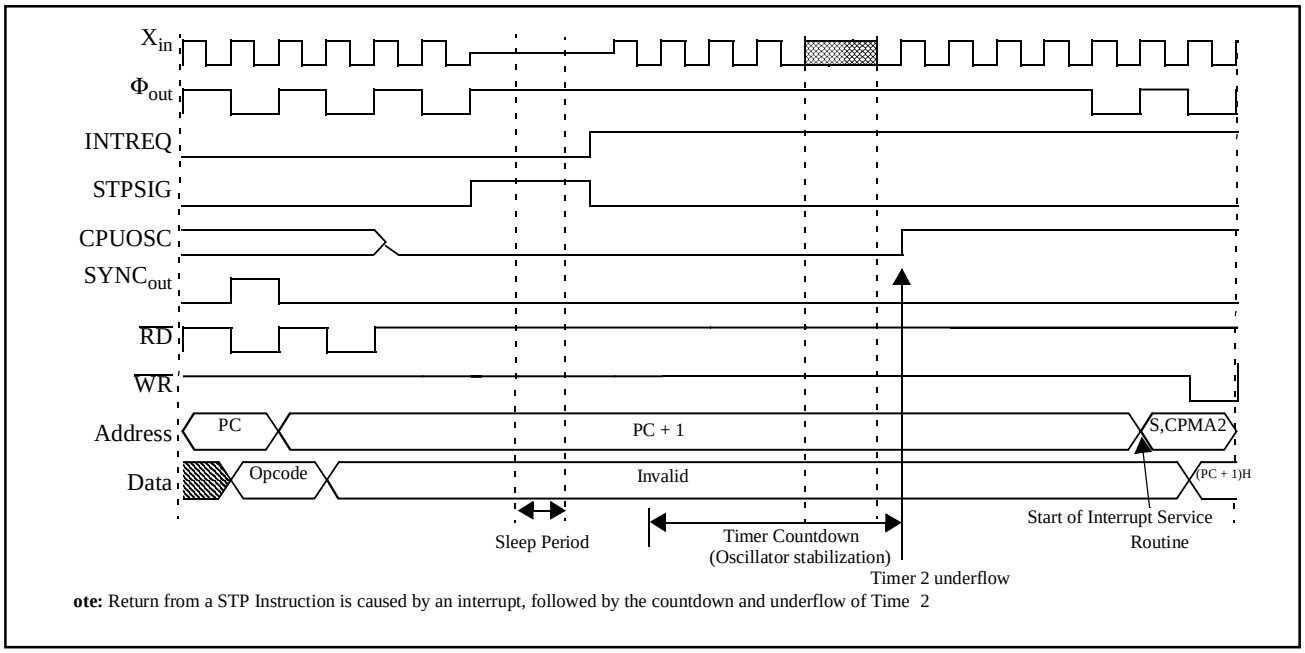


Fig. 1.97. STP Cycle Timing Diagram (STP)

1.25.2 Wait Mode

Use of the wait mode allows the microcomputer to be placed in a state where excitation of the CPU is stopped, but the clocks to the peripherals continue to oscillate. This mode provides lower power dissipation during the idle periods and quick wake-up time. The microcomputer enters the wait mode when the WIT instruction is executed.

Returning from wait mode is accomplished just as it is when returning from stop mode, with the exception that you need not provide time for the oscillator to stabilize, because the oscillation never stopped. Additionally, any peripheral interrupt can be used to bring the microcomputer out of the wait mode. The timing for the WIT instruction is shown in Figure 1.98.

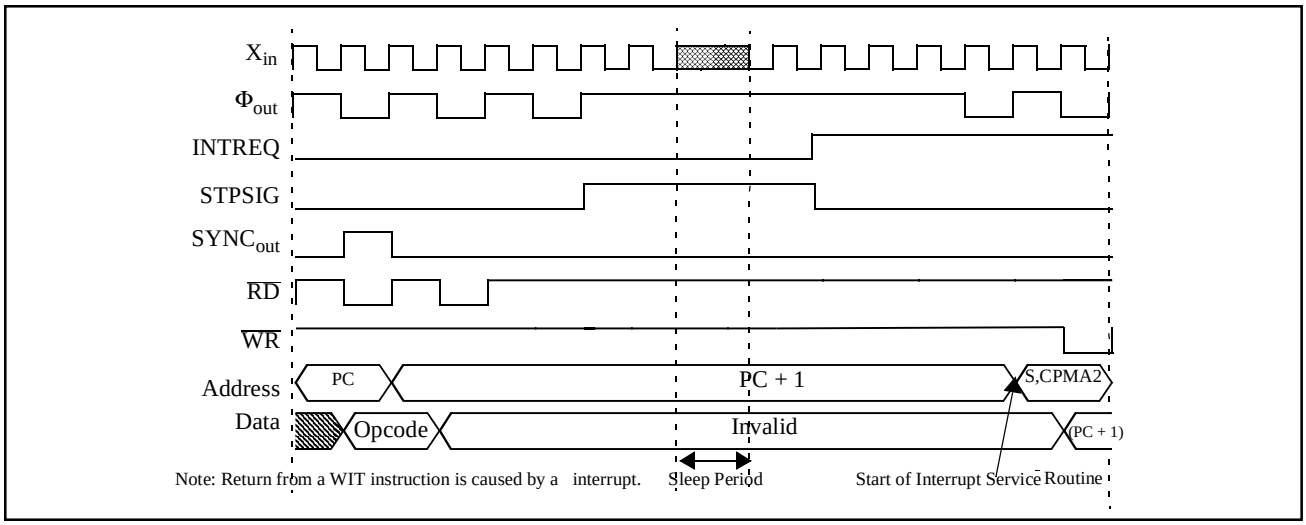


Fig. 1.98. WIT Cycle Timing Diagram (WIT)

1.26 RESET

This device is reset if the $\overline{\text{RESET}}$ pin is held low for a minimum of 2μs while the supply voltage is set between 4.15 and 5.25V. When the $\overline{\text{RESET}}$ pin returns high, the reset sequence commences (see Figure 1.99). To allow the oscillation source the time to stabilize, a delay is generated by the countdown of Timer 1 and Timer 2 cascaded with FF_{16} loaded in Timer 1 and 01_{16} loaded in Timer 2. After the reset sequence completes, program execution begins at the address whose high-order byte is the contents of address $FFFA_{16}$ and whose low-order byte is the contents of address $FFFB_{16}$.

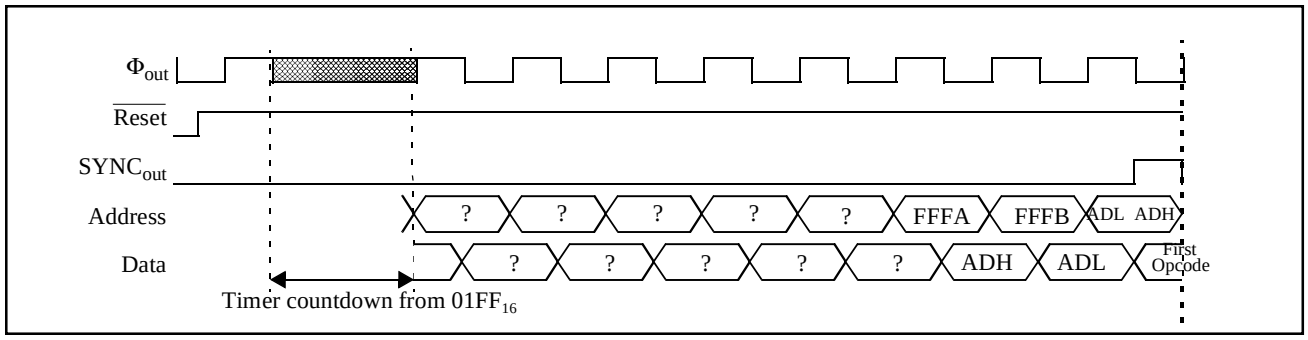


Fig. 1.99. Internal Processing Sequence after $\overline{\text{RESET}}$

2.1 ABSOLUTE MAXIMUM RATINGS

Table 2.1 Absolute Maximum Ratings

Symbol	Parameter	Conditions	Limits	Unit
V_{CC}	Power supply		-0.3 to 6.5	V
AV_{CC}	Analog power supply		-0.3 to V_{CC} + 0.3	V
V_I	Input voltage P0, P1, P2, P3, P4, P5, P6, P7, P8	Values are with respect to V_{SS}. Output transistors are in off state.	-0.3 to V_{CC} + 0.3	V
V_I	Input voltage $\overline{RESET}$, X_{in}, XC_{in}		-0.3 to V_{CC} + 0.3	V
V_I	Input voltage CNV_{SS}		-0.3 to 13	V
V_I	Input voltage USB D+, D-		-0.5 to 3.8	V
V_O	Output voltage P0, P1, P2, P3, P4, P5, P6, P7, P8, X_{out}, XC_{out}, LPF		-0.3 to V_{CC} + 0.3	V
V_O	Output voltage USB D+, D-		-0.5 to 3.8	V
P_D	Power dissipation (Note)	Ta = 25°C	750	mW
T_{OPR}	Operating temperature		-20 to +85	°C
T_{STG}	Storage temperature		-40 to +125	°C

Note: Maximum power dissipation is based on heat dissipation characteristics not chip power consumption.

2.2 RECOMMENDED OPERATING CONDITIONS

Table 2.2. Recommended Operating Conditions (Vcc = 4.15 to 5.25V, Vss = 0V, Ta = -20 to 85°C unless otherwise noted)

Symbol	Parameter		Limits			Unit
			Min.	Typ.	Max.	
V_{CC}	Supply voltage		4.15	5	5.25	V
AV_{CC}	Analog supply voltage		4.15	5	V_{CC}	V
V_{SS}	Supply voltage			0		V
AV_{SS}	Analog supply voltage			0		V
V_{IH}	H input voltage	$\overline{RESET}$, X_{in}, XC_{in}, CNV_{SS}	$0.8V_{CC}$		V_{CC}	V
V_{IH}	H input voltage	P0, P1, P2, P3, P4, P5, P6, P7, P8	$0.8V_{CC}$		V_{CC}	V
V_{IH}	H input voltage	P2 (When PTC6 = "0")	$0.5V_{CC}$		V_{CC}	V
V_{IH}	H input voltage	$P5_7$-$P5_4$, P6, $P7_2$ (When MBI inputs and PTC7 = "1")	2.0		V_{CC}	V
V_{IH}	H input voltage	USB D+, D-	2.0		3.8	V
V_{IL}	L input voltage	$\overline{RESET}$, X_{in}, XC_{in}, CNV_{SS}	0		$0.2V_{CC}$	V
V_{IL}	L input voltage	P0, P1, P2, P3, P4, P5, P6, P7, P8	0		$0.2V_{CC}$	V
V_{IL}	L input voltage	P2 (When PTC6 = "0")	0		$0.16V_{CC}$	V
V_{IL}	L input voltage	$P5_7$-$P5_4$, P6, $P7_2$ (When MBI inputs and PTC7 = "1")	0		0.8	V
V_{IL}	L input voltage	USB D+, D-	0		0.8	V
I_{OL} (peak)	L peak output current Note 1	P0, P1, P2, P3, P4, P5, P6, P7, P8			10	mA
I_{OL} (avg)	L average output current Note 2	P0, P1, P2, P3, P4, P5, P6, P7, P8			5	mA
I_{OH} (peak)	H peak output current Note 1	P0, P1, P2, P3, P4, P5, P6, P7, P8			-10	mA
I_{OH} (avg)	H average output current Note 2	P0, P1, P2, P3, P4, P5, P6, P7, P8			-5	mA
ΣI_{OL} (peak)	L total peak output current Note 3	P0, P1, P2, P3, P4, P5, P6, P7, P8			80	mA
ΣI_{OL} (avg)	L total average output current Note 4	P0, P1, P2, P3, P4, P5, P6, P7, P8			40	mA
ΣI_{OH} (peak)	H total peak output current Note 3	P0, P1, P2, P3, P4, P5, P6, P7, P8			-80	mA
ΣI_{OL} (avg)	H total average output current Note 4	P0, P1, P2, P3, P4, P5, P6, P7, P8			-40	mA
f(CNTR0)	TimerX - input frequency Note 5				5	MHz
f(CNTR1)	TimerY - input frequency Note 5				5	MHz
$f(X_{in})$	Clock frequency Note 5,7		4		24	MHz
$f(XC_{in})$	Clock frequency Note 5,6			32.768	50/5.0	KHz/MHz

Note 1. The peak output current is the peak current flowing through any pin of the listed ports.
Note 2. The average output current is an average current value measured over 100 ms.
Note 3. The total peak output current is the peak current flowing throught all pins of the listed ports.
Note 4. The total average output current is an average current value measured over 100 ms.
Note 5. The oscillation frequency has a 50% cycle.
Note 6. The maximum oscillation frequency of 50 KHz is for a crystal oscillator connected between XCin and XCout. An external signal having a maximum frequency of 5 MHz can be input to XCin.
Note 7. When using Frequency Synthesizer Circuit, minimum limit is 4 MHz. And when using USB, put internal clock f(Φ)to more than 6 MHz.

2.3 ELECTRICAL CHARACTERISTICS

Table 2.3. Electrical Characteristics (Vcc = 4.15 to 5.25V, Vss = 0V, Ta = -20 to 85°C unless otherwise noted)

Symbol	Parameters		Test Conditions	Limits			Unit
				Min	Typ.	Max	
V_{OH}	H output voltage	P0, P1, P2, P3, P4, P5, P6, P7, P8	Ioh = -10mA	V_{CC} - 2.0			V
V_{OH}	H output voltage	USB D+, D-	USB D+, USB D- pins pull down to Vss by 15 KΩ ± 5% and USB D+ pin pull up to ExtCap pin by 1.5 KΩ ± 5%	2.8		3.6	V
V_{OL}	L output voltage	P0, P1, P2, P3, P4, P5, P6, P7, P8	Iol = 10mA			2.0	V
V_{OL}	L output voltage	USB D+, D-	USB D+, USB D- pins pull down to Vss by 15 KΩ ± 5% and USB D+ pin pull up to ExtCap pin by 1.5 KΩ ± 5%			0.3	V
V_T + ~V_T-	Hysteresis	CNTR0, CNTR1, INT0, INT1, RDY, HOLD, P2			0.5		V
		URXD1, URXD2 (SCLK), CTS2 (SRXD), SRDY, CTS1			0.5		V
		RESET			0.5		V
I_{IH}	H input current	P0, P1, P2, P3, P4, P5, P6, P7, P8	$V_i = V_{CC}$			5	µA
		RESET, USB D+, USB D-, CNV_{SS}				5	µA
		X_{in}			9	20	µA
		XC_{in}				5	µA
I_{IL}	L input current	P0, P1, P3, P4, P5, P6, P7, P8	$V_i = V_{SS}$			-5	µA
		P2	$V_i = V_{SS}$ (Pullups off)			-5	µA
			V_{CC} = 5V, $V_i = V_{SS}$ (Pullups on)	-30	-75	-140	µA
		RESET, USB D+, USB D-	$V_i = V_{SS}$			-5	µA
		CNV_{SS}				-20	
		X_{in}			-9	-20	µA
		XC_{in}				-5	µA
V_{RAM}	RAM retention voltage		Clocks stopped	2.0		5.25	V
I_{CC}	Supply current (Output transistors are isolated)	Normal Mode	$f(X_{in})$ = 24MHz, Φ = 12MHz, USB operating, frequency synthesizer on, Note 1		70	90	mA
		Wait Mode	$f(X_{in})$ = 24MHz, Φ = 12MHz, USB suspended, frequency synthesizer on, USB clock disabled Note 2		7.5	10	mA
			$f(XC_{in})$ = 32KHz, Φ = 16KHz, USB disabled, frequency synthesizer off, transceiver voltage converter off Note 3		6	10	µA
			Transceiver voltage converter on with USBC3 = "1" (low current mode)		200	250	µA
		Stop Mode	T_a = 25°C, transceiver voltage converter off		0.1	1	µA
			T_a = 85°C, transceiver voltage converter off			10	µA

Note 1: Icc test conditions

Single chip mode (run state)
Square wave clock input on X_{in} (X_{out} drive disabled)
I/O pins isolated
Frequency synthesizer running
USB operating with transceiver voltage converter enabled
CPU and DMAC running
Timers and SCSG running
BothUARTs transmitting
MBI and SIO disabled

Note 2: Icc test conditions

Single chip mode (wait state)
Square wave clock input on X_{in} (X_{out} drive disabled)
I/O pins isolated
Frequency synthesizer running
USB in suspend state with USB clock disabled
Transceiver voltage converter enabled
Timers and SCSG running
CPU and DMAC not running
Both UARTs, SIO, and MBI disabled

Note 3: Icc test conditions

Single chip mode (wait state)
X_{in}/X_{out} oscillation disabled
Square wave clock input on XC_{in} (XC_{out} drive disabled)
I/O pins isolated
Frequency synthesizer disabled
USB and USB clock disabled
Transceiver voltage converter disabled
Timers and SCSG running
CPU and DMAC not running
Both UARTs, SIO, and MBI disabled

2.4 TIMING REQUIREMENTS AND SWITCHING CHARACTERISCTICS

Table 2.4 Timing Requirements and Switching Characterisctics
(Vcc = 4.15 to 5.25V, Vss = 0V, Ta = -20 to 85°C unless otherwise noted)

Symbol	Parameter	Limits			Unit
		Min	Typ.	Max	
Inputs					
tw($\overline{RESET}$)	$\overline{RESET}$ input "Low" pulse width	2			_s
tc(X_{in})	Clock input cycle time	41.66			ns
twh(X_{in})	Clock input "High" pulse width	0.4*tc(X_{in})			ns
twl(X_{in})	Clock input "Low" pulse width	0.4*tc(X_{in})			ns
tc(XC_{in})	Clock input cycle time	200			ns
twh(XC_{in})	Clock input "High" pulse width	0.4*tc(XC_{in})			ns
twl(XC_{in})	Clock input "Low" pulse width	0.4*tc(XC_{in})			ns
Interrupts					
tc($\overline{INT}$)	$\overline{INT}$0, $\overline{INT}$1 input cycle time	200			ns
twh($\overline{INT}$)	$\overline{INT}$0, $\overline{INT}$1 input "High" pulse width	90			ns
twl($\overline{INT}$)	$\overline{INT}$0, $\overline{INT}$1 input "Low" pulse width	90			ns
tc(CNTRI)	CNTR0, CNTR1 input cycle time	200			ns
twh(CNTRI)	CNTR0, CNTR1 input "High" pulse width	80			ns
twl(CNTRI)	CNTR0, CNTR1 input "Low" pulse width	80			ns
Timers					
td(Φ-TOUT)	TIMER TOUT delay time (Note)			15	ns
td(Φ-CNTR0)	TIMER CNTR0 delay time (pulse output mode) (Note)			15	ns
tc(CNTRE0)	TIMER CNTR0 input cycle time (event counter mode)	200			ns
twh(CNTRE0)	TIMER CNTR0 input "High" pulse width (event counter mode)	0.4*tc(CNTRE0)			ns
twl(CNTRE0)	TIMER CNTR0 input "Low" pulse width (event counter mode)	0.4*tc(CNTRE0)			ns
td(Φ-CNTR1)	TIMER CNTR1 delay time (pulse output mode) (Note)			15	ns
tc(CNTRE1)	TIMER CNTR1 input cycle time (event counter mode)	200			ns
twh(CNTRE1)	TIMER CNTR1 input "High" pulse width (event counter mode)	0.4*tc(CNTRE1)			ns
twl(CNTRE1)	TIMER CNTR1 input "Low" pulse width (event counter mode)	0.4*tc(CNTRE1)			ns
SIO					
tc(SCLKE)	SIO external clock input cycle time	400			ns
twh(SCLKE)	SIO external clock input "High" pulse width	190			ns
twl(SCLKE)	SIO external clock input "Low" pulse width	180			ns
tsu(SRXD-SCLKE)	SIO receive setup time (external clock)	15			ns
th(SCLKE-SRXD)	SIO receive hold time (external clock)	10			ns
td(SCLKE-STXD)	SIO transmit delay time (external clock)			25	ns
tv(SCLKE-$\overline{SRDY}$)	SIO $\overline{SRDY}$ valid time (external clock)			26	ns
tc(SCLKI)	SIO internal clock output cycle time	166.66			ns
twh(SCLKI)	SIO internal clock output "High" pulse width	0.5*tc(SCLKI)-5			ns
twl(SCLKI)	SIO internal clock output "Low" pulse width	0.5*tc(SCLKI)-5			ns
tsu(SRXD-SCLKI)	SIO receive setup time (internal clock)	20			ns
th(SCLKI-SRXD)	SIO receive hold time (internal clock)	5			ns
td(SCLKI-STXD)	SIO transmit delay time (internal clock)			5	ns

Table 2.4 Timing Requirements and Switching Characterisctics (continued)
(Vcc = 4.15 to 5.25V, Vss = 0V, Ta = -20 to 85°C unless otherwise noted)

Symbol	Parameter	Limits			Unit
MBI (Separate $\overline{R}$ and $\overline{W}$ Type Mode)		Min	Typ.	Max	
tsu(S-$\overline{R}$)	$\overline{S}_0$, $\overline{S}_1$ setup time for read	0			ns
tsu(S-$\overline{W}$)	$\overline{S}_0$, $\overline{S}_1$ setup time for write	0			ns
th($\overline{R}$-S)	$\overline{S}_0$, $\overline{S}_1$ hold time for read	0			ns
th($\overline{W}$-S)	$\overline{S}_0$, $\overline{S}_1$ hold time for write	0			ns
tsu(A-$\overline{R}$)	A_0 setup time for read	10			ns
tsu(A-$\overline{W}$)	A_0 setup time for write	10			ns
th($\overline{R}$-A)	A_0 hold time for read	0			ns
th($\overline{W}$-A)	A_0 hold time for write	0			ns
tw($\overline{R}$)	Read pulse width	50			ns
tw($\overline{W}$)	Write pulse width	50			ns
tsu(D-$\overline{W}$)	Data input setup time before write	25			ns
th($\overline{W}$-D)	Data input hold time after write	0			ns
ta($\overline{R}$-D)	Data output enable time after read			40	ns
tv($\overline{R}$-D)	Data output disable time after read	10			ns
tv($\overline{R}$-OBF)	OBF output transmission time after read			40	ns
td($\overline{W}$-$\overline{IBF}$)	$\overline{IBF}$ output transmission time after write			40	ns
MBI (R/$\overline{W}$ Type Mode)					
tsu(S-E)	$\overline{S}_0$, $\overline{S}_1$ setup time	0			ns
th(E-S)	$\overline{S}_0$, $\overline{S}_1$ hold time	0			ns
tsu(A-E)	A_0 setup time	10			ns
th(E-A)	A_0 hold time	0			ns
tsu(R/$\overline{W}$-E)	R/$\overline{W}$ setup time	10			ns
th(E-R/$\overline{W}$)	R/$\overline{W}$ hold time	10			ns
tw(E)	Enable pulse width	50			ns
tw(E-E)	Enable pulse interval	50			ns
tsu(D-E)	Data input setup time before write	25			ns
th(E-D)	Data input hold time after write	0			ns
ta(E-D)	Data output enable time after read			40	ns
tv(E-D)	Data output disable time after read	10			ns
tv(E-OBF)	OBF output transmission time after E inactive			40	ns
td(E-$\overline{IBF}$)	$\overline{IBF}$ output transmission time after E inactive			40	ns

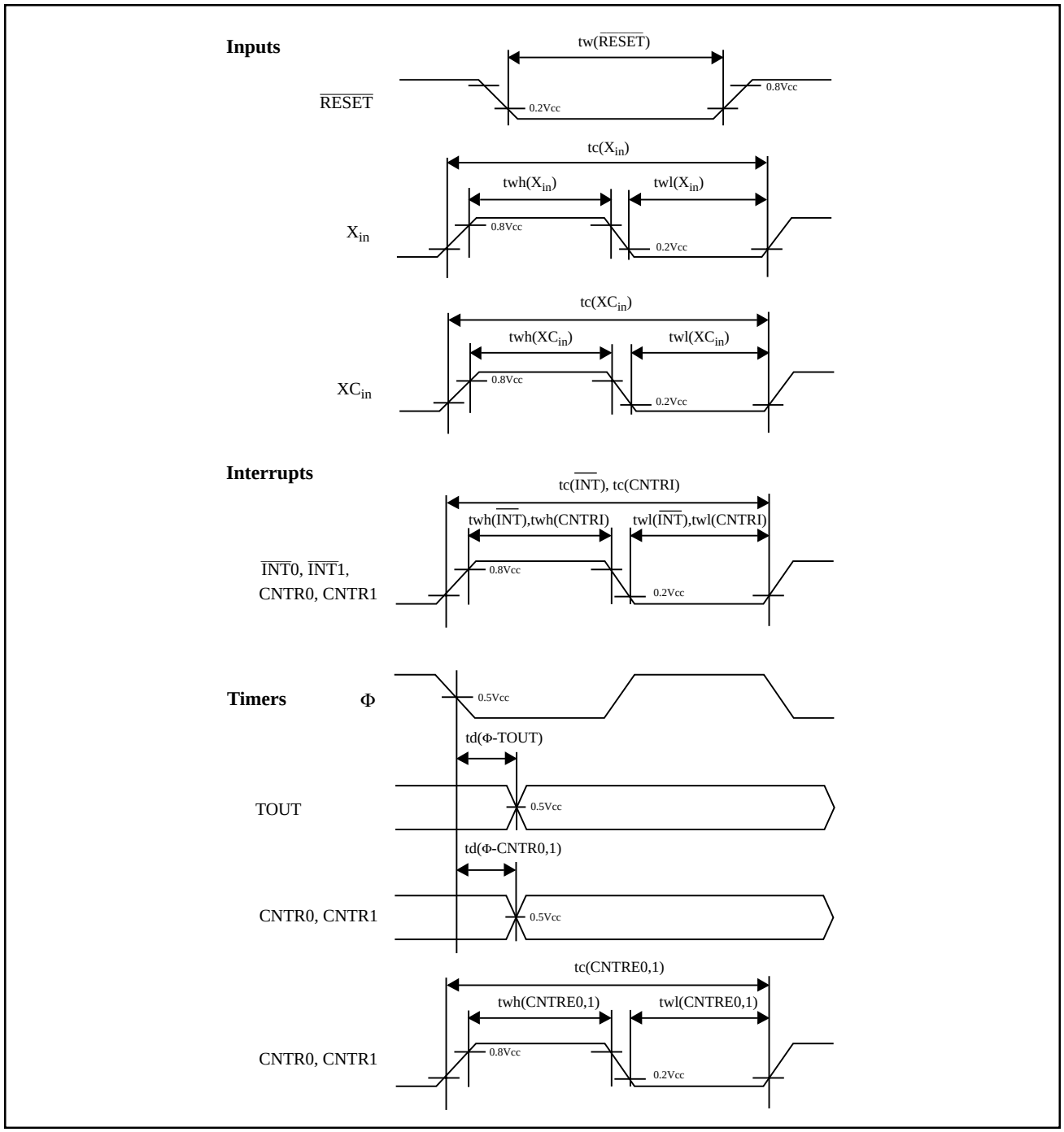
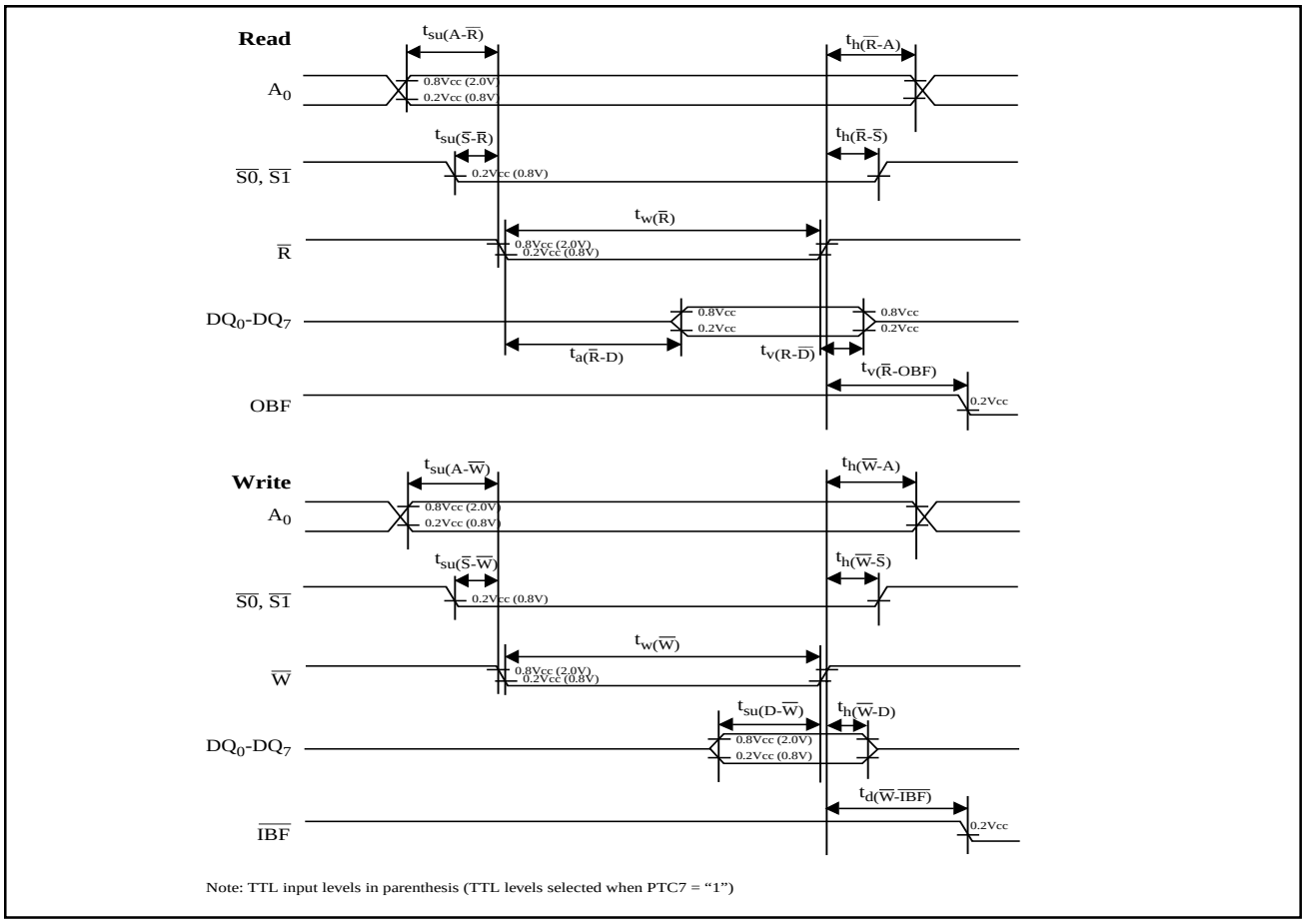


Fig. 2.1. Reset, Clock, Interrupts and Timers Timing Diagram



Note: TTL input levels in parenthesis (TTL levels selected when PTC7 = "1")

Fig. 2.2. MBI Timing Diagram (Separate $\overline{R}$ and $\overline{W}$ Type Mode)

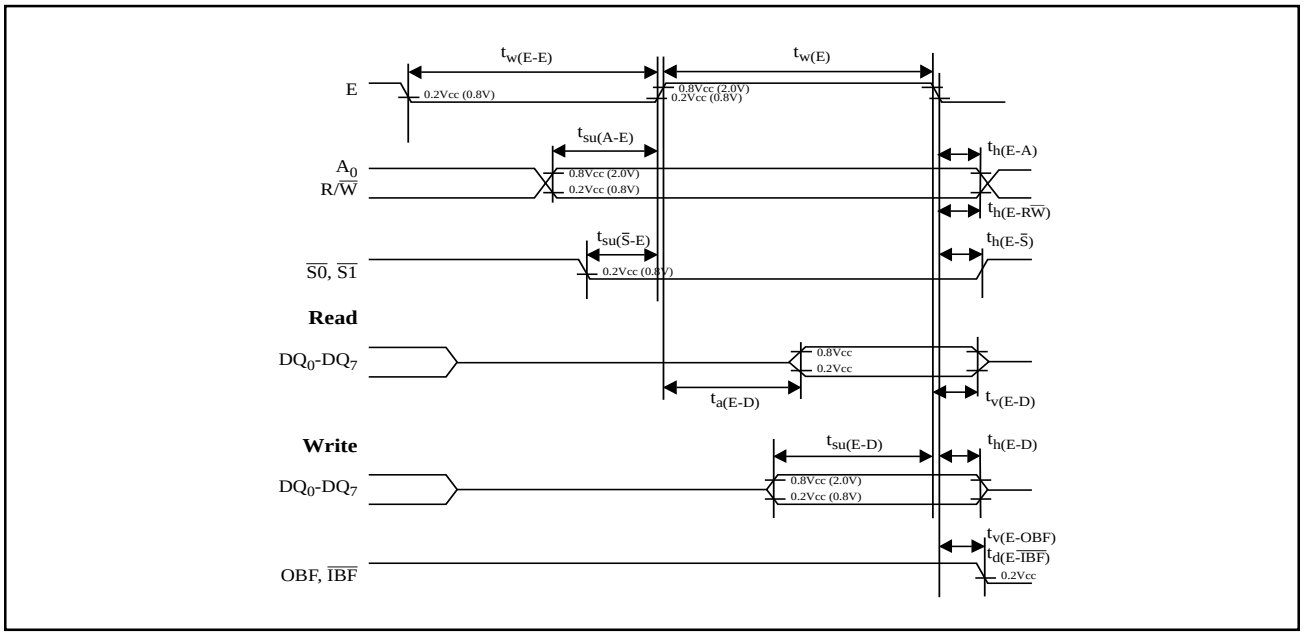


Fig. 2.3. MBI Timing Diagram ($\overline{R}$/$\overline{W}$ Type Mode)

Table 2.5. Memory Expansion Mode and Microprocessor Mode Timing
(Vcc = 4.15 to 5.25V, Vss = 0V, Ta = -20 to 85°C unless otherwise noted)

Symbol	Parameter	Limits			Unit
		Min.	Typ.	Max.	
tc(Φ)	Φ clock cycle time	83.33			ns
twh(Φ)	Φ clock "H" pulse width	0.5*tc(Φ)-5			ns
twl(Φ)	Φ clock "L" pulse width	0.5*tc(Φ)-5			ns
td(Φ-AH)	Address bus AB15-AB8 delay time with respect to Φ			31	ns
tv(Φ-AH)	Address bus AB15-AB8 valid time with respect to Φ	5			ns
td(Φ-AL)	Address bus AB7-AB0 delay time with respect to Φ			33	ns
tv(Φ-AL)	Address bus AB7-AB0 valid time with respect to Φ	5			ns
td(Φ-$\overline{WR}$)	$\overline{WR}$ delay time			6	ns
tv(Φ-$\overline{WR}$)	$\overline{WR}$ valid time	3			ns
td(Φ-$\overline{RD}$)	$\overline{RD}$ delay time			6	ns
tv(Φ-$\overline{RD}$)	$\overline{RD}$ valid time	3			ns
td(Φ-SYNC)	$SYNC_{OUT}$ delay time			6	ns
tv(Φ-SYNC)	$SYNC_{OUT}$ valid time	4			ns
td(Φ-DMA)	DMA_{OUT} delay time			25	ns
tv(Φ-DMA)	DMA_{OUT} valid time	5			ns
tsu(RDY-Φ)	RDY setup time with respect to Φ	21			ns
th(Φ-RDY)	RDY hold time with respect to Φ	0			ns
tsu($\overline{HOLD}$-Φ)	$\overline{HOLD}$ setup time	21			ns
th(Φ-$\overline{HOLD}$)	$\overline{HOLD}$ hold time	0			ns
td(Φ-$\overline{HLDAL}$)	$\overline{HLDAL}$ delay time			25	ns
tv(Φ-$\overline{HLDAH}$)	$\overline{HLDAH}$ delay time			25	ns
tsu(DB-Φ)	Data bus setup time with respect to Φ	7			ns
th(Φ-DB)	Data bus hold time with respect to Φ	0			ns
td(Φ-DB)	Data bus delay time with respect to Φ			22	ns
tv(Φ-DB)	Data bus valid time with respect to Φ (Note)	13			ns
twl($\overline{WR}$)	$\overline{WR}$ pulse width	0.5*tc(Φ)-5			ns
twl($\overline{RD}$)	$\overline{RD}$ pulse width	0.5*tc(Φ)-5			ns
td(AH-$\overline{WR}$)	$\overline{WR}$ delay time after stable address AB15-AB8	0.5*tc(Φ)-28			ns
td(AL-$\overline{WR}$)	$\overline{WR}$ delay time after stable address AB7-AB0	0.5*tc(Φ)-30			ns
tv($\overline{WR}$-AH)	Address bus AB15-AB8 valid time with respect to $\overline{WR}$	0			ns
tv($\overline{WR}$-AL)	Address bus AB7-AB0 valid time with respect to $\overline{WR}$	0			ns
td(AH-$\overline{RD}$)	$\overline{RD}$ delay time after stable address AB15-AB8	0.5*tc(Φ)-28			ns
td(AL-$\overline{RD}$)	$\overline{RD}$ delay time after stable address AB7-AB0	0.5*tc(Φ)-30			ns
tv($\overline{RD}$-AH)	Address bus AB15-AB8 valid time with respect to $\overline{RD}$	0			ns
tv($\overline{RD}$-AL)	Address bus AB7-AB0 valid time with respect to $\overline{RD}$	0			ns
tsu(RDY-$\overline{WR}$)	RDY setup time with respect to $\overline{WR}$	27			ns
th($\overline{WR}$-RDY)	RDY hold time with respect to $\overline{WR}$	0			ns
tsu(RDY-$\overline{RD}$)	RDY setup time with respect to $\overline{RD}$	27			ns
th($\overline{RD}$-RDY)	RDY hold time with respect to $\overline{RD}$	0			ns
tsu(DB-$\overline{RD}$)	Data bus setup time with respect to $\overline{RD}$	13			ns
th($\overline{RD}$-DB)	Data bus hold time with respect to $\overline{RD}$	0			ns
td($\overline{WR}$-DB)	Data bus delay time with respect to $\overline{WR}$			20	ns
tv($\overline{WR}$-DB)	Data bus valid time with respect to $\overline{WR}$ Note	10			ns
tr(D+), tr(D-)	USB output rise time, CL=50 pF	4		20	ns
tf (D+), tf(D-)	USB output fall time, CL=50 pF	4		20	ns

Note: Measurement conditions: Iohl = ±5ma, C_L = 50pF

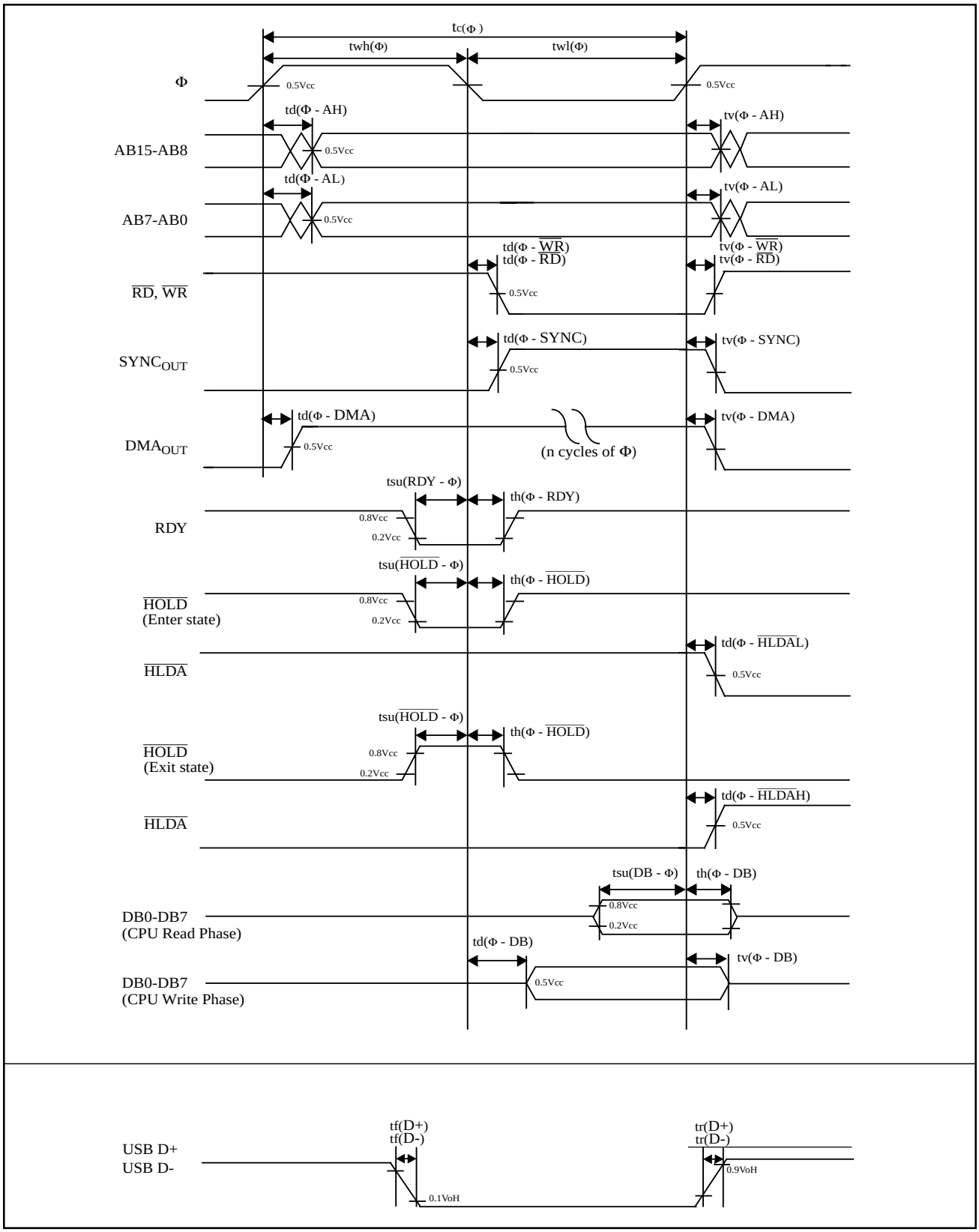


Fig. 2.4. Microprocessor and Memory Expansion Mode Timing Diagram 1

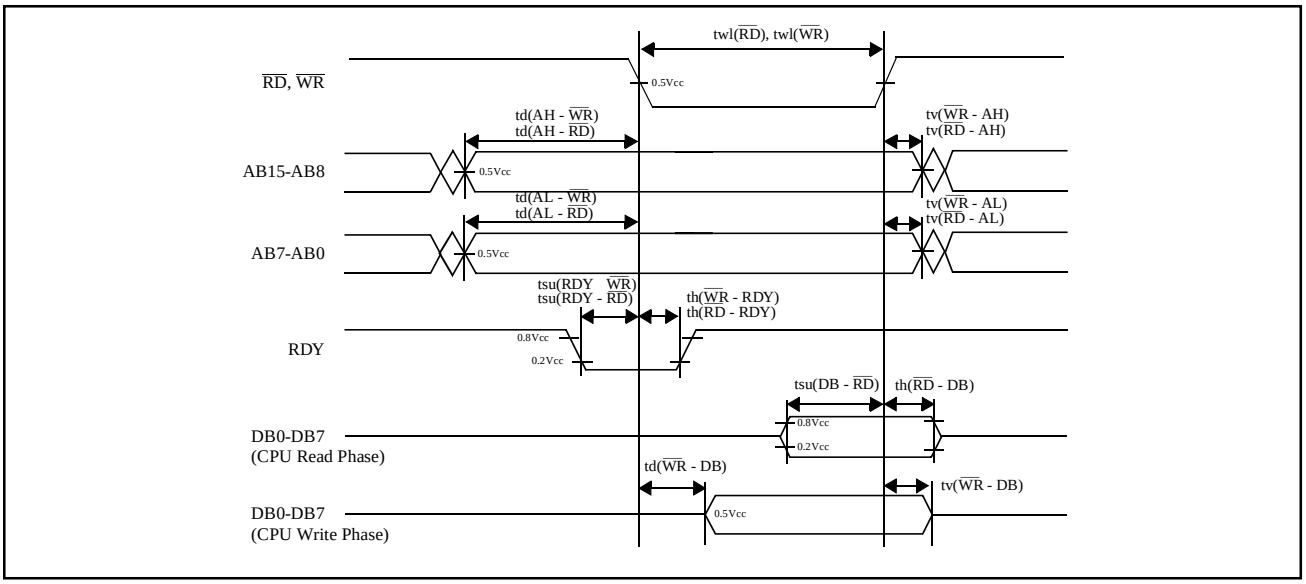


Fig. 2.5. Microprocessor and Memory Expansion Mode Timing Diagram 2

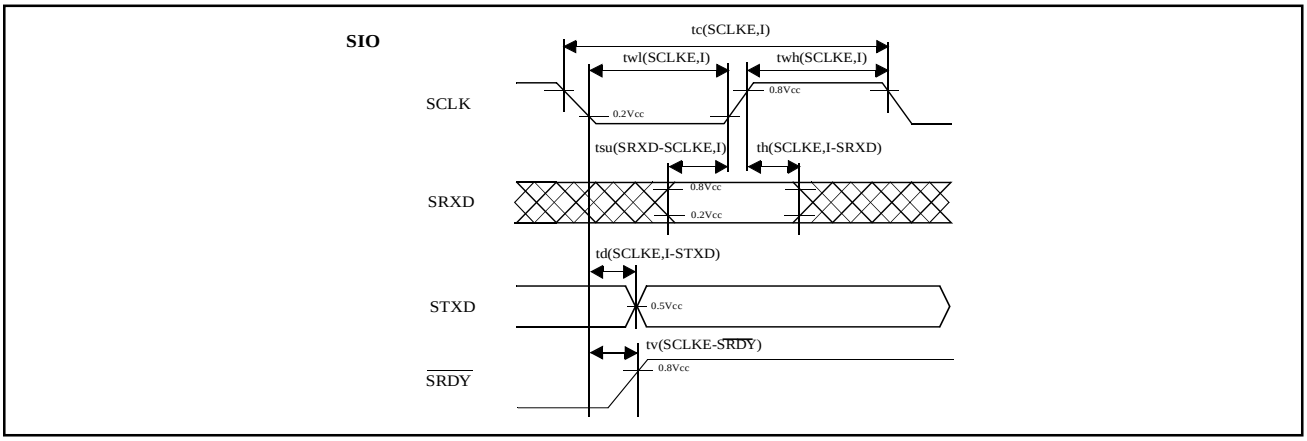


Fig. 2.6. SIO Timing Diagram

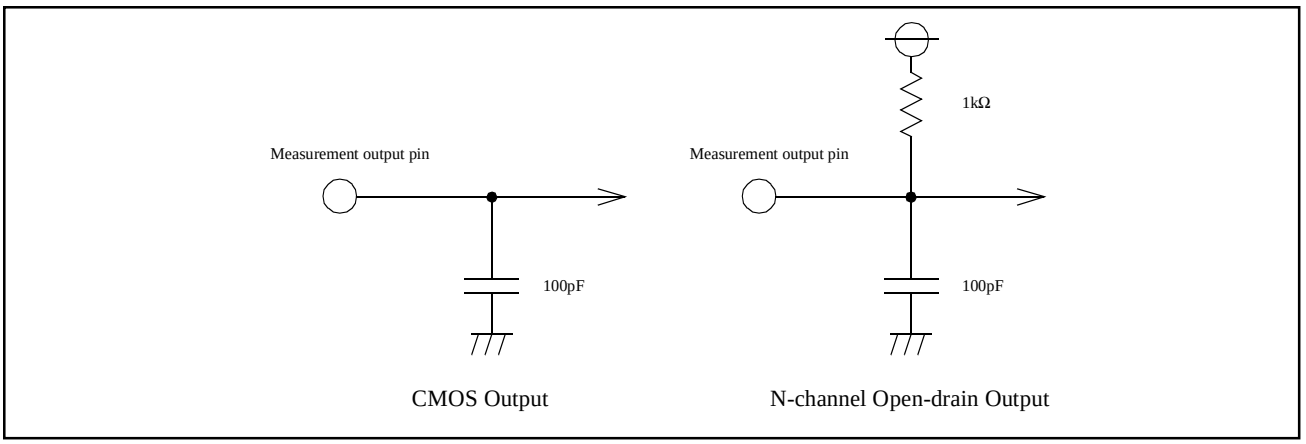


Fig. 2.7. Output Switching Characteristics Measurements Circuits

DATA REQUIRED FOR MASK ORDERS

The following are necessary when ordering a mask ROM production:

1. Mask ROM Order Confirmation Form
2. Mark Specification Form
3. Data to be written to ROM, in EPROM form three identical copies) or in floppy disk form.

ROM PROGRAMMING METHOD

The built-in PROM of the blank One-time PROM version and built-in EPROM version can be read or programmed with a general-purpose PROM programmer using a special programming adapter. (See Table 2.6).

Table 2.6. Programming adapter

Package	Name of Programming Adapter
80P6N-A	PCA7440FP
80D0	PCA7440FS

The PROM of the blank One Time PROM version is not tested or screened in the assembly process and following processes. To ensure proper operation after programming, the procedure shown in Figure 2.8 is recommended to verify programming.

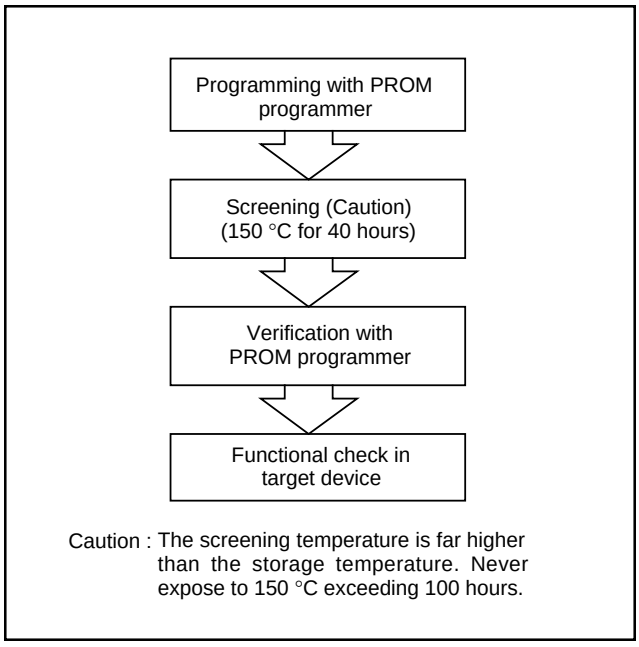


Fig. 2.8. Programming and testing of One-Time PROM version

GZZ-SH57-30B<9XA0>

Mask ROM number	

740 FAMILY MASK ROM CONFIRMATION FORM
SINGLE-CHIP MICROCOMPUTER M37640M8-XXXFP
MITSUBISHI ELECTRIC

We recommend the use of the following pseudo-command to set the start address of the assembler source program because ASCII codes of the product name are written to addresses 0000_{16} to 0008_{16} of EPROM.

EPROM type	27512
The pseudo-command	*=△ $0000 .BYTE △ 'M37640M8-'

Note : If the name of the product written to the EPROMs does not match the name of the mask confirmation form, the ROM will not be processed.

☐ **Ordering by floppy disk**

We will produce masks based on the mask file generating utility. We shall assume the responsibility for errors only if the mask ROM data on the products differs from this mask file. Therefore, extreme care must be taken to verify the accuracy of the mask file submitted.

The floppy disk must be 3.5-inch 2HD type and DOS/V format. Only one mask file per floppy disk should be submitted.

File Code ☐☐☐☐☐☐☐☐ (hexadecimal notation)

Mask file name ☐☐☐☐☐☐☐☐ .MSK (equal or less than eight characters)

*** 1. Mark specification**

Mark specification must be submitted using the correct form for the package being ordered. Fill out the appropriate mark specification form 80P6N and attach it to the mask ROM confirmation form.

*** 2. Usage conditions**

Please answer the following questions about usade for use in our product inspection.

(1) How will you use the X_{in}-X_{out} oscillator?

☐ Quartz crystal ☐ External clock input

☐ Other ( )

At what frequency? $f(X_{in})$ = ☐☐☐☐ MHz

(2) Which function will you use the pins $P5_0/XC_{in}$ and $P5_1/XC_{out}$ as $P5_0$ and $P5_1$, or XC_{in} and XC_{out} ?

☐ Ports $P5_0$ and $P5_1$ function ☐ XC_{in} and XC_{out} function (external resonator)

*** 3. Comments**

GZZ-SH57-30B<9XA0>

Mask ROM number	

740 FAMILY MASK ROM CONFIRMATION FORM
SINGLE-CHIP MICROCOMPUTER M37640M8-XXXFP
MITSUBISHI ELECTRIC

Receipt	Date:	
	Section head signature	Supervisor signature

Note : Please fill in all items marked *.

*	Customer	Company name	TEL ( )	Issuance signature	Submitted by	Supervisor
		Date issued	Date:			

*1 Confirmation

Three EPROMs are required for each pattern if this order is performed by EPROMs.

One floppy disk is required for each pattern if this order is performed by a floppy disk.

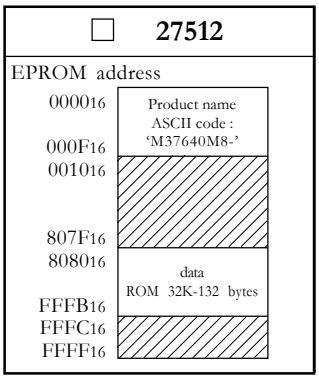
☐ Ordering by EPROMs

If at least two of the three sets of EPROMs submitted contain identical data, we will produce masks based on this data. We shall assume the responsibility for errors only if the mask ROM data on the products we produce differs from this data. Thus, extreme care must be taken to verify the data in the submitted EPROMs.

Checksum code for entire EPROM | | | | | (hexadecimal notation)

EPROM type (indicate the type used)

☐ 27512



In the address space of the microcomputer, the internal ROM area is from address 8080_{16} to $FFFB_{16}$. The reset vector is stored in addresses $FFFA_{16}$ and $FFFB_{16}$.

(1) Set the data in the unused area (the shaded area of the diagram) to "FF_{16}".
(2) The ASCII codes of the product name "M37640M8-" must be entered in addresses 0000_{16} to 0008_{16}. And set the data "FF_{16}" in addresses 0009_{16} to $000F_{16}$.
The ASCII codes and addresses are listed to the right in hexadecimal notation.

Address		
0000_{16}	'M' =	$4D_{16}$
0001_{16}	'3' =	33_{16}
0002_{16}	'7' =	37_{16}
0003_{16}	'6' =	36_{16}
0004_{16}	'4' =	34_{16}
0005_{16}	'0' =	30_{16}
0006_{16}	'M' =	$4D_{16}$
0007_{16}	'8' =	38_{16}

Address	
0008_{16}	'-' = $2D_{16}$
0009_{16}	FF_{16}
$000A_{16}$	FF_{16}
$000B_{16}$	FF_{16}
$000C_{16}$	FF_{16}
$000D_{16}$	FF_{16}
$000E_{16}$	FF_{16}
$000F_{16}$	FF_{16}

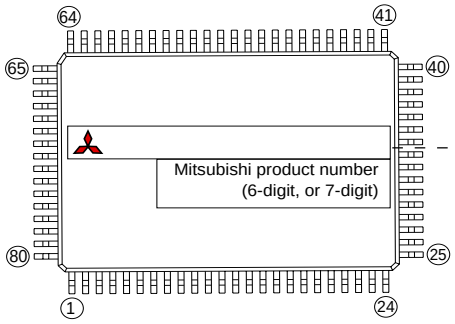
80P6n (80-PIN QFP) MARK SPECIFICATION FORM

Mitsubishi IC catalog name

Please choose one of the marking types below (A, B, C), and enter the Mitsubishi IC catalog name and the special mark (if needed).

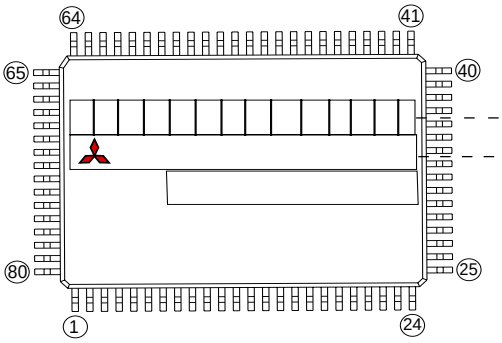
Notes

A. Standard Mitsubishi Mark



Mitsubishi IC catalog name

B. Customer's Parts Number + Mitsubishi IC Catalog Name



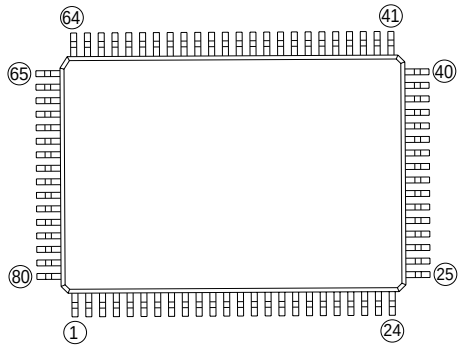
Customer's Parts Number
Note: The fonts and size of characters are standard Mitsubishi type.
Mitsubishi IC catalog name

Notes

1: The mark field should be written right aligned.
2: The fonts and size of characters are standard Mitsubishi type.
3: Customer's parts number can be up to 14 alphanumeric characters for capital letters, hyphens, commas, periods and so on.
4: If the Mitsubishi logo is not required, check the box below.

☐ Mitsubishi logo is not required

C. Special Mark Required



1: If special mark is to be printed, indicate the desired layout of the mark in Figure C. The layout will be duplicated technically as close as possible. Mitsubishi product number (6-digit, or 7-digit) and Mask ROM number (3-digit) are always marked for sorting the products.

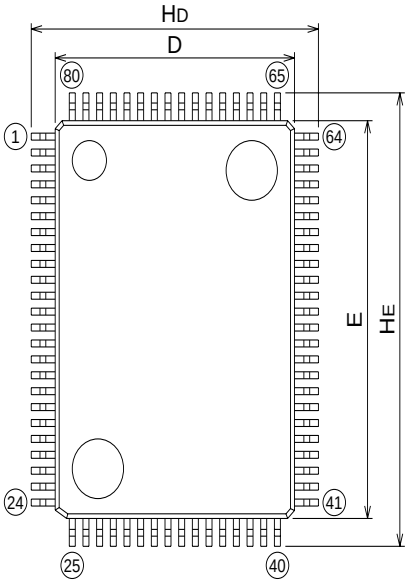
2: If special character fonts (e,g., customer's trade mark logo) must be used in Special Mark, check the box below. For the new special character fonts, a clean font original (ideally logo drawing) must be submitted.

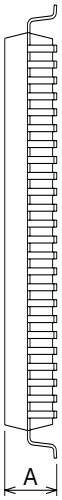
☐ Special character fonts required

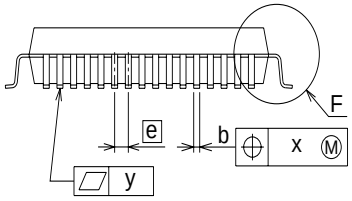
80P6N-A

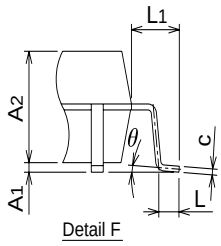
Plastic 80pin 14X20mm body QFP

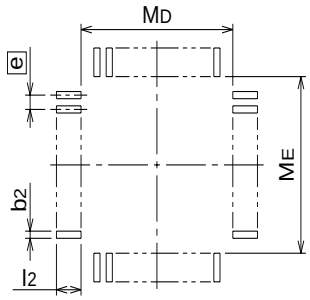
EIAJ Package Code	JEDEC Code	Weight(g)	Lead Material
QFP80-P-1420-0.80	–	1.58	Alloy 42











Recommended Mount Pad

Symbol	Dimension in Millimeters		
	Min	Nom	Max
A	–	–	3.05
A1	0	0.1	0.2
A2	–	2.8	–
b	0.3	0.35	0.45
c	0.13	0.15	0.2
D	13.8	14.0	14.2
E	19.8	20.0	20.2
e	–	0.8	–
HD	16.5	16.8	17.1
HE	22.5	22.8	23.1
L	0.4	0.6	0.8
L1	–	1.4	–
x	–	–	0.2
y	–	–	0.1
θ	0°	–	10°
b2	–	0.5	–
l2	1.3	–	–
MD	–	14.6	–
ME	–	20.6	–

MITSUBISHI MICROCOMPUTERS

7640 Group

Ver 1.4

SINGLE-CHIP 8-BIT CMOS MICROCOMPUTER

MITSUBISHI ELECTRIC CORPORATION
HEAD OFFICE: 2-2-3, MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN

MITSUBISHI ELECTRIC

REVISION HISTORY	7640 GROUP DATA SHEET

Rev. No.	Revision Description	Rev. date
1.4	First Edition	09/05/00