MiDAS3.0 Family:
Flash / ISP / IAP
8-bit Turbo Microcontrollers

Ver. 1.2

May 2008

- *CORERIVER Semiconductor reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time.*
- *CORERIVER shall give customers at least a three advance notice of intended discontinuation of a product or a service through its homepage.*
- *Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.*
- *The CORERIVER Semiconductor products listed in this document are intended for usage in general electronics applications. These CORERIVER Semiconductor products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury.*

Table of Contents

List of Figures

List of Tables

1 Product Overviews

The MiDAS3.0 family of CORERIVER is a group of fast 80C52 compatible microcontrollers without wasted system and memory clock cycles. Its processor core was redesigned for reducing the execution time of one machine cycle to four system clock cycles. As a result, almost all of its 8052 instructions are executed about 3 times faster than that of traditional 80C52.

The MiDAS3.0 family contains three timer/counters, two serial ports, maximum 32 programmable I/O ports, 32-channel 10-bit ADC (Analog to Digital Converter), 12 PWM (Pulse Width Modulator) outputs in two PCAs (Programmable Counter Array), one Watchdog timer, and one LVD (Low Voltage Detector) as peripherals. In addition, it contains an internal ring oscillator.

The MiDAS3.0 family provides power reduction modes and EMI noise reduction scheme.

To help a user develop a target system, on-chip hardware debugging engine and easy-to-use training kits are also provided.

1.1 Product Line-up of MiDAS3.0 Family

Table 1-1 MiDAS3.0 Family - GC89L5X1AE Series (ISP Flash MCU)

<table>
<tr><th>Product</th><th>Mask ROM (byte)</th><th>EPROM (byte)</th><th>Flash (byte)</th><th>RAM (byte)</th><th>Voltage (V)</th><th>Frequency (MHz)</th><th>ADC (bit x ch)</th><th>PWM (bit x ch)</th><th>I/O (pin)</th><th>Package</th></tr>
<tr><td>GC89L591A0</td><td>-</td><td>2K</td><td>62K</td><td rowspan="6">16K + 256</td><td rowspan="6">1.6~2.0 (Core)
3.0~3.6 (I/O)</td><td rowspan="6">80</td><td rowspan="6">10X32
10X21</td><td rowspan="6">8X12 or 16X6

8X6 Or 16X3</td><td rowspan="6">32
21</td><td rowspan="6">44-MQFP
32-MLF</td></tr>
<tr><td>GC81L591A0</td><td>62K</td><td>2K</td><td>-</td></tr>
<tr><td>GC89L581A0</td><td>-</td><td>2K</td><td>30K</td></tr>
<tr><td>GC81L581A0</td><td>30K</td><td>2K</td><td>-</td></tr>
<tr><td>GC89L541A0</td><td>-</td><td>2K</td><td>14K</td></tr>
<tr><td>GC81L541A0</td><td>14K</td><td>2K</td><td>-</td></tr>
</table>

2 Features

- CPU
  - 8-bit Turbo 80C52 Architecture
  - 4 clock cycles/1 machine cycle
  - Pin/instruction level compatible with Intel 80C52
- 0/14/30/62 Kbyte Mask ROM
- 0/14/30/62 Kbyte Flash
  - ISP by serial interface
  - IAP and virtual EEPROM for data (2 Kbyte)
  - Endurance : Typ. 50,000 write/erase cycles.
    Min. 10,000 write/erase cycles.
- 16 Kbyte on-chip RAM256 byte internal RAM
  - 256 bytes IRAM
  - 6,384 bytes AUXRAM (Accessed by MOVX)
- Supply voltage
  - Core: 1.62V ~ 1.98V
  - IO: 1.62V ~ 3.6V
- Operating Temperature & Frequency
  - Max. 100MHz @ -20°C ~ 85 °C
  - Max. 80 MHz @ -40°C ~ 125 °C
- Max. 32 Programmable I/O pins
  - Open-drain Intel compatible ports : P0
  - Quasi-bidirectional Intel type ports : P1 ~ P3
  - TTL and CMOS compatible logic levels : P0 ~ P3
  - Input/Output direction and Internal pull-up selectable : P0 ~ P3
  - All ports are initialized during asynchronous power-on-reset
- EMI reduction mode: Inhibit ALE
- Low Voltage Detector (LVD)
- 27-bit Programmable Watchdog Timer (WDT)
- Three 16-bit Timer/Counters
- Two sets of Full-Duplex UART
  - Automatic address recognition

- 12 Pulse Width Modulator (PWM) outputs provided from two Programmable Counter Arrays
  - 12-channels of 8-bit/16-bit dynamic PWM outputs
  - 12-channels of 16-bit capture/compare counter
  - 12-channels of high speed output
- 32-channel of 10-bit Analog to Digital Converter (ADC)
  - Max. 104K samples per second @ F_{ADC}=10MHz & 3V
  - Programmable clock frequency prescaler
- 16 interrupt sources including 6 external ones
  - Timer 0/1/2, UART 0/1, PCA 0/1, ADC, WDT, I2c, and six externals
  - Four/two level interrupt priority
- Reset scheme
  - On-chip Power-On-Reset (POR)
  - External reset
  - Low Voltage Detector (LVD) reset
  - Watchdog timer reset
- Internal counter to secure the stabilization time of a crystal oscillator
  - Maintains the power-on-reset state for 50 ms.
- On-chip PLL
  - VCO operating frequency: 70MHz ~ 130MHz
  - PFD comparison frequency: 2MHz ~ 20MHz
  - Support 2-bit output divider, 2-bit input divider
  - Support 8-bit feedback divider
- Power consumption
  - Active current: Typical 50 mA @ 1.8V, 100MHz
  - Stop current: Typical 10uA @ 1.8V
- Wake up from stop mode
  - On-chip power-on reset
  - External reset
  - External interrupt 0/1/2/3/4/5
  - Watchdog timer reset or interrupt
- ESD protection up to 2,000V
- Latch-up protection up to ±200mA
- Package : 32-MLF, 44-MQFP

3 Block Diagram

Figure 3-1 shows the block diagram of MiDAS3.0 family. The MiDAS3.0 family fetches instructions from the program memory (Mask ROM or Flash) and executes them. Data are read from or written to data memory (RAM) or special function registers (SFRs) of peripherals. The arithmetic-logic unit (ALU) inside its CPU processes data. Also several registers are used for processing or accessing data.

The MiDAS3.0 internal registers except program counter (PC) are configured as part of the on-chip RAM: therefore each register has an address. This is reasonable for the MiDAS 3.0, since it has so many registers. The bottom 32 locations of internal memory contain the register banks. The MiDAS3.0 instruction set supports 8 registers, R0 through R7, and by default (after a system reset) these registers are at addresses 00H-07H. As well as R0 to R7, there are special function registers (SFRs) at addresses from 80H to FFH. Most SFRs are accessed using direct addressing. However, the accumulator (ACC) is mostly accessed implicitly by the instruction set. The program counter (PC) has no address. So it is accessed only implicitly.

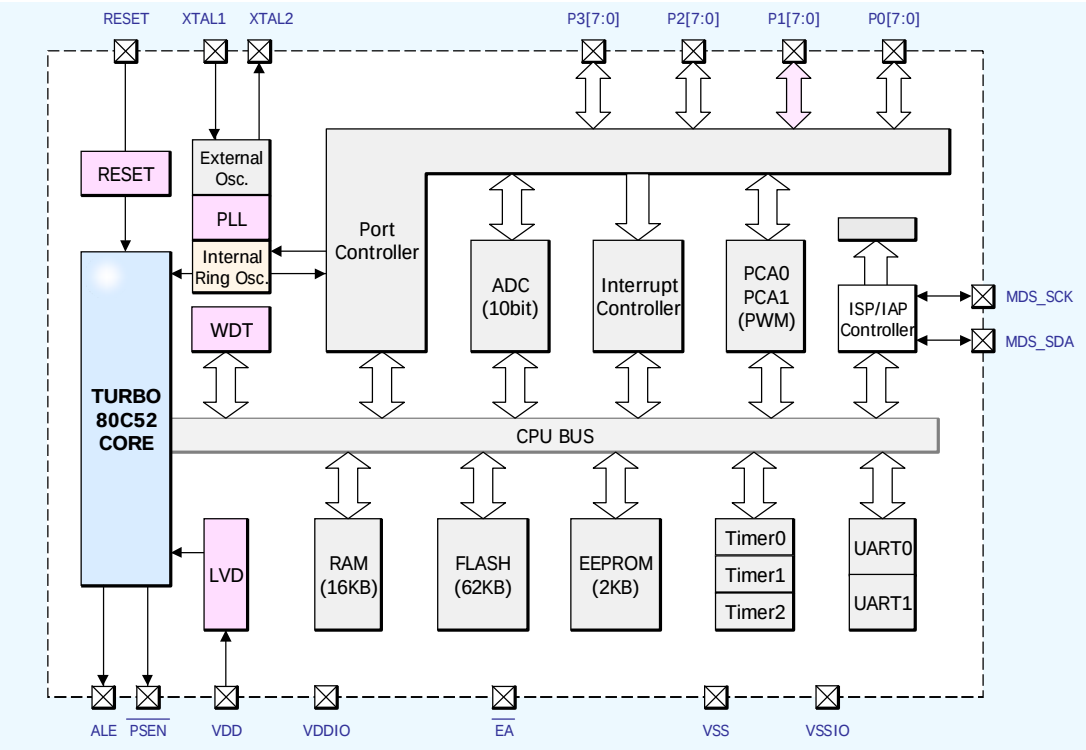
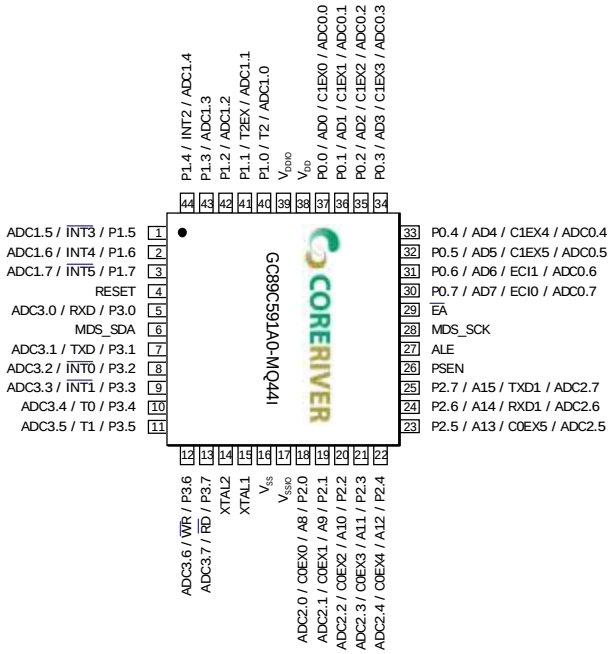


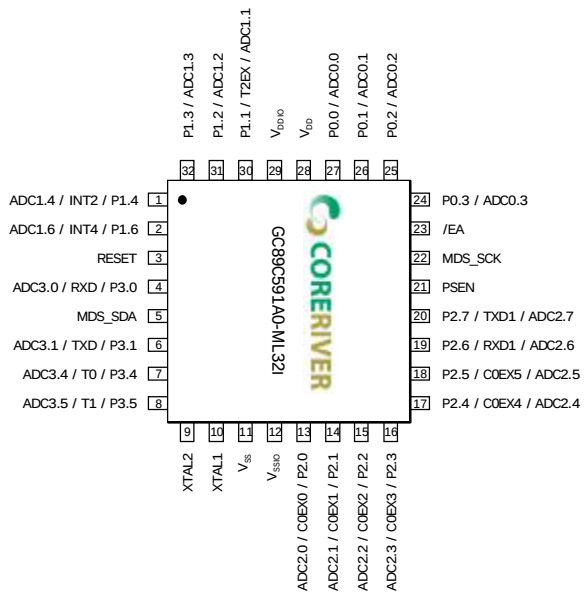
Figure 3-1 Block Diagram

4 Pin Configurations

The MiDAS3.0 family supports several packages, e.g. 32-pin MLF and 44-pin MQFP. The pin configurations are shown in Figure 4-1.



[44-pin MQFP]



[32-pin MLF]

Figure 4-1 Pin Configuration

5 Pin Description

Table 5-1 Pin Description

Symbol	Direction	Description	Share Pins
VDD	Input	Power Supply	-
VDDIO	Input	IO Power Supply	-
VSS	Input	Ground	-
VSSIO	Input	IO Ground	-
RESET	Input	External Reset	-
XTAL1	Input	Input to the inverting oscillator amplifier	-
XTAL2	Output	Output from the inverting oscillator amplifier	-
/EA	Input	External ROM Access Enable (The MiDAS3.0 family dose not use this pin.)	-
ALE	Input/ Output	Address Latch Enable (If ALEOFF is set, active only for external RAM access) This pin is also used for the parallel programming of FLASH memory.	-
PSEN	Input/ Output	Program Strobe Enable. Pull-up. Used for Special Input only. (MiDAS3.0 does not support the code fetch from external ROM.)	-
MDS_SDA, MDS_SCK	Input/ Output	I/O for MDS/ISP. The pull-up resistor is always switched on. This port is quasi-bidirectional.	-
P0[7:0]	Input/ Output	◆An 8-bitl open-drain I/O port or push-pull I/O port or ADC Input (3.3V). 5V Tolerant Input. ◆Note that the output is fully driven (push-pull) when P0 drives address/data to access external RAM or PCA1 drives output signals (C1EXn). • P0.0 ~ P0.7 → AD0 ~ AD7: Low address or data input/output • P0.0 ~ P0.5 → C1EX0 ~ C1EX5 for PCA1 • P0.6 → ECI1 for PCA1 • P0.7 → ECI0 for PCA0 • P0.0 → ADC0.0 : A/D converter Input 0 • P0.1 → ADC0.1 : A/D converter Input 1 • P0.2 → ADC0.2 : A/D converter Input 2 • P0.3 → ADC0.3 : A/D converter Input 3 • P0.4 → ADC0.4 : A/D converter Input 4 • P0.5 → ADC0.5 : A/D converter Input 5 • P0.6 → ADC0.6 : A/D converter Input 6 • P0.7 → ADC0.7 : A/D converter Input 7	P0.0 / AD0 / ADC0.0 / C1EX0 P0.1 / AD1 / ADC0.1 / C1EX1 P0.2 / AD2 / ADC0.2 / C1EX2 P0.3 / AD3 / ADC0.3 / C1EX3 P0.4 / AD4 / ADC0.4 / C1EX4 P0.5 / AD5 / ADC0.5 / C1EX5 P0.6 / AD6 / ADC0.6 / ECI1 P0.7 / AD7 / ADC0.7 / ECI0

P1[7:0]	Input/ Output	◆An 8-bitl open-drain I/O port or push-pull I/O port or ADC Input (3.3V). 5V Tolerant Input. • P1.0 → T2: External Input for Timer/Counter 2 • P1.1 → T2EX: Timer/Counter 2 Capture/Reload Trigger • P1.4 → INT2: External Interrupt 2 (Positive Edge) • P1.5 → /INT3: External Interrupt 3 (Negative Edge) • P1.6 → INT4: External Interrupt 4 (Positive Edge) • P1.7 → /INT5: External Interrupt 5 (Negative Edge) • P1.0 → ADC1.0: A/D converter Input 8 • P1.1 → ADC1.1: A/D converter Input 9 • P1.2 → ADC1.2: A/D converter Input 10 • P1.3 → ADC1.3: A/D converter Input 11 • P1.4 → ADC1.4: A/D converter Input 12 • P1.5 → ADC1.5: A/D converter Input 13 • P1.6 → ADC1.6: A/D converter Input 14 • P1.7 → ADC1.7: A/D converter Input 15	P1.0 / T2 / ADC1.0 P1.1 / T2EX / ADC1.1 P1.2 / ADC1.2 P1.3 / ADC1.3 P1.4 / INT2 / ADC1.4 P1.5 / /INT3 / ADC1.5 P1.6 / INT4 / ADC1.6 P1.7 / /INT5 / ADC1.7
P2[7:0]	Input/ Output	◆An 8-bitl open-drain I/O port or push-pull I/O port or ADC Input (3.3V). 5V Tolerant Input. ◆Note that the output is fully driven (push-pull) when P2 drives the high byte of address to access external RAM or PCA0 drives output signals (C0EXn). • P2.0~P2.7 → AD8 ~ AD15: High address output • P2.0~P2.5 → C0EX0 ~ C0EX5 for PCA0 • P2.6 → RXD1: Serial Port 1 Output • P2.7 → TXD1: Serial Port 1 Input • P2.0 → ADC2.0: A/D converter Input 16 • P2.1 → ADC2.1: A/D converter Input 17 • P2.2 → ADC2.2: A/D converter Input 18 • P2.3 → ADC2.3: A/D converter Input 19 • P2.4 → ADC2.4: A/D converter Input 20 • P2.5 → ADC2.5: A/D converter Input 21 • P2.6 → ADC2.6: A/D converter Input 22 • P2.7 → ADC2.7: A/D converter Input 23	P2.0 / AD8 / ADC2.0 / C0EX0 P2.1 / AD9 / ADC2.1 / C0EX1 P2.2 / AD10 / ADC2.2/C0EX2 P2.3 / AD11 / ADC2.3/C0EX3 P2.4 / AD12 / ADC2.4/C0EX4 P2.5 / AD13 / ADC2.5/C0EX5 P2.6 / AD14 / ADC2.6/RXD1 P2.7 / AD15 / ADC2.7/TXD1
P3[7:0]	Input/ Output	◆An 8-bitl open-drain I/O port or push-pull I/O port or ADC Input (3.3V). 5V Tolerant Input. • P3.0 → RXD: Serial Port 0 Input • P3.1 → TXD: Serial Port 0 Output • P3.2 → /INT0: External Interrupt Input 0 • P3.3 → /INT1: External Interrupt Input 1 • P3.4 → T0: Timer 0 External Input • P3.5 → T1: Timer 1 External Input • P3.6 → WR: External Data Memory Writer Strobe • P3.7 → RD: External Data Memory Read Strobe • P3.0 → ADC3.0: A/D converter Input 24 • P3.1 → ADC3.1: A/D converter Input 25 • P3.2 → ADC3.2: A/D converter Input 26 • P3.3 → ADC3.3: A/D converter Input 27	P3.0 / RXD / ADC3.0 P3.1 / TXD / ADC3.1 P3.2 / INT0 / ADC3.2 P3.3 / INT1 / ADC3.3 P3.4 / T0 / ADC3.4 P3.5 / T1 / ADC3.5 P3.6 / WR / ADC3.6 P3.7 / RD / ADC3.7

		• P3.4 → ADC3.4: A/D converter Input 28 • P3.5 → ADC3.5: A/D converter Input 29 • P3.6 → ADC3.6: A/D converter Input 30 • P3.7 → ADC3.7: A/D converter Input 31	

6 Functional Description

6.1 CPU Description

6.1.1 Memory Organization

Most microprocessors implement a shared memory space for data and programs. This is reasonable, since programs are usually stored on a disk and loaded into RAM for execution; thus both the data and programs reside in the system RAM. Microcontrollers, on the other hand, are rarely used as the CPU in “computer systems.” Instead, they are employed as the central component in control-orient designs. There is limited memory, and there is no disk drive.

For the reason, MiDAS3.0 family has separate address spaces for program and data memory. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by an 8-bit CPU.

Program Memory can be only read, not written to. There can be up to 62K bytes of Program Memory.

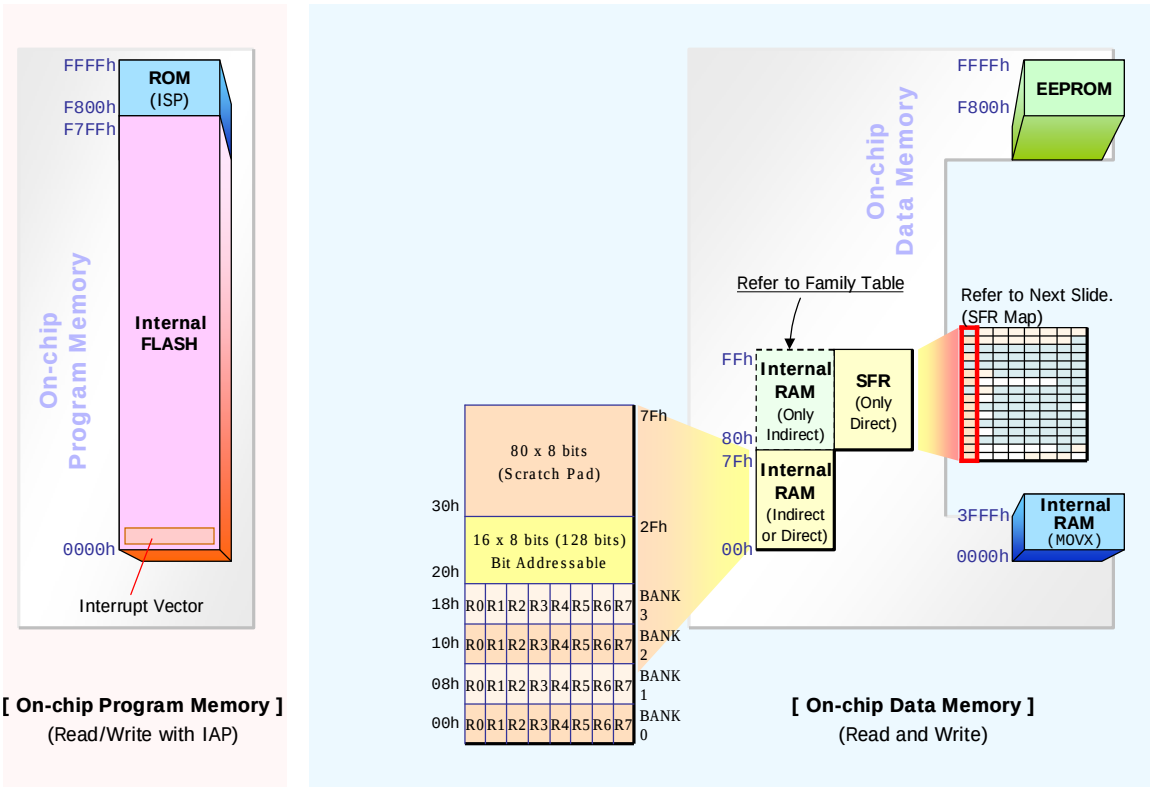


Figure 6-1 Memory Organization

6.1.1.1 Program Memory

MiDAS3.0 uses the on-chip flash or mask-ROM program memory. So no external program memory is necessary. The flash program memory is In-System Programmable. MiDAS3.0 contains the 14/30/62 Kbytes of the program memory.

The left diagram_of Figure 6-1 shows the map of the Program Memory. After reset, the CPU executes from location 0000H.

As shown in Figure 6-1, each interrupt has the fixed vector address in Program memory. When an interrupt is accepted, the vector address value is loaded into PC and the interrupt service routine starts. The vector address of External Interrupt 0, for example, is 0003H. If External Interrupt 0 is accepted, its service routine will begin at location 0003H. If the interrupt is not used, its service location is available as general purpose Program Memory.

The interrupt vector addresses are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1 and etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are used.

6.1.1.2 Data Memory

MiDAS3.0 contains two kinds of on-chip data memory. One is the normal internal data memory. The other is an on-chip external data memory accessible only with the MOVX instruction. From now on, this on-chip external data memory will be called ‘AUXRAM.’

The address space of the normal internal data memory is divided into three basic, physical separate and distinct blocks as shown in the right diagram of Figure 6-1:

- The lower 128byte of internal data RAM located to the address range 00h – 7Fh.
- The upper 128byte of internal data RAM located to the address range 80h – FFh.
- The 128byte area for special function register (SFR) located to the address range 80h – FFh.

While the upper internal RAM area and the SFR area share the same address locations (80h – FFh), they are accessed through different addressing modes. The upper internal RAM can only be accessed through indirect addressing mode (indirect addressing with general purpose registers R0 or R1), and the special function registers (SFRs) are accessible only by direct addressing mode (address is part of the

instruction).

MiDAS3.0 contains 16,384 bytes of AUXRAM. AUXRAM has the address space from 0000H to 3FFFH. If AUXRAM provides sufficient memory space, no external access to an off-chip data memory is needed. So P0 port and P2 port will be used exclusively for general I/O ports.

6.1.1.2.1 Register Banks

The lowest 32 bytes of the internal data RAM are grouped into 4 banks of 8 general purpose registers (GPRs). Instructions call out these registers as R0 through R7. RS1 and RS0 bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than direct addressing instructions.

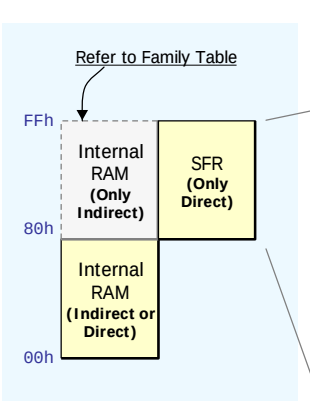
6.1.1.2.2 Bit-addressable Memory Space

The next 16 bytes, locations 20H through 2FH, above the register banks form a block of bit-addressable memory space. There are 128 bits in this area. The 128 bits can be directly addressed by single-bit instructions. The bit addresses in this area are 00H through 7FH.
Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 000B. The bit addresses in this area are 80H through FFH.

6.1.2 Special Function Registers (SFRs) Map and Description

The MiDAS3.0 family uses Special Function Registers (SFRs) to control and monitor peripherals and their modes.
The SFRs are located at 80h-FFh and are accessed by direct addressing only. Some of the SFRs are bit addressable. This is very useful in cases where one wishes to modify a particular bit without changing the others. The SFRs that are bit addressable are those whose addresses end in 0h or 8h. The MiDAS3.0 family contains all the SFRs included in the standard 80C52. However, some additional SFRs have been added. In some cases, unused bits of the original 80C52 have been given new functions. The list of SFRs is as follows. Refer to Appendix B for more details about SFRs.

Table 6-1 SFR map (* = bit addressable SFR)



: Newly added SFR at MiDAS3.0 Family

: Reserved for future use.

Bit addressable

F8h	EIP	UINDX	UDATA	CLKSEL					FFh
F0h	B							FAEN	F7h
E8h	EIE	P3SEL	C1L	C1H	ADCENB0	ADCENB1	ADCENB2	ADCENB3	EFh
E0h	ACC	P2SEL	C1CAPM0	C1CAPM1	C1CAPM2	C1CAPM3	C1CAPM4	C1CAPM5	E7h
D8h	WDCON	P1SEL	C1CAP0H	C1CAP1H	C1CAP2H	C1CAP3H	C1CAP4H	C1CAP5H	DFh
D0h	PSW	P0SEL	C1CAP0L	C1CAP1L	C1CAP2L	C1CAP3L	C1CAP4L	C1CAP5L	D7h
C8h	T2CON	T2MOD	RCAP2L	RCAP2H	TL2	TH2	C1CON	C1MOD	CFh
C0h		PLLCON	PLLNR	PLLFR	PMR	STATUS	OSCICN	IOCFG	C7h
B8h	IP	SADEN	ITSEL	P0DIR	P1DIR	P2DIR	P3DIR	AUXAD	BFh
B0h	P3	SCON1	IT	P0TYP	P1TYP	P2TYP	P3TYP	IPH	B7h
A8h	IE	SADDR	SADDR1	SADEN1	C0CON	C0MOD	C0L	C0H	AFh
A0h	P2	SBUF1	C0CAPM0	C0CAPM1	C0CAPM2	C0CAPM3	C0CAPM4	C0CAPM5	A7h
98h	SCON	SBUF	C0CAP0H	C0CAP1H	C0CAP2H	C0CAP3H	C0CAP4H	C0CAP5H	9Fh
90h	P1	EXIF	C0CAP0L	C0CAP1L	C0CAP2L	C0CAP3L	C0CAP4L	C0CAP5L	97h
88h	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	RINGCON	8Fh
80h	P0	SP	DPL	DPH	ADCON	ADCSEL	ADCR	PCON	87h
	0h/8h	1h/9h	2h/Ah	3h/Bh	4h/Ch	5h/Dh	6h/Eh	7h/Fh	

(for 32-MLF)

Table 6-2 Summary of SFRs (*: undefined value)

Name	Symbol	Address	Reset Value	Bit Addressable
Accumulator	ACC		00000000	
B Register	B		00000000	
Program Status Word	PSW		00000000	
Stack Pointer	SP		00000111	
Data Pointer Low byte	DPL		00000000	
Data Pointer High byte	DPH		00000000	
Port 0	P0		11111111	
Port 1	P1		11111111	
Port 2	P2		11111111	
Port 3	P3		11111111	
Interrupt Priority Low	IP		10000000	
Interrupt Priority High	IPH		10000000	

Interrupt Enable Control	IE		00000000	
T/C 0/1 Control	TCON		00000000	
T/C 0/1 Mode Control	TMOD		00000000	
T/C 2 Control	T2CON		00000000	
T/C 2 Mode Selection	T2MOD		******00	
T/C 0 High byte	TH0		00000000	
T/C 0 Low byte	TL0		00000000	
T/C 1 High byte	TH1		00000000	
T/C 1 Low byte	TL1		00000000	
T/C 2 High byte	TH2		00000000	
T/C 2 Low byte	TL2		00000000	
T/C 2 Capture Reg. High byte	RCAP2H		00000000	
T/C 2 Capture Reg. Low byte	RCAP2L		00000000	
Serial Port Control of UART0	SCON		00000000	
Serial Data Buffer of UART0	SBUF		00000000	
Slave Address Mask Enable of UART0	SADEN		00000000	
Slave Address of UART0	SADDR		00000000	
Power Control	PCON		00*10000	
Port 0 Pull-up Control	P0SEL		11111111	
Port 1 Pull-up Control	P1SEL		00000000	
Port 2 Pull-up Control	P2SEL		00000000	
Port 3 Pull-up Control	P3SEL		00000000	
Port 0 Input/Output Control	P0DIR		11111111	
Port 1 Input/Output Control	P1DIR		11111111	
Port 2 Input/Output Control	P2DIR		11111111	
Port 3 Input/Output Control	P3DIR		11111111	
Port 0 Type Control	P0TYP		11111111	
Port 1 Type Control	P1TYP		11111111	
Port 2 Type Control	P2TYP		11111111	
Port 3 Type Control	P3TYP		11111111	
Serial Port Control of UART1	SCON1		00000000	
Serial Data Buffer of UART1	SBUF1		00000000	
Slave Address of UART1	SADDR1		00000000	
Slave Address Mask Enable of UART1	SADEN1		00000000	

ADC Control & ADC Result Low	ADCON		0010**00	
ADC Result High	ADCR		00000000	
ADC Clock & MUX Selection	ADCSEL		000**000	
ADC Input Channel Enable Bar: ADC0.0 ~ 7	ADCENB0		11111111	
ADC Input Channel Enable Bar: ADC1.0 ~ 7	ADCENB1		11111111	
ADC Input Channel Enable Bar: ADC2.0 ~ 7	ADCENB2		11111111	
ADC Input Channel Enable Bar: ADC3.0 ~ 7	ADCENB3		11111111	
Extended Interrupt Priority	EIP		00000000	
Extended Interrupt Enable	EIE		00000000	
High Address for MOVX with Ri	AUXAD		00000000	
Watchdog Timer & Power Status	WDCON		*1010000	
IAP Routine Access Enable	FAEN		********0	
Power Management	PMR		****00**	
External Interrupt Flag	EXIF		00001001	
Clock Control	CKCON		11000*00	
Crystal Status	STATUS		***1****	
Internal RING Oscillator Control	OSCICN		*****100	
I/O Configuration	IOCFG		****0**0	
RING Control Register	RINGCON		01110000	
Internal Clock Selection	CLKSEL		0**00110	
PLL Control	PLLCON		00011010	
PLL NR Control	PLLNR		****0000	
PLL FR Control	PLLFR		00000000	
Low Byte of PCA0 Counter	C0L		00000000	
High Byte of PCA0 Counter	C0H		00000000	
PCA0 Counter Control	C0CON		00000000	
PCA0 Counter Mode	C0MOD		00*00000	
Mode Control of PCA0 MODULE0	COCAPM0		01000000	
Mode Control of PCA0 MODULE1	COCAPM1		01000000	
Mode Control of PCA0 MODULE2	COCAPM2		01000000	
Mode Control of PCA0 MODULE3	COCAPM3		01000000	
Mode Control of PCA0 MODULE4	COCAPM4		01000000	
Mode Control of PCA0 MODULE5	COCAPM5		01000000	
Low Capture/Compare of PCA0 MODULE0	COCAP0L		00000000	

Low Capture/Compare of PCA0 MODULE1	COCAP1L		00000000	
Low Capture/Compare of PCA0 MODULE2	COCAP2L		00000000	
Low Capture/Compare of PCA0 MODULE3	COCAP3L		00000000	
Low Capture/Compare of PCA0 MODULE4	COCAP4L		00000000	
Low Capture/Compare of PCA0 MODULE5	COCAP5L		00000000	
High Capture/Compare of PCA0 MODULE0	COCAP0H		00000000	
High Capture/Compare of PCA0 MODULE1	COCAP1H		00000000	
High Capture/Compare of PCA0 MODULE2	COCAP2H		00000000	
High Capture/Compare of PCA0 MODULE3	COCAP3H		00000000	
High Capture/Compare of PCA0 MODULE4	COCAP4H		00000000	
High Capture/Compare of PCA0 MODULE5	COCAP5H		00000000	
Low Byte of PCA1 Counter	C1L		00000000	
High Byte of PCA1 Counter	C1H		00000000	
PCA1 Counter Control	C1CON		00000000	
PCA1 Counter Mode	C1MOD		00*00000	
Mode Control of PCA1 MODULE0	C1CAPM0		01000000	
Mode Control of PCA1 MODULE1	C1CAPM1		01000000	
Mode Control of PCA1 MODULE2	C1CAPM2		01000000	
Mode Control of PCA1 MODULE3	C1CAPM3		01000000	
Mode Control of PCA1 MODULE4	C1CAPM4		01000000	
Mode Control of PCA1 MODULE5	C1CAPM5		01000000	
Low Capture/Compare of PCA1 MODULE0	C1CAP0L		00000000	
Low Capture/Compare of PCA1 MODULE1	C1CAP1L		00000000	
Low Capture/Compare of PCA1 MODULE2	C1CAP2L		00000000	
Low Capture/Compare of PCA1 MODULE3	C1CAP3L		00000000	
Low Capture/Compare of PCA1 MODULE4	C1CAP4L		00000000	
Low Capture/Compare of PCA1 MODULE5	C1CAP5L		00000000	
High Capture/Compare of PCA1 MODULE0	C1CAP0H		00000000	
High Capture/Compare of PCA1 MODULE1	C1CAP1H		00000000	
High Capture/Compare of PCA1 MODULE2	C1CAP2H		00000000	
High Capture/Compare of PCA1 MODULE3	C1CAP3H		00000000	
High Capture/Compare of PCA1 MODULE4	C1CAP4H		00000000	
High Capture/Compare of PCA1 MODULE5	C1CAP5H		00000000	
I2C Slave Access Index Register	UINDX		***00000	

I2C Slave Access Data Register	UDATA		01100111	
Interrupt Type Selection	IT		00001111	
Interrupt Polarity Selection	ITSEL		**010100	

CAUTION: Don't modify the bits marked by *. Modifying these bits can cause the malfunctions. Especially the following bits of SFRs should not be updated.

- **n** EXIF.3(XT) : This bit must always remain high which is the default value.
- **n** PMR.3 (*): This bit should not be modified.

6.1.3 External Data Memory Access

There are two types of MOVX instructions: the MOVX @Ri and MOVX @DPTR. In the MOVX @Ri, the address of the external data memory comes from two sources. The lower 8-bits of the address are stored in the Ri register of the selected register bank. The upper 8-bits of the address come from the port 2 SFR or the AUXAD SFR. In the MOVX @DPTR type, the full 16-bit address is supplied by the Data Pointers.

MiDAS3.0 contains 16,384 bytes of AUXRAM. AUXRAM has the address space from 0000H to 3FFFH. If AUXRAM provides sufficient memory space, no external access to an off-chip data memory is needed. When no off-chip external RAM is used, let's set up the ENAUX (IOCFG.3) to 1 in order to use the AUXAD SFR instead of P2 SFR as the upper 8-bit address source. Then the P2 SFR and P0 SFR are not used for an address source. As a result, the port 2 and port 0 are used exclusively as general I/O ports.

When off-chip external data memory is used, Port 0 is unavailable as an I/O port. It becomes a multiplexed address (A0 – A7) and data (D0 – D7) bus, with ALE latching the low-byte of the address at the beginning of each external memory cycle. Port 2 outputs the high byte of the off-chip external data memory address.

6.1.4 Instruction Set Summary

The MiDAS3.0 family executes all the instructions of the standard 80C52. The instructions of the MiDAS3.0 family produce the same result with those of the standard 80C52.

Table 6-3 Summary of the Instruction Set

Type	Instruction	Description
Arithmetic	ADD	Addition
	ADDC	Addition with Carry
	SUBB	Subtraction with Borrow
	INC	Increment
	DEC	Decrement
	MUL	Multiply
	DIV	Divide
	DA	Decimal Adjust
Logical	ANL	Logical AND
	ORL	Logical OR
	XRL	Logical Exclusive OR
	CLR	Clear
	CPL	Complement
	RL	Rotate Left
	RLC	Rotate Left with Carry
	RR	Rotate Right
	RRC	Rotate Right with Carry
	SWAP	Swap nibbles
Data Transfer	MOV	Move data
	MOVC	Move Code
	MOVX	Move data to external RAM
	PUSH	Push
	POP	Pop
	XCH	Exchange
	XCHD	Exchange low-digit
Boolean	CLR	Clear bit
	SETB	Set bit
	CPL	Complement bit
	ANL	AND bit
	ORL	OR bit
	MOV	Move bit
	JC	Jump if Carry is set

	JNC	Jump if Carry is not set
	JB	Jump if Bit is set
	JNB	Jump if Bit is not set
	JBC	Jump if Bit is set & Clear
Branch	ACALL	Absolute Call
	LCALL	Long Call
	RET	Return from subroutine
	RETI	Return from interrupt
	AJMP	Absolute Jump
	LJMP	Long Jump
	SJMP	Short Jump
	JMP	Jump with DPTR
	JZ	Jump if ACC is zero
	JNZ	Jump if ACC is not zero
	CJNE	Compare and Jump if not equal
	DJNZ	Decrement and Jump if not zero
	NOP	No Operation

6.1.4.1 Addressing Modes

The operands used in the instructions can be addressed in different modes. The MiDAS3.0 family uses six different addressing modes:

- Immediate
- Direct
- Indirect
- Index
- Register
- Register-specific

6.1.4.1.1 Immediate Addressing Mode

Immediate addressing mode means that an operand is the constant value included in the instruction.

Examples:	
MOV R3, #0FFh	R3 is loaded with constant value 0FFh
ORL PSW, #00000101b	Logical OR operation with PSW and 0000 0101b
MOV DPTR, #1243h	Load data pointer with constant value 1234h

6.1.4.1.2 Direct Addressing Mode

In direct addressing, the operand is specified by an 8-bit address field in the instruction.
Note: The lower 128 bytes of the normal internal RAM can be addressed in this mode. SFRs are accessible only with this addressing mode.

Examples:	
MOV A, TCON	TCON is the 8-bit address of the SFR TCON.
MOV R4, variable	The variable is the 8-bit address of a memory location in the lower part of the internal RAM.

6.1.4.1.3 Indirect Addressing Mode

In indirect addressing, the instruction specifies a register, which contains the address of the operand. Both internal and external RAM can be addressed indirectly.

The address register for 8-bit addresses can be R0 or R1 of the selected register bank, or the Stack Pointer. The address register for 16-bit addresses can be only the 16-bit “data pointer” register, DPTR.

Note: The lower and upper part of the internal RAM can be addressed by using 8-bit addresses. The external data memory can be addressed by 16-bit addresses.

Examples:	
MOV A, @R0	RAM is addressed by contents of R0 (8-bit address)
MOVX A, @DPTR	External data memory is addressed by contents of the data pointer DPTR (16-bit address)

6.1.4.1.4 Index Addressing Mode

Only program memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in program memory. A 16-bit base register (either DPTR or the PC) points to the base of the table, and the accumulator is set up with the table entry number. The address of the table entry in program memory is formed by adding the accumulator data to the base pointer data. Another type of indexed addressing is used in the “case jump” instruction. In this case, the destination address of a jump instruction is calculated as the sum of the base pointer data and the accumulator data.

Examples:

MOVC A, @A + DPTR	The address is the sum of DPTR and accumulator.
MOVC A, @A + PC	The address is the sum of PC and accumulator
JMP @A + DPTR	Jump to sum of accumulator and DPTR

6.1.4.1.5 Register Addressing Mode

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank selection bits (RS0 and RS1) in the PSW.

6.1.4.1.6 Register-Specific Addressing Mode

Some instructions are specific to a certain register. For example, some instructions always operate on the accumulator, or data pointer, etc., so no address byte is needed to point to it.

6.1.5 CPU Timing

The CPU timing for the MiDAS3.0 family is important, especially for those who wish to use software

instructions to generate timing delays. Also, it provides the insight into the timing differences between the MiDAS3.0 family and the standard 80C52 as shown in Figure 6-2. In the MiDAS3.0 family, each machine cycle is four clock periods long. Each clock period is designated a state. Thus each machine cycle is made up of four states, S1, S2, S3 and S4 in that order. In order to reduce the instruction execution time, both the clock edges are used for internal timing. Hence it is important that the duty cycle of the clock should be as close to 50% as possible. Since the MiDAS3.0 family fetches one byte per machine cycle, in most of the instructions, the number of machine cycles needed to execute the instruction is equal to the number of bytes in the instruction.

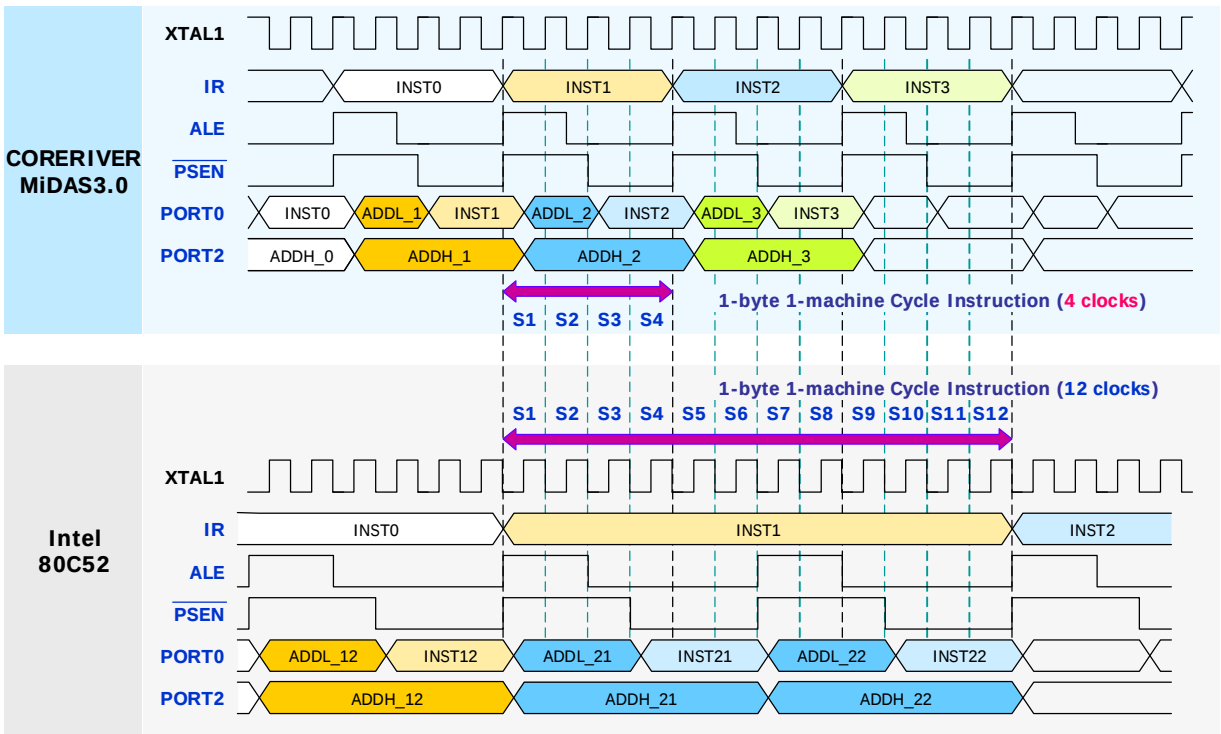


Figure 6-2 Timing comparison of the MiDAS3.0 family and Intel 80C52

6.1.5.1 MOVX Instruction

The MiDAS3.0 family, like the standard 80C52, uses the MOVX instruction to access on-chip and off-chip external Data Memories. This Data Memory includes both off-chip memory as well as memory-mapped peripherals. While the results of the MOVX instruction are the same with those of the standard 80C52, the operation and the timing of the strobe signals have been modified in order to give the user much greater flexibility as shown in Figure 6-3 and Figure 6-4.

There are two type of MOVX instructions: the MOVX @Ri and MOVX @DPTR. In the MOVX @Ri, the

address of the external data RAM comes from two sources. The lower 8-bits of the address are stored in the Ri register of the selected register bank. The upper 8-bits of the address come from the port 2 SFR or the AUXAD SFR. In the MOVX @DPTR type, the full 16-bit address is supplied by the Data Pointers.

The MiDAS3.0 family contains 16,384 byte of RAM which can be accessed only the MOVX instruction like the off-chip RAM. So no off-chip RAM is needed for most cases. When no off-chip external RAM is used, let's set up the ENAUX (IOCFG.3) to 1 in order to use the AUXAD SFR as the upper 8-bit address source. Then the port 2 and port 0 are not used for an address source. As a result, the port 2 and port 0 are used exclusively as general I/O ports.

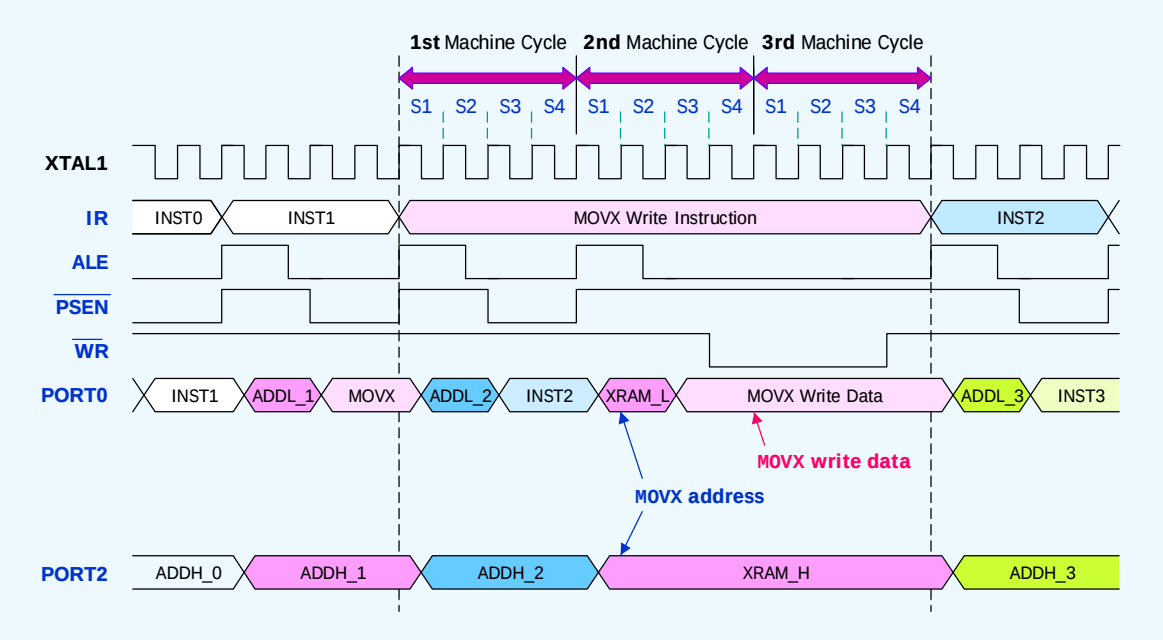


Figure 6-3 Write Timing of MOVX instruction

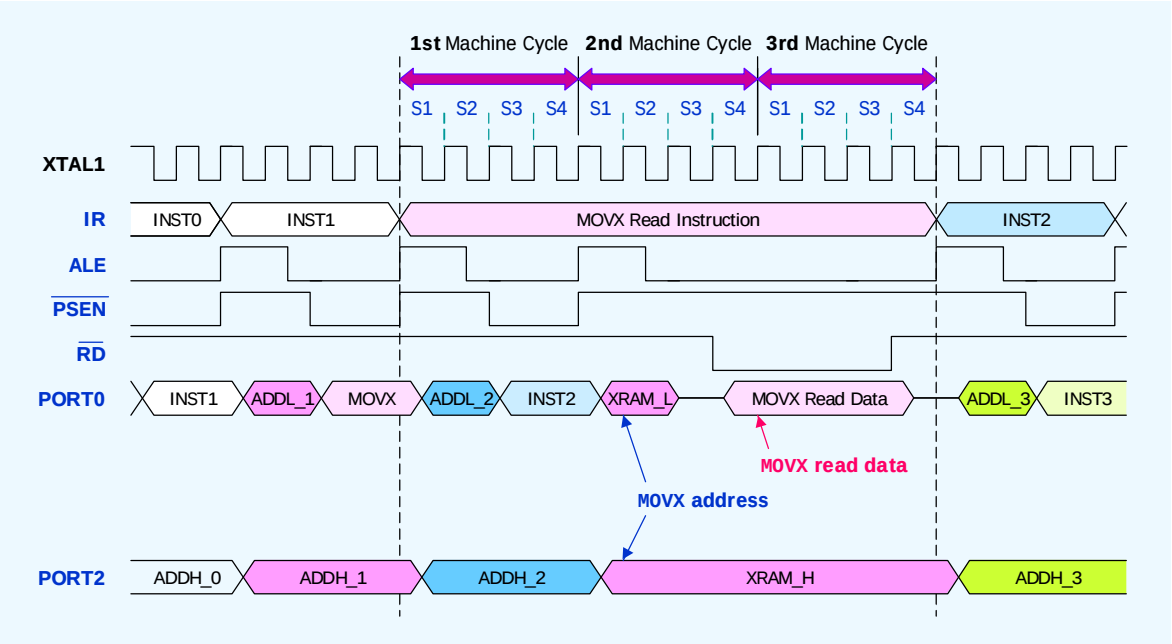


Figure 6-4 Read Timing of MOVX instruction

6.2 Peripheral Description

6.2.1 I/O Ports

The I/O ports of MiDAS3.0 are bi-directional I/O ports. They consist of a latch (Special Function Register P0 through P3), an output driver, and an input buffer. All the latches are 8-bits, byte-addressable, and bit-addressable. Bit operation uses different instructions from byte operations.

The output drivers of Ports 0 and 2 and the input buffers of Port 0 are used for access to off-chip external data memory. When off-chip data memory is accessed, Port 0 is unavailable as a general I/O port. It becomes a multiplexed address (A0 – A7) and data (D0 – D7) bus, with ALE latching the low-byte of the address at the beginning of each external memory cycle. Port 2 outputs the high byte of the external memory address.

6.2.1.1 PORT 0

Port 0 is an 8-bit, open-drain or push-pull, bidirectional I/O port, which has 5V tolerant input and 3.3V output. A bit of P0TYP Register determines in which mode (0: push-pull, 1: Open-drain by default) the corresponding output driver operates. When it is used for external memory access or PCA output, it drives output strongly in push-pull mode. So Port 0 pins that have 1s are pulled high strongly. A bit of P0DIR Register determines the I/O direction of the corresponding pin (0: Output, 1: Input by default). All bits of P0SEL Register are cleared to 0 by default. Setting a bit of P0SEL Register to 1 switches the corresponding pin pull-up off.

It is not recommendable that Port 0 is used as a general purpose I/O port if it also accesses off-chip data memory. It is because the CPU will write FFh to the port 0 latch during any access to off-chip data memory. To access off-chip data memory, the type of Port 0 output driver should operate in open-drain mode.

When off-chip data memory is accessed, Port 0 is unavailable as a general I/O port. It becomes a multiplexed address (A0 – A7) and data (D0 – D7) bus, with ALE latching the low-byte of the address at the beginning of each external memory cycle. For this, Port 0 is disconnected from its own port latch, and the address/data signal drives push-pull FETs in its output buffer. In this application, Port 0 uses strong

internal pull-ups when emitting 1s. During any access to off-chip data memory, the CPU writes FFh to the port 0 latch, thus obliterating whatever information the port SFR may have been holding.

All Port 0 pins also serve the alternate functions specified below. For preventing errors, let the corresponding I/O pin operate in the open-drain mode and the corresponding bit of Port 0 Register be set to logic 1**.**

Port	Alternate Functions
P0.0	(AD0.0) Multiplexed Address/Data bus or (C1EX0) PCA1 Module 0 I/O or (ADC0.0) A/D Converter Input 0
P0.1	(AD0.1) Multiplexed Address/Data bus or (C1EX1) PCA1 Module 1 I/O or (ADC0.1) A/D Converter Input 1
P0.2	(AD0.2) Multiplexed Address/Data bus or (C1EX2) PCA1 Module 2 I/O or (ADC0.2) A/D Converter Input 2
P0.3	(AD0.3) Multiplexed Address/Data bus or (C1EX3) PCA1 Module 3 I/O or (ADC0.3) A/D Converter Input 3
P0.4	(AD0.4) Multiplexed Address/Data bus or (C1EX4) PCA1 Module 4 I/O or (ADC0.4) A/D Converter Input 4
P0.5	(AD0.5) Multiplexed Address/Data bus or (C1EX5) PCA1 Module 5 I/O or (ADC0.5) A/D Converter Input 5
P0.6	(AD0.6) Multiplexed Address/Data bus or (ECI1) PCA1 Counter Input or (ADC0.6) A/D Converter Input 6
P0.7	(AD0.7) Multiplexed Address/Data bus or (ECI0) PCA0 Counter Input or (ADC0.7) A/D Converter Input 7

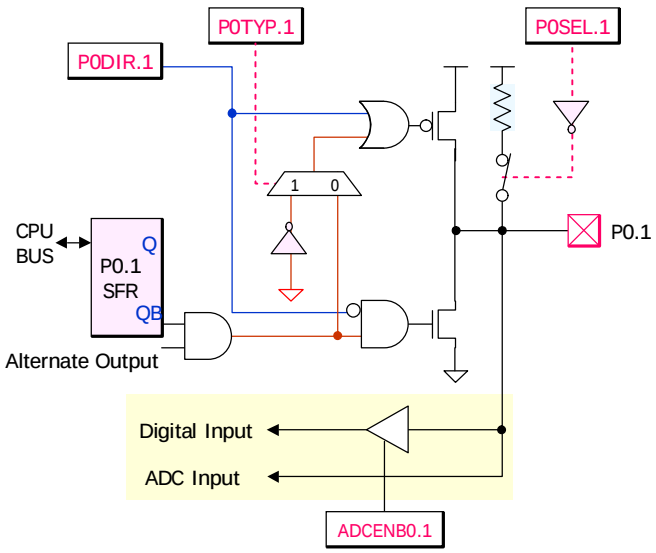


Figure 6-5 Configuration of PORT 0

Note: During off-chip data memory access, P0 SFR will be automatically set to “FFh.”

Note: Read-Modify-Write instructions do not read port pin but SFR register.

l P0TYPE (B3h): Port 0 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	P0TYPE.7	P0TYPE.6	P0TYPE.5	P0TYPE.4	P0TYPE.3	P0TYPE.2	P0TYPE.1	P0TYPE.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Push-pull Output / 1 = Open-drain Output (Default)

l P0DIR (BBh): Port 0 Input/Output Control Register

Bit No.	7	6	5	4	3	2	1	0
	P0DIR.7	P0DIR.6	P0DIR.5	P0DIR.4	P0DIR.3	P0DIR.2	P0DIR.1	P0DIR.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Output / 1 = Input (Default)

l P0SEL (D1h): Port 0 Pull-up Control Register

Bit No.	7	6	5	4	3	2	1	0
	P0SEL.7	P0SEL.6	P0SEL.5	P0SEL.4	P0SEL.3	P0SEL.2	P0SEL.1	P0SEL.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Pull-up resistor ON / 1 = Pull-up resistor OFF (Default)

l P0 (80h): Port 0 Register

Bit No.	7	6	5	4	3	2	1	0
	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

All pins of Port 0 can be used as an analog input pin (ADC input channel) by clearing the corresponding bit of ADCENB0 to 0. For this, the corresponding pull-up should be switched off and the corresponding I/O pin driver should operate in open-drain mode. So the corresponding bits of P0TYP and P0SEL should be set to 1. Also the corresponding bit of P0 Register should be set to 1.

l ADCENB0 (ECh): ADC Channel Enable Bar Register (P0 port)

Bit No.	7	6	5	4	3	2	1	0
	ADCENB0.7	ADCENB0.6	ADCENB0.5	ADCENB0.4	ADCENB0.3	ADCENB0.2	ADCENB0.1	ADCENB0.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = ADC0.X channel ON / 1 = ADC0.X channel OFF (Default)

6.2.1.2 PORT 1

Port 1 is an 8-bit, bidirectional I/O port, which has 5V tolerant input and 3.3V output. A bit of P1TYP Register determines in which mode (0: push-pull by default, 1: quasi-bidirectional) the corresponding output driver operates. A bit of P1DIR Register determines the I/O direction of the corresponding pin (0: Output, 1: Input by default). All bits of P1SEL Register are cleared to 0 by default. Setting a bit of P1SEL Register to 1 switches the corresponding pin pull-up off. When the P1.0 pin is used for Timer2 input/output (T2), it drives strongly in the push-pull mode. So it is pulled high strongly. Port 1 pins can be

also used alternate functions. When we use the alternate functions, the corresponding bit of the Port 1 pin should be set to high and Port 1 should operate in the quasi-bidirectional mode for preventing errors. The alternate functions will be selected according to the combination of the SFR values. Assignment of any pin to alternate functions has no influence on the operation of other pins.

Port	Alternate Functions
P1.0	(T2) Timer2 input/output or (ADC1.0) ADC1 input channel 0
P1.1	(T2EX) Timer 2 capture/reload input or (ADC1.1) ADC1 input channel 1
P1.2	(ADC1.2) ADC1 input channel 2
P1.3	(ADC1.3) ADC1 input channel 3
P1.4	(INT2) External Interrupt 2, rising edge triggered or (ADC1.4) ADC1 input channel 4
P1.5	(/INT3) External Interrupt 3, falling edge triggered or (ADC1.5) ADC1 input channel 5
P1.6	(INT4) External Interrupt 4, rising edge triggered or (ADC1.6) ADC1 input channel 6
P1.7	(/INT5) External Interrupt 5, falling edge triggered or (ADC1.7) ADC1 input channel 7

All pins of Port 1 can be used as an analog input pin (ADC1 input channel) by setting the corresponding bit of ADCENB1 to 1. When it is used an analog input pin, its pull-up is switched off.

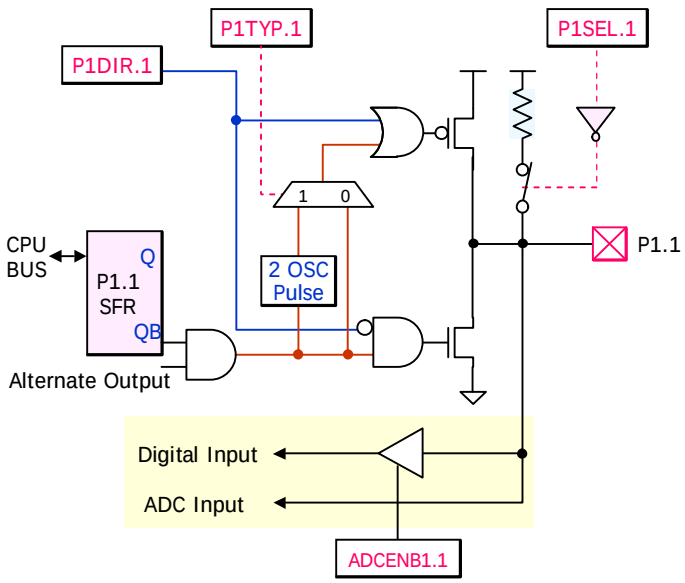


Figure 6-6 Configuration of Port 1

l P1TYPE (B4h): Port 1 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	P1TYPE.7	P1TYPE.6	P1TYPE.5	P1TYPE.4	P1TYPE.3	P1TYPE.2	P1TYPE.1	P1TYPE.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Push-pull Output / 1 = Quasi-bidirectional Output (Default)

l P1DIR (BCh): Port 1 Input/Output Control Register

Bit No.	7	6	5	4	3	2	1	0
	P1DIR.7	P1DIR.6	P1DIR.5	P1DIR.4	P1DIR.3	P1DIR.2	P1DIR.1	P1DIR.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Output / 1 = Input (Default)

l P1SEL (D9h): Port 1 Pull-up Control Register

Bit No.	7	6	5	4	3	2	1	0
	P1SEL.7	P1SEL.6	P1SEL.5	P1SEL.4	P1SEL.3	P1SEL.2	P1SEL.1	P1SEL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Pull-up resistor ON (Default) / 1 = Pull-up resistor OFF

l P1 (90h): Port 1 Register

Bit No.	7	6	5	4	3	2	1	0
	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

All pins of Port 1 can be used as an analog input pin (ADC input channel) by clearing the corresponding bit of ADCENB1 to 0. For this, the corresponding pull-up should be switched off and the corresponding I/O pin driver should operate in quasi-bidirectional output mode. So the corresponding bits of P1TYP and P1SEL should be set to 1. Also the corresponding bit of P1 Register should be set to 1.

I ADCENB1 (EDh): ADC Channel Enable Bar Register (P1 port)

Bit No.	7	6	5	4	3	2	1	0
	ADCENB1.7	ADCENB1.6	ADCENB1.5	ADCENB1.4	ADCENB1.3	ADCENB1.2	ADCENB1.1	ADCENB1.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = ADC1.X channel ON / 1 = ADC1.X channel OFF (Default)

6.2.1.3 PORT 2

Port 2 is an 8-bit, bidirectional I/O port, which has 5V tolerant input and 3.3V output. A bit of P2TYP Register determines in which mode (0: push-pull by default, 1: quasi-bidirectional) the corresponding output driver operates. A bit of P2DIR Register determines the I/O direction of the corresponding pin (0: Output, 1: Input by default). All bits of P2SEL Register are cleared to 0 by default. Setting a bit of P2SEL Register to 1 switches the corresponding pin pull-up off. When Port 2 is used for PCA output, it drives output strongly in the push-pull mode. So Port 2 pins that have 1s are pulled high strongly.

Port 2 emits the high address byte during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). It is not recommendable that a MiDAS3.0 embedded system using off-chip data memory should use Port 2 as a general purpose I/O port. It is because the state of the Port 2 latch will change during any access to the off-chip data memory. During accesses to off-chip data memory using 8-bit addresses (MOVX @Ri), Port 2 uses the P2 Register as the high address byte buffer.

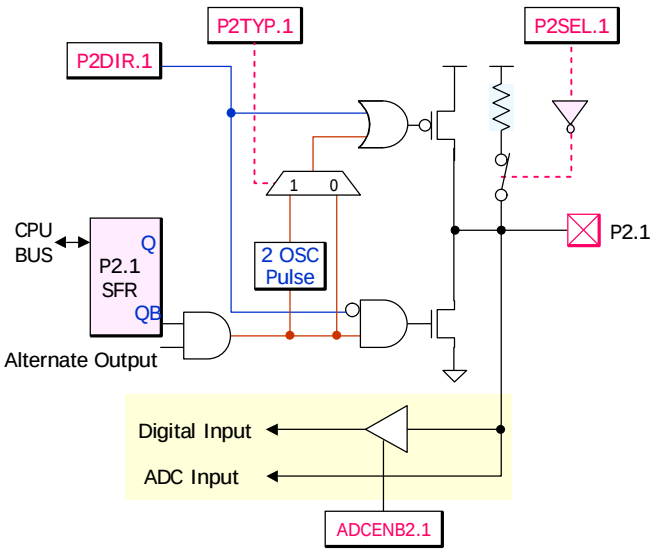


Figure 6-7 Configuration of PORT2

MiDAS3.0 contains 16,384 bytes of on-chip external data memory (AUXRAM). If no more external data memory is needed, let's set up the ENAUX (IOCFG.3) to 1 in order to use the AUXAD Register as the upper 8-bit address source instead of P2 Register. As a result, Port 2 can be used exclusively as a general I/O port.

l AUXAD (BFh) : High Address Register for MOVX with Ri

Bit No.	7	6	5	4	3	2	1	0
	AUXAD.7	AUXAD.6	AUXAD.5	AUXAD.4	AUXAD.3	AUXAD.2	AUXAD.1	AUXAD.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

If ENAUX bit (IOCFG.3) is set, "MOVX A, @Ri" and "MOVX @Ri, A" instructions refer to AUXAD instead of P2 register for the high address byte3.

l IOCFG (C7h) : I/O Configuration Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	ENAUX	-	-	-
					R/W(0)			

ENAUX: Select AUXAD for MOVX with Ri.
1 = AUXAD register serves high address for MOVX with Ri.
0 = P2 register serves high address for MOVX with Ri.

All Port 2 pins also serve the alternate functions specified below. When we use the alternate functions, the corresponding bit of the Port 2 pin should be set to high and Port 2 should operate in the quasi-bidirectional mode, for preventing errors.

Port	Alternate Functions
P2.0	(AD8) Address output 8 or (C0EX0) PCA0 Module 0 I/O or (ADC2.0) A/D converter Input 16
P2.1	(AD9) Address output 9 or (C0EX1) PCA0 Module 1 I/O or (ADC2.1) A/D converter Input 17
P2.2	(AD10) Address output 10 or (C0EX2) PCA0 Module 2 I/O or (ADC2.2) A/D converter Input 18
P2.3	(AD11) Address output 11 or (C0EX3) PCA0 Module 3 I/O or (ADC2.3) A/D converter Input 19
P2.4	(AD12) Address output 12 or (C0EX4) PCA0 Module 4 I/O or (ADC2.4) A/D converter Input 20
P2.5	(AD13) Address output 13, or (C0EX5) PCA0 Module 5 I/O or (ADC2.5) A/D converter Input 21

P2.6	(AD14) Address output 14 or (RXD1) Serial Receive UART 1 or (ADC2.6) A/D converter Input 22
P2.7	(AD15) Address output 15 or (TXD1) Serial Transmit UART 1 or (ADC2.7) A/D converter Input 23

l P2TYPE (B5h): Port 2 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	P2TYPE.7	P2TYPE.6	P2TYPE.5	P2TYPE.4	P2TYPE.3	P2TYPE.2	P2TYPE.1	P2TYPE.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Push-pull Output / 1 = Quasi-bidirectional Output (Default)

l P2DIR (BDh): Port 2 Input/Output Control Register

Bit No.	7	6	5	4	3	2	1	0
	P2DIR.7	P2DIR.6	P2DIR.5	P2DIR.4	P2DIR.3	P2DIR.2	P2DIR.1	P2DIR.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Output / 1 = Input (Default)

l P2SEL (E2h): Port 2 Pull-up Control Register

Bit No.	7	6	5	4	3	2	1	0
	P2SEL.7	P2SEL.6	P2SEL.5	P2SEL.4	P2SEL.3	P2SEL.2	P2SEL.1	P2SEL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Pull-up resistor ON (Default) / 1 = Pull-up resistor OFF

l P2 (A0h): Port 2 Register

Bit No.	7	6	5	4	3	2	1	0
	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

All pins of Port 2 can be used as an analog input pin (ADC input channel) by clearing the corresponding bit of ADCENB2 to 0. For this, the corresponding pull-up should be switched off and the corresponding I/O pin driver should operate in quasi-bidirectional output mode. So the corresponding bits of P2TYP and P2SEL should be set to 1. Also the corresponding bit of P2 Register should be set to 1.

I ADCENB2 (EEh): ADC Channel Enable Bar Register (P2 port)

Bit No.	7	6	5	4	3	2	1	0
	ADCENB2.7	ADCENB2.6	ADCENB2.5	ADCENB2.4	ADCENB2.3	ADCENB2.2	ADCENB2.1	ADCENB2.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = ADC2.X channel ON / 1 = ADC2.X channel OFF (Default)

6.2.1.4 PORT 3

Port 3 is an 8-bit, bidirectional I/O port, which has 5V tolerant input and 3.3V output. A bit of P3TYP Register determines in which mode (0: push-pull by default, 1: quasi-bidirectional) the corresponding output driver operates. A bit of P3DIR Register determines the I/O direction of the corresponding pin (0: Output, 1: Input by default). All bits of P3SEL Register are cleared to 0 by default. Setting a bit of P3SEL Register to 1 switches the corresponding pin pull-up off.

Port 3 pins can be also used alternate functions. When we use the alternate functions, the corresponding bit of the Port 3 pin should be set to high and Port 3 should operate in the quasi-bidirectional mode, for preventing errors. The alternate functions will be selected according to the combination of the SFR values.

Table 6-4 Alternate Functions

Port	Alternate Functions
P3.0	(RXD) Serial Receive UART or (ADC3.0) A/D converter Input 24
P3.1	(TXD) Serial Transmit UART or (ADC3.1) A/D converter Input 25
P3.2	(/INT0) External Interrupt 0 active low or (ADC3.2) A/D converter Input 26
P3.3	(/INT1) External Interrupt 1 active low or (ADC3.3) A/D converter Input 27
P3.4	(T0) Timer 0 input or (PWM0) PWM0 output or (ADC3.4) A/D converter Input 28
P3.5	(T1) Timer 1 input or (ADC3.5) A/D converter Input 29
P3.6	(/WR) External RAM Write strobe or (ADC3.6) A/D converter Input 30

P3.7	(/RD) External RAM Read strobe or (ADC3.7) A/D converter Input 31

The logic 1 output changes a pin status with a strong pull-up and maintains the logic 1 status with a weak pull-up. More than one port pins are assigned to alternate functions without changing the functions of the other pins.

As an example, P3.6 is used as the alternate function of /WR and P3.7 as the alternate function of /RD. These strobes are used for access to external data memory. If only the /RD signal is used, P3.6 would still function as an I/O pin.

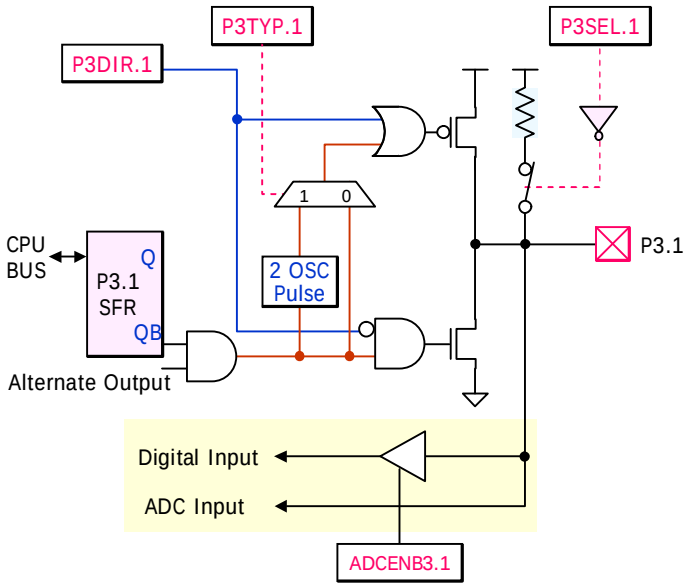


Figure 6-8 Configuration of PORT3

For another practical example, P3.2 is used as the /INT0 input pin. If external interrupt 0 is enabled, a logic 0 input to P3.2 will generate external interrupt 0. Then by disabling the external interrupt 0, P3.2 will return to a general purpose I/O pin. So the interrupt is used to “wake-up” the system from power saving modes.

I P3TYPE (B5h): Port 3 Type Control Register

Bit No.	7	6	5	4	3	2	1	0
	P3TYPE.7	P3TYPE.6	P3TYPE.5	P3TYPE.4	P3TYPE.3	P3TYPE.2	P3TYPE.1	P3TYPE.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Push-pull Output / 1 = Quasi-bidirectional Output (Default)

l P3DIR (BDh): Port 3 Input/Output Control Register

Bit No.	7	6	5	4	3	2	1	0
	P3DIR.7	P3DIR.6	P3DIR.5	P3DIR.4	P3DIR.3	P3DIR.2	P3DIR.1	P3DIR.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = Output / 1 = Input (Default)

l P3SEL (E2h): Port 3 Pull-up Control Register

Bit No.	7	6	5	4	3	2	1	0
	P3SEL.7	P3SEL.6	P3SEL.5	P3SEL.4	P3SEL.3	P3SEL.2	P3SEL.1	P3SEL.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

0 = Pull-up resistor ON (Default) / 1 = Pull-up resistor OFF

l P3 (A0h): Port 3 Register

Bit No.	7	6	5	4	3	2	1	0
	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

All pins of Port 3 can be used as an analog input pin (ADC input channel) by clearing the corresponding bit of ADCENB3 to 0. For this, the corresponding pull-up should be switched off and the corresponding I/O pin driver should operate in the quasi-bidirectional mode. So the corresponding bits of P3TYP and P3SEL should be set to 1. Also the corresponding bit of P3 Register should be set to 1.

l ADCENB3 (EEh): ADC Channel Enable Bar Register (P3 port)

Bit No.	7	6	5	4	3	2	1	0
	ADCENB3.7	ADCENB3.6	ADCENB3.5	ADCENB3.4	ADCENB3.3	ADCENB3.2	ADCENB3.1	ADCENB3.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = ADC3.X channel ON / 1 = ADC3.X channel OFF (Default)

6.2.1.5 Output Functions in open-drain or quasi-bidirectional operation (Port 0, 1, 2, and 3)

When software writes logic 0 to a port, strong pull-downs are used. However, in the case of logic 1, Port 1, 2, 3 or 4 use weak pull-up resistors and Port 0 will float (after the strong transition from 0 to 1). Thus as long as the ports are not pulled low by another output driver, the pin status will be high. The voltage of a port pin is determined by the DC current flowing out from it.
A strong pull-up FET is activated for two clock cycles if a 0-to-1 transition is programmed to a port pin, i.e. logic 1 is programmed to a port latch which was set to logic 0. This causes a fast transition of the logic levels. A weak pull-up transistor is always activated when the port latch is set to logic 1, thus maintain the output logic level. This pull-up current is much lower than that of the pull-up FET for a 0-to-1 transition; therefore logic 0 input forces the port pin to be pulled down to ground.

6.2.1.6 Input Functions in open-drain or quai-bidirectional operation (Port 0, 1, 2, and 3)

To be used as an input, the port latch must be set to logic 1, which turns off the output driver except the weak pull-up resistor. Then, the port pin is pulled high weakly by the internal pull-up resistor, but can be pulled low by an external source. Thus when the port latch is set to logic 1, the port pin will be configured as an input. When externally pulled high, it will maintain logic 1. When externally pulled low, current will flow from it and its logic state will drop to 0. So it can be used as an input. All instructions except the read-modify-write instructions read the port pin.

6.2.1.7 ESD protection of I/O ports

As electrostatic discharge (ESD) problems become more common in electronic circuits, various devices have been used to protect circuits from ESD. A pin of MiDAS3.0 family also uses two diodes and one resistor for ESD protection. The ESD protection scheme is shown in Figure 6-9.

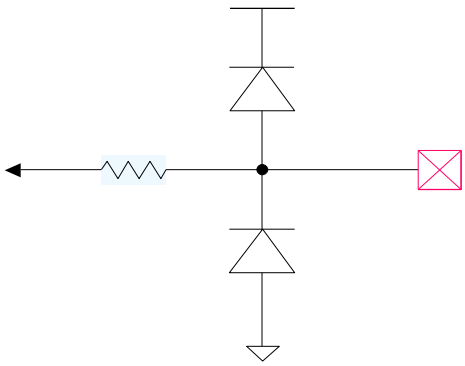


Figure 6-9 ESD protection scheme

As voltage-clamping devices, the two diodes and a resistor limit the surge voltage to a safe level for the circuit being protected.

6.2.1.8 Read-Modify-Write Instructions

Some instructions that read a port read the latch and others read the pin. The instructions that read the latch rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the latch. These are called “read-modify-write” instructions. The instructions listed below are read-modify-write instructions.

Table 6-5 Read-Modify-Write Instructions

Instruction	Description
ANL	Logical AND
ORL	Logical OR
XRL	Logical Exclusive OR (XOR)
JBC	Branch if bit is set then clear bit
CPL	Complement bit
INC	Increment
DEC	Decrement
DJNZ	Decrement and branch if not zero
MOV PX.Y, C	Move the carry bit to bit Y of Port X
CLR PX.Y	Clear bit Y of Port X
SETB PX.Y	Set bit Y of Port X

It is not obvious that the last three instructions in this list are read-modify-write instructions, but they are. They read the port byte, all 8 bits, modify the addressed bit, and then write the new byte back to the latch. The reason that read-modify-write instructions are directed to the latch rather than the pin is to avoid a possible misinterpretation of the voltage level at the pin. For example, a port bit might be used to drive the base of transistor. When logic 1 is written to the bit, the transistor is turned on. If the CPU the reads the same port bit at the pin rather than the latch, it will read the base voltage of the transistor and interpret it as logic 0. Reading the latch rather than the pin will return the correct value of logic 1.

6.2.2 WDT (Watchdog Timer)

The Watchdog timer is a free-running timer with programmable time-out intervals. It is available for a system monitor, a time-base generator or an event timer. It allows an automatic recovery from execution errors due to some cause including external noise. It is a kind of counter with the programmable bit length. It counts the system clock pulses. Software can always initialize the watchdog timer. When the count rolls over from all 1s to all 0s, the time-out occurs. At that time, the watchdog interrupt flag is set. The watchdog interrupt will occur if EWDT (watchdog interrupt enable) and EA (global enable) are set to 1. If the watchdog reset is enabled by setting the bit EWT to logic 1, the watchdog reset will occur after 512 clock cycles since the watchdog timer interrupt flag is set, However, the watchdog timer can be initialized by setting the bit RWT to logic 1 during the delay of 512 clock cycles. Then the watchdog reset will not occur. The watchdog interrupt and reset may be used independently of each other. So you can use them separately or together.

Figure 6-10 shows the block diagram for Watchdog timer.

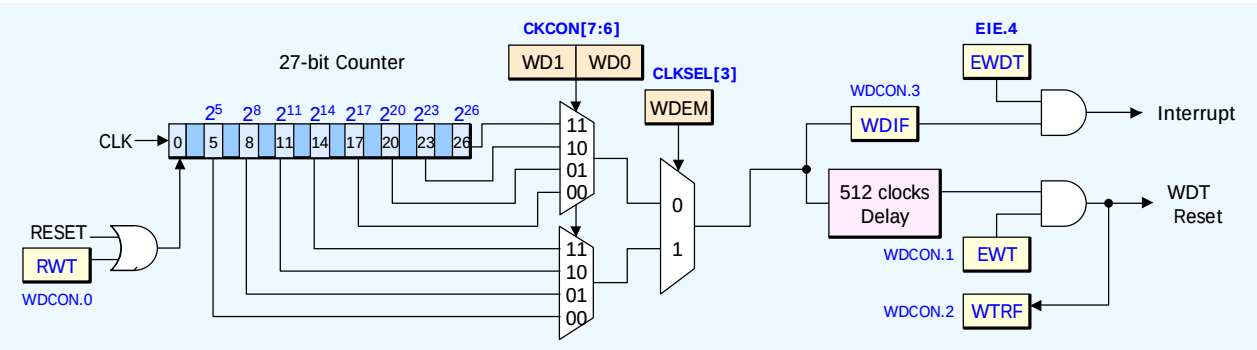


Figure 6-10 Block Diagram for Watchdog Timer

l WDCON (D8h) : Watchdog Timer Timer & Power Status Register

Bit No.	7	6	5	4	3	2	1	0
	-	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT
		R/W(1)	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

POR: Power-on Reset Flag
WDIF: Watchdog Timer Interrupt Flag
WTRF: Watchdog Timer Reset Flag. Only cleared by S/W.
EWT: Watchdog Timer Reset Enable
RWT: Restart Watchdog Timer

l CKCON (8Eh) : Clock Control Register

Bit No.	7	6	5	4	3	2	1	0
	WD1	WD0	T2M	T1M	T0M	-	U1T2DIS	U0T2DIS
	R/W(1)	R/W(1)	R/W(0)	R/W(0)	R/W(0)		R/W(0)	R/W(0)

WD1, WD0: Watchdog Timer Mode Select

l CLKSEL (FBh) : Wakeup/I2C Data Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	XR/HF	WDEM	XR/PL	RG/PR	OSC32EB
				R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

WDEM: Watchdog Timer Extension Mode Select

First, the watchdog timer should be initialized by setting the bit RWT (WDCON.0) to 1. Then it restarts from 0. After clearing the Watchdog timer, the hardware clears the RWT bit automatically. The time-out interval is selected by the values of the three bits, WD1, WD0 (CKCON.7, CKCON.6), and WDEM (CLKSEL.3). When the Watchdog timer reaches to the selected time-out, the Watchdog interrupt flag WDIF (WDCON.3) is set to 1. If the watchdog reset is enabled by setting the bit EWT to logic 1, the watchdog reset will occur after 512 clock cycles since the watchdog timer interrupt flag is set, However, the watchdog timer can be initialized by setting the bit RWT to logic 1 during the delay of 512 clock cycles. Then the watchdog reset will not occur. The reset process will last for thirty clock cycles, and the Watchdog timer reset flag WTRF (WDCON.2) will be set. This flag indicates to software that the reset was caused by the Watchdog timer.

If the reset and the interrupt are disabled, the watchdog timer can be used as a simple timer. Every time the watchdog timer reaches to the selected time-out, the WDIF flag is set. Software can detect the watchdog time-out by the WDIF flag and restart the timer by setting RWT bit. The watchdog timer can also be used as a timer with a long time-out interval. The interrupt is enabled in this case. Every time the time-out occurs, the interrupt service routine will be called if the global interrupt enable bit EA is set.
The main purpose of the watchdog timer is detection of execution errors. This is important for real-time control applications. It is because some power glitches or electro-magnetic interferences may cause execution errors.

The watchdog time-out interval varies with the clock frequency. The watchdog timer reset will occur after 512 clocks from the end of the time-out interval.

Table 6-6 Time-out intervals for the watchdog timer

WDEM	WD1	WD0	Interrupt time-out (@25MHz)		Reset time-out (@25MHz)	
0 (Default)	0	0	2^{17} clocks	5.243ms	2^{17} + 512 clocks	5.263ms
	0	1	2^{20} clocks	41.943ms	2^{20} + 512 clocks	41.964ms
	1	0	2^{23} clocks	335.544ms	2^{23} + 512 clocks	335.565ms
	1	1	2^{26} clocks	2684.355ms	2^{26} + 512 clocks	2684.375ms
1	0	0	2^{5} clocks	0.001ms	2^{5} + 512 clocks	0.022ms
	0	1	2^{8} clocks	0.010ms	2^{8} + 512 clocks	0.031ms
	1	0	2^{11} clocks	0.082ms	2^{11} + 512 clocks	0.102ms
	1	1	2^{14} clocks	0.665ms	2^{14} + 512 clocks	0.676ms

The watchdog timer will be disabled by a power-on/fail reset. The watchdog timer reset does not disable the watchdog timer, but will restart it. The default watchdog time-out interval is 2^{26} clocks, which is the longest.

6.2.3 Timer 0/1/2

The MiDAS3.0 family has three 16-bit programmable timer/counters.

6.2.3.1 Timer/Counters 0 & 1

Timer/Counter 0 or 1 has two 8-bit registers that form the 16-bit counting register. For Timer/Counter 0, they are the upper 8-bit register TH0 and the lower 8-bit register TL0. Similarly, Timer/Counter 1 has two 8-bit registers, TH1 and TL1. They operate as either a timer or an event counter.

When operating as a timer, the registers counts clock pulses. The timer clock frequency can be 1/12 or 1/4 of the system clock frequency. In the “Counter” function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 pin for Timer 0 and T1 pin for Timer 1. The T0 and T1 external inputs are sampled during S3 state of every 4-clock system or every 12-clock system. If the sampled value is high in one machine cycle and low in the next, a valid high-to-low transition on the pin is recognized and the count register is incremented. Since it takes 8 clocks (4-clock system) or 24 clocks (12-clock system) to recognize a negative transition on the pin, the maximum counting rate is 1/8 or 1/24 of the master clock frequency. In either the “Timer” or “Counter” mode, the count register will be updated at S2 state. Therefore, in the “Counter” mode, the recognized negative transition on T0 or T1 pin causes the register value to increase in the next machine cycle after the negative edge was detected.

The “Timer” or “Counter” mode is selected by the control bit “C/T” in the TMOD register; bit 2 of TMOD selects the mode for Timer/Counter 0 and bit 6 of TMOD the mode for Timer/Counter 1. In addition, each Timer/Counter has four operating modes. The modes are selected by bits M0 and M1 in the TMOD register.

Table 6-7 Operation modes of timers 0 and 1

Mode / Timer	Mode 0 (M1, M0 = 00)	Mode 1 (M1, M0 = 01)	Mode 2 (M1, M0 = 10)	Mode 3 (M1, M0 = 11)
Timer 0	13-bit T/C	16-bit T/C	8-bit T/C With automatic reload (TL0 <- TH0)	8-bit T/C (TL0) -> Timer 0 interrupt 8-bit T/C (TH0) -> Timer 1 interrupt
Timer 1	13-bit T/C	16-bit T/C	8-bit T/C With automatic reload (TL1 <- TH1)	Halt

l TCON (88h) : Timer/Counter Control Register

Bit No.	7	6	5	4	3	2	1	0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

TF1: Timer 1 Overflow Flag.
TR1: Timer 1 Run Enable.
TF0: Timer 0 Overflow Flag.
TR0: Timer 0 Run Enable.

l TMOD (89h) : Timer/Counter 0 & 1 Mode Control Register

Bit No.	7	6	5	4	3	2	1	0
	GATE	C/T	M1	M0	GATE	C/T	M1	M0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

Timer 1: GATE[7], C/T[6], M1:M0[5:4]
Timer 0: GATE[3], C/T[2], M1:M0[1:0]
GATE: Timer 0 Gate control.
When TRx (in TCON) is set and GATE=1,
Timer x will run only while INTx pin is high (hardware control).
When GATE=0 Timer x will run only while TRx=1 (software control).
C/T : Counter / Timer Selector.
0 for Timer operation (input from internal system clock).
1 for Counter operation (input from Tx input pin).
M1, M0: Mode Selection bits

l TL0 (8Ah) : Timer/Counter 0 Low Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TL0.7	TL0.6	TL0.5	TL0.4	TL0.3	TL0.2	TL0.1	TL0.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

l TH0 (8Ch) : Timer/Counter 0 High Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TH0.7	TH0.6	TH0.5	TH0.4	TH0.3	TH0.2	TH0.1	TH0.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

l TL1 (8Bh) : Timer/Counter 1 Low Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TL1.7	TL1.6	TL1.5	TL1.4	TL1.3	TL1.2	TL1.1	TL1.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

l TH1 (8Dh) : Timer/Counter 1 High Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TH1.7	TH1.6	TH1.5	TH1.4	TH1.3	TH1.2	TH1.1	TH1.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

6.2.3.1.1 Time-base Selection

The MiDAS3.0 family provides the two time-bases for the timer/counter. Its timers can operate at the same speed with those of the standard 80C52 family. This can ensure that timer programs of the MiDAS3.0 family have the same timing with those of the standard 80C52 family. This is the default time-base of the MiDAS3.0 family. However, the timers of the MiDAS3.0 family can also operate in the turbo mode, where they increments at the rate of 1/4 clock speed. This time-base is selected by the T0M or T1M bit in the CKCON register. A reset clears these bits to 0, and the timers then operate in the standard 80C52 mode. A user should set these bits to 1 if the timers are to operate in turbo mode.

l CKCON (8Eh) : Clock Control Register

Bit No.	7	6	5	4	3	2	1	0
	WD1	WD0	T2M	T1M	T0M	-	U1T2DIS	U0T2DIS
	R/W(1)	R/W(1)	R/W(0)	R/W(0)	R/W(0)		R/W(0)	R/W(0)

T1M: Timer 1 Time-base Selection
T1M=1, Time-base 4 clocks not 12 clocks
T0M: Timer 0 Time-base Selection
T0M=1, Time-base 4 clocks not 12 clocks

6.2.3.1.2 Mode 0

In Mode 0, the timer/counters operate as an 8-bit counter with a divide-by-32 prescaler. This 13-bit counter consists of 8 bits of THx and the lower 5 bits of TLx. The upper 3 bits of TLx are indeterminate and should be ignored.

The value in the TLx register increases when the timer/counters detect a 1-to-0 transition. Every time the value of the fifth bit in TLx changes from 1 to 0, the count in the THx register increases. When the count in THx rolls over from FFh to 00h, the overflow flag TFx in the TCON register is set to 1. The timer/counters operate only if TRx is set and either GATE=0 or /INTx=1. When C/T is cleared to 0, it will count clock pulses, and if C/T is set to 1, it will count 1-to-0 transitions on T0 (P3.4) for Timer 0 and T1 (P3.5) for Timer 1.

When the 13-bit counter rolls over from 1FFFh to 000H, the timer overflow flag TFx is set to 1. If enabled, a timer interrupt will occur. When used as a timer, the time-base is selected as either {clock cycles/12} or {clock cycles/4} by the bits TxM of the CKCON register.

6.2.3.1.3 Mode 1

Mode 1 is the same as Mode 0, except that the timer register is being run with all 16 bits counter. This means that all the bits of THx and TLx are used. Roll-over occurs when the timer changes from FFFFh to 0000h. The timer overflow flag TFx is set to 1 and a timer interrupt will occur if enabled. The selection of the time-base is the same as Mode 0. The function of GATE bit is the same as Mode 0.

6.2.3.1.4 Mode 2

In Mode 2, the Timer/Counters operate as 8-bit counters with automatic reload. TLx operates as an 8-bit counter, while THx holds the reload value. When the TLx register overflows from FFH to 00H, the TFx bit in TCON is set to 1 and the value of THx is reloaded into TLx. The counting process continues from the reloaded value. The reload operation doesn't change the values of the THx register. Counting is enabled when the TRx = 1 and either GATE = 0 or /INTx =1. As Mode 0 and 1, Mode 2 counts either clock pulses (clock/12 or clock/4) or external input pulses on pin Tx.

6.2.3.1.5 Mode 3

In Mode 3, Timer/Counters 0 and 1 operate differently from each other. Timer/Counter 1 in Mode 3 simply maintains its value. TL0 and TH0 of Timer/Counter 0 operate independently as two separate 8-bit counters. TL0 uses the Timer/Counter 0 control bits: C/T, GATE, TR0, /INT0 and TF0. It is determined by C/T (TMOD.2) whether the TL0 can count clock pulses (clock/12 or clock/4) or 1-to-0 transitions on pin T0. TH0 operates only as a timer (clock/12 or clock/4) and uses the control bits TR1 and TF1 of Timer/Counter 1. Mode 3 is used when an extra 8-bit timer is needed. With Timer 0 in Mode 3, Timer 1 can still be used in Modes 0, 1 and 2. However, it no longer has control over its overflow flag TF1 and the enable bit TR1. Timer 1 can still be used as a timer/counter and retains the use of GATE and INT1 pin. In this condition, it can be turned on and off by switching it out of and into its own Mode 3. It can also be used as a baud rate generator for the UART. Figure 6-11 illustrates the various operation modes for Timer/Counter 0/1.

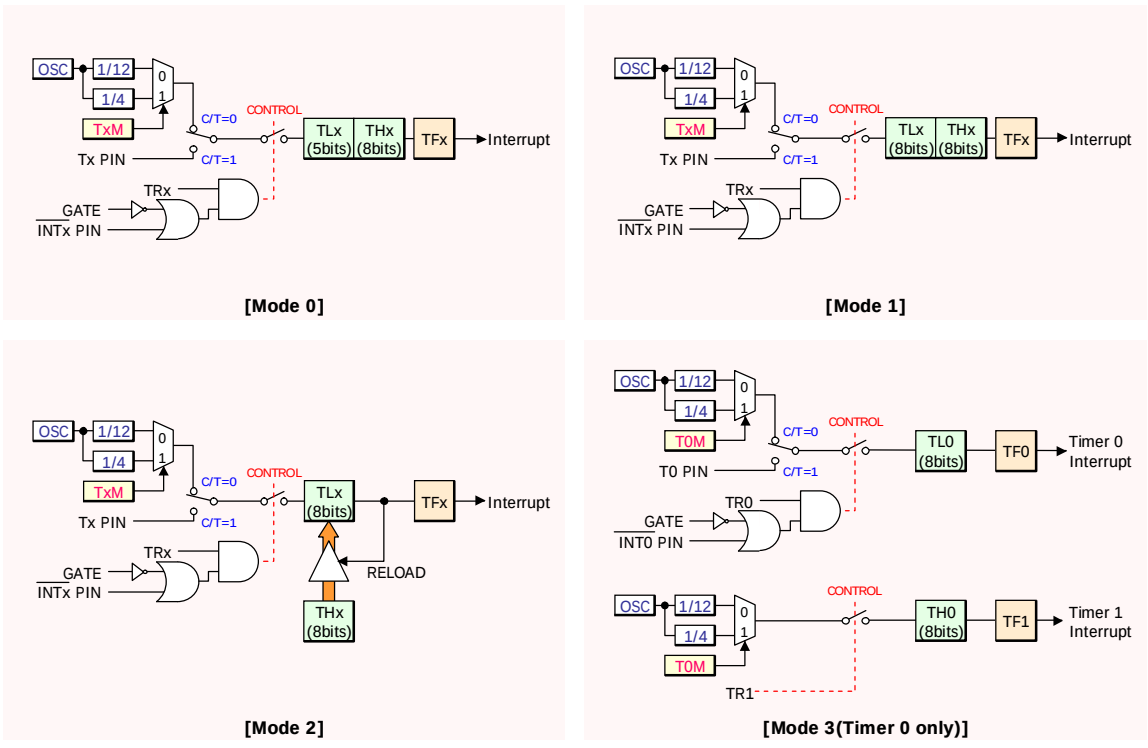


Figure 6-11 Timer/Counter 0/1 in Mode 0/1/2/3

6.2.3.2 Timer/Counter 2

Timer/Counter 2 is a 16-bit up/down counter that is configured by the T2MOD register and controlled by

the T2CON register. Like Timers 0 and 1, it can operate either as a timer or as an event counter. Timer/Counter 2 counts either the external pulses of T2 pin (C/T2=1) or the prescaled clock(C/T2=0), which is divided by 12 or 4. If T2M = 0, 1/12 clock will be selected. Timer 2 counts when TR2 is 1. It stops when TR2 is 0.

Table 6-8 Operation modes of timer 2

1. 16-bit Auto-reload [RCLK+TCLK=0, CP/RL2=0, T2OE=0]	16-bit Timer/Counter With Automatic Reload (TH2, TL2｜RCAP2H, RCAP2L)
2. 16-bit Capture [RCLK+TCLK=0, CP/RL2=1, T2OE=0]	16-bit Timer/Counter with Capture (RCAP2H, RCAP2L｜TH2, TL2)
3. Baud Rate Generator [RCLK+TCLK=1, CP/RL2=X, T2OE=X]	Baud Rate Generation * Timer 2 Interrupt Disable
4. Programmable Clock Out [RCLK+TCLK=X, CP/RL2=0, T2OE=1]	Clock-out on P1.0

6.2.3.2.1 Capture Mode

The capture mode is established by setting the CP/RL2 bit in the T2CON register to 1. In the capture mode, Timer/Counter 2 operates as a 16-bit up counter. When the counter rolls over from FFFFh to 0000h, the TF2 bit is set to 1, which will generate an interrupt request. If the EXEN2 bit is set, a negative transition of T2EX pin will cause the value in the TL2 and TH2 registers to be captured by the RCAP2L and RCAP2H registers. This action also causes the EXF2 bit in T2CON to be set to 1, which will also generate an interrupt.

6.2.3.2.2 Auto-reload Mode, Counting Up

The auto-reload mode as an up counter is enabled by clearing the CP/RL2 bit in the T2CON SFR and clearing the DCEN bit in T2MOD SFR. In this mode, Timer/Counter 2 is a 16-bit up counter. When the counter rolls over from FFFFh to 0000h, TL2 and TH2 are reloaded with the 16-bit value in SFRs RCAP2L and RCAP2H. The TF2 bit is also set to 1. If the EXEN2 bit is set, then a negative transition of T2EX pin will also cause a reload. The EXF2 bit in T2CON is also set to 1.

6.2.3.2.3 Auto-reload Mode, Counting Up/Down

Timer/Counter 2 operates in auto-reload mode as an up/down-counter if CP/RL2 bit in T2CON is cleared and the DCEN bit in T2MOD is set. In this mode, Timer/Counter 2 is an up/down-counter whose direction is controlled by the T2EX pin. When T2EX is high, Timer 2 counts up. Timer overflow occurs at FFFFh, which sets the TF2 flag. The overflow also causes the 16-bit value in RCAP2H and RCAP2L registers to be reloaded into the timer registers TH2 and TL2. When T2EX is low, Timer 2 counts down. Timer underflow occurs when the count in the timer registers TH2 and TL2 equals the value stored in RCAP2H and RCAP2L registers. The underflow sets TF2 flag and reloads FFFFh into the timer registers.
The EXF2 bit toggles when Timer 2 overflows or underflows according to the direction of the count. EXF2 does not generate any interrupt in this mode. This bit can be used to provide 17-bit resolution.

6.2.3.2.4 Baud Rate Generator Mode

The baud rate generator mode is selected by setting either the RCLK or TCLK bits in the T2CON register to 1. In this mode, Timer/Counter 2 is a 16-bit counter with automatic reload when the count rolls over from FFFFh to 0000h. However, rolling-over does not set the TF2 bit. If EXEN2 bit is set, a negative transition of the T2EX pin will set EXF2 bit in T2CON and cause an interrupt request. Timer 2 interrupt caused by TF2 will be ignored in this mode.

6.2.3.2.5 Programmable Clock-out

In the clock-out mode, Timer 2 operates as a 50% duty cycle, programmable clock generator. TL2 increases by the rate of (Oscillator frequency) / 2. Timer 2 repeatedly counts to overflow from a reloaded value. At overflow, the Timer 2 registers, TL2 and TH2, will be reloaded with the 16-bit value in registers RCAP2L and RCAP2H, which are preset by software. In this mode, Timer 2 overflows do not generate interrupts as not in the baud rate generator mode. So Timer 2 can be used as a baud rate generator and a clock generator simultaneously. For this configuration, the baud rates and clock frequencies are not independent since both functions use the values in the RCAP2H and RCAP2L registers.
Timer 2 is programmed for the clock-out mode as follows:

- Set the T2OE bit in the T2MOD register.

- Clear the C/T2 bit in the T2CON register.
- Determine the 16-bit reload value from the formula and enter it in RCAP2H/RCAP2L registers.
- Enter a 16-bit initial value in timer registers TH2/TL2. It can be the same as the reload value or a different one depending on the application.
- To start the timer, set the TR2 run control bit in the T2CON register.

The formula gives the clock-out frequency as a function of the system oscillator frequency and the value in the RCAP2H and RCAP2L registers:

(Clock-out frequency) = (Oscillator frequency) / [4 * {65536 - (RCAP2H, RCAP2L)}]

The generated clock signal is brought out to T2 pin (P1.0)

Figure 6-12 and Figure 6-13 illustrate the operating modes supported by Timer/Counter 2.

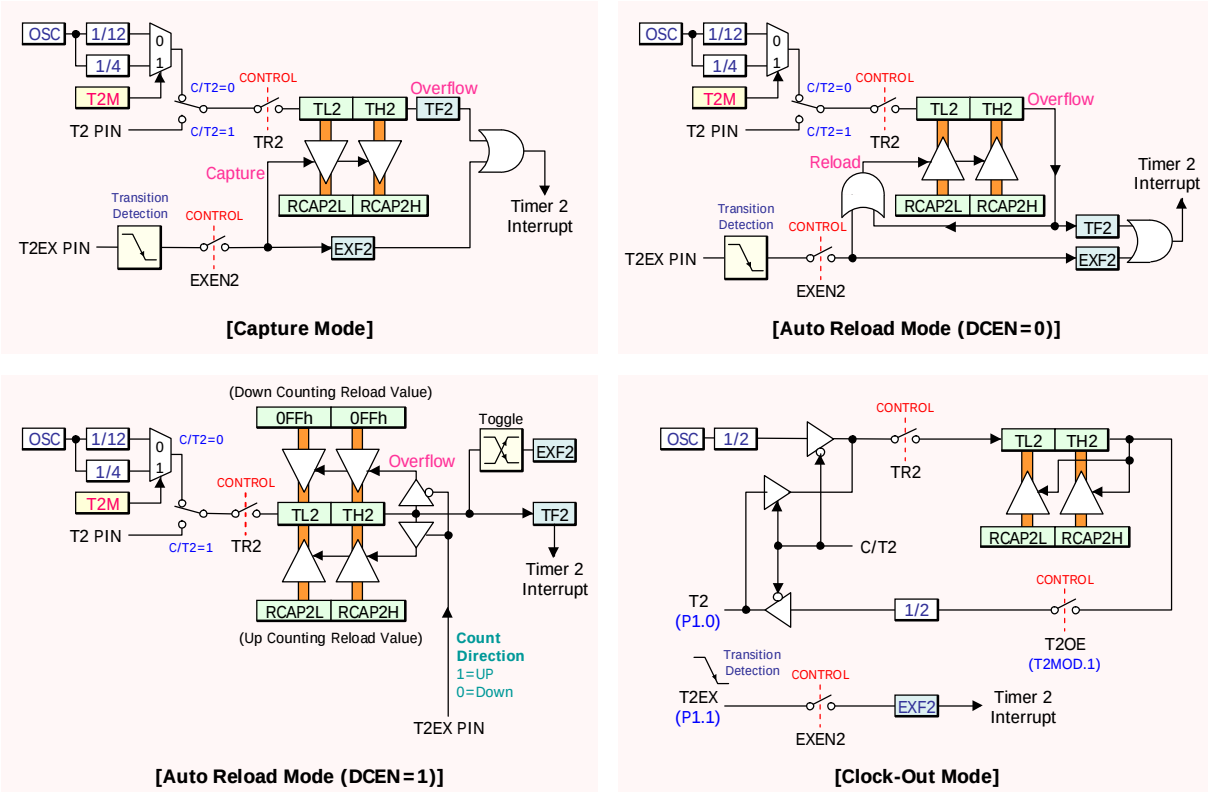


Figure 6-12 Timer/Counter 2 in Capture, Auto reload, and Clock-out Mode

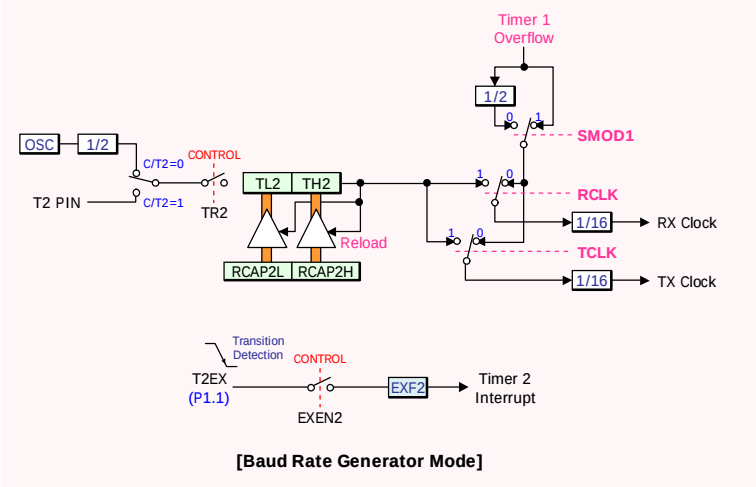


Figure 6-13 Timer/Counter 2 in Baud Rate Generator Mode

I T2CON (C8h) : Timer 2 Control Register

Bit No.	7	6	5	4	3	2	1	0
	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

TF2: Timer 2 Overflow Flag
EXF2: Timer 2 External Flag
RCLK: Receive Clock Flag
TCLK: Transmit Clock Flag
EXEN2: Timer 2 External Enable Flag
TR2: Timer 2 Run Enable
C/T2 : Timer or Counter Selection. If C/T2=0, Timer Operation.
CP/RL2: Capture/Reload Flag.

I CKCON (8Eh) : Clock Control Register

Bit No.	7	6	5	4	3	2	1	0
	WD1	WD0	T2M	T1M	T0M	-	U1T2DIS	U0T2DIS
	R/W(1)	R/W(1)	R/W(0)	R/W(0)	R/W(0)		R/W(0)	R/W(0)

T2M: Timer 1 Time-base Selection
T2M=1, Time-base 4 clocks not 12 clocks

l T2MOD (C9h) : Timer 2 Mode Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	T2OE	DCEN
							R/W(0)	R/W(0)

T2OE: Timer 2 Clock Output to P1.0
DCEN: Timer 2 Down Counter Enable

l TL2 (CCh) : Timer 2 Low Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TL2.7	TL2.6	TL2.5	TL2.4	TL2.3	TL2.2	TL2.1	TL2.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

l TH2 (CDh) : Timer 2 High Byte Register

Bit No.	7	6	5	4	3	2	1	0
	TH2.7	TH2.6	TH2.5	TH2.4	TH2.3	TH2.2	TH2.1	TH2.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

l RCAP2L (CAh) : Timer 2 Capture/Reload Low Byte Register

Bit No.	7	6	5	4	3	2	1	0
	RCAP2L.7	RCAP2L.6	RCAP2L.5	RCAP2L.4	RCAP2L.3	RCAP2L.2	RCAP2L.1	RCAP2L.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

l RCAP2H (CBh) : Timer 2 Capture/Reload High Byte Register

Bit No.	7	6	5	4	3	2	1	0
	RCAP2H.7	RCAP2H.6	RCAP2H.5	RCAP2H.4	RCAP2H.3	RCAP2H.2	RCAP2H.1	RCAP2H.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

6.2.4 UART (UART 0/1: Universal Asynchronous Rx/Tx)

The MiDAS3.0 family has two serial full duplex ports. ‘Full duplex’ means the port can transmit and receive simultaneously. The two ports are named UART 0 and UART 1, respectively. It includes additional features such as the Automatic Address Recognition. The UART provides both synchronous and asynchronous communication modes. In the synchronous mode, the MiDAS3.0 family generates the communication clock and operates in a half duplex mode. In the asynchronous mode, the UART operates as a full duplex port. The transmit buffer and the receive buffer are both addressed as the SBUF register for UART 0 or the SBUF1 register for UART 1. However, writing to SBUF for UART 0 or SBUF1 for UART 1 loads the transmit buffer, and reading SBUF for UART 0 or SBUF1 for UART1 accesses a physically separate receive buffer. The UART ports can operate in four different modes.

6.2.4.1 UART 0 and UART 1

UART 1 has the completely same functions with UART 0. Besides, all registers and I/O pins of UART 1 play the same role with those of UART 0. The registers listed below are the registers for UART 0 and UART 1 communication.

- **PCON (87h) : Power Control Register**

Bit No.	7	6	5	4	3	2	1	0
	SMOD1	SMOD0	-	POF	GF1	GF0	PD	IDL
	R/W(0)	R/W(0)		R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

SMOD1: Timer 1 baud rate double in UART mode 1, 2, and 3
SMOD2: Enable SM0 access. Don’t modify this bit

- **SCON (98h) : Serial Port Control Register for UART0**

Bit No.	7	6	5	4	3	2	1	0
	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

SM0, SM1: Serial Port Mode Selection (See Table 6-9)
SM2: Enable the Automatic Address Recognition in Mode 2 and 3.
Clear after receiving the address.
In Mode 1, Valid Stop Bit Check if SM2=1.

In Mode 0, SM2 should be 0.
REN: Serial Reception Enable.
TB8: 9th data bit that will be transmitted in Mode 2 and 3.
RB8: 9th data bit that was received in Mode 2 and 3.
In Mode 1, RB8 is equal to Stop Bit if SM2=0.
In Mode 0, RB8 is not used.
TI: Transmission Interrupt Flag. This bit must be cleared by software.
RI: Reception Interrupt Flag. This bit must be cleared by software.

Table 6-9 Summary of Modes in UART

Mode	SM1	SM0	Data Size	Data Format	Baud Rate Formulas
0	0	0	8 bit	8 data bit	1/4 * oscillator clock
0	0	1	10 bit	Start bit, 8 data bit, stop bit	1/32 * Timer 1 overflow (SMOD1=0) 1/16 * Timer 1 overflow (SMOD1=1) 1/16 * Timer 2 overflow rate
2	1	0	11 bit	Start bit, 8 data bit, programmable bit, stop bit	1/32 * oscillator clock (SMOD1=0) 1/16 * oscillator clock (SMOD1=1)
0	1	1	11 bit	Start bit, 8 data bit, programmable bit, stop bit	1/32 * Timer 1 overflow (SMOD1=0) 1/16 * Timer 1 overflow (SMOD1=1) 1/16 * Timer 2 overflow rate

l SBUF (99h) : Serial Data Buffer Register for UART0

Bit No.	7	6	5	4	3	2	1	0
	SBUF.7	SBUF.6	SBUF.5	SBUF.4	SBUF.3	SBUF.2	SBUF.1	SBUF.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

The transmission buffer and the reception buffer are separated.
However, the transmission/reception buffers have the same address.

l SADDR(A9h) : Slave Address Register of UART0

Bit No.	7	6	5	4	3	2	1	0
	SADDR.7	SADDR.6	SADDR 5	SADDR.4	SADDR.3	SADDR.2	SADDR.1	SADDR.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

The given or broadcast address assigned to serial port0.

I SADEN(B9h) : Slave Address Mask Enable Register of UART0

Bit No.	7	6	5	4	3	2	1	0
	SADEN.7	SADEN.6	SADEN.5	SADEN.4	SADEN.3	SADEN.2	SADEN.1	SADEN.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I CKCON (8Eh) : Clock Control Register

Bit No.	7	6	5	4	3	2	1	0
	WD1	WD0	T2M	T1M	T0M	-	U1T2DIS	U0T2DIS
	R/W(1)	R/W(1)	R/W(0)	R/W(0)	R/W(0)		R/W(0)	R/W(0)

U1T2DIS disables RCLK/TCLK control for UART1
U0T2DIS disables RCLK/TCLK control for UART0

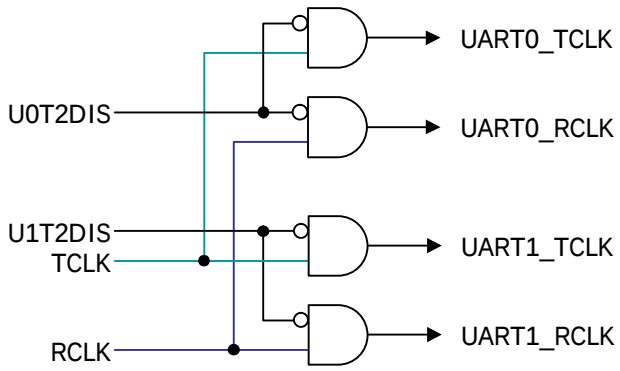


Figure 6-14 UART baud rate source control by U1T2DIS and U0T2DIS

When using the U1T2DIS and U0T2DIS bits, you can generate the baud rates of UART 1 and UART 0 by different timers. For more information, see Table 6-10. Let's assume that the values of RCLK and TCLK are 1.

Table 6-10 Timer Selection by U1T2DIS and U0T2DIS

U1T2DIS	U0T2DIS	UART 1 baud rate	UART 0 baud rate
0	0	Determined by Timer 2	Determined by Timer 2
0	1	Determined by Timer 2	Determined by Timer 1
1	0	Determined by Timer 1	Determined by Timer 2
1	1	Determined by Timer 1	Determined by Timer 1

l SCON1 (B1h) : Serial Port Control Register for UART1

Bit No.	7	6	5	4	3	2	1	0
	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

SM0, SM1: Serial Port Mode Selection (See Table 6-9)
SM2: Enable the Automatic Address Recognition in Mode 2 and 3.
Clear after receiving the address.
In Mode 1, Valid Stop Bit Check if SM2=1.
In Mode 0, SM2 should be 0.
REN: Serial Reception Enable.
TB8: 9th data bit that will be transmitted in Mode 2 and 3.
RB8: 9th data bit that was received in Mode 2 and 3.
In Mode 1, RB8 is equal to Stop Bit if SM2=0.
In Mode 0, RB8 is not used.
TI: Transmission Interrupt Flag. This bit must be cleared by software.
RI: Reception Interrupt Flag. This bit must be cleared by software.

l SBUF1 (A1h) : Serial Data Buffer Register for UART1

Bit No.	7	6	5	4	3	2	1	0
	SBUF1.7	SBUF1.6	SBUF1.5	SBUF1.4	SBUF1.3	SBUF1.2	SBUF1.1	SBUF1.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

The transmission buffer and the reception buffer are separated.
However, the transmission/reception buffers have the same address.

l SADDR1(AAh) : Slave Address Register of UART1

Bit No.	7	6	5	4	3	2	1	0
	SADDR1.7	SADDR1.6	SADDR1.5	SADDR1.4	SADDR1.3	SADDR1.2	SADDR1.1	SADDR1.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

The given or broadcast address assigned to serial port0.

l SADEN1(ABh) : Slave Address Mask Enable Register of UART1

Bit No.	7	6	5	4	3	2	1	0
	SADEN1.7	SADEN1.6	SADEN1.5	SADEN1.4	SADEN1.3	SADEN1.2	SADEN1.1	SADEN1.0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

It is possible to use UART1 instead of UART0 only by exchanging SCON/SBUF/SADDR/SADEN in the program code for SCON1/SBUF1/SADDR1/SADEN1. After that, be sure to use RXD1/TXD1 instead of RXD/TXD as I/O pins.

For these reasons, only UART 0 will be explained.

6.2.4.2 Mode 0

This mode provides synchronous communication for external devices. Serial data enters and exits through the RxD. TxD outputs the shift clock. This mode, therefore, supports a half duplex mode of serial communication. 8 bits are transmitted/received: 8 data bits (LSB first). Because the baud rate is fixed at 1/4 of the oscillator frequency, SM2 bit (SCON.5) should be 0 in mode 0.

The UART port sends and receives data through the RxD line. The TxD line outputs the shift clock. The clock shifts data bits into and out of the devices at the either end of the line. Transmission is initiated by any instruction that uses SBUF as a destination buffer. The shift clock will be activated and data will be shifted out to the RxD pin until all 8 bits are transmitted. The data on RxD will appear four clock periods at the center of the falling edge of TxD pin. This ensures that at the receiving end the data on RxD line can either be clocked on the rising edge of the shift clock on TxD.

The TI flag is set to 1 during the S1 state following the end of transmission of the last bit. The UART port will receive data when REN is 1 and RI is zero. The shift clock (TxD) will be activated and the serial port will latch data at the low state of shift clock. The external device should therefore present data at the rising edge on the shift clock. This process will be repeated until all the 8 bits have been received. The RI flag is set in S1 state following the last rising edge of the shift clock on TxD. This will stop reception and hold stop state until the RI is cleared by software.

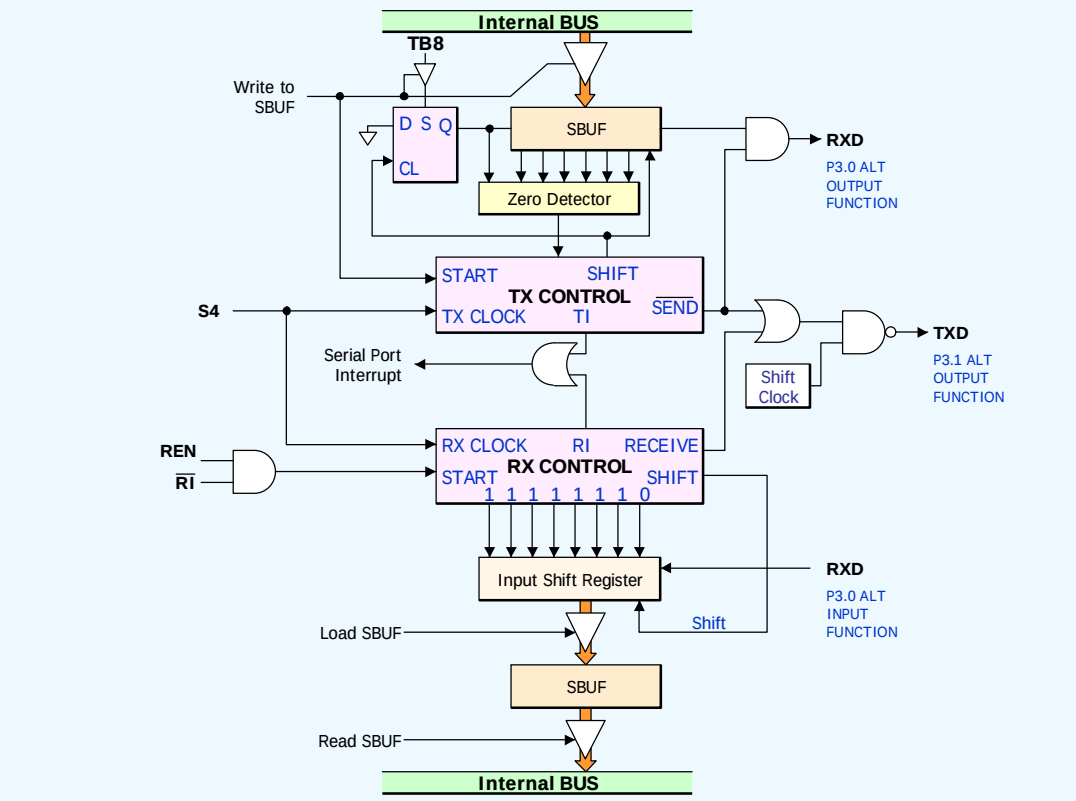


Figure 6-15 UART Mode 0

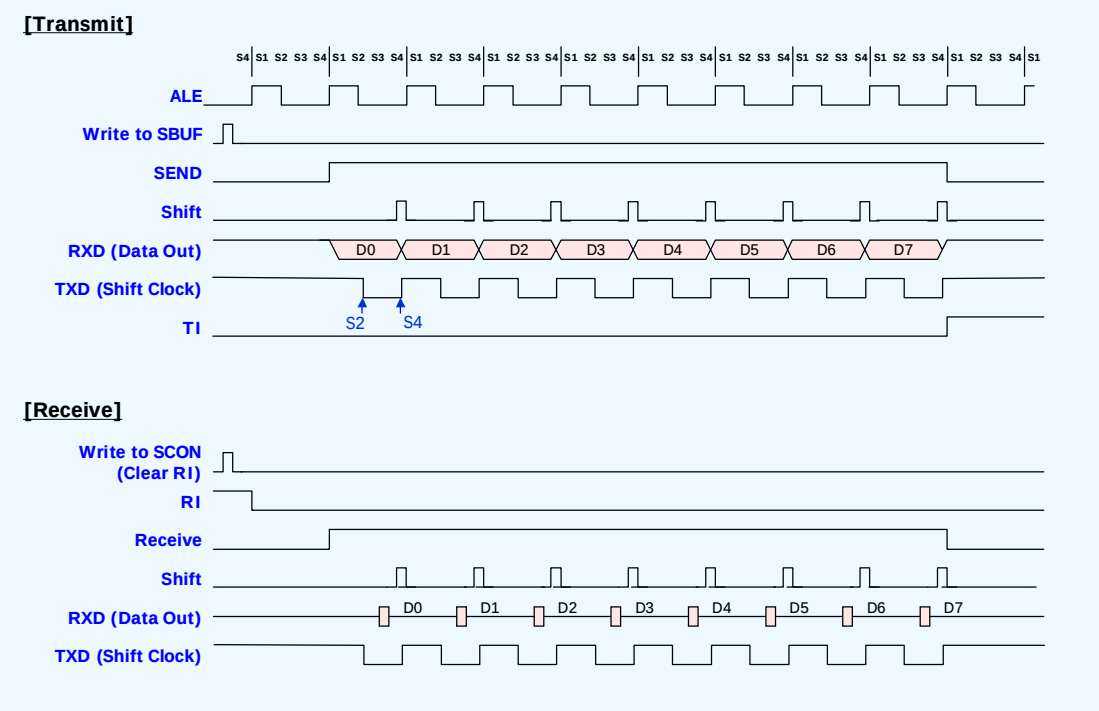


Figure 6-16 UART Mode 0 Timing

6.2.4.3 Mode 1

In Mode 1, the UART communicates asynchronously in the full duplex. Ten bits are transmitted (through TXD), or received (through RXD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receiving, the stop bit goes into RB8 in the SCON. The baud rate can be programmed to be 1/16 or 1/32 of the Timer 1 overflow or 1/16 of the Timer 2 overflow rate. It varies widely according to the automatic reload value of Timer 1 or Timer 2.

Transmission is initiated by any instruction that uses SBUF as a destination register. The serial data is sent to TxD pin at S1 state after the first roll-over of the divide-by-16 counter. The next bit is placed on TxD pin at S1 state after the next roll-over of the divide-by-16 counter. Thus, the transmission is synchronized to the divide-by-16 counter, not directly to the "write to SBUF" signal. After transmission of all 8-bit data, the stop bit is transmitted. The TI flag is set in the S1 state after the stop bit has been sent to TxD pin. This will occur at the 10^{th} roll-over of the divide-by-16 counter after a write to SBUF.
Reception is enabled only if REN is "1." It is initiated by detecting 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times whatever baud rate established. When a transition is detected, the divide-by-16 counter is immediately reset. This aligns its roll-overs with the boundaries of the incoming bit times.

The 16 states of the counter divide each bit time into 16 ths. At the 8^{th}, 9^{th}, and 10^{th} counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of three samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

After shifting in 8-bit data, there is one more shift to do, after which the SBUF and RB8 are loaded and RI is set. However certain conditions must be met before the loading and setting of RI can be done.

1. RI must be 0 and
2. Either SM2 = 0, or the received stop bit = 1

If these conditions are met and the stop bit goes to RB8, the 8-bit data go into SBUF and RI is set. Otherwise the received frame may be lost. After the middle of the stop bit, the receiver goes back to looking for a 1-to-0 transition in RxD.

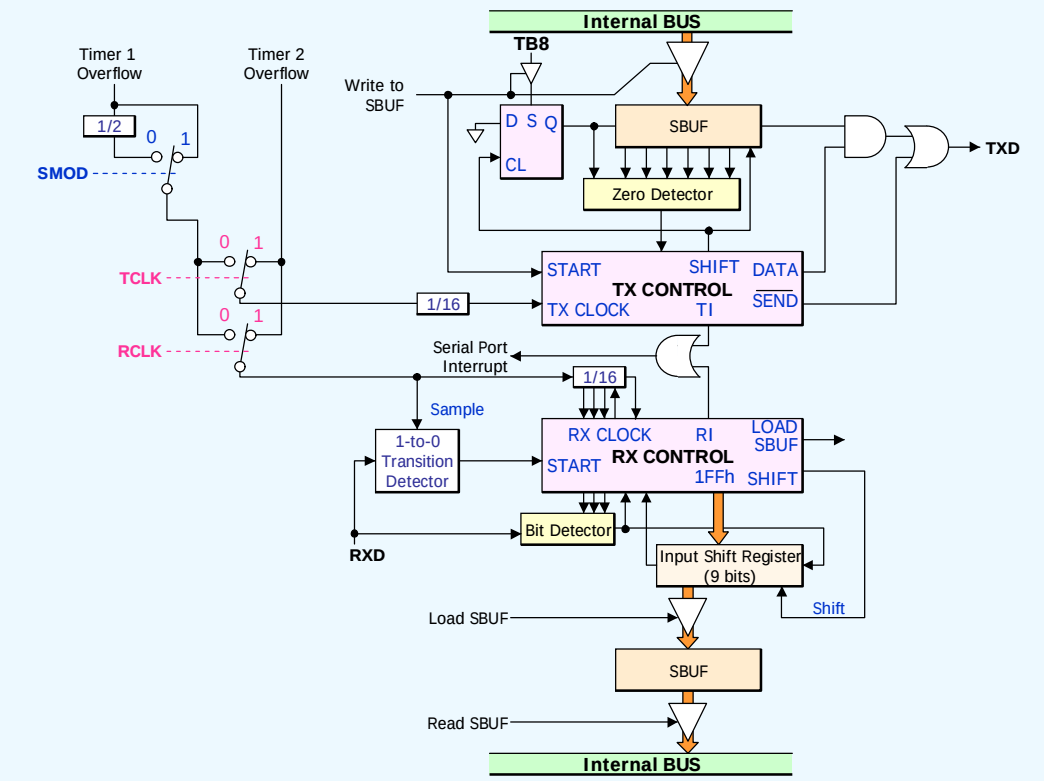


Figure 6-17 UART Mode 1

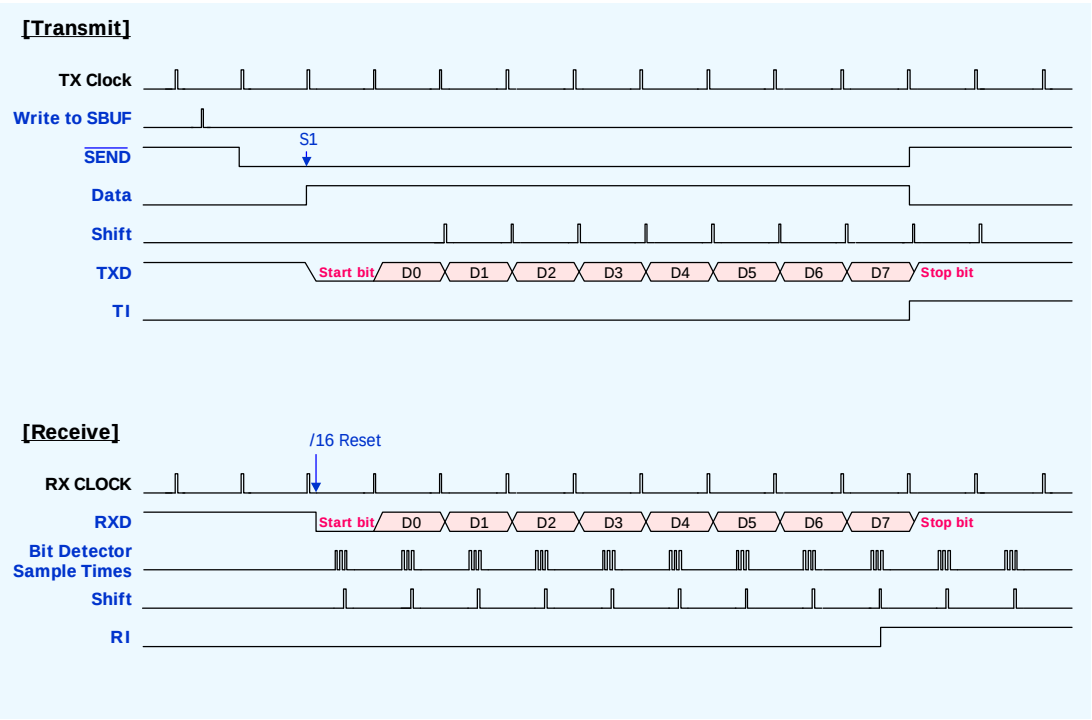


Figure 6-18 UART Mode 1 Timing

6.2.4.4 Mode 2

In Mode 2, the UART communicates asynchronously in full-duplex mode. Eleven bits are transmitted (through TXD), or received (through RXD): start bit (0), 8 data bits (LSB first), a programmable 9th bit (TB8) and a stop bit (0). On transmission, the 9th data bit (TB8) can be assigned the value of 0 or 1. On reception, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to 1/32 or 1/64 of the oscillator frequency, which is determined by the SMOD1 bit (PCON.7). Transmission is initiated by any instruction that uses SBUF as a destination register. The serial data is sent to TxD at S1 state after the first roll-over of the divide-by-16 counter. The next bit is placed on TxD at S1 state after the next roll-over of the divid-by-16 counter. Thus the transmission is synchronized to the divide-by0-16 counter, and not directly to the write to SBUF signal. After transmission of all 9 bits of data, the stop bit is transmitted. The TI flag is set at the S1 state after the stop bit has been put out on TxD. This will occur at the 11th roll-over of the divide-by-16 counter after a write to SBUF. Reception is enabled only if REN is "1." Reception is initiated by detecting 1-to-0 transition of RXD. For this purpose RXD is sampled at a rate of 16 times whatever baud rate has been established. When a 1-to-0 transition of RXD is detected, the divide-by-16 counter is immediately reset. This aligns its roll-overs with the boundaries of the incoming bit times. The 16 states of the counter divide each bit time into 16 ths.

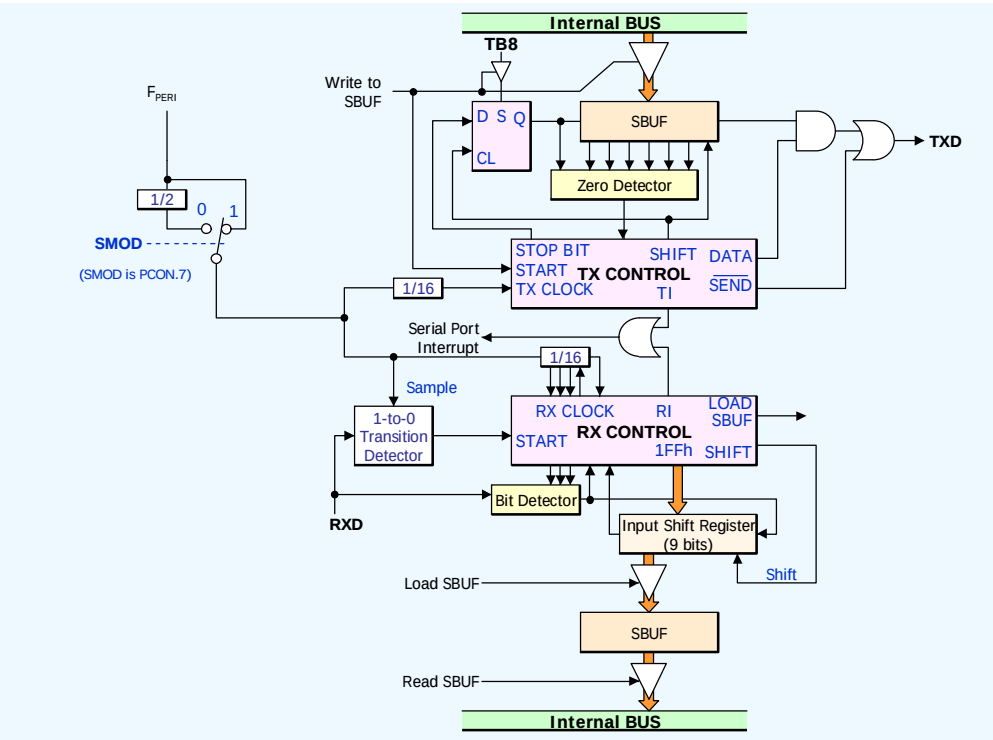


Figure 6-19 UART Mode 2

The 16 states of the counter divide each bit time into 16 ths. At the 8th, 9th, and 10th counter states of each bit time, the bit detector samples the value of RXD. The value accepted is the value that was seen in at least 2 of three samples. This is done for noise rejection. If the value accepted during the first bit time is not 0, the receive circuits are reset and the unit goes back to looking for another 1-to-0 transition. This is to provide rejection of false start bits. If the start bit proves valid, it is shifted into the input shift register, and reception of the rest of the frame will proceed.

After shifting in 9 data bits, there is one more shift to do, after which the SBUF and RB8 are loaded and RI is set. However certain conditions must be met before the loading and setting of RI can be done.

1. RI must be 0 and
2. Either SM2=0, or the received stop bit=1

If these conditions are met, then the stop bit goes to RB8, the 8 data bits go into SBUF and RI is set. Otherwise the received frame may be lost. After the middle of the stop bit, the receiver goes back to looking for a 1-to-0 transition on the RxD pin.

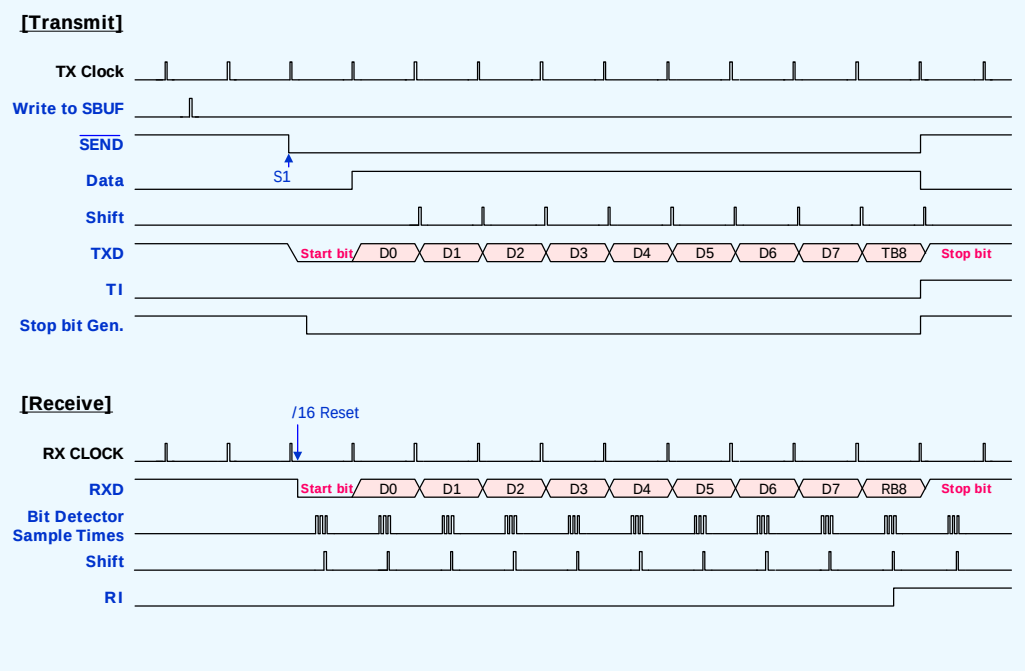


Figure 6-20 UART Mode 2 Timing

6.2.4.5 Mode 3

This mode is the same as Mode 2, except that the baud rate is programmable. Mode 3 may have a variable baud rate generated from either Timer 1 or 2 depending on the state of TCLK and RCLK. Timer 1 should also be initialized if Mode 1 and 3 are used. In all four modes, transmission is started by any instruction that uses SBUF as a destination register. Reception is initiated in Mode 0 when RI=0 and REN=1. This will shift in 8 bits on the RxD pin. Reception is initiated in the other modes by the incoming start bit if REN=1. The external device will start the communication by transmitting the start bit.

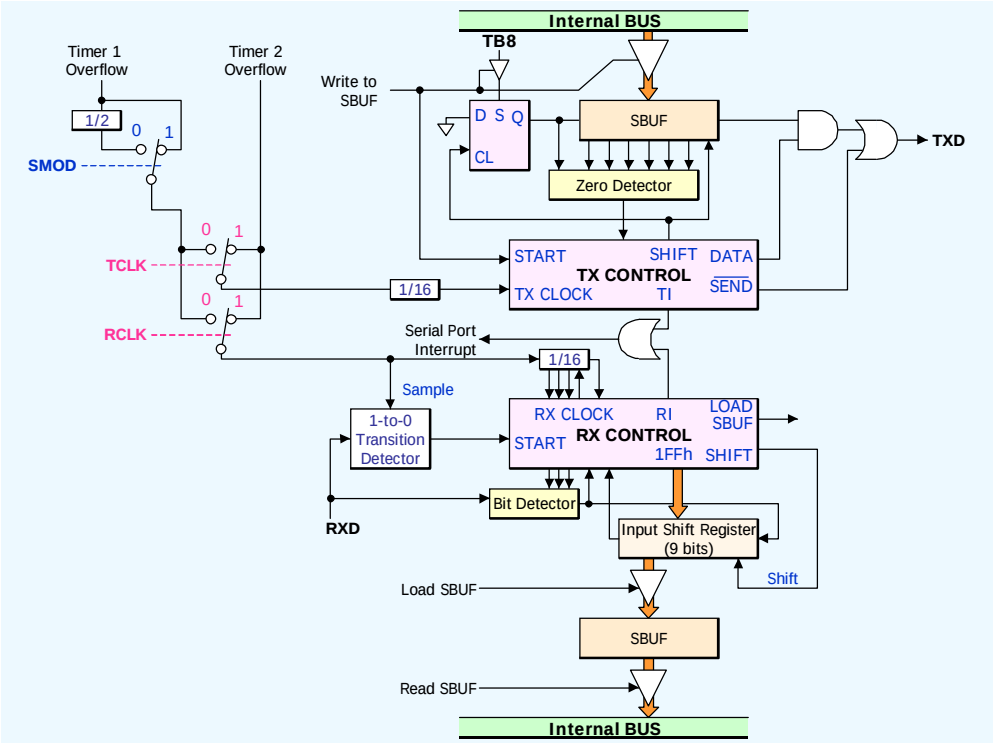


Figure 6-21 UART Mode 3

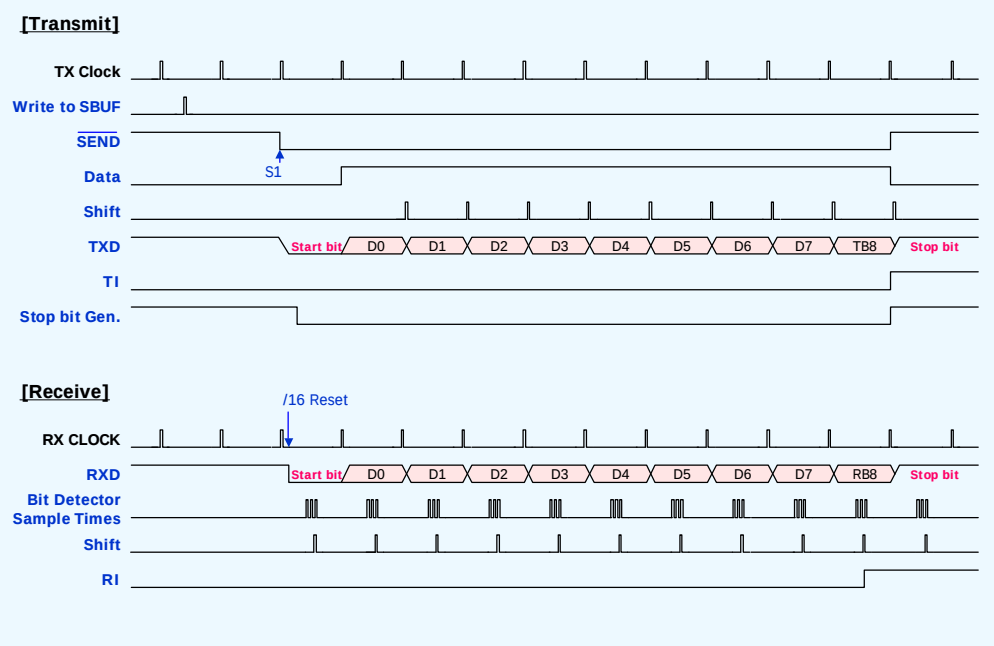


Figure 6-22 UART Mode 3 Timing

6.2.4.6 Automatic Address Recognition

Automatic address recognition makes use of the 9th data bit in Mode 2 and 3. In the MiDAS3.0 family, the RI flag is set only if the received byte corresponds to the given or broadcast address. This hardware feature eliminates the software overhead required for checking every received byte, and greatly reduces the software programmer task.

In the multiprocessor communication, the address byte is distinguished from the data byte by setting its 9th bit to 1. When the master processor wants to transmit data to a slave, first it sends the address byte of the targeted slave (or slaves). All the slave processors have their SM2 bit set high when waiting for its address byte. So they will be interrupted only by the reception of their address. By automatic address recognition feature, only the addressed slave will be interrupted. The address was recognized by hardware, not software.

The addressed slave clears the SM2 bit and waits for data bytes. If SM2=0, the slave will be interrupted every time it receive a complete frame of data. The unaddressed slaves will be still waiting for their address. If SM2 is 1, RI is set only if a valid frame is received and the received byte matches the given or broadcast address.

The master processor can selectively communicate with a group of slaves by using a given address. The address for each slave is formed with the SADDR and SADEN SFRs. The slave address is 8-bit value specified in the SADDR. The SADEN is actually a mask byte for the byte value in SADDR. If a bit in SADEN is 0, the corresponding bit in SADDR is don't care. By only the bits in SADDR whose corresponding bits in SADEN are 1, the given address is determined. So a user can address many slaves flexibly without changing the slave address in SADDR.

The following example shows how to address different slaves with a given address.

Slave 1:
SADDR 1010 0100
SADEN 1111 1010
Given 1010 0X0X

Slave 2:
SADDR 1010 0111
SADEN 1111 1001
Given 1010 0XX1

The given addresses for slave 1 and 2 have the different LSB. For slave 1: Don't care for Slave 1, 1 for Slave 2. Thus to communicate only with slave 1, the master must send 0 for the LSB of address value (1010 0000). Similarly the bit 1 is 0 for slave 1 and don't care for slave 2. Hence to communicate only with slave 2, the master has to transmit 1 for the bit 1 of address value (1010 0011). If the master wishes to communicate with both slaves simultaneously, then the LSB two bits of address must be set to '01'. The bit 3 position is a don't care for both the slaves. So the two different addresses (1010 0001 and 1010 0101) can be selected.

The master can communicate with all the slaves simultaneously with the broadcast address. This address is formed from the logical OR of the SADDR and SADEN registers with zeros defined as don't care bits. In most applications, a broadcast address is FFh. In the previous example, the broadcast address is (1111 111X) for slave 1 and (1111 1111) for slave 2.

The SADDR and SADEN are located at address A9H and B9H, respectively. By reset, these two SFRs are initialized to 00H. The given address and broadcast address are set to XXXX XXXX (i.e. all bits are don't care). As a result, the master cannot communicate with the slaves selectively.

6.2.4.7 Baud Rates

The baud rate in Mode 0 is fixed:

Mode 0 Baud rate = (Oscillator Frequency) * 1/4

The baud rate in Mode 2 depends on the value of bit SMOD1 in PCON SFR. If SMOD1 = 0 (which is the value on reset), the baud rate 1/32 the oscillator frequency. If SMOD = 1, the baud rate is 1/16 the oscillator frequency.

Mode 2 Baud Rate = $2^{SMOD1}/32$ * (Oscillator Frequency)

In the MiDAS2.0 family, the baud rates in Modes 1 and 3 can be determined by the overflow rate of Timer 1, or by that of Timer 2, or by both (one for transmit and the other for receive).

Using Timer 1 to Generate Baud Rates

When Timer 1 is used as the baud rate generator, the baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD1 as follows:

Modes 1, 3 Baud Rate = $2^{SMOD1}/32$ * (Timer 1 Overflow Rate)

The Timer 1 interrupt should be disabled in this application. The Timer itself can be configured for either “timer” or “counter” operation, and in any of its 3 running modes. In the most typical applications, it is configured for “timer” operation, in the auto-reload mode (high nibble of TMOD = 0010B). In that case, the baud rate is given by the formula

Modes 1, 3 Baud Rate = $2^{SMOD1}/32$ * (Oscillator Frequency) * $3^{T1M}/12$ * 1/[256-(TH1)]

Using Timer 2 to Generate Baud Rates

When U0T2DIS and/or U1T2DIS = 0, Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK. Note then the baud rates for transmit and receive can be simultaneously different.
The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the the Timer 2 registers to be reloaded with 16-bit value in registers RCAP2H and RCAP2L, which are preset by

software.

Now, the baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate as follows:

Modes 1,3 Baud Rate = (Timer 2 Overflow Rate) / 16

The Timer can be configured for either "timer" or "counter" operation. In the most typical applications, it is configured for "timer" operation (C/T2 = 0). "Timer" operation is a little different for Timer 2 when it's being used as a baud rate generator. Normally, as a timer it would increment every machine cycle or every 3 machine cycles (thus at 1/4 or 1/12 the oscillator frequency). As a baud rate generator, however, it increments every 2 clock cycles (thus at 1/2 the oscillator frequency). In that case the baud rate is given by the formula

Modes 1, 3 Baud Rate = 1/32 * (Oscillator frequency) / [65536 – (RCAP2H, RCAP2L)]

Where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as baud rate generator is shown in Figure 6-23. This is valid only if UART0_TCLK+UART0_RCLK =1 or UART1_TCLK+UART1_RCLK =1. If UART0_TCLK+UART0_RCLK, UART 0 uses Timer 2 as a baud rate generator. Otherwise, UART 1 uses Timer 2 as a baud rate generator.

l CKCON (8Eh) : Clock Control Register

Bit No.	7	6	5	4	3	2	1	0
	WD1	WD0	T2M	T1M	T0M	-	U1T2DIS	U0T2DIS
	R/W(1)	R/W(1)	R/W(0)	R/W(0)	R/W(0)		R/W(0)	R/W(0)

T2M: Timer 2 Time-base Selection
T2M=1, Time-base 4 clocks not 12 clocks
T1M: Timer 1 Time-base Selection
T1M=1, Time-base 4 clocks not 12 clocks
T0M: Timer 0 Time-base Selection
T0M=1, Time-base 4 clocks not 12 clocks
U1T2DIS disables RCLK/TCLK control for UART1
U0T2DIS disables RCLK/TCLK control for UART0

I T2CON (C8h) : Timer 2 Control Register

Bit No.	7	6	5	4	3	2	1	0
	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

TF2: Timer 2 Overflow Flag
EXF2: Timer 2 External Flag
RCLK: Receive Clock Flag
TCLK: Transmit Clock Flag
EXEN2: Timer 2 External Enable Flag
TR2: Timer 2 Run Enable
C/T2: Timer or Counter Selection. If C/T2=0, Timer Operation.
CP/RL2: Capture/Reload Flag.

I PCON (87h) : Power Control Register

Bit No.	7	6	5	4	3	2	1	0
	SMOD1	SMOD0	-	POF	GF1	GF0	PD	IDL
	R/W(0)	R/W(0)		R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

SMOD1: Timer 1 baud rate double in UART mode 1, 2, and 3
SMOD2: Enable SM0 access. Don't modify this bit

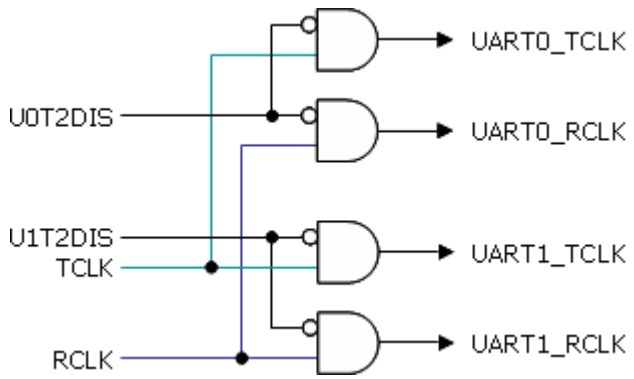


Figure 6-23 Using different baud rate generators between UART 0 and UART 1

Note that a rollover in TH2 does not set TF2, and will not generate an interrupt. Therefore, the Timer 2 interrupt does not have to be disabled when Timer 2 is in the baud rate generator mode. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H,

RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt, if desired.

It should be noted that when Timer 2 is running (TR2 = 1) in “timer” function in the baud rate generator mode, one should not try to read or write TH2 or TL2. Under these conditions the timer is being incremented every two clock cycle, and the results of a read or write may not be accurate. The RCAP registers may be read, but shouldn’t be written to, because a write might overlap a reload and cause write and/or reload errors. Turn the Timer off (clear TR2) before accessing the Timer 2 or RCAP registers, in this case.

Baud rate		UART Mode	FOSC	SMOD1	Timer 1		
T1M=0	T1M=1				C/T	Mode	Reload Value (TH1)
Max : 3 MHz	Max : 3 MHz	Mode 0	12 MHz	X	X	X	X
Max : 750 KHz	Max : 750 KHz	Mode 2	12 MHz	1	X	X	X
62.5 KHz	187.5 KHz		12 MHz	1	0	2	FFh
19.2 KHz	57.6 KHz		11.0592 MHz	1	0	2	FDh
9.6 KHz	28.8 KHz		11.0592 MHz	0	0	2	FDh
4.8 KHz	14.4 KHz		11.0592 MHz	0	0	2	FAh
2.4 KHz	7.2 KHz	Mode 1 & 3	11.0592 MHz	0	0	2	F4h
1.2 KHz	3.6 KHz		11.0592 MHz	0	0	2	E8h
137.5 Hz	412.5 Hz		11.0592 MHz	0	0	2	1Dh
110 Hz	330 Hz		6 MHz	0	0	2	72h
110 Hz	330 Hz		12 MHz	0	0	1	FEEBh

6.2.5 PCA (Programmable Counter Array)

The MiDAS3.0 family contains two PCAs (PCA0 and PCA1) as shown in Figure 6-24. The two PCAs have the completely same architecture and functions. Each PCA consists of a 16-bit timer/counter and six 16-bit compare/capture modules as shown in Figure 6-24. The registers for PCA0 are C0L, C0H, C0MOD, C0CON, C0CAPM0, C0CAPM1, C0CAPM2, C0CAPM3, C0CAPM4, and C0CAPM5; those for PCA1 are C1L, C1H, C0M1D, C1CON, C1CAPM0, C1CAPM1, C1CAPM2, C1CAPM3, C1CAPM4, and C1CAPM5. PCA1 has the corresponding registers for those of PCA0. A pair of corresponding registers has the same architecture and function.

Each PCA timer/counter serves as a common time base for the six modules and is the only timer that can service the PCA. Its clock input can be programmed to count any one of the following signals:

- Prescaled oscillator frequency: The prescale factor is 1, or 2, 4, 8, 12, 16, 32, 64, 128, 256
- Timer 0 overflow
- External input on ECI0(P0.7) or ECI1(P0.6)

Each compare/capture module can be programmed in any one of the following modes:

- Rising and/or falling edge capture
- Software timer
- High speed output
- Pulse width modulator

When the compare/capture modules are programmed in the capture mode, software timer, or high speed output mode, an interrupt can be generated when the module executes its function. All six modules plus the PCA timer overflow share one interrupt vector (more about this in the PCA Interrupt section).

From now on, “n” is used instead of 0 or 1 to denote one of two PCAs. Likewise, “m” stands for one of numbers from 0 to 5, which represents a capture/compare module in a PCA.

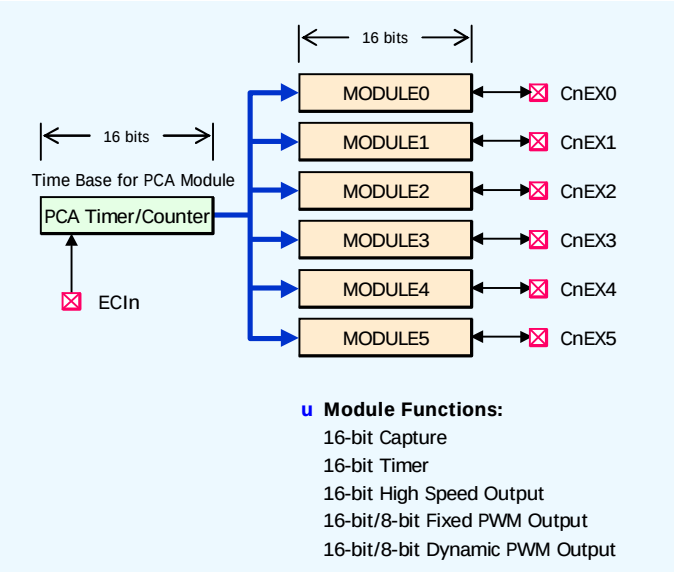


Figure 6-24 Programmable Counter Array

6.2.5.1 PCA 16-bit Timer/Counter

Each PCA has a free-running up 16-bit timer/counter consisting of registers CnH and CnL (the high and low bytes of the count value). CnH represents C0H or C1H, and CnL does C0L or C1L. These two registers can be read or written to at any time. Figure 6-25 shows a block diagram of this timer. The clock input can be selected from the following three modes:

- Prescaled oscillator frequency

  The CnL register is incremented at overflow of the prescaling counter value.
- Timer 0 overflows

  The CnL register is incremented when Timer 0 overflows.
- External input

  The CnL register is incremented when a 1-to-0 transition is detected on the ECIn (ECI0 or ECI1). The maximum input frequency in this mode is (oscillator frequency / 2).

The clock input of the registers of CnH and CnL is determined by the values of GPS3, CPS2, GPS1, and GPS0. The CnH register is incremented immediately when the CnL register overflows. The PCA timer/counter is a common time base for all six 16-bit compare/capture modules.

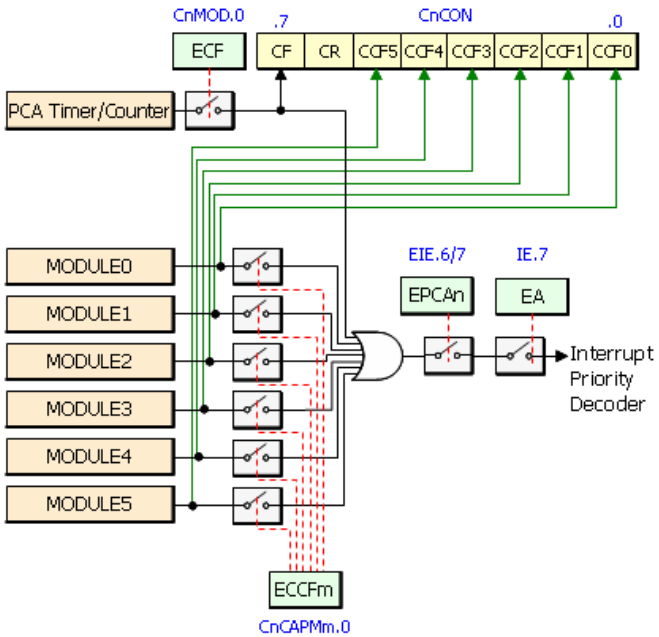


Figure 6-25 PCA Timer/Couter

The CnMOD register has four more bits. They are CIDL, PWMDYN, PWM16, and ECF. When CIDL = 1,

PCAn stops during the idle mode. By setting PWMDYN to 1, the PCA timer operates as an auto-reset counter. When PWM16 is set to 1, 16-bit PWM output is generated. Setting ECF to 1 causes an interrupt when the PCA overflow flag CF (in the CnCON register) is set to 1.

Table 6-11 PCA counter operation mode

PWMDYN	PWM16	PCA Counter Mode
0	0	16-bit free running mode. The PCA overflow flag is set when the PCA counter rolls over to 0 from 0xFFFF. This is the default operation mode.
0	1	16-bit free running mode. The PCA overflow flag is set when the PCA counter rolls over to 0 from 0xFFFF. This is the default operation mode. In this mode, module 0, 2, and 4 can generate 16-bit fixed PWM.
1	0	8-bit auto-reset mode. When CnL reaches to CnH, the PCA overflow flag is set to 1 and CnL reset to 0. At that time, the value of 8-bit PWM register is updated.
1	1	16-bit auto-reset mode. When the PCA counter reaches to the capture registers of module 0 (CnCAP0H, CnCAP0L), the PCA overflow flag is set to 1 and the PCA counter reset to 0. At that time, the value of 16-bit PWM register (the capture register of module 3 or 5) is updated. Module 2 and 4 are available for the PWM output.

l C0MOD (ADh) : PCA0 Counter Mode Register

Bit No.	7	6	5	4	3	2	1	0
	CIDL	PWMDYN	PWM16	CPS3	CPS2	CPS1	CPS0	ECF
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

CIDL: Counter idle control.
When CIDL = 0, the PCA counter continues to operate during the idle mode.
Otherwise (CIDL = 1), it stops operating during the idle mode.

PWMDYN: Dynamic PWM bit. When this bit is set to 1, the counter registers, CnL and CnH, operate in the auto-reset mode.

PWM16: 16-bit PWM generation. When this bit is set to 1, 16-bit PWM output is generated in the pulse width modulator mode.

* Refer to Table 6-11 if you want to know more about PWMDYN and PWM16.

CPS[3:0]: PCA prescaler rate (FPCA) selection.

ECF: Enable PCA counter overflow interrupt.
When ECF = 1, PCA counter overflow interrupt is enabled.
Otherwise, the interrupt is disabled.

I C1MOD (CFh) : PCA1 Counter Mode Register

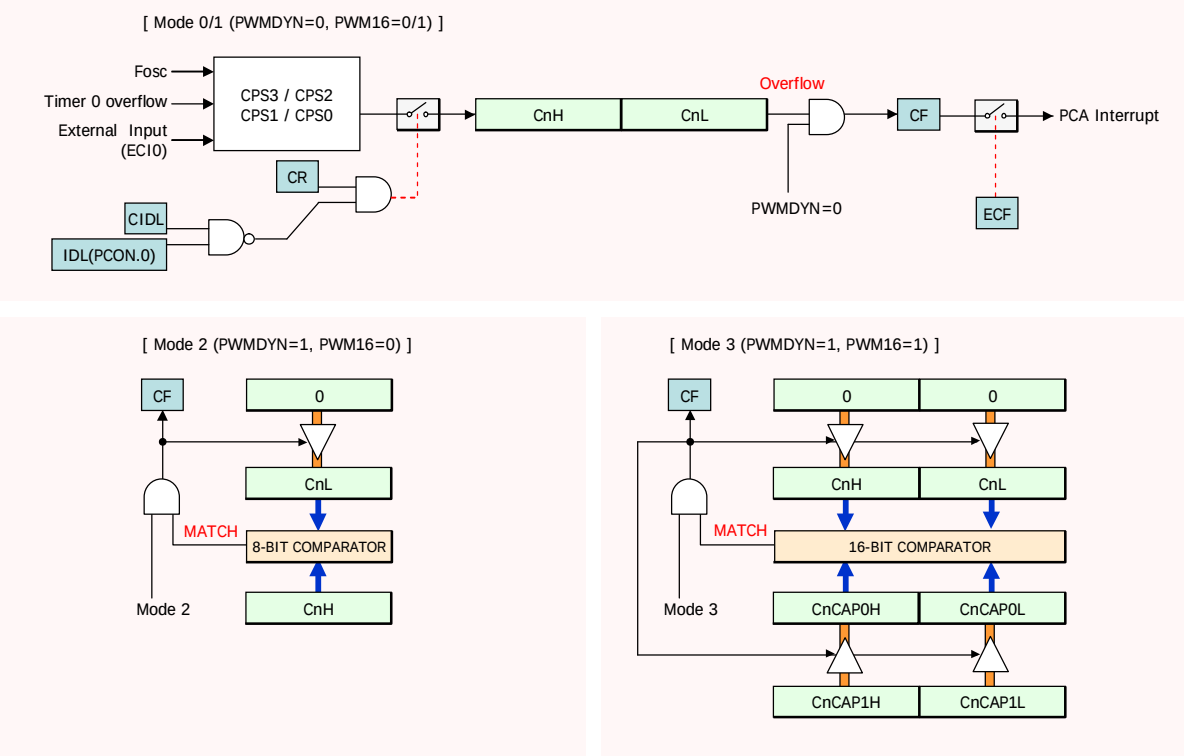


Figure 6-26 Operation of A PCA counter

If PWMDYN = 0, the PCA timer/counter operates as a free-running up 16-bit counter. If PWMDYN = 1 and PWM16 =0, the PCA counter overflow flag (CF) is set when CnL reaches to CnH. And the CnL register is cleared to 0 by the flag. Assuming that both PWMDYN and PWM16 are set to 1, the PCA counter overflow flag is set when CnH and CnL reach to CnCAP0H and CnCAP0L. And the CnH and CnL registers are cleared to 0 by the flag. At that time, the CnCAP0H and CnCAP0L registers are updated by the CnCAP1H and CnCAP1L registers.

A mode register CnMOD (C0MOD or C1MOD) contains the Count Pulse Select bits (CPS3, CPS2, CPS1, and CPS0) to specify the clock input. CnMOD is shown in Table 6-12. This register also contains the ECF bit which enables the PCA counter overflow to generate the PCA interrupt.

The CnCON register contains two more bits which are associated with the PCA timer/counter. The PCA

counter overflow flag, CF, gets set by hardware when the PCA counter overflows. The CR bit is The PCA counter run control bit. Setting CR starts the PCA counter. The counter stops by clearing CR. The other six bits in this register are the event flags for the compare/capture modules and will be discussed in the next section.

- **C0CON (ACh): PCA0 Counter Control Register**

Bit No.	7	6	5	4	3	2	1	0
	CF	CR	CCF5	CCF4	CCF3	CCF2	CCF!	CCF0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

CF: PCA counter overflow flag.
Set by hardware when the counter rolls over.
CF flags an interrupt if bit ECF in COMOD is set.
CF may be set by either hardware or software but
can only be cleared by software

CR: PCA counter run control bit.
Set by software to turn the PCA counter on.
Must be cleared by software to turn the PCA counter off.

CCF5: PCA module 5 interrupt flag.
Set by hardware when a match or capture occurs.
Must be cleared by software.

CCF4: PCA module 4 interrupt flag.

CCF3: PCA module 3 interrupt flag.

CCF2: PCA module 2 interrupt flag.

CCF1: PCA module 1 interrupt flag.

CCF0: PCA module 0 interrupt flag.

- **C1CON (CEh): PCA1 Counter Control Register**

6.2.5.2 Capture/Compare Modules

Each of the six compare/capture modules has six possible functions it can perform:

- 16-bit Capture, positive-edge triggered
- 16-bit Capture, negative-edge triggered
- 16-bit Capture, both positive and negative-edge triggered
- 16-bit Software Timer
- 16-bit High Speed Output
- 16-bit / 8-bit Pulse Width Modulator

Each module has a mode register called CnAPMm (m = 0, 1, 2, 3, 4, or 5) to select which function it will perform. The CnCAPMm register is shown in Table 6-12. Note the ECCFm bit which enables the PCA interrupt when a module's event flag is set. The event flags (CCFm) are located in the CnCON register and get set when a capture event, software timer, or high speed output event occurs for a given module.

Table 6-12 shows the combinations of bits in the CnCAPMm register that are valid and have a defined function. Invalid combinations will produce undefined results.

Each module also has a pair of 8-bit compare/capture registers (CnCAPmH and CnCAPmL) associated with it. These registers store the time when a capture event occurred or when a compare event should occur. For the PWM mode, the high byte register CnCAPmH controls the duty cycle of the waveform.

Table 6-12 CnCAPMm: PCAn Module Compare/Capture Registers

IPWMm	ECOMm	CAPPm	CAPNm	MATm	TOGm	PWMm	ECCFm	Module Function
X	0	0	0	0	0	0	0	No operation
X	X	1	0	0	0	0	X	1) 16-bit capture by a positive-edge trigger on CnEXm
X	X	0	1	0	0	0	X	1) 16-bit capture by a negative-edge trigger on CnEXm
X	X	1	1	0	0	0	X	1) 16-bit capture by any transition on CnEXm
X	1	0	0	1	0	0	X	2) 16-bit software timer
X	1	0	0	1	1	0	X	3) 16-bit high speed output
0	1	0	0	0	0	1	0	4) 5) 8-bit PWM normal

								output
1	1	0	0	0	0	1	0	4) 5) 8-bit PWM inverted output

- **C0CAPM0 (A2h): Mode Control Register of PCA0 Module0**

Bit No.	7	6	5	4	3	2	1	0
	IPWM0	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

IPWM0: Inverted PWM output.
If this bit is set, the PWM output is high when C0L ≥ C0CAPmL.
The change of this bit will take effect from the next overflow / match time of PWM.

ECOM0: Enable comparator.
ECOM0 = 1 enables the comparator function.

CAPP0: Capture positive.
CAPP0 = 1 enables positive edge capture.

CAPN0: Capture negative.
CAPN0 = 1 enables negative edge capture.

MAT0: Match.
When MAT0 = 1, a match of the PCA counter with this module's comparator/capture register causes the CCF0 bit in C0CON to be set, flagging an interrupt.

TOG0: Toggle.
When TOG0 = 1, a match of the PCA counter with this module's comparator/capture register causes the C0EX0 pin to toggle.

PWM0: Pulse width modulation mode.
PWM0 = 1 enables the C0EX0 pin to be used as a pulse width modulated output.

ECCF0: Enable CCF interrupt.
Enables compare/capture flag CCF0 in the C0CON register to generate an interrupt

- **C0CAPM1 (A3h): Mode Control Register of PCA0 Module1**
- **C0CAPM2 (A4h): Mode Control Register of PCA0 Module2**
- **C0CAPM3 (A5h): Mode Control Register of PCA0 Module3**
- **C0CAPM4 (A6h): Mode Control Register of PCA0 Module4**
- **C0CAPM5 (A7h): Mode Control Register of PCA0 Module5**
- **C1CAPM0 (E2h): Mode Control Register of PCA1 Module0**
- **C1CAPM1 (E3h): Mode Control Register of PCA1 Module1**
- **C1CAPM2 (E4h): Mode Control Register of PCA1 Module2**

- **C1CAPM3 (E5h): Mode Control Register of PCA1 Module3**
- **C1CAPM4 (E6h): Mode Control Register of PCA1 Module4**
- **C1CAPM5 (E7h): Mode Control Register of PCA1 Module5**

The next four sections describe each of the compare/capture mode modes in detail.

6.2.5.3 16-Bit Capture Mode

Both positive and negative transitions can trigger a capture with the PCA. This gives the PCA the flexibility to measure periods, pulse widths, duty cycles, and phase differences on up to six separate inputs. Setting the CAPPm and/or CAPNm in a CnCAPMm mode register select the input trigger – positive and/or negative transition – for module m. Refer to Figure 6-27.

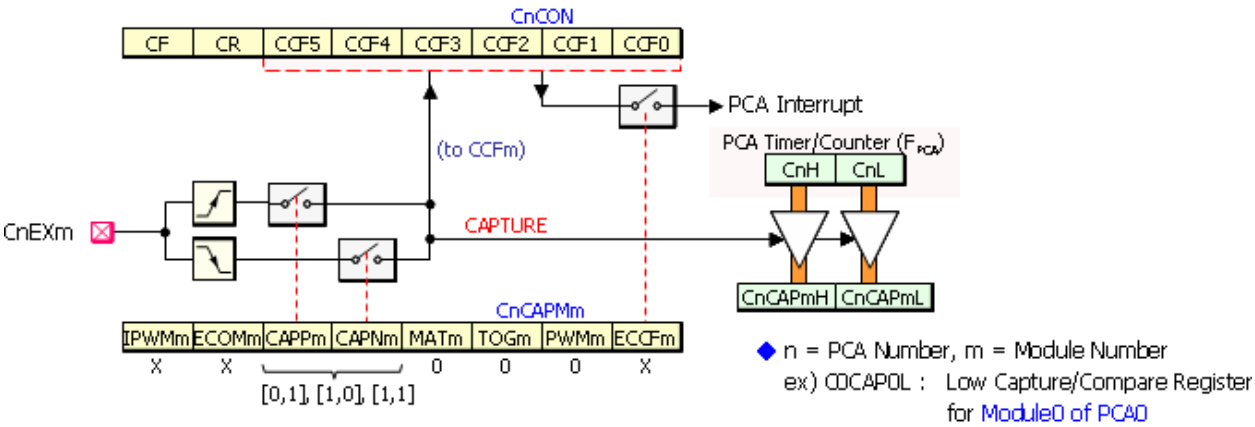


Figure 6-27 PCA 16-Bit Capture Mode

The external input pins, C0EX0 through C0EX5 and C1EX0 through C1EX5, are sampled for a transition. When a valid transition is detected (positive and/or negative edge), hardware loads the 16-bit value of the PCA timer (CnH and CnL) into the module's capture registers (CnCAPmH and CnCAPmL). The resulting value in the capture registers reflects the PCA timer value at the time a transition was detected on the CnEXm or CnEXm pin.

Upon a capture, the module's event flag (CCFm) in CnCON is set, and an interrupt is flagged if the ECCFm bit mode register CnCAPMm is set. The PCA interrupt will then be generated if it is enabled. Since the hardware does not clear an event flag when the interrupt is vectored to, the flag must be cleared in software.

In the interrupt service routine, the 16-bit capture value must be saved in RAM before the next capture event occurs. A subsequent capture on the same CnEXm pin will write over the first capture value in CnCAPmH and CnCAPmL. The PWMDYN bit has to be set to 0.

6.2.5.4 16-Bit Software Timer Mode

In the compare mode, the 16-bit value of the PCA timer is compared with a 16-bit value pre-loaded in the module's compare registers (CnCAPmH and CnCAPmL). The comparison occurs every time the PCA timer value is incremented. Setting the ECOMm bit in a mode register CnCAPMm enables the comparator function as shown in Figure 6-28.

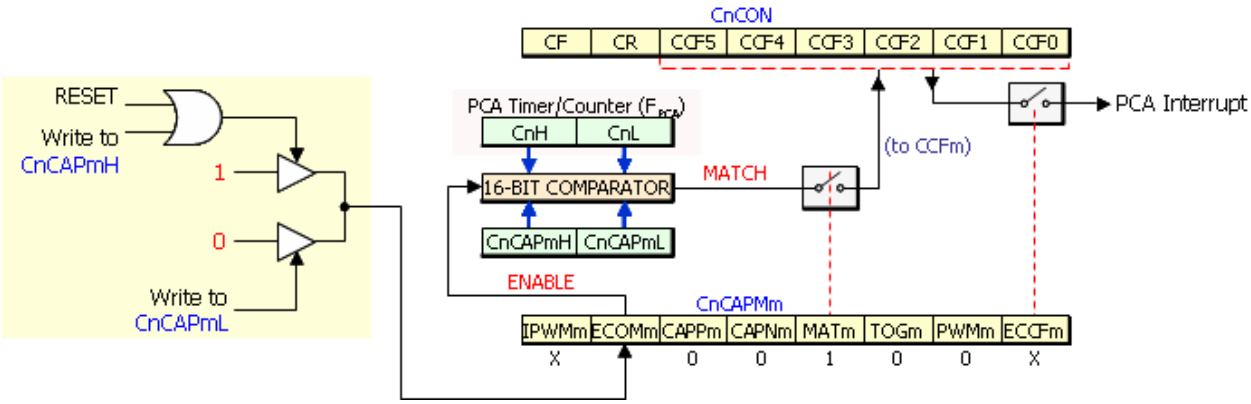


Figure 6-28 PCA 16-Bit Comparator Mode: Software Timer

For the Software Timer mode, the MATm bit also needs to be set. When a match occurs between the PCA timer and the compare registers, a math signal is generated and the module's event flag (CCFm) is set. An interrupt is then flagged if the ECCFm bit is set. The PCA interrupt is generated only if it has been properly enabled. Software must clear the event flag before the next interrupt will be flagged.

During the interrupt routine, a new 16-bit compare value can be written to the compare registers (CnCAPmH and CnCAPmL). Notice, however, that a write to CnCAPmL clears the ECOMm bit which temporarily disables the comparator function while these registers are being updated so an invalid match does not occur. A write to CnCAPmH sets ECOMm bit and re-enables the comparator. For this reason, user software should write to CnCAPmL first, then CnCAPmH. The PWMDYN bit has to be set to 0.

6.2.5.5 High Speed Output Mode

The High Speed Output (HSO) mode toggles a CnEXm pin when a match occurs between the PCA timer and a pre-loaded value in a module's compare registers. For this mode, the TOGm bit needs to be set in addition to the ECOMm and MATm bits as seen in Figure 6-28. One also has the option of flagging an interrupt when a match event occurs by setting the ECCFm bit.

The HSO mode is more accurate than toggling port pins in software because the toggle occurs before branching to an interrupt. That is, interrupt latency will not affect the accuracy of the output. If one does not change the compare registers in an interrupt routine, the next toggle will occur when the PCA timer rolls over and matches the last compare value. The PWMDYN bit has to be set to 0.

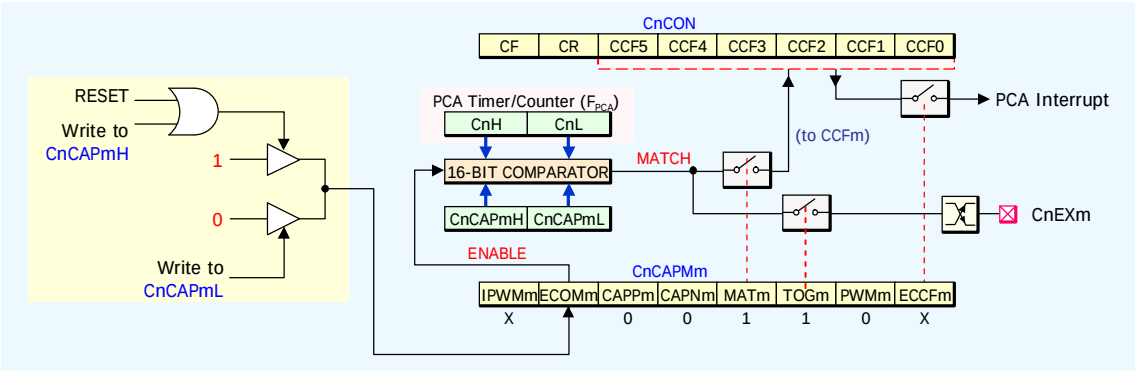


Figure 6-29 PCA 16-Bit Comparator Mode: High Speed Output

6.2.5.6 Pulse Width Modulator Mode

Any or all of the 12 PCA modules in PCA0 and PCA1 can be programmed to be a Pulse Width Modulator. The PWM output can be used to convert digital data to an analog signal by simple external circuitry. There are four modes to generate the PWM output.

6.2.5.6.1 The 8-bit Fixed PWM mode (PWM16=0, PWMDYN = 0)

CnL counts from 00H to FFH. The PCA generates 8-bit PWMs by comparing the low byte of a PCA timer (CnL) with the low byte of the module's compare registers (CnCAPmL) Refer to Figure 6-30. There are two methods to generate the PWM output according to the value of the IPWMm bit.

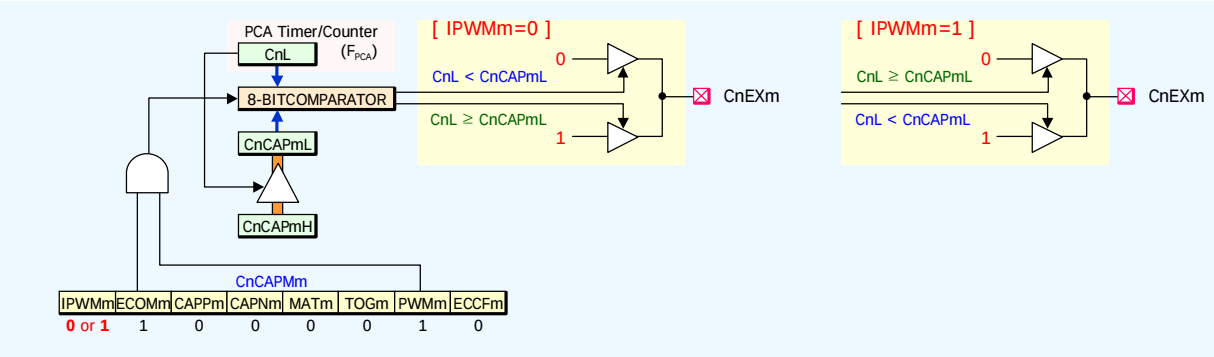


Figure 6-30 The 8-bit Fixed PWM mode

1) Normal PWM Output (IPWMm = 0)

When CnL < CnCAPmL, the output is low. When CnL >= CnCAPmL, the output is high. The value in CnCAPmL controls the duty cycle of the waveform. To change the value in CnCAPmL without output glitches, one must write to the high byte register (CnCAPmH). This value is then shifted into CnCAPmL by hardware when CnL rolls over from 0FFH to 00H, which corresponds to the next period of the output. CnCAPmH can contain any integer from 0 to 255 to vary the duty cycle from 100% to 0.4% (See Figure 6-31).

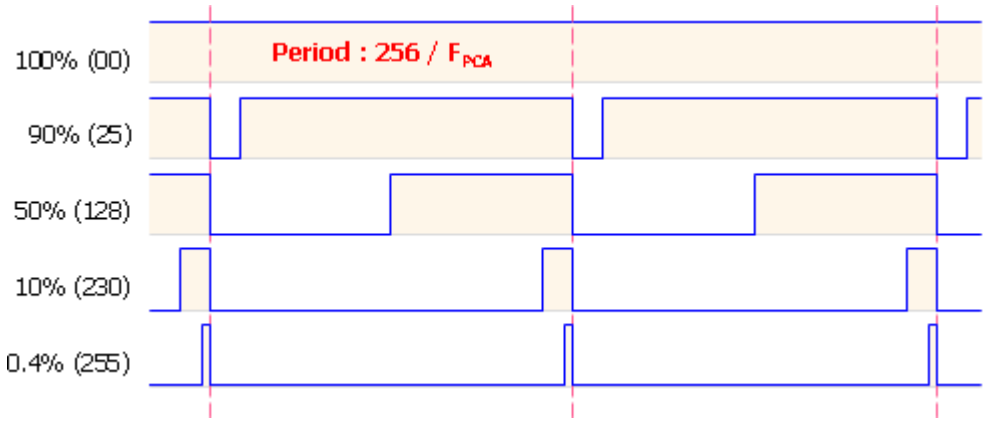


Figure 6-31 Duty Cycles of 8-bit Fixed Normal PWM Output.

2) Inverted PWM Output (IPWMm = 1)

When CnL < CnCAPmL, the output is high. When CnL >= CnCAPmL, the output is low. The value in CnCAPmL controls the duty cycle of the waveform. To change the value in CnCAPmL without output glitches, one must write to the high byte register (CnCAPmH). This value is then shifted into CnCAPmL

by hardware when CnL rolls over from 0FFH to 00H, which corresponds to the next period of the output. CnCAPmH can contain any integer from 0 to 255 to vary the duty cycle from 0% to 99.6% (See Figure 6-32).

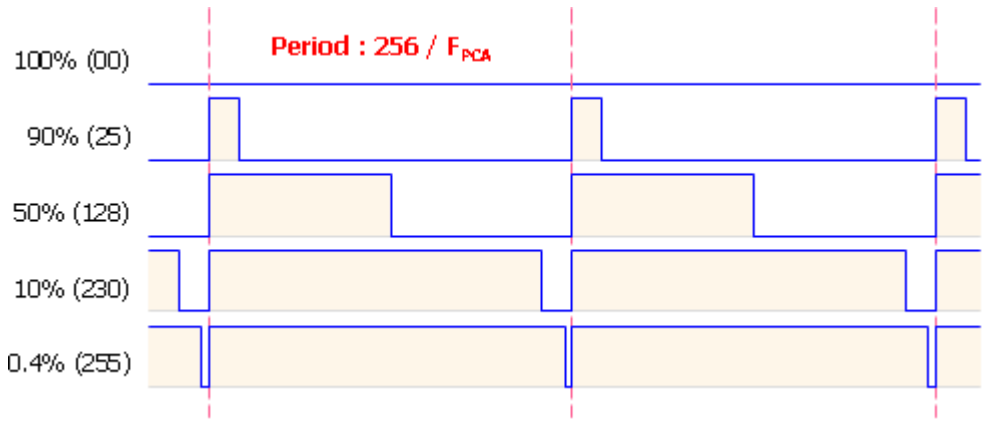


Figure 6-32 Duty Cycles of 8-bit Fixed Inverted PWM Output.

6.2.5.6.2 The 8-bit Dynamic PWM mode (PWM16 = 0, PWMDYN = 1)

At first, the counting limit value of CnL is loaded into CnH in order to adjust the period of PWM output. CnL counts from 00H to the value of CnH. After the value of CnL reaches to that of CnH, CnL will be cleared to 0 at the next input clock. The PCA generates 8-bit PWMs by comparing the low byte of the PCA timer (CnL) with the low byte of the module's compare registers (CnCAPmL) Refer to Figure 6-33. There are two methods to generate the PWM output according to the value of the IPWMm bit.

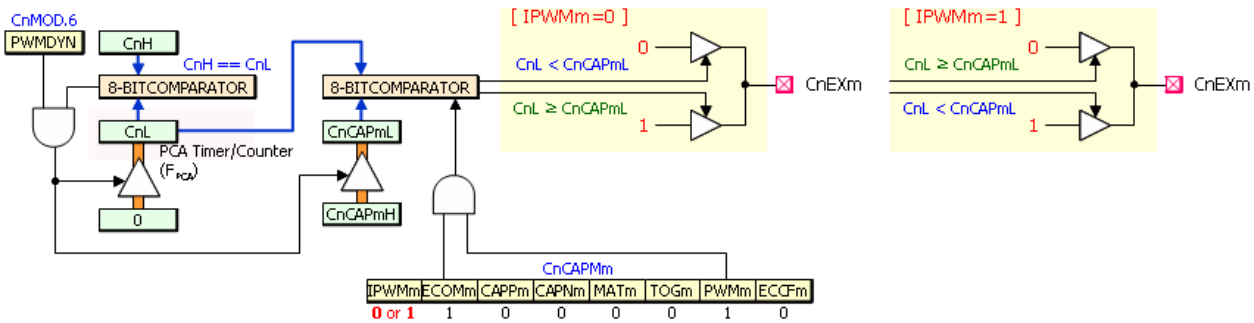


Figure 6-33 The 8-bit Dynamic PWM mode

1) Normal PWM Output (IPWMm = 0)

When CnL < CnCAPmL, the output is low. When CnL >= CnCAPmL, the output is high. The value in

CnCAPmL controls the duty cycle of the waveform. To change the value in CnCAPmL without output glitches, one must write to the high byte register (CnCAPmH). This value is then shifted into CnCAPmL by hardware when CnL is cleared to 0 from the value of CnH, which corresponds to the next period of the output. When CnH = 47, CnCAPmH can contain any integer from 0 to 47 to vary the duty cycle from 100% to 2% (See Figure 6-34).

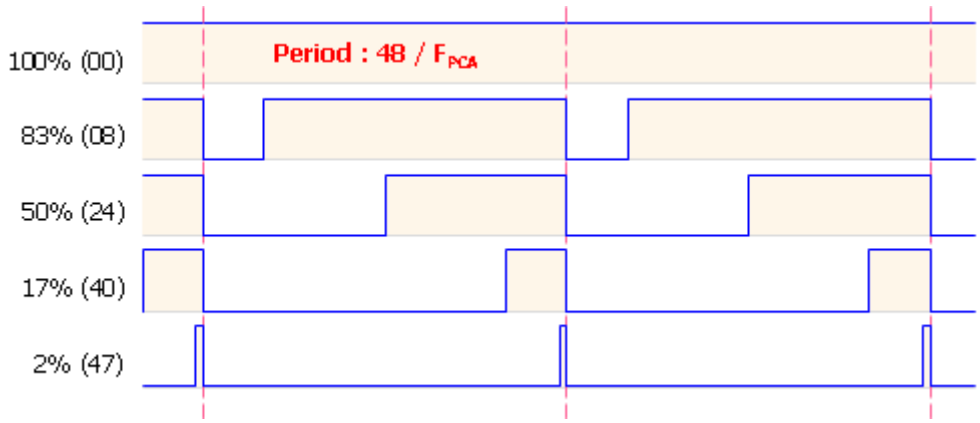


Figure 6-34 Duty Cycles of the 8-bit Dynamic Normal PWM Output.

2) Inverted PWM Output (IPWMm = 1)

When CnL < CnCAPmL, the output is high. When CnL >= CnCAPmL, the output is low. The value in CnCAPmL controls the duty cycle of the waveform. To change the value in CnCAPmL without output glitches, one must write to the high byte register (CnCAPmH). This value is then shifted into CnCAPmL by hardware when CnL is cleared to from the value of CnH, which corresponds to the next period of the output. When CnH = 47, CnCAPmH can contain any integer from 0 to 47 to vary the duty cycle from 0% to 98% (See Figure 6-35).

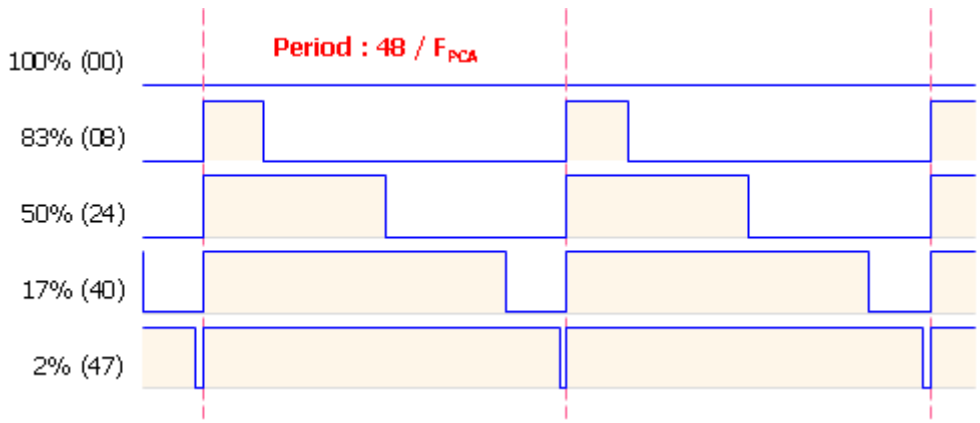


Figure 6-35 Duty Cycles of the 8-bit Dynamic Inverted PWM Output.

6.2.5.6.3 The 16-bit Fixed PWM mode (PWM16=1, PWMDYN = 0)

CnL and CnH counts from 0000H to FFFFH. The PCA generates 16-bit PWMs by comparing the value of a PCA timer (CnL and CnH) with that of the module's compare registers (CnCAPmL and CnCAPmH, m = 0, 2, or 4) Refer to Figure 6-36. There are two methods to generate the PWM output according to the value of the IPWMm bit.

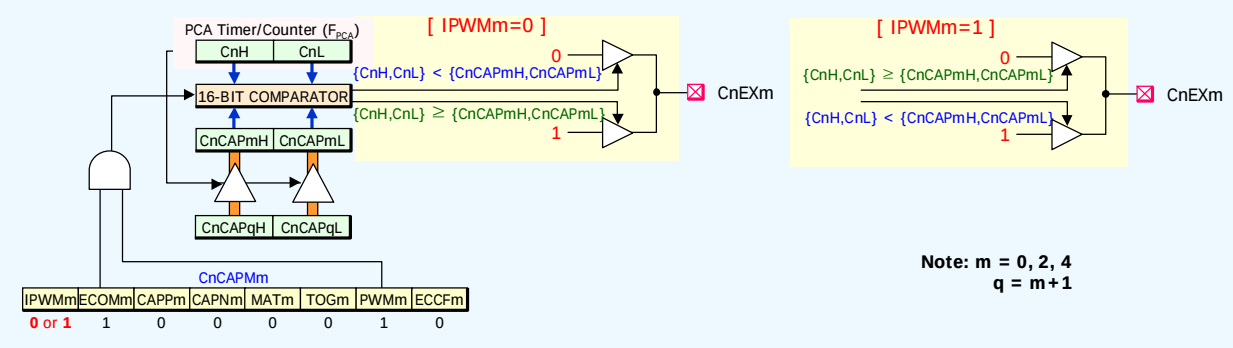


Figure 6-36 The 16-bit Fixed PWM mode

1) Normal PWM Output (IPWMm = 0)

When (CnH, CnL) < (CnCAPmH, CnCAPmL), the output is low. When (CnH, CnL) >= (CnCAPmH, CnCAPmL), the output is high. The value of (CnCAPmH, CnCAPmL) controls the duty cycle of the waveform. To change the value without output glitches, one must write it to the module p register (CnCAPpH, CnCAPpL). Here, p means m+1. This value is then shifted into (CnCAPmH, CnCAPmL) by hardware when (CnH, CnL) rolls over from FFFFH to 0000H, which corresponds to the next period of the output. Let CnH = 47(0x2F). Then, (CnCAPpH, CnCAPpL) can contain any integer from 0 to 65535 to vary the duty cycle from 100% to 0.002% (See Figure 6-31).

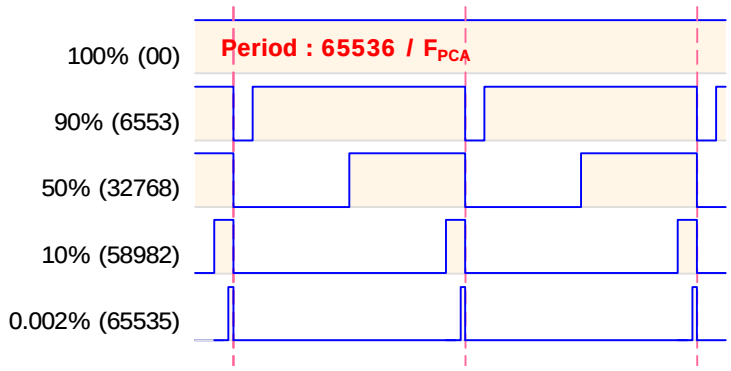


Figure 6-37 Duty Cycles of 16-bit Fixed Normal PWM Output.

2) Inverted PWM Output (IPWMm = 1)

When (CnH, CnL) < (CnCAPmH, CnCAPmL), the output is low. When (CnH, CnL) >= (CnCAPmH, CnCAPmL), the output is high. The value of (CnCAPmH, CnCAPmL) controls the duty cycle of the waveform. To change the value without output glitches, one must write it to the module p register (CnCAPpH, CnCAPpL). Here, p means m+1. This value is then shifted into (CnCAPmH, CnCAPmL) by hardware when (CnH, CnL) rolls over from FFFFH to 0000H, which corresponds to the next period of the output. Let CnH = 47(0x2F). Then, (CnCAPpH, CnCAPpL) can contain any integer from 0 to 65535 to vary the duty cycle from 0.002% to 99.998% (See Figure 6-32).

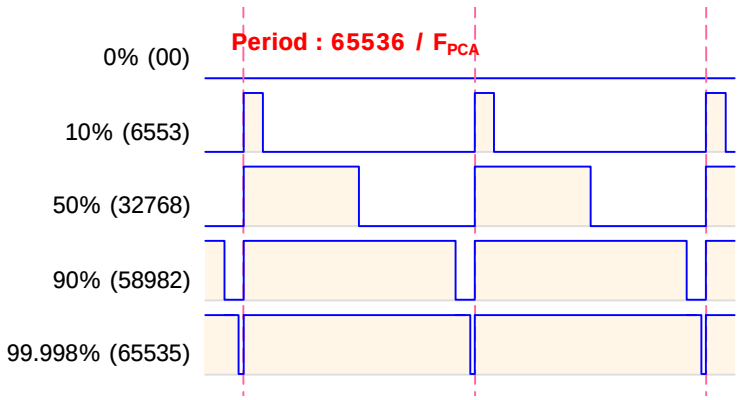


Figure 6-38 Duty Cycles of 16-bit Fixed Inverted PWM Output.

6.2.5.6.4 The 16-bit Dynamic PWM mode (PWM16 = 1, PWMDYN = 1)

At first, the counting limit value of (CnH, CnL) is loaded into (CnCAP0H, CnCAP0L) in order to adjust the

period of PWM output. (CnH, CnL) counts from 0000H to the value of (CnCAP0H, CnCAP0L). After the value of (CnH, CnL) reaches to that of (CnCAP0H, CnCAP0L), (CnH, CnL) will be cleared to 0 at the next input clock. At that time, the value of (CnCAP0H, CnCAP0L) is updated with that of the value of (CnCAP1H, CnCAP1L). The PCA generates 16-bit PWMs by comparing the value of the PCA counter with that of the module 0 registers (CnCAP0H, CnCAP0L) Refer to Figure 6-33. There are two methods to generate the PWM output according to the value of the IPWMm bit.

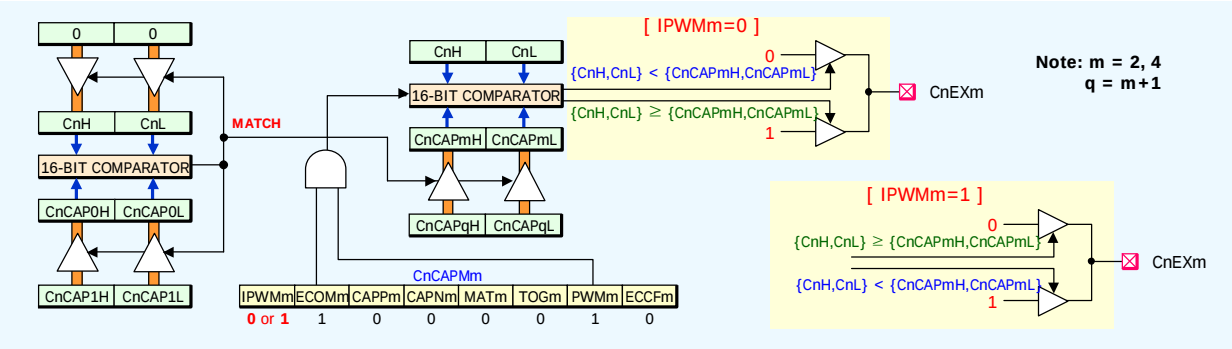


Figure 6-39 The 16-bit Dynamic PWM mode

1) Normal PWM Output (IPWMm = 0)

When (CnH, CnL) < (CnCAPmH, CnCAPmL), the output is low. When (CnH, CnL) >= (CnCAPmH, CnCAPmL), the output is high. The value of (CnCAPmH, CnCAPmL) controls the duty cycle of the waveform. To change the value without output glitches, one must write it to the module p register (CnCAPpH, CnCAPpL). Here, p means (m+1). This value is then shifted into (CnCAPmH, CnCAPmL) by hardware when (CnH, CnL) is cleared to 0, which corresponds to the next period of the output. Let CnCAP0H = CnCAP1H = 4, CnCAP0L = CnCAP1L = 0. Then, (CnCAPmH, CnCAPmL) can contain any integer from 0 to 1023 to vary the duty cycle from 100% to 0.002% (See Figure 6-34).

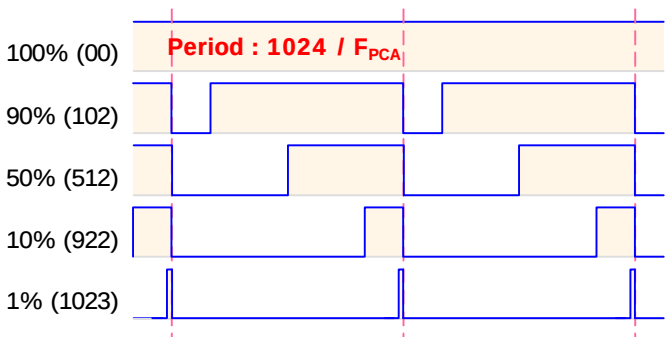


Figure 6-40 Duty Cycles of the 16-bit Dynamic Normal PWM Output.

2) Inverted PWM Output (IPWMm = 1)

When (CnH, CnL) < (CnCAPmH, CnCAPmL), the output is high. When (CnH, CnL) >= (CnCAPmH, CnCAPmL), the output is low. The value of (CnCAPmH, CnCAPmL) controls the duty cycle of the waveform. To change the value without output glitches, one must write it to the module p register (CnCAPpH, CnCAPpL). Here, p means (m+1). This value is then shifted into (CnCAPmH, CnCAPmL) by hardware when (CnH, CnL) is cleared to 0, which corresponds to the next period of the output. Let CnCAP0H = CnCAP1H = 4, CnCAP0L = CnCAP1L = 0. Then, (CnCAPmH, CnCAPmL) can contain any integer from 0 to 1023 to vary the duty cycle from 0.1% to 100% (See Figure 6-35).

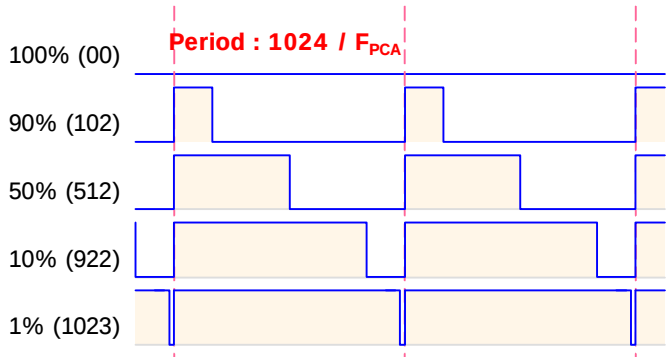


Figure 6-41 Duty Cycles of the 16-bit Dynamic Inverted PWM Output.

6.2.6 A/D Converter (Analog to Digital Converter)

The 10-bit A/D converter converts analog input voltages to 10-bit digital values with successive approximation logic. The analog input voltages can come in through eight input channels. The analog input voltage must be in the range between the V_{DD} and V_{SS}.

The A/D converter consists of the following components:

- One analog comparator with successive approximation logic
- One D/A converter logic
- One A/D converter control register (ADCON)
- One A/D converter input channel enable register (ADCHENB0, ADCHENB1, ADCHENB2, ADCHENB3)
- One A/D converter clock and mux selection register (ADCSEL)

- Thirty two multiplexed analog data input pins (ADC0.0–ADC0.7, ADC1.0–ADC1.7, ADC2.0–ADC2.7, ADC3.0–ADC3.7)
- One 10-bit A/D conversion result data buffer register (ADCR[7:0] and ADCON[1:0])

- **ADCENB0 (ECh) : ADC Channel Enable Bar Register (P0 Port)**

Bit No.	7	6	5	4	3	2	1	0
	ADCENB0.7	ADCENB0.6	ADCENB0.5	ADCENB0.4	ADCENB0.3	ADCENB0.2	ADCENB0.1	ADCENB0.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = ADC0 channel ON / 1 = ADC0 channel OFF (Default)

- **ADCENB1 (EDh) : ADC Channel Enable Bar Register (P1 Port)**

Bit No.	7	6	5	4	3	2	1	0
	ADCENB1.7	ADCENB1.6	ADCENB1.5	ADCENB1.4	ADCENB1.3	ADCENB1.2	ADCENB1.1	ADCENB1.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = ADC1 channel ON / 1 = ADC1 channel OFF (Default)

- **ADCENB2 (EEh) : ADC Channel Enable Bar Register (P2 Port)**

Bit No.	7	6	5	4	3	2	1	0
	ADCENB2.7	ADCENB2.6	ADCENB2.5	ADCENB2.4	ADCENB2.3	ADCENB2.2	ADCENB2.1	ADCENB2.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = ADC2 channel ON / 1 = ADC2 channel OFF (Default)

- **ADCENB3 (EFh) : ADC Channel Enable Bar Register (P3 Port)**

Bit No.	7	6	5	4	3	2	1	0
	ADCENB3.7	ADCENB3.6	ADCENB3.5	ADCENB3.4	ADCENB3.3	ADCENB3.2	ADCENB3.1	ADCENB3.0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

0 = ADC3 channel ON / 1 = ADC3 channel OFF (Default)

ADCR (86h) : ADC Result High Register

Bit No.	7	6	5	4	3	2	1	0
	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADCON (84h) : ADC Control & ADC Result Low Register

Bit No.	7	6	5	4	3	2	1	0
	AD_EN	AD_REQ	AD_END	ADCF	-	-	SAR1	SAR0
	R/W(0)	R/W(0)	R(1)	R/W(0)			R/W(0)	R/W(0)

AD_EN: ADC Enable

AD_REQ: ADC Start. This bit is cleared by hardware when AD_END drop to 1 from 0.

AD_END: Current ADC Status. 0 = ADC is running now.
Software must check the ADCF instead of AD_END.

ADCF: ADC Interrupt Flag. This bit must be cleared by software.

SAR[1:0]: Low Bits of ADC Result Value. (Total 10 bits)

ADCSEL (85h) : ADC Clock and MUX Selection Register

Bit No.	7	6	5	4	3	2	1	0
	ADIV2	ADIV1	ADIV0	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

ADIV[2:0]: ADC clock selection.
[000]: FSYS / 2. [001]: FSYS / 4.
[010]: FSYS / 8. [011]: FSYS / 16.
[100]: FSYS / 32. [101]: FSYS / 64.
[110]: FSYS / 128. [111]: FSYS / 256.

ADCS[4:0]: ADC channel selection
[00000]: ADC0.0 channel selection.
[00001]: ADC0.1 channel selection.
[00010]: ADC0.2 channel selection.
[00011]: ADC0.3 channel selection.
[00100]: ADC0.4 channel selection.
[00101]: ADC0.5 channel selection.
[00110]: ADC0.6 channel selection.
[00111]: ADC0.7 channel selection.
[01000]: ADC1.0 channel selection.
……

[10111]: ADC2.7 channel selection.
[11000]: ADC3.0 channel selection.
[11001]: ADC3.1 channel selection.
[11010]: ADC3.2 channel selection.
[11011]: ADC3.3 channel selection.
[11100]: ADC3.4 channel selection.
[11101]: ADC3.5 channel selection.
[11110]: ADC3.6 channel selection.
[11111]: ADC3.7 channel selection.

An analog-to-digital conversion is started as follows:

- Write the channel selection data to the ADCSEL register in order to select one analog input channel among thirty two analog pins (ADCXn, X= 0 ~ 3, n = 0 ~ 7).
- Set the corresponding bit of the ADCHENBX register in order to enable the selected analog input channel.
- Set AD_EN bit and AD_REQ bit in ADCON register.

At first, A/D converter logic sets the successive approximation register to 000h. This register will be updated automatically for every conversion step. The successive approximation block performs 10-bit conversions for one input channel at a time. You can select different channels by changing the analog input enable value in ADCHENBX[7:0] and the channel selection value in (ADCSEL[4:0]).

To start the A/D conversion, one should set AD_EN bit (ADCON[7]) and AD_REQ bit (ADCON[6]) to 1. When a conversion is completed, the end-of-conversion (AD_END and ADCF) bits are automatically set to 1. Also AD_REQ bit is cleared to 0 by hardware. AD_END bit is ADCON[5]. ADCF (ADCON[4]) is used for interrupt flag. Conversion result is sent to the ADCR[7:0] and ADCON[1:0] register. Then the A/D converter enters an idle state.

Remember to read the value of ADCR and ADCON before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

NOTE: Because the A/D converter does not contain sample-and-hold circuitry, it is important that any fluctuations in the analog voltage of the input pins should be kept in the absolute minimum range during a conversion procedure. Any change in the input voltage, perhaps due to circuit noise, will invalidate the result.

The A/D converter input pins are alternately used for digital input. For the A/D conversion the input pins must be configured as analog input.

6.2.6.1 INTERNAL REFERENCE VOLTAGE LEVELS

In the A/D converter block diagram

(

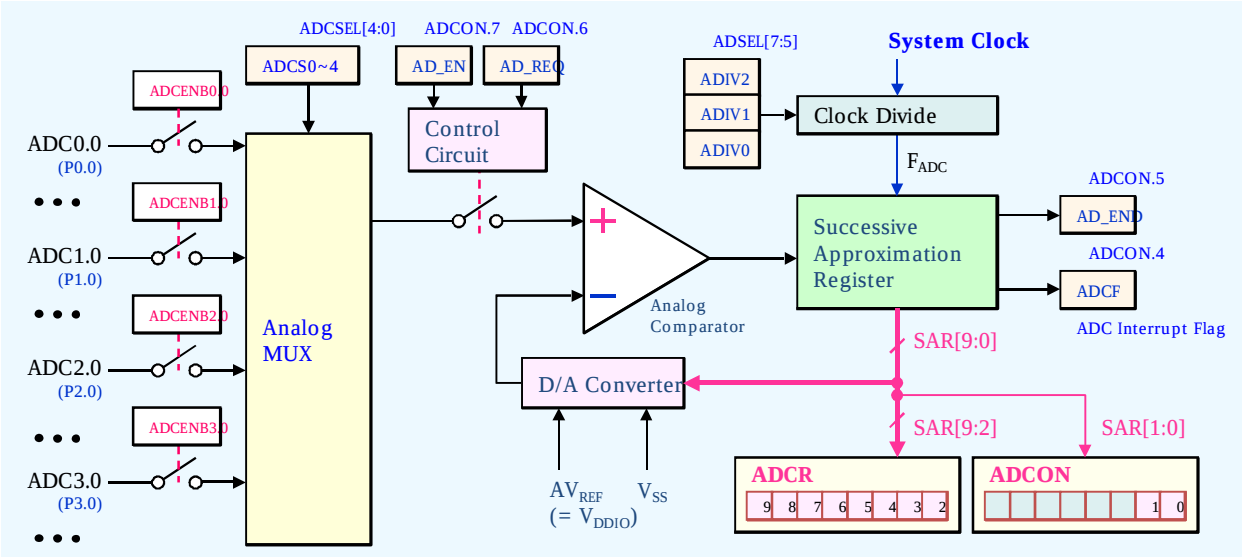


Figure 6-42), the analog input voltage is compared to the reference voltage. The analog input voltage must be within the range from VSS to VDD.

The resistor tree generated different reference internally during the analog-to-digital conversion process for each conversion step. The reference voltage level for the first bit conversion is always 1/2 VDD.

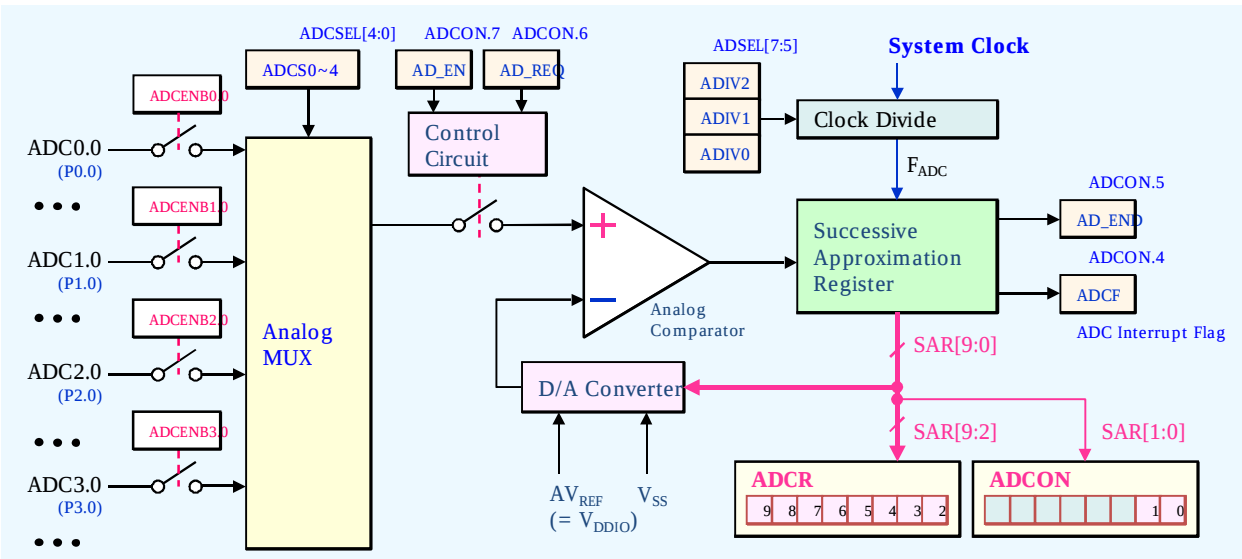


Figure 6-42 A/D Converter block Diagram

6.2.6.2 CONVERSION TIME

The A/D conversion process requires 96 A/D conversion clock cycles.

- Setup time: 8 cycles
- Conversion time: 10bit x 8 cycles = 80 cycles
- Hold time: 8 cycles

Each conversion time for each system clock frequency is presented in

Table 6-13.

Table 6-13. Example of conversion time vs. clock frequency

System Clock (F_{PERI})	Divide (ADCSEL[7:5] = 0)	F_{ADC}	T ($1/F_{ADC}$)	1 sample conversion time
20MHz@3V	F_{PERI} / 2	10MHz	100ns	9.6 us
10MHz@3V	F_{PERI} / 2	5MHz	200ns	19.2 us
10MHz@3V	F_{PERI} / 2	5MHz	200ns	19.2 us
5MHz@3V	F_{PERI} / 2	2.5MHz	400ns	38.4 us

6.2.6.3 INTERNAL A/D CONVERSION PROCEDURE

The sequences execute ADC conversion as follows:

1. Analog input voltage must be in the voltage range between V_{SS} and V_{DD}.
2. Enable an analog input channel by setting the corresponding bit of ADCHENBX[7:0] to 0.
3. Select the analog input pin connected to the enabled channel by setting ADCSEL[4:0] to the appropriate value.

4. Enable A/D conversion by setting AD_EN to 1.
5. Start A/D conversion by setting AD_REQ to 1.
6. When the conversion is completed, AD_END and ADCF will be set to 1.
7. Check AD_END or ADCF flag to see if the conversion is finished or not.
8. The converted digital value is loaded to the A/D conversion result register, ADCR[7:0] and ADCON[1:0]. Then the ADC enters an idle state.
9. The A/D conversion result can now be read from the ADCR and ADCON register.

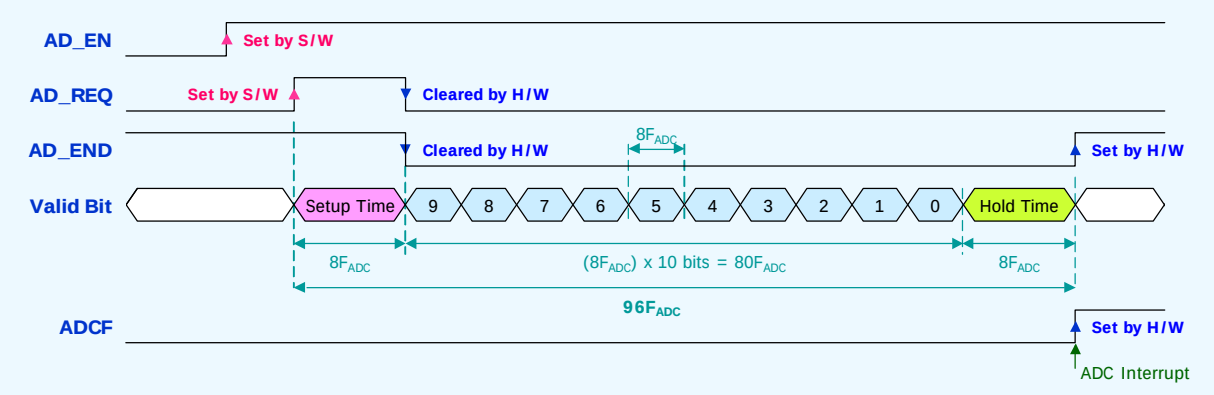


Figure 6-43 A/D Converter Timing Diagram

6.2.7 I2C Slave Logic

The I2C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the bus are:

- Bidirectional data transfer between masters and slaves
- Multimaster bus (no centeral master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- The I2C bus may be used for test and diagnostic purposes

In order to enable the I2C slave bus, the output latches of P3.4 and P3.5 must be set to logic 1. Their output type must be quasi-bidirectional and their pull-ups switched on.

The MiDAS3.0 on-chip I2C Slave logic provides a serial interface that meets the I2C bus specification and

supports 1.2MHz communication (100MHz internal clock) and 7-bit Slave address. The I2C logic handles bytes transfer autonomously. It also keeps track of serial transfers, and the control registers (IT, UINDX) reflect the status of the I2C logic and bus. The I2C logic has 16 byte TX/RX independent FIFO buffers. It uses only device address as protocol. Device address mismatch, RX buffer overflow, or TX buffer emptiness irrelevant to multi-byte reading cause 'NO ACK' state. The I2C logic supports the state.

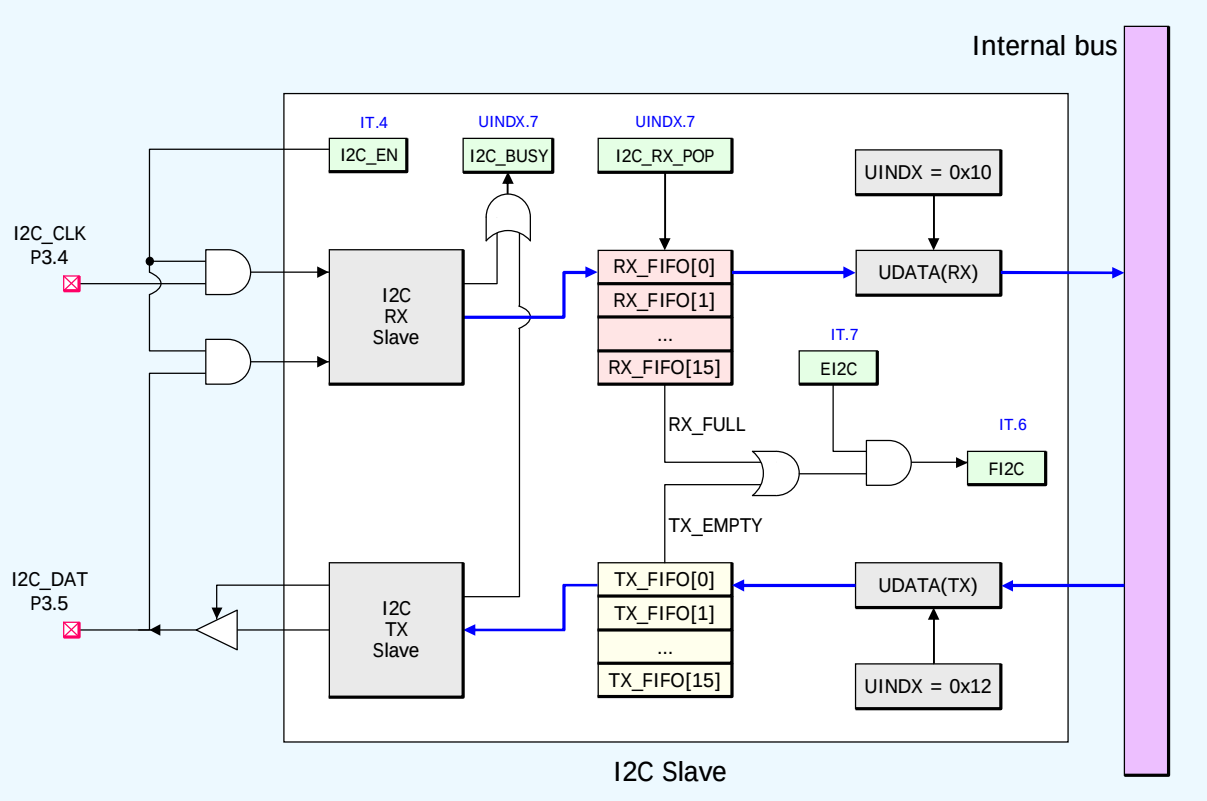


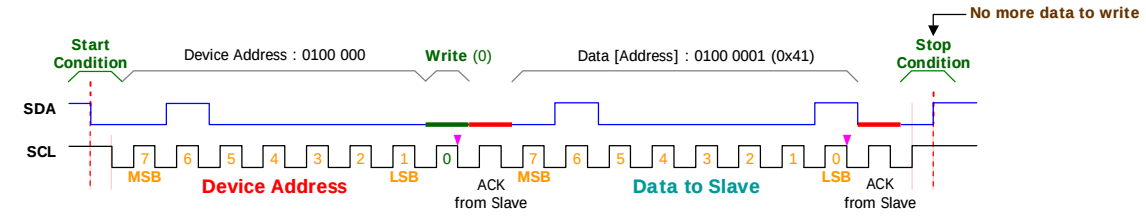
Figure 6-44 I2C Slave Logic Architecture

The CPU interfaces to the I2C Slave logic via the following three function registers: IT (Interrupt Type Selection Register), UINDX (Wakeup/I2C Index Register), and UDATA (Wakeup/I2C Data Register).

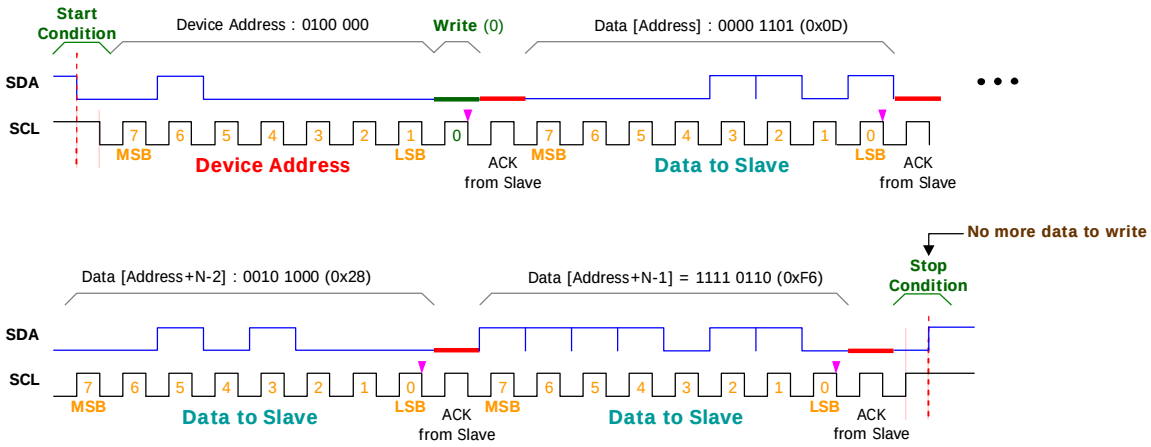
Two types of data transfers are possible on the I^2C bus:

1. Figure 6-45: Slave receiver. The first byte transmitted by the master is the slave address. The data direction bit is logic 0. Next follow a number of data bytes. The slave returns an acknowledge bit after each received byte.
2. Figure 6-46: Data transfer from a slave transmitter to a master receiver. The first byte (the slave

address) is transmitted by the master. The slave then returns an acknowledge bit. Next follow the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “no more data to read” is returned.

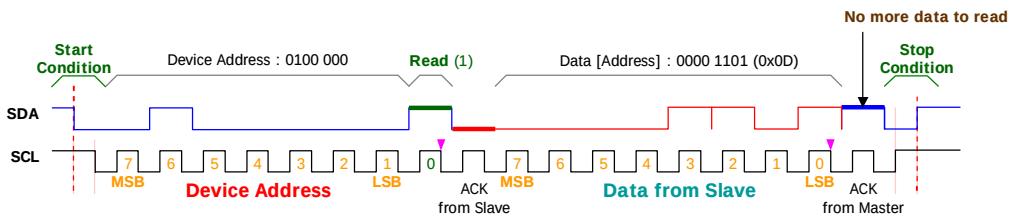


(a) One byte receiving

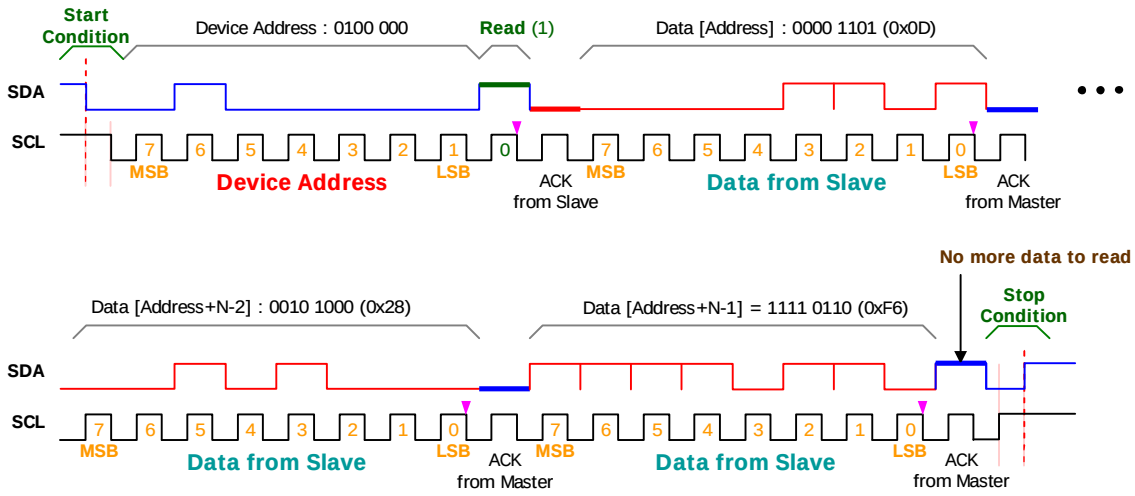


(b) Multi-byte receiving

Figure 6-45 Slave receiver



(a) One byte transmitting



(b) Multi-byte transmitting

Figure 6-46 Slave Transmitter

The master device generates all of the serial clock pulses and the Start, and Stop conditions. A transfer is ended with a Stop condition.

6.2.7.1 Mode of Operation

The on-chip I2C Slave logic may operate in the following two modes:

1. Slave Receiver Mode:

   Serial data and the serial clock are received through P3.5/SDA and P3.4/SCL. After each byte is received, an acknowledge bit is transmitted. Start and Stop conditions are recognized as the beginning and the end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.

2. Slave Transmitter Mode:

   After a Start condition or a Restart condition, the first byte is received and handled as a slave. However, in this mode, the direction bit will indicate that transfer direction is reversed. Serial data is transmitted through P3.5/SDA while the serial clock is input through P3.4/SCL. Start, Restart, and

Stop conditions are recognized as the beginning and the end of a serial transfer.

The on-chip I2C module may operate as a slave. In the slave mode, the I^2C hardware looks for its own address and the general call address. If one of these addresses is detected, an interrupt is requested.

1. IT (B2h): Interrupt Type Selection Register

Bit No.	7	6	5	4	3	2	1	0
	EI2C	FI2C	PI2C	I2C_EN	IT5	IT4	IT3	IT2
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

EI2C: I2C Interrupt Enable

FI2C: I2C Interrupt Flag.

PI2C: I2C Interrupt Priority

I2C_EN: Normal I2C Enable

0: Normal I2C Disable 1: Normal I2C Enable.

2. UINDX (F9h): Wake-up / I2C INDEX Register

Bit No.	7	6	5	4	3	2	1	0
	I2C_BS	I2C_RXP	-	UINDX4	UINDX3	UINDX2	UINDX1	UINDX0
	R(0)	R/W(1)	-	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

I2C_BS: I2C Busy

0: Idle 1: Busy

I2C_RXP: I2C RX FIFO pop

0: Idle

1: Pop FIFO, and move data to UDATA SFR

(cleared automatically by hardware).

UINDX[4:0]: Wake-up Index Register

[10000]: I2C RX FIFO read indirect address

[10001]: I2C TX FIFO write indirect address

[10010]: I2C RX FIFO pointer indirect address

[10011]: I2C TX FIFO pointer indirect address

[0XXXX]: reserved

3. UDATA (FAh): Wake-up / I2C Data Register

Bit No.	7	6	5	4	3	2	1	0
	UDAT7	UDAT6	UDAT5	UDAT4	UDAT3	UDAT2	UDAT1	UDAT0
	R/W(0)	R/W(1)	R/W(1)	R/W(0)	R/W(0)	R/W(1)	R/W(1)	R/W(1)

UDATA[7:0]: I2C Data Register

6.2.8 Interrupt

The MiDAS3.0 family has 16 interrupt sources with a four or two priority level. Each interrupt source has individual priority bits, a flag, an interrupt vector and an enable bit. All interrupts can be globally enabled or disabled. The MiDAS3.0 family has 10 interrupt-related registers: TCON, IT, ITSEL, IE, EIE, IP, IPH, EIP, and WDCON.

6.2.8.1 Interrupt Sources

The two external interrupts /INT0 and /INT1 can be either edge or level triggered, depending on bits IT0 and IT1 in the TCON register. The bits IE0 and IE1 in the TCON register are the interrupt request bits. The other four external interrupts of INT2, /INT3, INT4 and /INT5 are edge or level triggered, depending on the bits ITx in the IT register.

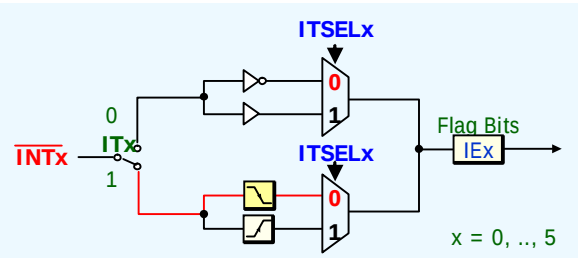


Figure 6-47 External Interrupt Detection

The six external interrupts are triggered by positive edge or high level input when the ITSELx bit in the ITSEL register is set. Otherwise, they are triggered by negative edge or low level input. The external interrupt inputs are sampled in every machine cycle. If the sampling result is high in one cycle and low in the next, a high to low transition is detected and the interrupt request flag IEx in TCON or EXIF is set. Similarly, a low-to-high transition on the positive interrupt inputs is detected and the interrupt request flag is set. Since the external interrupt inputs are sampled every machine cycle, they have to be held high or

low for at least one complete machine cycle. The IEx in TCON is cleared automatically when the service routine is called. But, IEx in EXIF must be cleared by software. For a level triggered interrupt, the interrupt input has to be kept low till the interrupt is serviced. The request bits, IE0 and IE1, are not cleared by the hardware for the level triggered interrupt when the interrupt is serviced. If the interrupt input continues to be held low even after the service routine is completed, the device may acknowledge another interrupt request from the same source. The individual interrupt flag corresponding to external interrupt 2 to 5 must be cleared manually by software.

1. ITSEL(BAh): Interrupt Polarity Selection Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	ITSEL5	ITSEL 4	ITSEL 3	ITSEL.2	ITSEL 1	ITSEL 0
			R/W(0)	R/W(1)	R/W(0)	R/W(1)	R/W(0)	R/W(0)

ITSELx: Interrupt x Polarity Selection

0 : low level or negative edge, 1 : high level or positive edge

2. IT(B2h): Interrupt Type Selection Register

Bit No.	7	6	5	4	3	2	1	0
	EI2C	FI2C	PI2C	I2C_EN	IT5	IT4	IT3	IT2
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(1)	R/W(1)	R/W(1)	R/W(1)

EI2C: I2C Interrupt Enable

FI2C: I2C Interrupt Flag

PI2C: I2C Interrupt Priority

I2C_EN: Normal I2C Enable Flag

0: Normal I2C Disable, 1: Normal I2C Enable

ITx: Interrupt x Type Selection

0: Level triggered, 1: Edge triggered

The TF0 and TF1 flags generate the Timer 0 and 1 interrupts. These flags are set by the overflow in the Timer 0 and 1. The TF0 and TF1 flags are cleared automatically by the hardware when the timer interrupt is serviced. The Timer 2 interrupt is generated by a logical OR of the TF2 and the EXF2 flags. These flags are set by overflow or capture/reload events in the Timer 2 operation. The hardware does not clear these flags when a Timer 2 interrupt is executed. Software has to distinguish the cause of the interrupt between TF2 and EXF2 and clear the appropriate flag.

The Watchdog timer can be used as a system monitor or a simple timer. In either case, when the time-out is reached, the Watchdog timer interrupt flag WDIF (WDCON.3) is set. If the enable bit EIE.4 enables the interrupt, then it will occur.

The UART 0 or the UART 1 can generate interrupts on reception or transmission. There are two interrupt sources from the UART, which are obtained by the RI and TI bits in the SCON or SCON1. These bits are not cleared automatically by the hardware, and software has to clear these bits.

ADCF flag and AD_END flag are set to 1 when A/D conversion is finished. Then the ADC interrupt will be generated if it is enabled. The ADCF flag must be cleared by software in ADC interrupt routine.
PFI flag is set when Vdd drops below V_{PFI}. The flag generates LVD interrupt if EPFI flag is set. This interrupt is not disabled by EA that is a global interrupt enable bit.

The PCA interrupts are generated by the logical OR of six event flags CCFm and the PCA timer overflow flag CF in the registers C0CON and C1CON. None of these flags are cleared by hardware when the service routine is vectored to. Normally the service routine will have to determine which bit flagged the interrupt and clear that bit in software. This allows one to define the priority of servicing each PCA module. The PCA0 interrupt is enabled by the bit EPCA0 in the EIE register. The PCA1 interrupt is enabled by the bit EPCA1 in the EIE register. In addition, the CF flag and each of the CCFm flags must also be individually enabled by bits ECF and ECCFm in registers C0MOD, C1MOD, C0CAPMm, and C0CAPMm, in order for that flag to be able to cause an interrupt.

The I2C interrupt is enabled by the bit EI2C in the register IT. The I2C interrupt flag is the FI2C bit in the IT register.
Each individual interrupt can be enabled or disabled by setting or clearing a bit in the IE register. IE also has a global enable/disable bit EA that can be cleared to disable all interrupts.

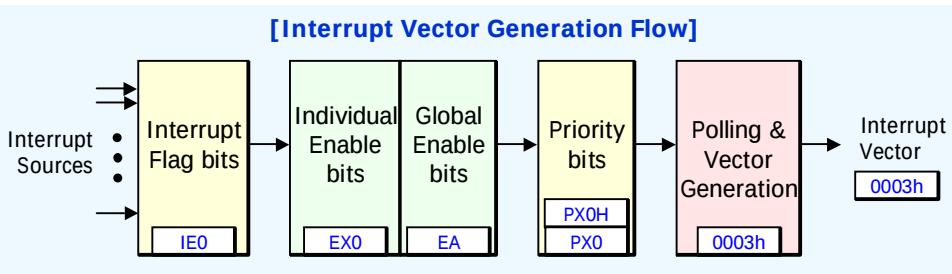


Figure 6-48 Interrupt Vector Generation Flow

6.2.8.2 Priority Level Structure

There are four or two priority levels for the interrupts. Naturally, a higher priority interrupt cannot be interrupted by a lower priority interrupt. However there exists a pre-defined hierarchy amongst the interrupts themselves. This hierarchy comes into play when the interrupt controller has to resolve simultaneous requests having the same priority level. This hierarchy is defined as shown below; the interrupts are numbered starting from the highest priority to the lowest.

Table 6-14. Priority Structure of Interrupts

Hierarchy	Sources	Vector Address	Priority Level
1	/INT0	0003h	4 levels
2	TF0	000Bh	4 levels
3	/INT1	0013h	4 levels
4	TF1	001Bh	4 levels
5	RI + TI	0023h	4 levels
6	TF2	002Bh	4 levels
7	ADC	003Bh	4 levels
8	INT2	0043h	2 levels
9	/INT3	004Bh	2 levels
10	INT4	0053h	2 levels
11	/INT5	005Bh	2 levels
12	WDT	0063h	2 levels
13	RI1 + TI1	006Bh	2 levels
14	PCA0	0073h	2 levels
15	PCA1	007Bh	2 levels

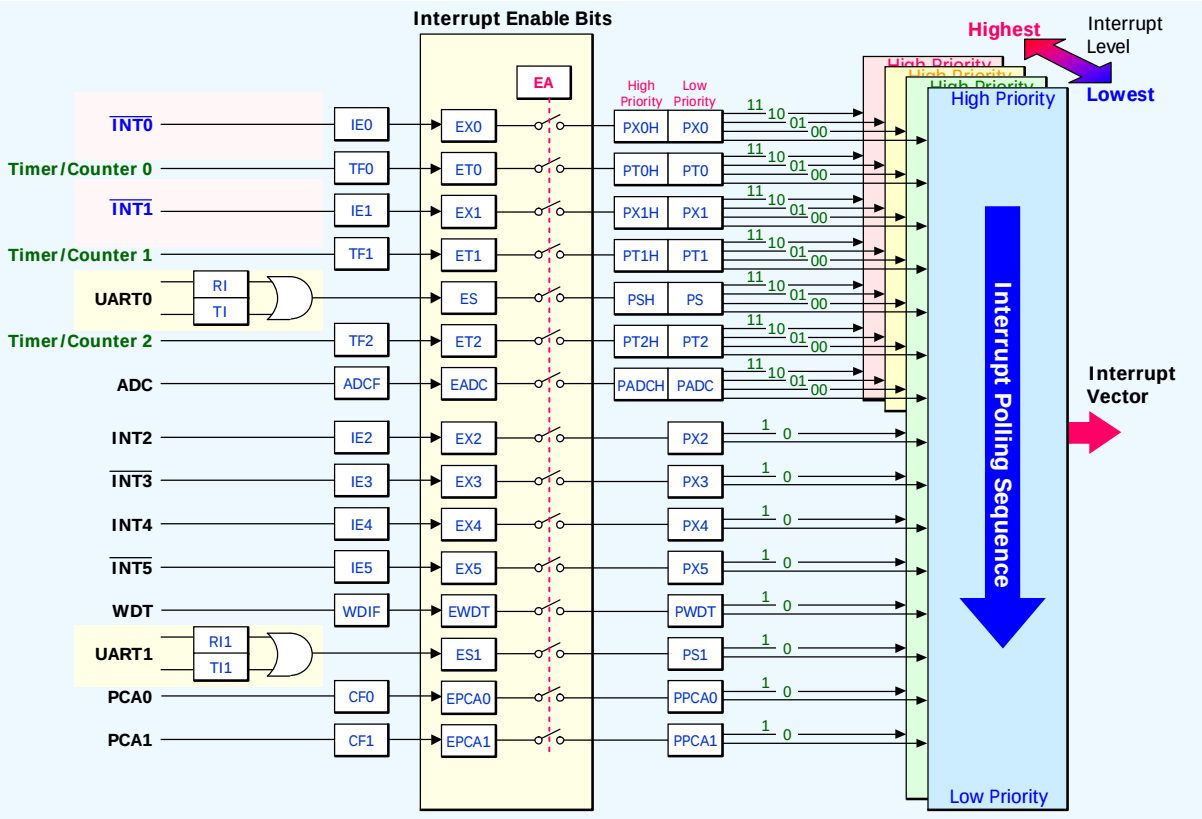


Figure 6-49 Hierarchy of Interrupt Priority

The interrupt flags are sampled at every machine cycle. In the same machine cycle, the sampled interrupts are polled and their priority is resolved. If certain conditions are met then the hardware will execute an internally generated LCALL instruction that will vector the process to the appropriate interrupt vector address. The conditions for generating the LCALL are

1. Neither an equal priority nor a higher priority interrupt is currently being serviced.
2. The current polling cycle is the last machine cycle of the instruction currently being executed.
3. The instruction in progress is neither RETI nor any write to IP, IE, EIP,EIE, IPH or EXIF.

If any of these conditions are not met, then the LCALL will not be generated. The polling cycle is repeated every machine cycle, with the interrupts sampled in the same machine cycle. Note that if an interrupt flag is active but not being responded to for one of the above conditions, and is not still active when the blocking condition is removed, the denied interrupt will not be serviced. This means that active interrupts are not remembered; every polling cycle is new.

The processor responds to a valid interrupt by executing an LCALL instruction to the appropriate service routine. This may or may not clear the flag that caused the interrupt. In case of timer interrupts, the TF0 or TF1 flags are cleared by hardware whenever the processor vectors to the appropriate timer service routine. In case of external interrupts, /INT0 and /INT1, the flags are cleared only if they are edge triggered. In case of UART interrupts, the flags are not cleared by hardware. In the case of Timer 2 interrupt, the flags are not cleared by hardware. Watchdog timer interrupt flag WDIF have to be cleared by software. The hardware LCALL behaves exactly like the software LCALL instruction. This instruction saves the Program Counter contents onto the stack, but does not save the PSW. The Program Counter is reloaded with the vector address of that interrupt which caused the LCALL.

Execution proceeds from the vector address until an RETI instruction is encounter. RETI instruction informs the processor that this interrupt routine is no longer in progress, then pops the top two bytes from the stach and reloads the Program Counter. Execution of the interrupted program continues from where it left off. Note that a simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system thinking that the service routine was still in progress.

6.2.8.3 Interrupt Response Time

The response time for each interrupt source depends on several factors, such as the nature of the interrupt and the instruction in progress. The external interrupt inputs, /INT0 to /INT5, are sampled at S3 state of every machine cycle. After that, their interrupt flags IEx will be set. The Timer 0 and 1 overflow flags are set at S3 state of the machine cycle in which overflow has occurred. These flag values are polled in the next machine cycle. If a request is active and all three conditions are met, then the hardware generated LCALL is executed. This LCALL takes four machine cycles to be completed. Thus there is a minimum of five complete machine cycles between activation of an external interrupt request and the beginning of execution of the service routine's first instruction.

A longer response time would result if any of the three conditions are not met. If a higher or equal priority level is already in progress, the additional wait time obviously depends on the nature of the currently executed service routine. If the polling cycle is not in the last machine cycle of the instruction in progress, then an additional delay is introduced. The maximum response time (if no other interrupt is in service) occurs if the MiDAS3.0 family is performing any write to IE, IP, EIE, EIP, IPH or EXIF and then executes a 4-machine cycle instruction. From the time an interrupt source is activated, the longest reaction time is 11 machine cycles. This includes 1 machine cycle to detect the interrupt, 2 machine cycles to complete the

IE, IP, EIE, EIP, IPH or EXIF access, 4 machine cycles to complete the instruction and 4 machine cycles to complete the hardware LCALL to the interrupt vector location.

Thus in a single-interrupt system, the interrupt response time will always be more than 5 machine cycles and not more than 11 machine cycles. The maximum latency of 11 machine cycles is 44 clock cycles. Note that in the standard 80C52, the maximum latency is 8 machine cycles that equals 96 clock cycles. This is more than 50% reduction in terms of clock periods.

6.2.9 Reset

The MiDAS3.0 family has several reset mechanisms. In general, reset values irrespective of the current ones are assigned to most registers, but there are several flag bits, the reset value of which depends on the reset source. These flags inform software what the reset source was. There are three mechanisms to reset the MiDAS3.0 family as shown in Figure 6-50: Power on/fail reset, External reset, and Watchdog reset.

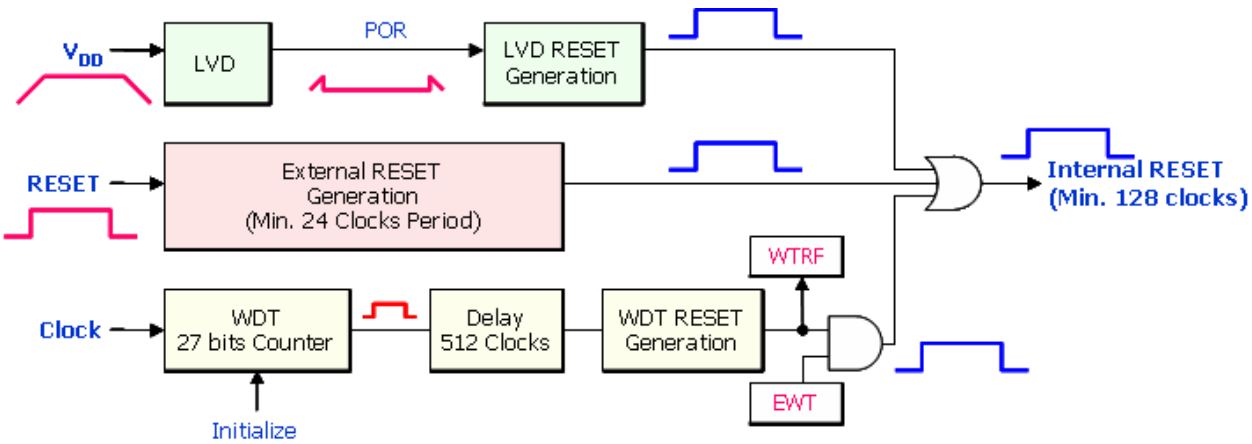


Figure 6-50 Three Reset Mechanisms

6.2.9.1 Power On/Fail Reset

The MiDAS3.0 family has a band-gap voltage reference to recognize that V_{CC} is deviated from the normal operation range. When power is turned on, its internal circuit detects the rise of V_{CC} above V_{RST}, the reset threshold (1.4V) voltage. Once V_{CC} is above this voltage, its oscillator starts oscillation. Then its internal reset circuit waits for 50 ms until the supply voltage are stabilized. Next, the MiDAS3.0 device will exit the

reset state. No external components are needed to generate a power-on-reset pulse. During power-down or during a severe power glitch, if V_{CC} falls below V_{RST}, the device will set Power-On-Reset flag again. This reset state is maintained as long as the power voltage remains below the threshold. This will occur automatically, needing no action from the user or from the software.

6.2.9.2 External Reset

The reset input is the RST pin. The external input signal is asynchronous to the internal clock. The RST pin is sampled during state S4 of every machine cycle. The RST pin must be held for at least 6 machine cycles (24 clocks) to accomplish an external reset. The reset circuitry synchronously generates the internal reset signal. Thus the reset procedure operates synchronously, while the clock is running.
The state will be maintained so as long as RST is 1. Even after the reset input is deactivated, the device will continue to be in reset state for up to three machine cycles, and then begin program execution from 0000h. There is no flag associated with the external reset condition. However, since the other two reset sources have flags, the external reset can be recognized as the default reset if those two flags are cleared.

6.2.9.3 Watchdog Timer Reset

The Watchdog timer is a free running timer with programmable time-out intervals. Software can clear the watchdog timer at any time and restart the timer. When the time-out interval is reached, an interrupt flag is set. If the Watchdog timer reset is enabled and the Watchdog timer is not cleared, Watchdog timer will generate a reset after 512 clock cycles. This reset condition is maintained by hardware for thirty clock cycles. Once the reset is removed the device will begin execution from 0000h.

6.2.10 Clock Circuits

The MiDAS3.0 family supports three clock sources as shown in Figure 6-51: 1) crystal oscillator, and 2) external oscillator module, 3) internal ring oscillator.

l PCON (87h) : Power Control Register

Bit No.	7	6	5	4	3	2	1	0
	SMOD1	SMOD0	-	POF	GF1	GF0	PD	IDL
	R/W(0)	R/W(0)	-	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)

PD: Power-down (Stop) mode enable.
IDL: IDLE Mode Enable.

l EXIF (91h) : External Interrupt Flag Register

Bit No.	7	6	5	4	3	2	1	0
	IE5	IE4	IE3	IE2	XT/RG	RGMD	RGSL	BGS
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R(1)	R/W(0)	R/W(1)

XT/RG: System clock selection.
0 = Internal Ring oscillator is selected as system clock.
1 = External clock is selected as system clock.
RGMD: Ring mode. Now system clock is Ring or XTAL.
Generally RGMD is the invert of XT/RG.
RGSL: Ring select bit when power-down wake-up.
1 = When wake-up from power-down mode in XTAL clock, use Ring oscillator as system clock during 65,536 XTAL clocks.
BGS: Band-gap select. (Default = 1)
0 = Band-gap block (LVD) will do not run in power-down mode, but function during normal mode.
It will support the significant power savings in power-down mode.
1 = Band-gap block (LVD) will run in power-down mode.

l PMR (C4h): Power Management Control Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	XTOFF	ALEOFF	WCLKE	WIOE
			-		R/W(0)	R/W(0)	R/W(0)	R/W(0)

XTOFF: Internal amplifier disable for external crystal oscillator.
1 = External crystal oscillator disable.
0 = External crystal will run (Default).
ALEOFF: 1 = ALE toggling disable.
0 = ALE toggling enable (Default).
WUCLK_EN: 1 = Wakeup CLK Enable
0 = Wakeup CLK Disable (Default)
WUIO_EN: 1 = Wakeup IO Enable
0 = Wakeup IO Disable (Default)

l STATUS (C5h): Crystal Status Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	XTUP	-	-	-	-
				R(1)				

XTUP: Crystal oscillator warm-up status. (External crystal oscillator)
It represents the crystal clock is stable (1) or not (0).
Cleared by H/W when Power-on reset and all kinds of reset.
Cleared by H/W when XTOFF bit is set.

l OSCICN (C6h): Internal RING Oscillator Control Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	DIV2	RINGON	DIV1	DIV0
					R/W(0)	R/W(1)	R/W(0)	R/W(0)

RINGON: 1 = Internal ring oscillator is running.
0 = Internal ring oscillator is killed.
Don't clear RINGON bit when XTRG = 0.

DIV[3:0]: Ring oscillator divider.

[0,0,0] = FOSC/1	[0,0,1] = FOSC/2
[0,1,0] = FOSC/4	[0,1,1] = FOSC/8
[1,0,0] = FOSC/16	[1,0,1] = FOSC/32
[1,1,0] = FOSC/64	[1,1,1] = FOSC/128

l CLKSEL (FBh): Clock Selection

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	XT/HF	WDEM	XR/PL	RG/PR	OSC32EB
				R/W(0)	R/W(0)	R/W(1)	R/W(1)	R/W(0)

XT/HF: XTAL division flag
0 = XTAL bypass. 1 = XTAL/2 division

XR/PL: PLL clock / XTRG clock selection
0 = PLL clock 1 = XTAL / RING mux clock

RG/PR : Ring clock selection.
0 = 32KHz ring clock for WDT power down.
1 = 4MHz ring clock for normal operation.

The MiDAS3.0 family has the on-chip oscillator circuitry, which is a single stage linear inverter intended for use as a crystal-controlled, positive reactance oscillator. Software can turn off its crystal or external oscillator by writing a 1 to the XTOFF bit in PMR. To drive the processor with an external clock source,

apply the external clock signal to XTAL1 and leave XTAL2 float, as shown in 6-42. After the crystal oscillator is stabilized, XTUP in the STATUS register will be set to 1.

The MiDAS3.0 family also contains the internal ring oscillator. It generates 4 MHz frequency signal. Clearing the RINGON bit to 0 disables the ring oscillator. The generated clock is prescaled by the bits DIV2, DIV1, and DIV0 in the OSCIN register. One can select the crystal oscillator or the internal ring oscillator as the system clock with the bit XT/RG in EXIF.

The MiDAS3.0 family contains an on-chip PLL to use it for generation of high frequency system clock. When using the PLL, the clock system of the MiDAS3.0 can generate 100 MHz system clock signal.

$$F_{SYS} = F_{VCO} / O_{DIV} = (N_{DIV} \text{ X } F_{REF}) / (O_{DIV} \text{ X } R_{DIV})$$

Where F_{SYS} is the system clock, R_{DIV} the prescaling factor, N_{DIV} the feedback devider, F_{REF} the frequency of the selected oscillator, O_{DIV} the output frequency divider.

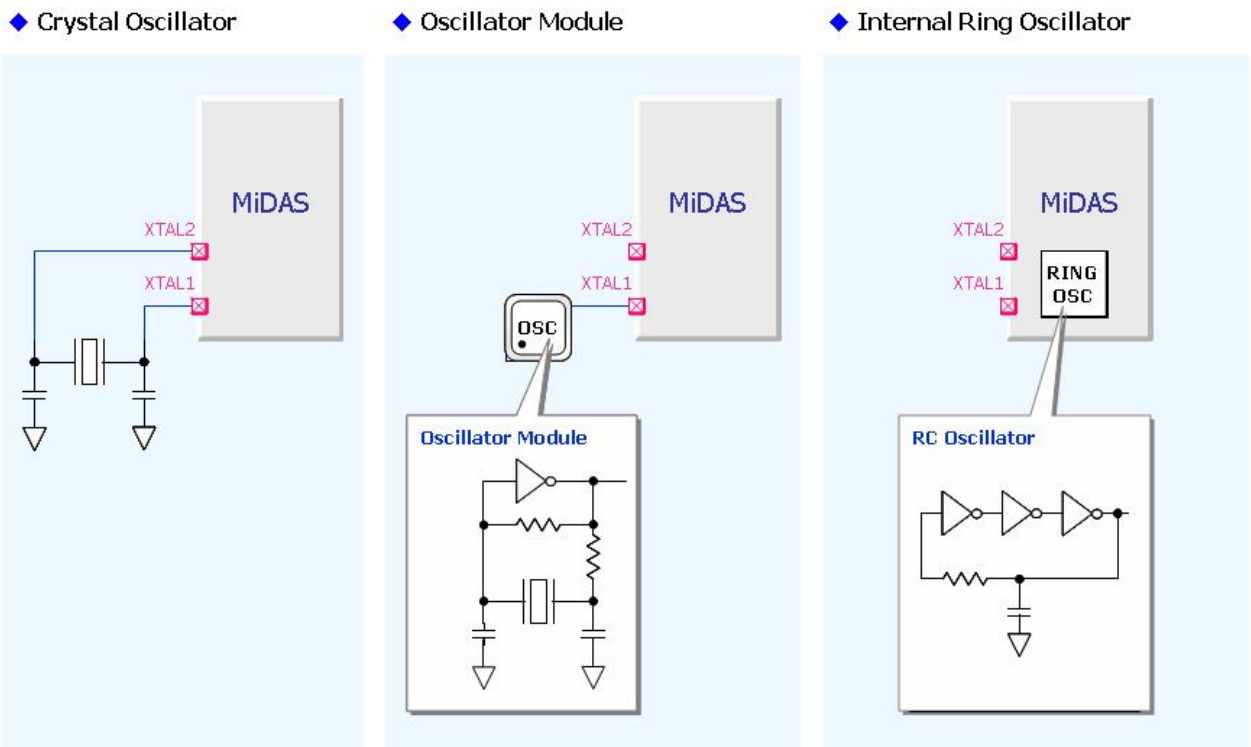


Figure 6-51 Clock Generators

l PLLCON (C1h): PLL Control Register

Bit No.	7	6	5	4	3	2	1	0
	Lock	-	icp1	icp0	Dly_ctr	Ph_sel	PLLPD	PLLBP
	R(0)		R/W(0)	R/W(1)	R/W(1)	R/W(0)	R/W(1)	R/W(0)

PLLBP: 1 = PLL Bypass Mode 0: PLL Normal Mode
PLLPD: 1 = PLL Power Down 0 = PLL Active
Ph_sel: PFD phase control
Dly_ctr: PFD delay control
icp[1:0]: CP current control
Lock: 1 = PLL Lock 0 = PLL unlock

l PLLNR (C2h): PLL Input Divider Register

Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	Odiv1	Odiv0	Rdiv1	Rdiv0
					R/W(1)	R/W(0)	R/W(1)	R/W(0)

Rdiv[1:0]: Input 2-bit divider
Odiv[1:0]: Output 2-bit divider

l PLLFR (C3h): PLL Feedback Divider Register

Bit No.	7	6	5	4	3	2	1	0
	Ndiv7	Ndiv6	Ndiv5	Ndiv4	Ndiv3	Ndiv2	Ndiv1	Ndiv0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(1)	R/W(0)

Ndiv[7:0]: Feedback 8-bit divider

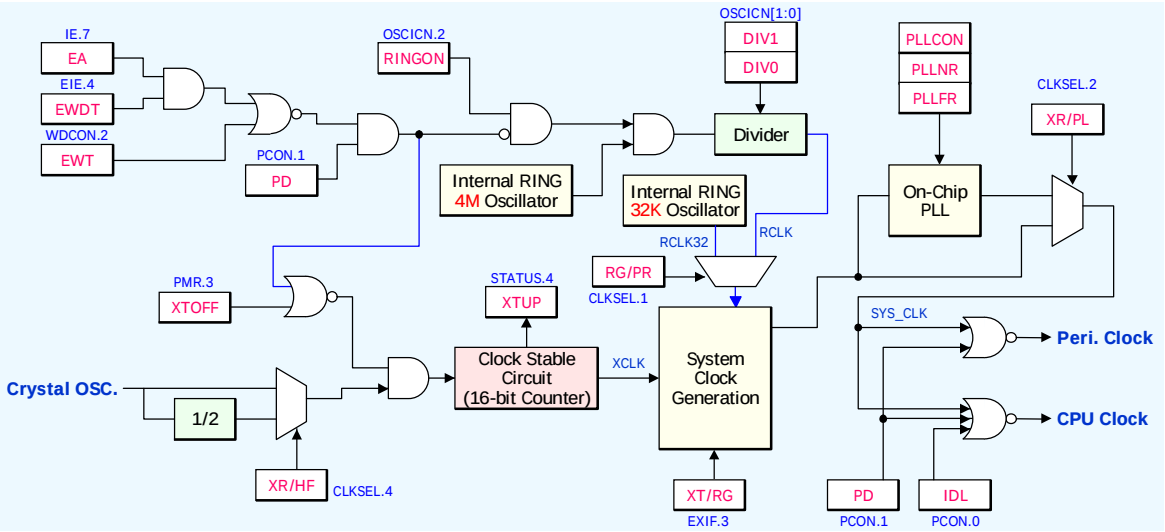


Figure 6-52 Clock Circuit

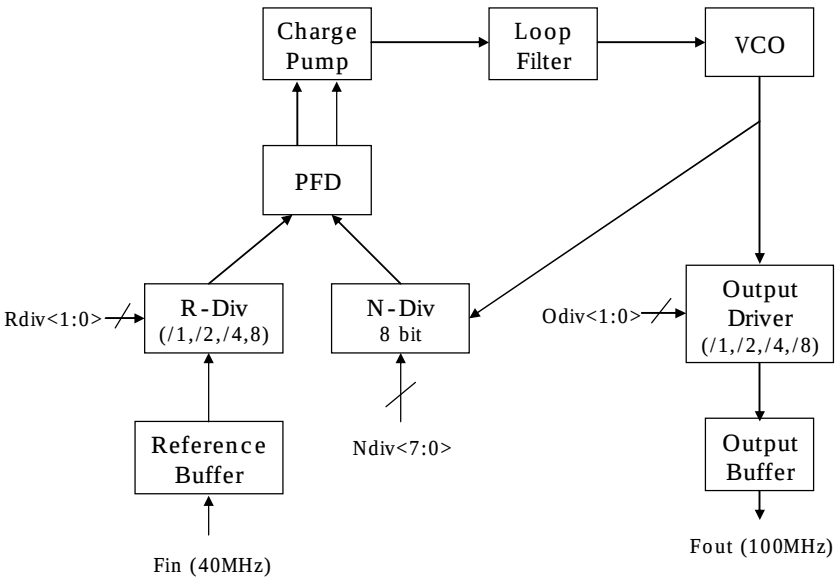


Figure 6-53 Block diagram of PLL

PLL consists of a programmable reference divider (R-Div), a phase/frequency detector (PFD), a charge pump, the loop filter, a voltage controlled oscillator (VCO), an output divider, and VCO divider (N-div).

The reference input (Fin) is divided by R-Div. The division rate is shown in Table 6-15. The divided signal is applied to the phase detector for comparison with the divided VCO signal. The phase/frequency detector produces an error signal between the divided reference signal and the divided VCO signal. This error signal controls the charge pump that generates the phase correction pulses, which are applied to the loop filter. The loop filter converts the pulses into the phase correction voltage signal. The voltage

signal controls the output frequency of VCO. The VCO frequency is divided by N-Div and Odiv as shown in Table 6-15. The signal divided by Odiv is used for the system operation. The signal divided by N-Div is applied to PFD to compare it with the divided reference frequency.

Table 6-15. Frequency Table (F_{REF} = 10MHz)

Rdiv[1:0]	Ndiv[7:0] range	F_{VCO} range [MHz]	Odiv [1:0]	F_{SYS} range [MHz]
1 (00)	0x07 ~ 0x0D	70 ~ 130	1 (00)	70 ~ 100
			2 (01)	35 ~ 65
			4 (10)	17.5 ~ 32.5
			8 (11)	8.75 ~ 16.25
2 (01)	0x0D ~ 0x1A	70 ~ 130	1 (00)	70 ~ 100
			2 (01)	35 ~ 65
			4 (10)	17.5 ~ 32.5
			8 (11)	8.75 ~ 16.25
4 (10)	0x1C ~ 0x34	70 ~ 130	1 (00)	70 ~ 100
			2 (01)	35 ~ 65
			4 (10)	17.5 ~ 32.5
			8 (11)	8.75 ~ 16.25
8 (11)	0x38 ~ 0x68	70 ~ 130	1 (00)	70 ~ 100
			2 (01)	35 ~ 65
			4 (10)	17.5 ~ 32.5
			8 (11)	8.75 ~ 16.25

Table 6-16. Frequency Table (F_{REF} = 20MHz)

Rdiv[1:0]	Ndiv[7:0] range	F_{VCO} range [MHz]	Odiv [1:0]	F_{SYS} range [MHz]
1 (00)	0x04 ~ 0x06	70 ~ 130	1 (00)	70 ~ 100
			2 (01)	35 ~ 65
			4 (10)	17.5 ~ 32.5
			8 (11)	8.75 ~ 16.25
2 (01)	0x07 ~ 0x0D	70 ~ 130	1 (00)	70 ~ 100
			2 (01)	35 ~ 65

			4 (10)	17.5 ~ 32.5
			8 (11)	8.75 ~ 16.25
4 (10)	0x0D ~ 0x1A	70 ~ 130	1 (00)	70 ~ 100
			2 (01)	35 ~ 65
			4 (10)	17.5 ~ 32.5
			8 (11)	8.75 ~ 16.25
8 (11)	0x1C ~ 0x34	70 ~ 130	1 (00)	70 ~ 100
			2 (01)	35 ~ 65
			4 (10)	17.5 ~ 32.5
			8 (11)	8.75 ~ 16.25

Table 6-17. PLL Operation Mode

Mode	Condition		Description
	PD	BP	
Power Down	1	-	Power down mode disables the PLL and pulls the output clock low
Bypass	0	1	Bypass mode enables some of the block in PLL and let the R-divider output directly go out.
Normal	0	0	The PLL is full enabled

6.2.11 Power Management

The MiDAS3.0 family has two power saving methods (POWER DOWN mode and IDLE mode) that help one to reduce the power consumption of the device.

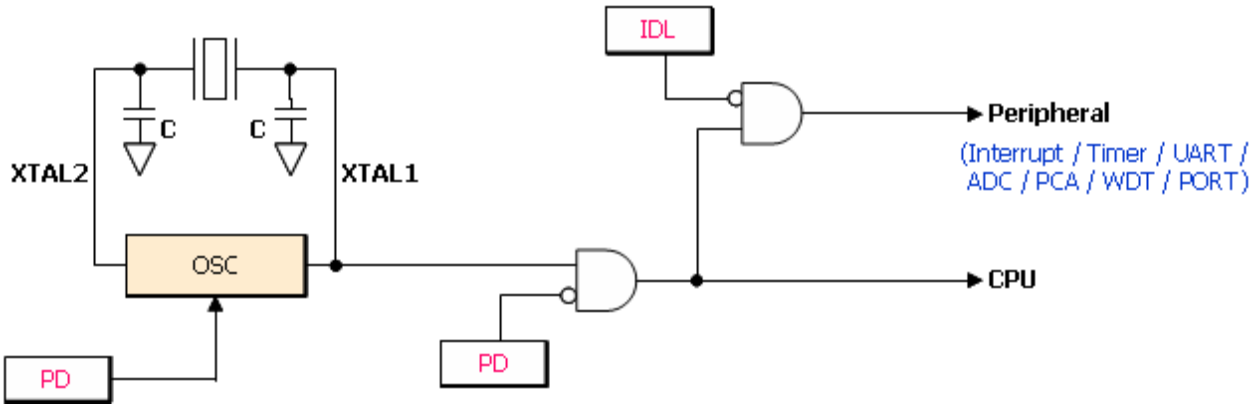


Figure 6-54 Power Management Circuit

6.2.11.1 IDLE Mode

One can put the device into the idle mode by set the IDL bit (PCON.0) to 1. Setting the IDL bit to 1 by an instruction causes the device to enter the IDLE mode after executing the instruction. In the idle mode, the internal clock signal is gated off to the CPU, but not to the interrupt and peripherals such as Timers, a watchdog timer and serial port blocks. The CPU status is preserved in its entirety; the Program counter, the Stack Pointer, the Program Status Word, the Accumulator and the other registers maintain their data during the Idle mode. The ALE and /PSEN pins hold at logic high levels. The port pins hold the logical states they had at the time the IDLE state had started.

There are two ways to terminate the idle mode. Activation of any enabled interrupt will cause the IDL bit to be cleared by hardware, terminating the idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into idle mode.

The other way of terminating the idle mode is with all kinds of resets. The device can be put into the reset by applying a high on the external RST pin, a Power-On-Reset condition or a Watchdog timer time-out. The external reset pin has to hold high logic level for at least six machine cycles i.e. 24 clock periods to complete the reset. By the reset, the program counter is cleared to 0000h and all the SFRs are set to the default value. Since the clock has been already running, without delay, execution starts immediately. In the IDLE mode, the Watchdog timer continues to run, and if enabled, a time-out will cause a watchdog timer interrupt that will wake up the device. After the MiDAS2.0 family exits from the idle mode by a reset, the execution starts from address 0000h.

6.2.11.2 Power Down Mode

An instruction that sets the PD bit (PCON.1) causes that to be the last instruction executed before going into the Power Down mode. In this mode, all the clock circuits are stopped. With the clock are frozen, all functions are stopped, but the on-chip data memory and SFRs are held. However, if EA (IE.7) = 1 and EWDT(EIE.4)=1, or if EWT(WDCON.1)=1, either a crystal oscillator or the internal ring oscillator operates in the power down mode. As a result, only the watchdog timer works and a little more power is consumed. In the power down mode, ALE and /PSEN outputs lows.

The MiDAS3.0 family can exit the Power Down mode with a hardware reset or the six external interrupts (INT0 to INT5). To wake it from the Power Down mode by an external interrupt, the interrupt has to be enabled (EA = 1, IE0, …, or IE5 = 1) and configured as level-sensitive (IT0, IT1, IT2,.., or IT5 = 0).

If EA(IE.7) = 1 and EWDT(EIE.4)=1, it can exit the Power Down mode with a watchdog reset. If EWT(WDCON.1)=1, it can be awakened by a watchdog interrupt. Reset redefines all the SFRs but does not change the on-chip data memory. An interrupt allows both the SFRs and the on-chip data memory to retain their value.

If the MiDAS3.0 family uses the crystal oscillator as the system clock, the crystal stabilization time is needed. So, the external interrupt pin should be held to “0” for at least 65,536clocks. After 65,536 clocks, system clock will be running normally and interrupt logic will start the external interrupt. If you want the device to respond immediately, set the RGSL bit (EXIF.1) to 1. Then it uses the internal ring oscillator for exiting the power down mode. After 65,536 clock cycles, the crystal oscillator operates as the system clock. If the internal ring oscillator is used as the system clock, the external interrupt service routine will be executed immediately. An interrupt will be serviced, and following RETI the next to be executed will be the one following the instruction that put the device into the power down mode. To prevent I/O port errors, insert the NOP instruction twice after the power down mode setting instruction.

An external reset can be used to exit the Power Down state. The high on RST pin terminates the Power Down mode, and restarts the system clock. At that time, the crystal oscillator or the external oscillator needed clock stabilization time. The program execution will restart from 0000H.

The PLL doesn’t operate in the power down mode. As a result, MiDAS3.0 should not use the PLL clock for system operation when entering or exiting from the power down mode. So you have to select the

crystal oscillation signal as the system clock before entering the power down mode. Simiarly, MiDAS3.0 wakes up from the power down mode with the crystal clock. Then we switch the system clock from the crystal oscillator to the PLL output signal (Fout).

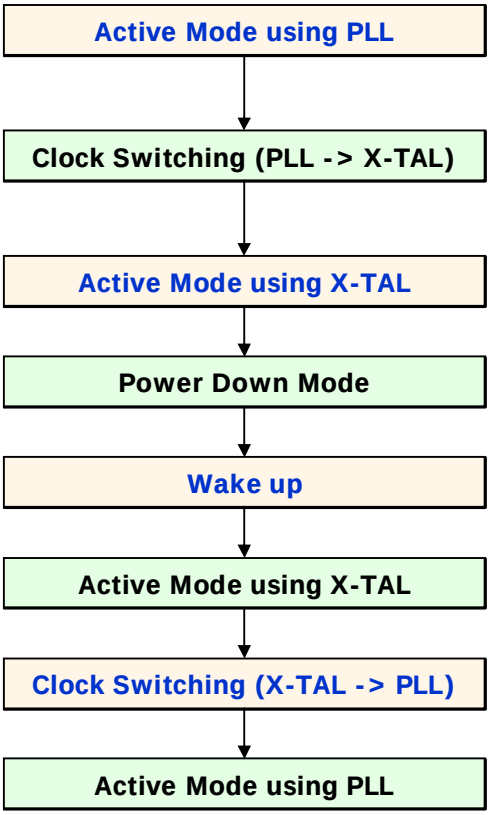


Figure 6-55 Clock System Control for Entering of Exiting from the Power Down Mode

Figure 6-55 shows how to control the clock system when using the PLL clock in the active mode. MiDAS3.0 operates with the PLL clock by executing the following code.

```
PLLNR = 0x01;                    // RDIV = 1, ODIV = 0,
PLLFR = 0x08;                    // 08: 80 MHz @ 22.118s MHz
PLLCON &= 0xFD;                  // PLL ON
While (!(PLLCON & 0x80));        // To check PLL lock
CLKSEL &= 0xFB;                  // Clock switch: XTAL -> PLL clock
```

To enter the power down mode, you should switch the system clock from the PLL clock to the crystal oscillator clock. Then PLL power is blocked. Here is an example code to do so.

```
CLKSEL |= 0x01;                 // 4 MHz Ring Clock Selection
EXIF |= 0x08;                   // Crystal Clock Selection
EXIF &= 0xFE;                   // LVD OFF
CLKSEL = 0x04;                  // Crystal / RING Clock
PLLCON = 0x02;                  // PLL power is blocked.
```

MiDAS3.0 enters the power down mode by assigning 0x02 to PCON and wakes up by external interrupts. After waking up, MiDAS3.0 uses the internal ring oscillator as the system clock. So you need to switch the system clock from the ring oscillator to the crystal oscillator.

```
EXIF = 0x00;                    // To select the internal ring oscillator
CLKSEL = 0x06;                  // To select the crystal or ring oscillator
While (!(STATUS&0x10));         // To wait for stabilization of the crystal oscillator
EXIF = 0x08;                    // To select the crystal oscillator
```

Finally, the system clock is switched from the crystal oscillator to the PLL clock.

```
PLLNR = 0x01;                   // RDIV = 1, ODIV = 0
PLLFR = 0x08;                   // 08 : 80 MHz @ 22.1184 MHz
PLLCON &= 0xFD;                 // PLL ON
While (!(PLLCON & 0x80));       // To check PLL lock
CLKSEL &= 0xFB;                 // Clock switch: Crystal -> PLL
```

6.2.12 Programming

The many MiDAS3.0 devices are shipped with the on-chip Flash memory array ready to be programmed.

6.2.12.1 Program Memory Lock Bits

A MiDAS3.0 device has six lock bits that can be left unprogrammed (1) or can be programmed (0) to obtain the protection features. When all the lock bits are left unprogrammed, no protection works. Program code (Flash memory) and user data (EEPROM) can be protected by these lock bits. According

to programmed lock bits, the six protection modes are specified in Table 6-18.

Table 6-18. Protection Modes

Level	Lock bits						Feature					
	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Flash		EEPROM		MDS	IAP
							Program	Read	Program	Read		
0	1	1	1	1	1	1	Enable	Enable	Enable	Enable	Enable	Enable
1	0	1	1	1	1	1	Disable	Enable	Enable	Enable	Enable	Enable
2	X	0	1	1	1	1	Disable	Disable	Enable	Enable	Enable	Enable
3	X	X	0	1	1	1	Disable	Disable	Disable	Enable	Enable	Enable
4	X	X	X	0	1	1	Disable	Disable	Disable	Disable	Enable	Enable
5	X	X	X	X	0	1	Disable	Disable	Disable	Disable	Disable	Enable
6	X	X	X	X	X	0	Disable	Disable	Disable	Disable	Disable	Disable

6.2.12.2 In-System Programming (ISP)

The program code array and the user data array can be programmed using the serial ISP interface through the EJTAG port in a target system. The EJTAG Port consists of the power supply pins and the serial ISP interface pins: the power supply pins (V_{DD}, V_{SS}) and the serial ISP interface pins (MDS_SCK, MDS_SDA, /PSEN).

Command	Function
Blank	▪ Confrims that the code and the user data memory are left unprogrammed.
Erase Chip	▪ Erases the code and the user data memory. ▪ Code: Flash ▪ User data: EEPROM
	▪ The device memory will return to the unprogrammed status.
Read code/EEPROM	▪ Reads the code/data from the device memory.
	▪ Reads code/data are load into the CORERIVER ISP software buffer and displayed on the screen.
Write code/EEPROM	▪ Sends the data in the CORERIVER ISP software buffer to all the device memory.

Verify Chip	▪ Compares the data of the CORERIVER ISP software buffer with those of the device memory.
	▪ If no difference between two data is found, the success signal is returned.
	▪ If an differences are found, the failure signal and the number of mismatched bytes are returned

6.2.12.3 In-Application Programming (IAP)

The MiDAS3.0 family can store operation status/data in the EEPROM (user data memory) or update program code in the Flash (code memory). The code memory (62KB) can be programmed or erased during the execution of the program code. So can the user data memory (2KB). This is called “In-Application Programming (IAP).”

IAP can program code/data memory by the byte but erase them by the sector (512 byte). If you want to erase only one byte, move one sector data to the on-chip external data memoryn first. Then, update the moved data and erases the target sector. Finally, program the erased sector with the updated data again.

The IAP routine begins at FFF0H. The IAP routine shows the execution result through the return value in the accumulator. 8XH means ‘success’; FCH ‘program fail; FDH ‘address fail’; FEH ‘lock fail’; FFh ‘lock fail’. Before executing the IAP routine, FLASH_AEN flag must to be set to 1. As a result of executing the IAP routine, the value of PSW SFR can be changed. Any interrupt service routine will not be executed timely since the CPU is suspended for tens of milliseconds during executing the IAP routine.

- **FAEN (F7h): IAP Routine Access Enable Register**

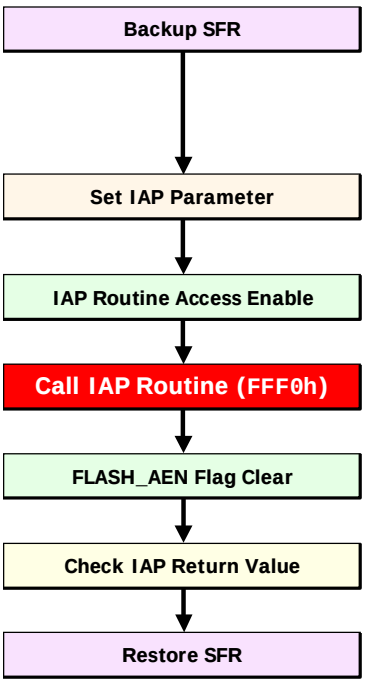
Bit No.	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	FLASH_AEN
								R/W(0)

FLASH_AEN: IAP Routine Access Enable

The IAP sequences shown in Figure 6-56 are as follows:

1. At first, back up the SFRs
2. Set IAP parameters.

3. Set the FLASH_AEN flag to 1.
4. Call the IAP routine.
5. Clear the FLASH_AEN flag to 0.
6. Check the return value of the IAP routine.
7. Restore the SFR values.




```
PUSH A                  ; backup acc
PUSH B                  ; backup b
PUSH DPL                ; backup dptr
PUSH DPH
MOV R1, IE              ; backup IE SFR
CLR IE.7                ; Interrupt disable

MOV B, #03h             ; IAP Function setting
MOV A, #55h             ; Programmed Data
MOV DPTR, #08000h       ; Programmed Address

ORL  EEAEN, #01h        ; IAP routine access enable

CALL FFF0h              ; Call IAP routine

MOV FAEN, #000h         ; IAP routine access disable

CJNE A, #83h, IAP_FAIL  ; Check return message

MOV IE, r1              ; restore IE SFR
POP DPH                 ; restore acc, b, dptr
POP DPL
POP B
POP A
```

Figure 6-56 IAP sequences

7 Absolute Maximum Ratings

Table 7-1 Absolute Maximum Ratings

Symbol	Parameter	Rating	Unit
V_{DD}	DC supply voltage	1.62 to 1.98	V
V_{DDIO}	DC IO supply voltage	0.3 to 3.6	V
V_{IN}	DC input voltage	- 0.3 to 5.5	V
I_{IN}	DC input current	+/- 10	mA
T_{STG}	Storage temperature	- 40 to 125	^{o}C

Table 7-2 Recommended Operating Conditions

Symbol	Parameter	Rating	Unit
V_{DD}	DC supply voltage	- 0.3 to 1.98	V
V_{DDIO}	DC IO supply voltage	- 0.3 to 3.6	V
T_A	Industrial temperature range	- 20 to 85	^{o}C

- Notes
  - All electrical characteristics are applied to digital cell blocks without any analog core.

8 DC Characteristics

(T_A = -20°C ~ +85°C, V_{DD}=2.7V ~ 3.6V unless otherwise specified)

Parameter	Symbol	Pin	Conditions	Value			Unit
				Min.	Typ.	Max.	
Input Low Voltage	V_{IL1}	RESETB, P0, P1, P2,P3	V_{DDIO} = 1.68V~3.6V	-0.5	-	$0.2V_{DDIO}$ -0.1	V
Input High Voltage	V_{IH1}	RESETB, P0, P1, P2,P3	V_{DDIO} = 1.68V~3.6V	$0.2V_{DDIO}$+1.0	-	V_{DDIO}+0.5	V
Output Low Voltage	V_{OL}	All pins	I_{OL}=20mA @V_{DDIO}=3.3V	-	-	$0.3V_{DDIO}$	V
Output High Voltage	V_{OH}	All pins	I_{OH}=-15mA @V_{DDIO}=3.3V	$0.7V_{DD}$	-	-	V
	V_{OH2}	Pull-up	I_{OH}=-10uA @V_{DDIO}=3.3V	$0.7V_{DD}$	-	-	V
Input Leakage Current	I_{IL}	All pins except XTAL1, XTAL2	V_{IN} = V_{IH} or V_{IL}	-	-	±1.0	uA
Pin Capacitance	C_{IO}	All pins	V_{DDIO} = 3.3V	-	10	-	pF

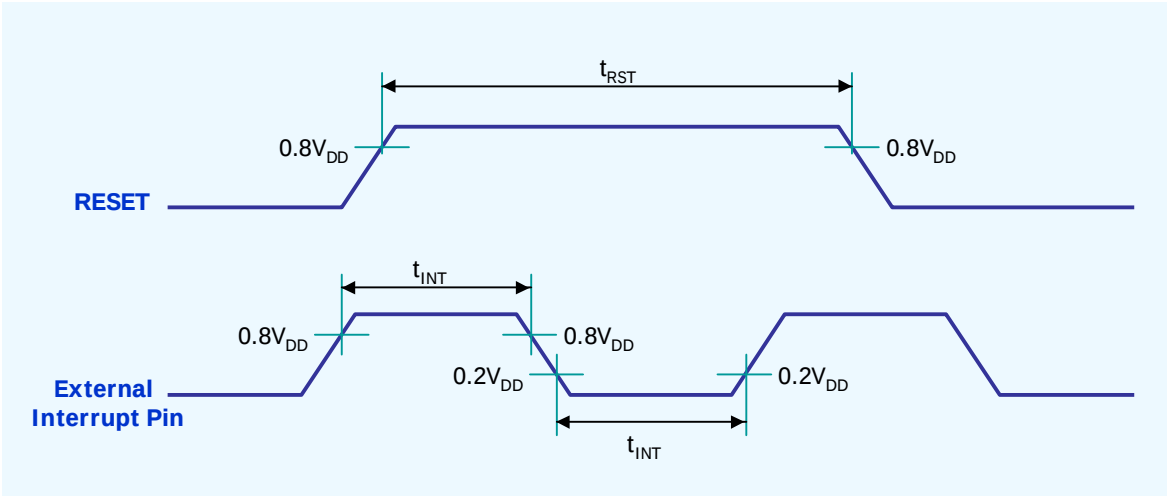
u PLL Clock DC Characteristic

Parameter	Symbol	Min	Typ	Max	Unit	Condition
Input Frequency	Fref	2		40	MHz	
Comparison Frequency	Fcomp	2	20	20	MHz	Fcomp = Fref / Rdiv
VCO Frequency	Fvco	70	100	130	MHz	Fvco = Ndiv * Fcomp
Output System clock Frequency	Fsys	8.75	-	100	MHz	Fsys = Fvco / Odiv

9 AC Characteristics

(T_A = -20°C ~ +85°C unless otherwise specified)

Parameter	Symbol	Pin	Conditions	Value			Unit
				Min.	Typ.	Max.	
Operating Frequency	F_{OSC}	XTAL1, XAL2	V_{DDIO} = 3.3V ±10%	-	-	40	MHz
RESET Input Width	t_{RST}	RESET	V_{DDIO} = 3.3V ±10%	24	-	-	F_{OSC}
External interrupt Input Width	t_{INT}	External Interrupt	V_{DDIO} = 3.3V ±10%	4	-	-	F_{OSC}



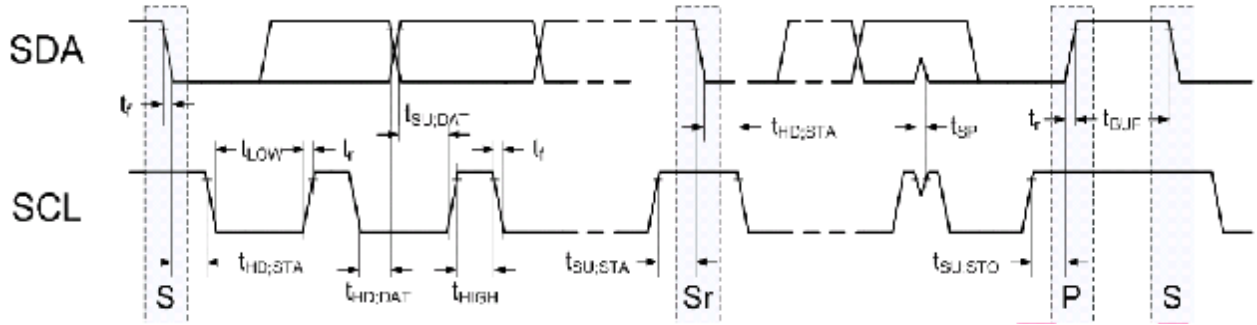
10 ADC Specifications

Parameter		Symbol	Conditions	Value			Unit
				Min.	Typ.	Max.	
Supply Voltage		V_{DDADC}	-	1.68	-	3.3	V
Input Voltage		V_{INADC}	-	V_{SSIO}	-	V_{DDIO}	V
Resolution		RES_{ADC}	-	-	10	-	Bit
Operating Frequency		F_{ADC}	V_{DDIO} = 3.0V ~ 3.6V V_{DDIO} = 1.68 ~ 1.92V	-	-	10 5	MHz
Conversion Time		t_{ADC}	-	-	$96/F_{ADC}$	-	sec
Overall Accuracy		OA_{ADC}	V_{DDIO} = 3.3V, F_{ADC} = 10MHz V_{DDIO} = 1.8V, F_{ADC} = 5MHz	-	±2.0	±4.0	LSB
Integral Nonlinearity		INL_{ADC}	V_{DDIO} = 3.3V, F_{ADC} = 10MHz V_{DDIO} = 1.8V, F_{ADC} = 5MHz	-	±2.0	±4.0	LSB
Differential Nonlinearity		DNL_{ADC}	V_{DDIO} = 3.3V, F_{ADC} = 10MHz V_{DDIO} = 1.8V, F_{ADC} = 5MHz	-	±0.5	±1.0	LSB
Zero Input Error		ZIE_{ADC}	V_{DDIO} = 3.3V, F_{ADC} = 10MHz V_{DDIO} = 1.8V, F_{ADC} = 5MHz	-	±2.0	±4.0	LSB
Full Scale Error		FSE_{ADC}	V_{DDIO} = 3.3V, F_{ADC} = 10MHz V_{DDIO} = 1.8V, F_{ADC} = 5MHz	-	±2.0	±4.0	LSB
Analog Input Capacitance		C_{INADC}	-	-	10	15	pF
ADC Current	Active	I_{ADC}	V_{DDIO} = 3.3V, F_{ADC} = 10MHz	-	1.0	2.0	mA
			V_{DDIO} = 1.8V, F_{ADC} = 5MHz	-	0.3	0.6	mA
	Power-down		V_{DDIO} = 3.3V	-	-	100	nA

11 I2C Characteristics (Normal I/O)

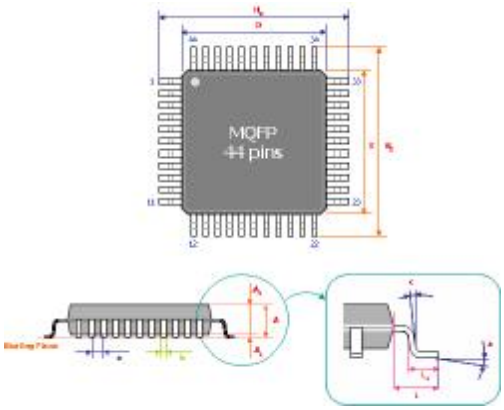
Symbol	Description	Value		Unit
		Min.	Max.	
F_{SCL}	SCL Clock frequency	0	400	KHz
t_{LOW}	Low period of the SCL Clock	1.3	-	us
t_{HIGH}	High period of the SCL Clock	0.6	-	us
t_r	Rise time of both SDA and SCL signals	20+0.1 C_b	300	us
t_f	Fall time of both SDA and SCL signals	20+0.1 C_b	250	Us
$t_{HD;STA}$	Hold time START condition	0/6	-	us
$t_{SU;STO}$	Set-up time for STOP condition	0/6	-	us
$t_{HD;DAT}$	Data hold time	0	0.9	us
$t_{SU;DAT}$	Data set-up time	100	-	us
$t_{SU;STA}$	Set-up time for a repeated START condition	0.6	-	us

C_b = Capacitance of one bus in pF



S : Start Condition / Sr : Repeated Start Condition / P : Stop Condition

12 Package Dimension



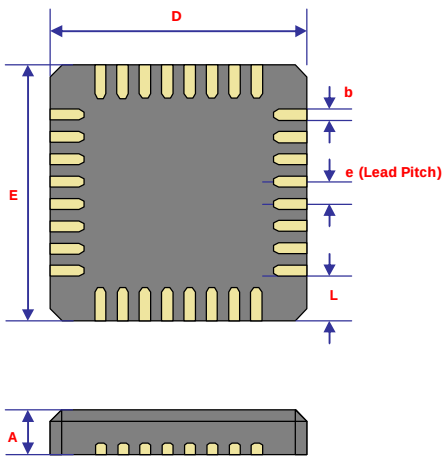
[44-MQFP]

Symbol	Dimension in Inches			Dimension in mm		
	Min.	Nom.	Max.	Min.	Nom.	Max.
A	-	-	0.091	-	-	2.30
A_1	0.002	-	0.006	0.05	-	0.15
A_2	0.077	0.081	0.085	1.95	2.05	2.15
b	0.012	0.015	0.018	0.30	0.37	0.45
D		0.394 BSC			10.00 BSC	
E		0.394 BSC			10.00 BSC	
e		0.031 BSC			0.80 BSC	
H_D		0.520 BSC			13.20 BSC	
H_E		0.520 BSC			13.20 BSC	
L	-	0.063	-	-	1.60	-
L_1	0.024	0.031	0.039	0.60	0.80	1.00
a	0°	-	8°	0°	-	8°
c	0°	-	-	0°	-	-

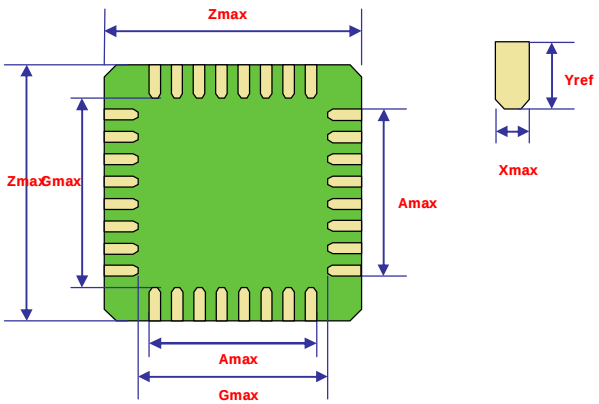
Notes:
1. Dimension D & E do not include interlead flash.
2. Controlling dimension: Inches
3. General appearance spec. should be based on final visual inspection spec

Figure 12-1 MQFP 44-pin Package Dimension

[MLF 32]



[PCB dimension]



Unit : mm

Package				Package Dimensions with Tolerance										Board Land Pattern Dimensions				
Size	I/O	Leads / Side	Lead Pitch	D(min)	D(max)	E(min)	E(max)	b(min)	b(max)	L(min)	L(max)	A(typ)	A(max)	Xmax	Yref	Amax	Gmin	Zmax
5X5	32	8	0.50	4.90	5.10	4.90	5.10	0.18	0.30	0.4	0.6	0.85	0.90	0.28	0.69	3.78	3.93	5.31

Figure 12-2 MLF 44-pin Package Dimension

13 Product Numbering System

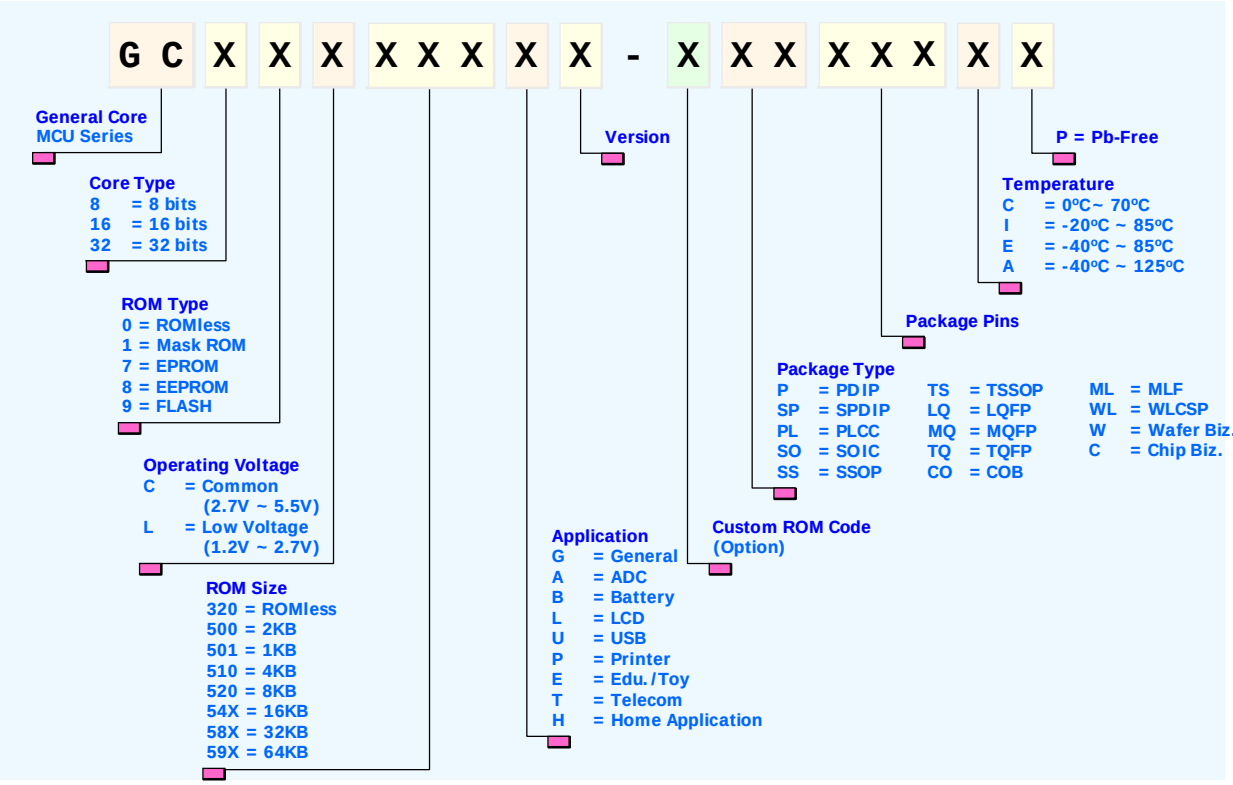


Figure 13-1 Product Numbering System

c

14 Appendix A: Instruction Set

Table 14-1 Note on instruction set and addressing modes

Notation	Description
Rn	Register R0-R7 of the currently selected Register Bank.
direct	8-bit internal data location's address. This could be an IRAM location (0-127) or a SFR (128-255).
@Ri	8-bit IRAM location (0-255) addressed indirectly through register R0 or R1.
#data	8-bit constant included in instruction
#data16	16-bit constant included in instruction
addr16	16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64Kbyte Program Memory address space.
Addr11	11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2Kbyte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is –128 to +127 bytes relative to first byte of the following instruction.
bit	Direct addressed bit in IRAM or SFR.

ACALL addr11

Function: Absolute Call

Description: ACALL unconditionally calls a subroutine located at the indicated address. The instruction increments the PC twice to obtain the address of the following instruction, then pushes the 16-bit result onto the stack (low-order byte first) and increments the Stack Pointer twice. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, opcode bits 7-5, and the second byte of the instruction. The called subroutine must therefore start within the same 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected.

Example: Initially SP equals 07h. The label “SUBRTN” is at program memory location 0345h. After executing the instruction,

ACALL SUBRTN

at location 0123h, SP will contain 09h, internal RAM locations 08h and 09h will contain 25h and 01h, respectively, and the PC will contain 0345h.

Bytes: 2

Cycles: 3

Encoding:

a10 a9 a8 1	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation:
ACALL
(PC) ← (PC)+ 2
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (PC_{15-8})
(PC_{10-0}) ← page address

ADD A, <src-byte>

Function: Add

Description: ADD adds the byte variable indicated to the Accumulator, leaving the result in the Accumulator. The carry and auxiliary-carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands, or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3h (11000011b) and register 0 holds 0AAh (10101010b). The instruction,

ADD A, R0

will leave 6Dh (01101101b) in the Accumulator with the AC flag cleared and both the carry flag and OV set to 1.

ADD A, Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	1	r	r	r

Operation: ADD
(A) ← (A) + (Rn)

ADD A, direct

Bytes: 2

Cycles: 2

Encoding:

0	0	1	0	0	1	0	1	direct address

Operation: ADD
(A) ← (A) + (direct)

ADD A, @Ri

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	0	1	1	i

Operation: ADD
(A) ← (A) + ((Ri))

ADD A, #data

Bytes: 2

Cycles: 2

Encoding:

0	0	1	0	0	1	0	0	immediate data

Operation: ADD
(A) ← (A) + #data

ADDC A, <src-byte>

Function: Add with Carry

Description: ADDC simultaneously adds the byte variable indicated, the carry flag and the Accumulator contents, leaving the result in the Accumulator. The carry and auxiliary–carry flags are set, respectively, if there is a carry-out from bit 7 or bit 3, and cleared otherwise. When adding unsigned integers, the carry flag indicates an overflow occurred.

OV is set if there is a carry-out of bit 6 but not out of bit 7, or a carry-out of bit 7 but not out of bit 6; otherwise OV is cleared. When adding signed integers, OV indicates a negative number produced as the sum of two positive operands or a positive sum from two negative operands.

Four source operand addressing modes are allowed: register, direct, register-indirect, or immediate.

Example: The Accumulator holds 0C3h (11000011b) and register 0 holds 0AAh (10101010b) with the carry flag set. The instruction,

ADDC A, R0

will leave 6Eh (01101110b) in the Accumulator with AC cleared and both the carry flag and OV set to 1.

ADDC A, Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	1	r	r	r

Operation: ADDC
(A) ← (A) + (C) + (Rn)

ADDC A, direct

Bytes: 2

Cycles: 2

Encoding:

0	0	1	1	0	1	0	1	direct address

Operation: ADDC
(A) ← (A) + (C) + (Rn)

ADDC A, @Ri

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	1	1	i

Operation: ADDC
(A) ← (A) + (C) + ((Ri))

ADDC A, #data

Bytes: 2

Cycles: 2

Encoding:

0	0	1	1	0	1	0	0	immediate data

Operation: ADDC
(A) ← (A) + (C) + #data

AJMP addr11

Function: Absolute Jump

Description: AJMP transfers program execution to the indicated address, which is formed at run-time by concatenating the high-order five bits of the PC (after incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must therefore be within the same 2K block of program memory as the first byte of the instruction following AJMP.

Example: The label “JMPADR” is at program memory location 0123h. The instruction,

AJMP JMPADR

is at location 0345h and will load the PC with 0123h.

Bytes: 2

Cycles: 3

Encoding:

a10 a9 a8 0	0 0 0 1	a7 a6 a5 a4	a3 a2 a1 a0

Operation: AJMP
(PC) ← (PC) + 2
(PC_{10-0}) ← page address

ANL <dest-byte>, <src-byte>

Function: Logical-AND for byte variables

Description: ANL performs the bitwise logical-AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.
The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register, indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.
Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch not the input pins.

Example: If the Accumulator holds 0C3h (11000011b) and register 0 holds 55h (0101010b) then the instruction,

ANL A, R0

will leave 41h (01000001b) in the Accumulator.

When the destination is a directly addressed byte, this instruction will clear combinations of bits in any RAM location or hardware register. The mask byte determining the pattern of bits to be cleared would either be a constant contained in the instruction or a value computed in the Accumulator at run-time. The instruction,

ANL P1, # 01110011b

will clear bits 7, 3, and 2 of output port 1.

ANL A, Rn

Bytes: 1

Cycles: 1

Encoding:

0	1	0	1	1	r	r	r

Operation: ANL
(A) ← (A) ∧ (Rn)

ANL A, direct

Bytes: 2

Cycles: 2

Encoding:

0	1	0	1	0	1	0	1	direct address

Operation: ANL
(A) ← (A) ∧ (direct)

ANL A, @Ri

Bytes: 1

Cycles: 1

Encoding:

0	1	0	1	0	1	1	i

Operation:
ANL
(A) ← (A) ∧ ((Ri))

ANL A, #data

Bytes: 2

Cycles: 2

Encoding:

0	1	0	1	0	1	0	0	immediate data

Operation:
ANL
(A) ← (A) ∧ #data

ANL direct, A

Bytes: 2

Cycles: 2

Encoding:

0	1	0	1	0	0	1	0	direct address

Operation:
ANL
(direct) ← (direct) ∧ (A)

ANL direct, #data

Bytes: 3

Cycles: 3

Encoding:

0	1	0	1	0	0	1	1	direct address	immediate data

Operation:
ANL
(direct) ← (direct) ∧ #data

ANL C, <src-bit>

Function: Logical-AND for bit variables

Description: If the boolean value of the source bit is a logical 0 then clear the carry flag; otherwise leave the carry flag in its current state. A slash ("/") preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Only direct addressing is allowed for the source operand.

Example: Set the carry flag if, and only if, P1.0 = 1, ACC. 7 = 1, and OV = 0:

```
MOV     C, P1.0         ; LOAD CARRY WITH INPUT PIN STATE
ANL     C, ACC.7        ; AND CARRY WITH ACCUM. BIT 7
ANL     C, /OV          ; AND WITH INVERSE OF OVERFLOW FLAG
```

ANL C, bit

Bytes: 2

Cycles: 2

Encoding:

1 0 0 0	0 0 1 0	bit address

Operation: ANL
(C) ← (C) ∧ (bit)

ANL C, /bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 1	0 0 0 0	bit address

Operation: ANL
(C) ← (C) ∧ ~(bit)

CJNE <dest-byte>, <src-byte>, rel

Function: Compare and Jump if Not Equal.

Description: CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The branch destination is computed by adding the signed relative-displacement in the last instruction byte to the PC, after incrementing the PC to the start of the next instruction.
The carry flag is set if the unsigned integer value of <dest-byte> is less than the unsigned integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected.

The first two operands allow four addressing mode combinations: the Accumulator may be compared with any directly addressed byte or immediate data, and any indirect RAM location or working register can be compared with an immediate constant.

Example: The Accumulator contains 34h. Register 7 contains 56h. The first instruction in the sequence,

```
                CJNE    R7, #60h, NOT_EQ
;               • • •. .  • • • •          ; R7 = 60h.
NOT_EQ:         JC      REQ_LOW            ; IF R7 < 60h.
;               • • •     • • • •          ; R7 > 60h.
```

sets the carry flag and branches to the instruction at label NOT_EQ. By testing the carry flag, this instruction determines whether R7 is greater or less than 60h.
If the data being presented to Port 1 is also 34h, then the instruction,

```
WAIT:           CJNE    A, P1, WAIT
```

clears the carry flag and continues with the next instruction in sequence, since the Accumulator does equal the data read from P1. (If some other value was being input on P1, the program will loop at this point until the P1 data changes to 34h.)

CJNE A, direct, rel

Bytes: 3

Cycles: 4

Encoding:

1 0 1 1	0 1 0 1	direct address	relative

Operation:
(PC) ← (PC) + 3
IF (A) < > (direct)
THEN (PC) ← (PC) + relative offset
IF (A) < (direct)
THEN (C) ← 1
ELSE (C) ← 0

CJNE A, #data, rel

Bytes: 3

Cycles: 4

Encoding:

1 0 1 1	0 1 0 0	immediate data	relative

Operation:
(PC) ← (PC) + 3
IF (A) < > data
THEN (PC) ← (PC) + relative offset
IF (A) < data
THEN (C) ← 1
ELSE (C) ← 0

CJNE Rn, #data, rel

Bytes: 3

Cycles: 4

Encoding:

1 0 1 1	1 r r r	immediate data	relative

Operation:
(PC) ← (PC) + 3
IF (Rn) < > data
THEN (PC) ← (PC) + relative offset
IF (Rn) < data
THEN (C) ← 1
ELSE (C) ← 0

CJNE @Ri, #data, rel

Bytes: 3

Cycles: 4

Encoding:

1 0 1 1	0 1 1 i	immediate data	relative

Operation:
(PC) ← (PC) + 3
IF ((Ri)) < > data
THEN (PC) ← (PC) + relative offset
IF ((Ri)) < data
THEN (C) ← 1
ELSE (C) ← 0

CLR A

Function: Clear Accumulator

Description: The Accumulator is cleared (all bits set on zero). No flags are affected.

Example: The Accumulator contains 5Ch (01011100b). The instruction,

CLR A

will leave the Accumulator set to 00h (00000000b).

Bytes: 1

Cycles: 1

Encoding:

1	1	1	0	0	1	0	0

Operation: CLR
$A \leftarrow 0$

CLR bit

Function: Clear bit

Description: The indicated bit is cleared (reset to zero). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

Example: Port 1 has previously been written with 5Dh (01011101b). The instruction,

CLR P1.2

will leave the port set to 59h (01011001b).

CLR C

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	0	0	1	1

Operation:
CLR
(C) ← 0

CLR bit

Bytes: 2

Cycles: 2

Encoding:

1	1	0	0	0	0	1	0	bit address

Operation:
CLR
(bit) ← 0

CPL A

Function: Complement Accumulator

Description: Each bit of the Accumulator is logically complemented (one's complement). Bits which previously contained a one are changed to a zero and vice-versa. No flags are affected.

Example: The Accumulator contains 5Ch (01011100b). The instruction,

CPL A

will leave the Accumulator set to 0A3h(10100011b).

Bytes: 1

Cycles: 1

Encoding:

1	1	1	1	0	1	0	0

Operation: CPL
(A) ← ~(A)

CPL bit

Function: Complement bit

Description: The bit variable specified is complemented. A bit which had been a one is changed to zero and vice-versa. No other flags are affected. CPL can operate on the carry or any directly addressable bit.

Note: When this instruction is used to modify an output pin, the value used as the original data will be read from the output data latch, not the input pin.

Example: Port 1 has previously been written with 5Bh (01011101b). The instruction sequence,

CPL P1.1

CPL P1.2

will leave the port set to 5Bh (01011011b).

CPL C

Bytes: 1

Cycles: 1

Encoding:

1	0	1	1	0	0	1	1

Operation: CPL
(C) ← ~(C)

CPL bit

Bytes: 2

Cycles: 2

Encoding:

1	0	1	1	0	0	1	0	bit address

Operation: CPL
(bit) ← ~(bit)

DA A

Function: Decimal-adjust Accumulator for Addition

Description: DA A adjusts the eight-bit value in the Accumulator resulting from the earlier addition of two variables (each in packed-BCD format), producing two four-bit digits. Any ADD or ADDC instruction may have been used to perform the addition.

If Accumulator bits 3-0 are greater than nine (xxxx1010-xxxx1111), or if the AC flag is one, six is added to the Accumulator producing the proper BCD digit in the low-order nibble. This internal addition would set the carry flag if a carry-out of the low-order four-bit field propagated through all high-order bits, but it would not clear the carry flag otherwise.

If the carry flag is now set, or if the four high-order bits now exceed nine (1010xxxx-1111xxxx), these high-order bits are incremented by six, producing the proper BCD digit in the high-order nibble. Again, this would set the carry flag if there was a carryout of the high-order bits, but wouldn't clear the carry. The carry flag thus indicates if the sum of the original two BCD variables is greater than 100, allowing multiple precision decimal addition. OV is not affected.

All of this occurs during the one instruction cycle. Essentially, this instruction performs the decimal conversion by adding 00h, 06h, 60h, or 66h to the Accumulator, depending on initial Accumulator and PSW conditions.

Note: DA A cannot simply convert a hexadecimal number in the Accumulator to BCD notation, nor does DA A apply to decimal subtraction.

Example: The Accumulator holds the value 56h (01010110b) representing the packed BCD digits of the decimal number 56. Register 3 contains the value 67h (01100111b) representing the packed BCD digits of the decimal number 67. The carry flag is set. The instruction sequence.

```
ADDC    A, R3
DA      A
```

will first perform a standard two's-complement binary addition, resulting in the value 0BEh (10111110b) in the Accumulator. The carry and auxiliary carry flags will be cleared.

The Decimal Adjust instruction will then alter the Accumulator to the value 24h (00100100b), indicating the packed BCD digits of the decimal number 24, the low-order two digits of the decimal sum of 56, 67, and the carry-in. The carry flag will be set by the Decimal Adjust instruction, indicating that a decimal overflow occurred. The true sum 56, 67, and 1 is 124.

BCD variables can be incremented or decremented by adding 01h or 99h. If the Accumulator initially holds 30h (representing the digits of 30 decimal), then the instruction sequence,

```
ADD     A, #99h
DA      A
```

will leave the carry set and 29h in the Accumulator, since 30 + 99 = 129. The low-order byte of the sum can be interpreted to mean 30 – 1 = 29.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	0	0

Operation:

DA
- contents of Accumulator are BCD

IF {[$(A_{3-0}) > 9$] $\vee$ [(AC) = 1]}
THEN $(A_{3-0}) \leftarrow (A_{3-0}) + 6$
AND
IF {[$(A_{7-4}) > 9$] $\vee$ [(C) = 1]}
THEN $(A_{7-4}) \leftarrow (A_{7-4}) + 6$

DEC byte

Function: Decrement

Description: The variable indicated is decremented by 1. An original value of 00h will underflow to 0FFh.
No flags are affected. Four operand addressing modes are allowed: accumulator, register, direct or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: Register 0 contains 7Fh (01111111b). Internal RAM locations 7Eh and 7Fh contain 00h and 40h, respectively. The instruction sequence

DEC @R0

DEC R0

DEC @R0

will leave register 0 set to 7Eh and internal RAM locations7Eh and 7Fh set to 0FFh and 3Fh.

DEC A

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	1	0	0

Operation: DEC
(A) ← (A) – 1

DEC Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	1	r	r	r

Operation: DEC
(Rn) ← (Rn) – 1

DEC direct

Bytes: 2

Cycles: 2

Encoding:

0 0 0 1	0 1 0 1	direct address

Operation: DEC
(direct) ← (direct) – 1

DEC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 1	0 1 1 i

Operation: DEC
((Ri)) ← ((Ri)) – 1

DEC DPTR

Function: Decrement Data Pointer

Description: Decrement the 16-bit data pointer by 1. A 16-bit decrement (modulo 2^{16}) is performed; an underflow of the low-order byte of the data pointer (DPL) from 0FFH to 00H will decrement the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be decremented.

Example: Registers DPH and DPL contain 12h and 01h, respectively. The instruction sequence,

```
DEC  DPTR
DEC  DPTR
DEC  DPTR
```

will change DPH and DPL to 11h and 0FEh.

Bytes: 1

Cycles: 1

Encoding:

1	0	1	0	0	1	0	1

Operation: DEC
(DPTR) ← (DPTR) - 1

DIV AB

Function: Divide

Description: DIV AB divides the unsigned eight-bit integer in the Accumulator by the unsigned eight-bit integer in register B. The Accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and OV flags will be cleared.

Exception: if B had originally contained 00h, the values returned in the Accumulator and B register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

Example: The Accumulator contains 251 (0FBh or 11111011b) and B contains 18 (12h or 00010010b). The instruction,

DIV AB

will leave 13 in the Accumulator (0Dh or 00001101b) and the value 17 (11h or 00010001b) in B, since 251 = (13 X 18) + 17. Carry and OV will both be cleared.

Bytes: 1

Cycles: 3

Encoding:

1	0	0	0	0	1	0	0

Operation: DIV

$(A)_{15\text{-}8}$, $(B)_{7\text{-}0} \leftarrow (A) / (B)$

DJNZ <byte>, <rel-addr>

Function: Decrement and Jump if Not Zero

Description: DJNZ decrements the location indicated by 1, and branches to the address indicated by the second operand if the resulting value is not zero. An original value of 00h will underflow to 0FFh. No flags are affected. The branch destination would be computed by adding the signed relative-displacement value in the last instruction byte to the PC, after incrementing the PC to the first byte of the following instruction.

The location decremented may be a register or directly addressed byte.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: Internal RAM locations 40h, 50h and 60h contain the values 01h, 70h, and 15h, respectively. The instruction sequence,

```
DJNZ   40H,LABEL_1
DJNZ   50H,LABEL_2
DJNZ   60H,LABEL_3
```

will cause a jump to the instruction at label LABEL_2 with the values 00h, 6Fh, and 15h in the three RAM locations. The first jump was not taken because the result was zero.

This instruction provides a simple way of executing a program loop a given number of times, or for adding a moderate time delay (from 2 to 512 machine cycles) with a single instruction. The instruction sequence,

```
            MOV    R2, #8
TOGGLE:     CPL    P1.7
            DJNZ   R2, TOGGLE
```

will toggle P1.7 eight times, causing four output pukes to appear at bit 7 of output Port 1. Each pulse will last three machine cycles; two for DJNZ and one to alter the pin.

DJNZ Rn, rel

Bytes: 2

Cycles: 3

Encoding:

1	1	0	1	1	r	r	r	relative

Operation:

DJNZ
(PC) ← (PC) + 2
(Rn) ← (Rn) – 1
IF (Rn) > 0 or (Rn) < 0
THEN (PC) ← (PC) + rel

DJNZ direct, rel

Bytes: 3

Cycles: 4

Encoding:

0	1	0	1	0	0	1	1	direct address	relative

Operation:

DJNZ
(PC) ← (PC) + 2
(direct) ← (direct) – 1
IF (direct) > 0 or (direct) < 0
THEN (PC) ← (PC) + rel

INC <byte>

Function: Increment

Description: INC increments the indicated variable by 1. An original value of 0FFh will overflow to 00h. No flags are affected. Three addressing modes are allowed: register, direct, or register-indirect.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: Register 0 contains 7Eh (01111110b). Internal RAM locations 7Eh and 7Fh contain 0FFh and 40H, respectively. The instruction sequence,

```
INC  @R0
INC  R0
INC  @R0
```

will leave register 0 set to 7Fh and internal RAM locations 7Eh and 7Fh holding 00h and 41h, repectively.

INC A

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	1	0	0

Operation: INC
(A) ← (A) + 1

INC Rn

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	1	r	r	r

Operation: INC
(Rn) ← (Rn) + 1

INC direct

Bytes: 2

Cycles: 2

Encoding:

0 0 0 0	0 1 0 1	direct address

Operation: INC
(direct) ← (direct) + 1

INC @Ri

Bytes: 1

Cycles: 1

Encoding:

0 0 0 0	0 1 1 i

Operation: INC
((Ri)) ← ((Ri)) + 1

INC DPTR

Function: Increment Data Pointer

Description: Increment the 16-bit data pointer by 1. A 16-bit increment (modulo 2^{16}) is performed; an overflow of the low-order byte of the data pointer (DPL) from 0FFh to 00h will increment the high-order byte (DPH). No flags are affected.

This is the only 16-bit register which can be incremented.

Example: Registers DPH and DPL contain 12h and 0FEh, respectively. The instruction sequence,

```
INC   DPTR
INC   DPTR
INC   DPTR
```

will change DPH and DPL to 13h and 01h.

Bytes: 1

Cycles: 1

Encoding:

1	0	1	0	0	0	1	1

Operation: INC
(DPTR) ← (DPTR) + 1

JB bit, rel

Function: Jump if Bit set

Description: If the indicated bit is a one, jump to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. The bit tested is not modified. No flags are affected.

Example: The data present at input port 1 is 11001010b. The Accumulator holds 56h (01010110b). The instruction sequence,

```
JB   P1.2, LABEL1
JB   ACC.2, LABEL2
```

will cause program execution to branch to the instruction at label LABEL2.

Bytes: 3

Cycles: 4

Encoding:

0 0 1 0	0 0 0 0	bit address	relative

Operation:

```
JB
(PC) ← (PC) + 3
IF   (bit) = 1
     THEN   (PC) ← (PC) + rel
```

JBC bit, rel

Function: Jump if Bit is set and Clear bit

Description: If the indicated bit is one, branch to the address indicated; otherwise proceed with the next instruction. The bit will not be cleared if it is already a zero. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. No flags are affected.

Note: When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, not the input pin.

Example: The Accumulator holds 56h (01010110b). The instruction sequence,

```
JBC   ACC.3, LABELI
JBC   ACC.2, LABEL2
```

will cause program execution to continue at the instruction identified by the label LABEL2, with the Accumulator modified to 52h (01010010b).

Bytes: 3

Cycles: 4

Encoding:

0 0 0 1	0 0 0 0	bit address	relative

Operation:

```
JBC
(PC) ← (PC) + 3
IF   (bit) = 1
     THEN   (bit) ← 0
            (PC) ← (PC) + rel
```

JC rel

Function: Jump if Carry is set

Description: If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. No flags are affected.

Example: The carry flag is cleared. The instruction sequence,

```
JC LABEL1
CPL C
JC LABEL2
```

will set the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 3

Encoding:

0 1 0 0	0 0 0 0	relative

Operation: JC

(PC) ← (PC) + 2

IF (C) = 1

THEN (PC) ← (PC) + rel

JMP @A + DPTR

Function: Jump indirect

Description: Add the eight-bit unsigned contents of the Accumulator with the sixteen-bit data pointer, and load the resulting sum to the program counter. This will be the address for subsequent instruction fetches. Sixteen-bit addition is performed (modulo 2^{16}): a carry-out from the low-order eight bits propagates through the higher-order bits. Neither the Accumulator nor the Data Pointer is altered. No flags are affected.

Example: An even number from 0 to 6 is in the Accumulator. The following sequence of instructions will branch to one of four AJMP instructions in a jump table starting at JMP_TBL:

```
                MOV     DPTR, #JMP_TBL
                JMP     @A+DPTR
JMP_TBL:        AJMP    LABEL0
                AJMP    LABEL1
                AJMP    LABEL2
                AJMP    LABEL3
```

If the Accumulator equals 04h when starting this sequence, execution will jump to label LABEL2. Remember that AJMP is a two-byte instruction, so the jump instructions start at every other address.

Bytes: 1

Cycles: 2

Encoding:

0	1	1	1	0	0	1	1

Operation: JMP
(PC) ← (A) + (DPTR)

JNB bit, rel

Function: Jump if Bit Not set

Description: If the indicated bit is a zero, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the third instruction byte to the PC, after incrementing the PC to the first byte of the next instruction. The bit tested is not modified. No flags are affected.

Example: The data present at input port 1 is 11001010b. The Accumulator holds 56h (01010110b). The instruction sequence,

```
JNB     P1.3, LABEL1
JNB     ACC.3, LABEL2
```

will cause program execution to continue at the instruction at label LABEL2.

Bytes: 3

Cycles: 4

Encoding:

0 0 1 1	0 0 0 0	bit address	relative

Operation:

```
JNB
(PC) ← (PC) + 3
IF   (bit) = 0
     THEN   (PC) ← (PC) + rel
```

JNC rel

Function: Jump if Carry not set

Description: If the carry flag is a zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice to point to the next instruction. The carry flag is not modified.

Example: The carry flag is set. The instruction sequence,

```
JNC   LABEL1
CPL   C
JNC   LABEL2
```

will clear the carry and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 3

Encoding:

0 1 0 1	0 0 0 0	relative

Operation:

```
JNC
(PC) ← (PC) + 2
IF   (C) = 0
     THEN   (PC) ← (PC) + rel
```

JNZ rel

Function: Jump if Accumulator Not Zero

Description: If any bit of the Accumulator is a one, branch to the indicated address; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally holds 00h. The instruction sequence,

```
JNZ   LABEL1
INC   A
JNZ   LAEEL2
```

will set the Accumulator to 01h and continue at label LABEL2.

Bytes: 2

Cycles: 3

Encoding:

0 1 1 1	0 0 0 0	relative

Operation:

```
JNZ
(PC) ← (PC) + 2
IF   (A) ≠ 0
     THEN   (PC) ← (PC) + rel
```

JZ rel

Function: Jump if Accumulator Zero

Description: If all bits of the Accumulator are zero, branch to the address indicated; otherwise proceed with the next instruction. The branch destination is computed by adding the signed relative-displacement in the second instruction byte to the PC, after incrementing the PC twice. The Accumulator is not modified. No flags are affected.

Example: The Accumulator originally contains 01h. The instruction sequence,

```
JZ    LABEL1
DEC   A
JZ    LABEL2
```

will change the Accumulator to 00h and cause program execution to continue at the instruction identified by the label LABEL2.

Bytes: 2

Cycles: 3

Encoding:

0 1 1 0	0 0 0 0	relative

Operation:

```
JNZ
(PC) ← (PC) + 2
IF   (A) = 0
     THEN   (PC) ← (PC) + rel
```

LCALL addr16

Function: Long call

Description: LCALL calls a subroutine located at the indicated address. The instruction adds three to the program counter to generate the address of the next instruction and then pushes the 16-bit result onto the stack (low byte first), incrementing the Stack Pointer by two. The high-order and low-order bytes of the PC are then loaded, respectively, with the second and third bytes of the LCALL instruction. Program execution continues with the instruction at this address. The subroutine may therefore begin anywhere in the full 64Kbyte program memory address space. No flags are affected.

Example: Initially the Stack Pointer equals 07h. The label “SUBRTN” is assigned to program memory location 1234h. After executing the instruction,

LCALL SUBRTN

at location 0123h, the Stack Pointer will contain 09h, internal RAM locations 08h and 09h will contain 26h and 01h, and the PC will contain 1234h.

Bytes: 3

Cycles: 4

Encoding:

0 0 0 1	0 0 1 0	addr15 ~ addr8	addr7 ~ addr0

Operation:
LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (PC_{15-8})
(PC) ← (PC) + 3
(PC) ← $addr_{15-0}$

LJMP addr16

Function: Long Jump

Description: LJMP causes an unconditional branch to the indicated address, by loading the high-order and low-order bytes of the PC (respectively) with the second and third instruction bytes. The destination may therefore be anywhere in the full 64Kbyte program memory address space. No flags are affected.

Example: The label “JMPADR” is assigned to the instruction at program memory location 1234h. The instruction,

LJMP JMPADR

at location 0123h will load the program counter with 1234h.

Bytes: 3

Cycles: 4

Encoding:

0 0 0 0	0 0 1 0	addr15 ~ addr8	addr7 ~ addr0

Operation: LJMP
(PC) ← $addr_{15-0}$

MOV <dest-byte>, <src-byte>

Function: Move byte variable

Description: The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

This is by far the most flexible operation. Fifteen combinations of source and destination addressing modes are allowed.

Example: Internal RAM location 30h holds 40h. The value of RAM location 40h is 10h. The data present at input port 1 is 11001010b (0CAh).

```
MOV   R0, #30h  ;R0 < = 30h
MOV   A, @R0    ;A < = 40h
MOV   R1, A     ;R1 < = 40h
MOV   B, @R1    ;B < = 10h
MOV   @R1, P1   ;RAM (40h)< = 0CAh
MOV   P2, P1    ;P2   #0CAh
```

leaves the value 30h in register 0, 40h in both the Accumulator and register 1, 10h in register B, and 0Cah (11001010b) both in RAM Iocation 40h and output on port 2.

MOV A, Rn

Bytes: 1

Cycles: 1

Encoding:

1	1	1	0	1	r	r	r

Operation: MOV
(A) ← (Rn)

MOV A, direct

Bytes: 2

Cycles: 2

Encoding:

1	1	1	0	0	1	0	1	direct address

Operation: MOV
(A) ← (direct)

MOV A, ACC is not a valid instruction

MOV A, @Ri

Bytes: 1

Cycles: 1

Encoding:

1	1	1	0	0	1	1	i

Operation: MOV
(A) ← ((Ri))

MOV A, #data

Bytes: 2

Cycles: 2

Encoding:

0	1	1	1	0	1	0	0	immediate data

Operation: MOV
(B) ← #data

MOV Rn, A

Bytes: 1

Cycles: 1

Encoding:

1	1	1	1	1	r	r	r

Operation: MOV
(Rn) ← (A)

MOV Rn, direct

Bytes: 2

Cycles: 2

Encoding:

1	0	1	0	1	r	r	r	direct address

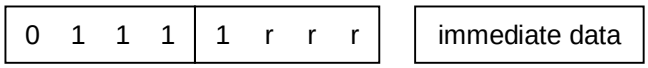
Operation: MOV
(Rn) ← (direct)

MOV Rn, #data

Bytes: 2

Cycles: 2

Encoding:



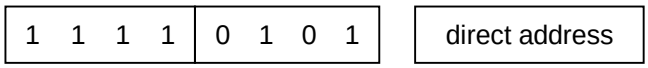
Operation: MOV
(Rn) ← #data

MOV direct, A

Bytes: 2

Cycles: 2

Encoding:



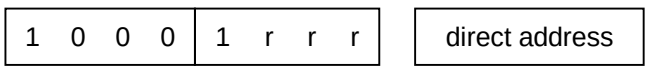
Operation: MOV
(direct) ← (A)

MOV direct, Rn

Bytes: 2

Cycles: 2

Encoding:



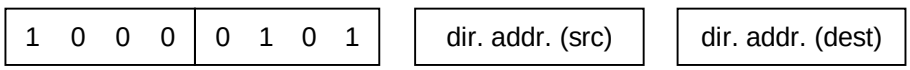
Operation: MOV
(direct) ← (Rn)

MOV direct, direct

Bytes: 3

Cycles: 3

Encoding:



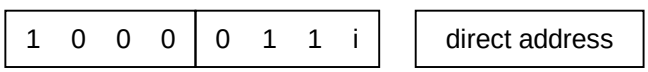
Operation: MOV
(direct) ← (direct)

MOV direct, @Ri

Bytes: 2

Cycles: 2

Encoding:



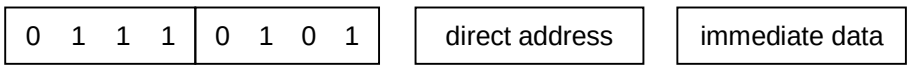
Operation: MOV
(direct) ← ((Ri))

MOV direct, #data

Bytes: 3

Cycles: 3

Encoding:



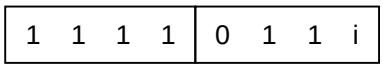
Operation: MOV
(direct) ← #data

MOV @Ri, A

Bytes: 1

Cycles: 1

Encoding:



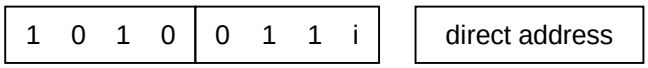
Operation: MOV
((Ri)) ← (A)

MOV @Ri, direct

Bytes: 2

Cycles: 2

Encoding:



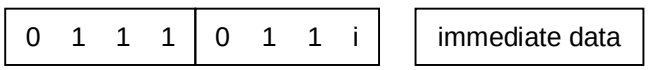
Operation: MOV
((Ri)) ← (direct)

MOV @Ri, #data

Bytes: 2

Cycles: 2

Encoding:



Operation: MOV
((Ri)) ← #data

MOV <dest-bit>, <src-bit>

Function: Move bit data

Description: The Boolean variable indicated by the second operand is copied into the location specified by the first operand. One of the operands must be the carry flag; the other may be any directly addressable bit. No other register or flag is affected.

Example: The carry flag is originally set. The data present at input Port 3 is 11000101b. The data previously written to output Port 1 is 35h (00110101b).

```
MOV  P1.3, C
MOV  C, P3.3
MOV  P1.2, C
```

will leave the carry cleared and change Port 1 to 39h (00111001b).

MOV C, bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 0 1 0	bit address

Operation:
MOV
(C) ← (bit)

MOV bit, C

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 0 1 0	bit address

Operation:
MOV
(bit) ← (C)

MOV DPTR, #data16

Function: Load Data Pointer with a 16-bit constant

Description: The Data Pointer is loaded with the 16-bit constant indicated. The 16-bit constant is loaded into the second and third bytes of the instruction. The second byte (DPH) is the high-order byte, while the third byte (DPL) holds the low-order byte. No flags are affected.

This is the only instruction, which moves 16 bits of data at once.

Example: The instruction,

MOV DPTR, # 1234h

will load the value 1234h into the Data Pointer: DPH will hold 12h and DPL will hold 34h.

Bytes: 3

Cycles: 3

Encoding:

1	0	0	1	0	0	0	0	imm. data 15~8	imm. data 7~0

Operation: MOV

(DPTR) $\leftarrow$ #data$_{15\text{-}0}$

{DPH, DPL} $\leftarrow$ {#data$_{15\text{-}8}$, #data$_{7\text{-}0}$}

MOVC A, @A + <base-reg>

Function: Move Code byte

Description: The MOVC instructions load the Accumulator with a code byte, or constant from program memory. The address of the byte fetched is the sum of the original unsigned eight-bit Accumulator contents and the contents of a sixteen-bit base register, which may be either the Data Pointer or the PC. In the latter case, the PC is incremented to the address of the following instruction before being added with the Accumulator; otherwise the base register is not altered. Sixteen-bit addition is performed so a carry-out from the low-order eight bits may propagate through higher-order bits. No flags are affected.

Example: A value between 0 and 3 is in the Accumulator. The following instructions will translate the value in the Accumulator to one of four values defined by the DB (define byte) directive.

```
REL-PC:   INC    A
          MOVC   A, @A+PC
          RET
          DB     66h
          DB     77h
          DB     88h
          DB     99h
```

If the subroutine is called with the Accumulator equal to 01h, it will return with 77h in the Accumulator. The INC A before the MOVC instruction is needed to “get around” the RET instruction above the table. If several bytes of code separated the MOVC from the table, the corresponding number would be added to the Accumulator instead.

MOVC A, @A + DPTR

Bytes: 1

Cycles: 2

Encoding:

1	0	0	1	0	0	1	1

Operation: MOV
(A) ← ((A) + (DPTR))

MOVC A, @A + PC

Bytes: 1

Cycles: 2

Encoding:

1	0	0	1	0	0	1	1

Operation: MOVC
PC ← PC + 1
(A) ← ((A) + (PC))

Note: It is recommended that all interrupts are disabled before using MOVC instruction.

MOVX <dest-byte>, <src-byte>

Function: Move External

Description: The MOVX instructions transfer data between the Accumulator and a byte of external data memory, hence the "X" appended to MOV. There are two types of instructions, differing in whether they provide an eight-bit or sixteen-bit indirect address to the external data RAM.

In the first type the contents of R0 or R1 in the current register bank provide an eight-bit address multiplexed with data on P0. Eight bits are sufficient for external I/O expansion decoding or for a relatively small RAM array. For somewhat larger arrays, any output port pins can be used to output higher-order address bits. These pins would be controlled by an output instruction preceding the MOVX.

In the second type of MOVX instruction, the Data Pointer generates a sixteen-bit address. P2 outputs the high-order eight address bits (the contents of DPH) while P0 multiplexes the low-order eight bits (DPL) with data. The P2 Special Function Register retains its previous contents while the P2 output buffers are emitting the contents of DPH. This form is faster and more efficient when accessing very large data arrays (up to 64Kbytes), since no additional instructions are needed to set up the output ports.

It is possible in some situations to mix the two MOVX types. A large RAM array with its high-order address lines driven by P2 can be addressed via the Data Pointer, or with code to output high-order address bits to P2 followed by a MOVX instruction using R0 or R1.

Example: An external 256byte RAM using multiplexed address/data lines (e.g., an Intel 8155 RAM / I/O / TIMER) is connected to the 8052 Port 0. Port 3 provides control lines for the external RAM. Ports 1 and 2 are used for normal I/O. Registers 0 and 1 contain 12h and 34h. Location 34h of the external RAM holds the value 56h. The instruction sequence,

```
MOVX   A, @R1
MOVX   @R0, A
```

copies the value 56h into both the Accumulator and external RAM location 12h.

MOVX A, @Ri

Bytes: 1

Cycles: 3

Encoding:

1	1	1	0	0	0	1	i

Operation: MOVX
(A) ← ((Ri))

MOVX A, @DPTR

Bytes: 1

Cycles: 3

Encoding:

1	1	1	0	0	0	0	0

Operation: MOVX
(A) ← ((DPTR))

MOVX @Ri, A

Bytes: 1

Cycles: 3

Encoding:

1	1	1	1	0	0	1	i

Operation: MOVX
((Ri)) ← (A)

MOVC @DPTR, A

Bytes: 1

Cycles: 3

Encoding:

1	1	1	1	0	0	0	0

Operation: MOVX
(DPTR) ← (A)

MUL AB

Function: Multiply

Description: MUL AB multiplies the unsigned eight-bit integers in the Accumulator and register B. The low-order byte of the sixteen-bit product is left in the Accumulator, and the high-order byte in B. If the product is greater than 255 (0FFh) the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.

Example: Originally the Accumulator holds the value 80 (50h). Register B holds the value 160 (0A0h). The instruction,

MUL AB

will give the product 12,800 (3200h), so B is changed to 32h (00110010b) and the Accumulator is cleared. The overflow flag is set, carry is cleared.

Bytes: 1

Cycles: 3

Encoding:

1	0	1	0	0	1	0	0

Operation: MUL

$(A)_{7\text{-}0}$, $(B)_{15\text{-}8} \leftarrow (A) \times (B)$

NOP

Function: No Operation

Description: Execution continues at the following instruction. Other than the PC, no registers or flags are affected.

Example: It is desired to produce a low-going output pulse on bit 7 of Port 2 lasting exactly 5 cycles. A simple SETB/CLR sequence would generate a one-cycle pulse, so four additional cycles must be inserted. This may be done (assuming no interrupts are enabled) with the instruction sequence,

```
CLR     P2.7
NOP
NOP
NOP
NOP
SETB    P2.7
```

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	0	0

Operation: NOP
(PC) ← (PC) + 1

ORL <dest-byte> <src-byte>

Function: Logical-OR for byte variables

Description: ORL performs the bitwise logical-OR operation between the indicated variables, storing the results in the destination byte. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct, register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.

Example: If the Accumulator holds 0C3h (11000011b) and R0 holds 55h (01010101b) then the instruction,

ORL A, R0

will leave the Accumulator holding the value 0D7h (11010111b).

When the destination is a directly addressed byte, the instruction can set combinations of bits in any RAM location or hardware register. The pattern of bits to be set is determined by a mask byte, which may be either a constant data value in the instruction or a variable computed in the Accumulator at run-time. The instruction,

ORL P1, # 00110010b

will set bits 5, 4, and 1 of output Port 1.

ORL A, Rn

Bytes: 1

Cycles: 1

Encoding:

0	1	0	0	1	r	r	r

Operation: ORL
(A) ← (A) ∨ (Rn)

ORL A, direct

Bytes: 2

Cycles: 2

Encoding:

0	1	0	0	0	1	1	i	direct address

Operation: ORL
(A) ← (A) ∨ (direct)

ORL A, @Ri

Bytes: 1

Cycles: 1

Encoding:

0	1	0	0	0	1	1	i

Operation: ORL
(A) ← (A) ∨ ((Ri))

ORL A, #data

Bytes: 2

Cycles: 2

Encoding:

0	1	0	0	0	1	0	0	immediate data

Operation: ORL
(A) ← (A) ∨ #data

ORL direct, A

Bytes: 2

Cycles: 2

Encoding:

0	1	0	0	0	0	1	0	direct address

Operation: ORL
(direct) ← (direct) ∨ (A)

ORL direct, #data

Bytes: 3

Cycles: 3

Encoding:

0	1	0	0	0	0	1	1	direct address	immediate data

Operation: ORL
(direct) ← (direct) ∨ #data

ORL C, <src-bit>

Function: Logical-OR for bit variables

Description: Set the carry flag if the Boolean value is a logical 1; leave the carry in its current state otherwise. A slash (“/”) preceding the operand in the assembly language indicates that the logical complement of the addressed bit is used as the source value, but the source bit itself is not affected. No other flags are affected.

Example: Set the carry flag if and only if P1.0 = 1, ACC. 7 = 1, or OV = 0:

```
MOV     C, P1.0         ;LOAD CARRY WITH INPUT PIN P1.0
ORL     C, ACC.7        ;OR CARRY WITH THE ACC.BIT 7
ORL     C, /OV          ;OR CARRY WITH THE INVERSE OF OV.
```

ORL C, bit

Bytes: 2

Cycles: 2

Encoding:

0 1 1 1	0 0 1 0	bit address

Operation: ORL
(C) ← (C) ∨ (bit)

ORL C, /bit

Bytes: 2

Cycles: 2

Encoding:

1 0 1 0	0 0 0 0	bit address

Operation: ORL
(C) ← (C) ∨ ~(bit)

POP direct

Function: Pop from stack.

Description: The contents of the internal RAM location addressed by the Stack Pointer is read, and the Stack Pointer is decremented by 1. The value read is then transferred to the directly addressed byte indicated. No flags are affected.

Example: The Stack Pointer originally contains the value 32h, and internal RAM locations 30h through 32h contain the values 20h, 23h, and 01h, respectively. The instruction sequence,

```
POP  DPH
POP  DPL
```

will leave the Stack Pointer equal to the value 30H and the Data Pointer set to 0123h. At this point the instruction,

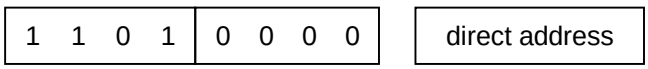
```
POP  SP
```

will leave the Stack Pointer set to 20h. Note that in this special case the Stack Pointer was decremented to 2Fh before being loaded with the value popped (20h).

Bytes: 2

Cycles: 2

Encoding:



Operation:

```
POP
(direct) ← ((SP))
(SP) ← (SP) – 1
```

PUSH direct

Function: Push onto stack

Description: The Stack Pointer is incremented by 1. The contents of the indicated variable is then copied into the internal RAM location addressed by the Stack Pointer. Otherwise no flags are affected.

Example: On entering an interrupt routine the Stack Pointer contains 09h. The Data Pointer holds the value 0123h.The instruction sequence,

```
PUSH DPL
PUSH DPH
```

will leave the Stack Pointer set to 0Bh and store 23h and 01h in internal RAM locations 0Ah and 0Bh, respectively.

Bytes: 2

Cycles: 2

Encoding:

1 1 0 1	0 0 0 0	direct address

Operation:

PUSH
(SP) ← (SP) + 1
((SP)) ← (direct)

RET

Function: Return from subroutine

Description: RET pops the high-and low-order bytes of the PC successively from the stack, decrementing the Stack Pointer by 2. Program execution continues at the resulting address, generally the instruction immediately following an ACALL or LCALL. No flags are affected.

Example: The Stack Pointer originally contains the value 0Bh. Internal RAM locations 0Ah and OBh contain the values 23h and 01h, respectively. The instruction,

RET

will leave the Stack Pointer equal to the value 09h. Program execution will continue at Iocation 0123h.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	0	0	0	1	0

Operation: RET

$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

RETI

Function: Return from interrupt

Description: RETI pops the high-and low-order bytes of the PC successively from the stack, and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. The Stack Pointer is left decremented by two. No other registers are affected; the PSW is not automatically restored to its pre-interrupt status. Program execution continues at the resulting address, which is generally the instruction immediately after the point at which the interrupt request was detected. If a lower-or same-level interrupt had been pending when the RETI instruction is executed, that one instruction will be executed before the pending interrupt is processed.

Example: The Stack Pointer originally contains the value 0Bh. An interrupt was detected during the instruction ending at location 0122h. Internal RAM locations 0Ah and 0Bh contain the values 23h and 01h, respectively. The instruction,

RETI

will leave the Stack Pointer equal to 09h and return program execution to location 0123h.

Bytes: 1

Cycles: 2

Encoding:

0	0	1	1	0	0	1	0

Operation:

RETI
$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

RL A

Function: Rotate Accumulator Left

Description: The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.

Example: The Accumulator holds the value 0C5h (11000101b). The instruction,

RL A

leaves the Accumulator holding the value 8Bh (10001011b) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	0	0	0	1	1

Operation:

RL
$(A_n + 1) \leftarrow (A_n)$ n = 0 ~ 6
(A0) ← (A7)

RLC A

Function: Rotate Accumulator Left through the Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

Example: The Accumulator holds the value 0C5h (11000101b), and the carry is zero. The instruction,

RLC A

leaves the Accumulator holding the value 8Bh (10001010b) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	1	1	0	0	1	1

Operation:

RLC
$(A_n + 1) \leftarrow (A_n)$ n = 0 ~ 6
(A0) ← (C)
(C) ← (A7)

RR A

Function: Rotate Accumulator Right

Description: The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.

Example: The Accumulator holds the value 0C5h (11000101b). The instruction,

RR A

leaves the Accumulator holding the value 0E2h (11100010b) with the carry unaffected.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	0	0	0	1	1

Operation:

RR
$(A_n) \leftarrow (A_n + 1)$ n = 0 ~ 6
(A7) $\leftarrow$ (A0)

RRC A

Function: Rotate Accumulator Right through Carry flag

Description: The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original value of the carry flag moves into the bit 7 position. No other flags are affected.

Example: The Accumulator holds the value 0C5h (11000101b), the carry is zero. The instruction,

RRC A

leaves the Accumulator holding the value 62h (01100010b) with the carry set.

Bytes: 1

Cycles: 1

Encoding:

0	0	0	1	0	0	1	1

Operation:

RRC
$(A_n) \leftarrow (A_n + 1)$ n = 0 ~ 6
(A7) ← (C)
(C) ← (A0)

SETB <bit>

Function: Set Bit

Description: SETB sets the indicated bit to one. SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

Example: The carry flag is cleared. Output Port 1 has been written with the value 34h (00110100b). The instructions,

```
SETB  C
SETB  PI.0
```

will leave the carry flag set to 1 and change the data output on Port 1 to 35h (00110101b).

SETB C

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	0	1	1

Operation:
SETB
(C) ← 1

SETB bit

Bytes: 2

Cycles: 2

Encoding:

1	1	0	1	0	0	1	0	bit address

Operation:
SETB
(bit) ← 1

SJMP rel

Function: Short Jump

Description: Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

Example: The label “RELADR” is assigned to an instruction at program memory location 0123h. The instruction,

SJMP RELADR

will assemble into location 0100h. After the instruction is executed, the PC will contain the value 0123h.

(Note: Under the above conditions the instruction following SJMP will be at 102h. Therefore, the displacement byte of the instruction will be the relative offset (0123h – 0102h) = 21h. Put another way, an SJMP with a displacement of 0FEH would be a one-instruction infinite loop.)

Bytes: 2

Cycles: 3

Encoding:

1 0 0 0	0 0 0 0	relative

Operation: SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

SUBB A, <src-byte>

Function: Subtract with borrow

Description: SUBB subtracts the indicated variable and the carry flag together from the Accumulator, leaving the result in the Accumulator. SUBB sets the carry (borrow) flag if a borrow is needed for bit 7, and clears C otherwise. (If C was set before executing a SUBB instruction, this indicates that a borrow was needed for the previous step in a multiple precision subtraction, so the carry is subtracted from the Accumulator along with the source operand.) AC is set if a borrow is needed for bit 3, and cleared otherwise. OV is set if a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not bit 6.

Example: When subtracting signed integers OV indicates a negative number produced when a negative value is subtracted from a positive value, or a positive result when a positive number is subtracted from a negative number.

The source operand allows four addressing modes: register, direct, register-indirect, or immediate.

The Accumulator holds 0C9h (11001001b), register 2 holds 54h (01010100b), and the carry flag is set. The instruction,

SUBB A, R2

will leave the value 74h (01110100b) in the accumulator, with the carry flag and AC cleared but OV set.

Notice that 0C9h minus 54h is 75h. The difference between this and the above result is due to the carry (borrow) flag being set before the operation. If the state of the carry is not known before starting a single or multiple-precision subtraction, it should be explicitly cleared by a CLR C instruction.

SUBB A, Rn

Bytes: 1

Cycles: 1

Encoding:

1	0	0	1	1	r	r	r

Operation: SUBB
(A) ← (A) – (C) – (Rn)

SUBB A, direct

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 1 0 1	direct address

Operation: SUBB
(A) ← (A) – (C) – (direct)

SUBB A, @Ri

Bytes: 1

Cycles: 1

Encoding:

1 0 0 1	0 1 1 i

Operation: SUBB
(A) ← (A) – (C) – ((Ri))

SUBB A, #data

Bytes: 2

Cycles: 2

Encoding:

1 0 0 1	0 1 0 0	immediate data

Operation: SUBB
(A) ← (A) – (C) – #data

SWAP A

Function: Swap nibbles within the Accumulator

Description: SWAP A interchanges the low- and high-order nibbles (four-bit fields) of the Accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a four-bit rotate instruction. No flags are affected.

Example: The Accumulator holds the value 0C5h (11000101b). The instruction,

SWAP A

leaves the Accumulator holding the value 5Ch (01011100b).

Bytes:

Cycles: 1

Encoding: 1

Operation:

1	1	0	0	0	1	0	0

XCH
$(A_{3-0}) \leftrightarrow (A_{7-4})$

XCH A, <byte>

Function: Exchange Accumulator with byte variable

Description: XCH leads the Accumulator with the contents of the indicated variable, at the same time writing the original Accumulator contents to the indicated variable. The source/destination operand can use register, direct, or register-indirect addressing.

Example: R0 contains the address 20h. The Accumulator holds the value 3Fh (00111111b). Internal RAM location 20h holds the value 75h (01110101b). The instruction,

XCH A, @R0

will leave RAM location 20h holding the values 3Fh (00111111b) and 75h (01110101b) in the accumulator.

XCH A, Rn

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	1	r	r	r

Operation:
XCH
(A) ↔ (Rn)

XCH A, direct

Bytes: 2

Cycles: 2

Encoding:

1	1	0	0	0	1	0	1	direct address

Operation:
XCH
(A) ↔ (direct)

XCH A, @Ri

Bytes: 1

Cycles: 1

Encoding:

1	1	0	0	0	1	1	i

Operation:
XCH
(A) ↔ ((Ri))

XCHD A, @Ri

Function: Exchange Digit

Description: XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), generally representing a hexadecimal or BCD digit, with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. No flags are affected

Example: R0 contains the address 20h. The Accumulator holds the value 36h (00110110b). Internal RAM location 20h holds the value 75h (01110101b). The instruction,

XCHD A, @R0

will leave RAM location 20h holding the value 76h (01110110b) and 35h (00110101b) in the Accumulator.

Bytes: 1

Cycles: 1

Encoding:

1	1	0	1	0	1	1	i

Operation: XCH
$(A_{3-0}) \leftrightarrow ((Rn_{3-0}))$

XRL <dest-byte>, <src-byte>

Function: Logical Exclusive-OR for byte variables

Description: XRL performs the bitwise logical Exclusive-OR operation between the indicated variables, storing the results in the destination. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the Accumulator, the source can use register, direct register-indirect, or immediate addressing; when the destination is a direct address, the source can be the Accumulator or immediate data.

(Note: When this instruction is used to modify an output port, the value used as the original port data will be read from the output data latch, not the input pins.)

Example: If the Accumulator holds 0C3h (11000011b) and register 0 holds 0Aah (10101010b) then the instruction,

XRL A, R0

will leave the Accumulator holding the value 69h (01101001b).

When the destination is a directly addressed byte this instruction can complement combinations of bits in any RAM location or hardware register. The pattern of bits to be complemented is then determined by a mask byte either a constant contained in the instruction or a variable computed in the Accumulator at run-time. The instruction,

XRL PI, #00110001b

will complement bits 5, 4, and 0 of output Port 1.

XRL A, Rn

Bytes: 1

Cycles: 1

Encoding:

0	1	1	0	1	r	r	r

Operation:
XRL
(A) ← (A) ⊗ (Rn)

XRL A, direct

Bytes: 2

Cycles: 2

Encoding:

0	1	1	0	0	1	0	1	direct address

Operation: XRL
(A) ← (A) ⊗ (direct)

XRL A, @Ri

Bytes: 1

Cycles: 1

Encoding:

0	1	1	0	0	1	1	i

Operation: XRL
(A) ← (A) ⊗ ((Ri))

XRL A, #data

Bytes: 2

Cycles: 2

Encoding:

0	1	1	0	0	1	0	0	immediate data

Operation: XRL
(A) ← (A) ⊗ #data

XRL direct, A

Bytes: 2

Cycles: 2

Encoding:

0	1	1	0	0	0	1	0	direct address

Operation: XRL
(direct) ← (direct) ⊗ (A)

XRL direct, #data

Bytes: 3

Cycles: 3

Encoding:

0	1	1	0	0	0	1	1	direct address	immediate data

Operation: XRL
(direct) ← (direct) ⊗ #data

15 Appendix B: SFR description

15.1 Port 0 Register (P0)

Bit No.	7	6	5	4	3	2	1	0	
80h	P0.7 ADC0.7	P0.6 ADC0.6	P0.5 ADC0.5	P0.4 ADC0.4	P0.3 ADC0.3	P0.2 ADC0.2	P0.1 ADC0.1	P0.0 ADC0.0	P0
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

This port functions as a multiplexed address/data bus during external memory access, and as a general purpose I/O port on devices which incorporate internal program memory. During external memory cycles, this port will contain the LSB of the address when ALE is high, and data when ALE is low. When used as a general purpose I/O, this port is open-drain and can select pull-up resistors optionally. If SFR P0SEL is set to “1”, writing a 1 to this port will place it in a high impedance mode, which is necessary if the pin is to be used as an input, and if not, pull-up resistor in port 0 is on. Pull-up resistors are not required when used as a memory interface.

Port 0 can be used for analog inputs. ADC0.x means the analog input x.

15.2 Stack Pointer Register (SP)

Bit No.	7	6	5	4	3	2	1	0	
81h	SP.7	SP.6	SP.5	SP.4	SP.3	SP.2	SP.1	SP.0	SP
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(1)	R/W(1)	R/W(1)	

The stack pointer identifies the location where the stack will begin. The stack pointer is incremented before every PUSH operation. This register defaults to 07H after reset.

15.3 Data Pointer Low Register (DPL)

Bit No.	7	6	5	4	3	2	1	0	
82h	DPL.7	DPL.6	DPL.5	DPL.4	DPL.3	DPL.2	DPL.1	DPL.0	DPL
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register is the low byte of the 16-bit data pointer. DPL and DPH are used to point to non-scratchpad data RAM.

15.4 Data Pointer High Register (DPH)

Bit No.	7	6	5	4	3	2	1	0	
83h	DPH.7	DPH.6	DPH.5	DPH.4	DPH.3	DPH.2	DPH.1	DPH.0	DPH
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register is the high byte of the 16-bit data pointer. DPL and DPH are used to point to non-scratchpad data RAM.

15.5 Power Control Register (PCON)

Bit No.	7	6	5	4	3	2	1	0	
87h	SMOD1	SMOD0	-	POF	GF1	GF0	PD	IDL	PCON
	R/W(0)	R(0)		R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
SMOD1	This bit doubles the serial port baud rate in mode 1, 2, and 3 when set to 1.
SMOD0	0: SCON.7 controls the SM0 function defined for the SCON register.
-	Not implemented, reserved for future use. **Note**: User software should not write 1s to reserved bits. These bits may be used in future products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.
POF	Power Off flag. When power-on, this bit will be set by H/W.
GF1	General purpose flag bit
GF0	General purpose flag bit
PD	Power down (Stop) mode enable. Setting this bit causes the MiDAS2.0 family to go into the POWER DOWN mode. In this mode all the clocks are stopped and program execution is frozen.
IDL	Idle mode enable. Setting this bit causes the MiDAS2.0 family to go into the IDLE mode. In this mode the clocks to the CPU are stopped, so program execution is frozen. But the clock to the peripherals and interrupt blocks is not stopped, and these blocks continue operating.

15.6 Timer 0/1 Control Register (TCON)

Bit No.	7	6	5	4	3	2	1	0	
88h	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	TCON
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
TF1	Timer 1 Overflow Flag. This bit indicates when Timer 1 overflows its maximum count as defined by the current mode. This bit can be cleared by software and is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine. 0: No Timer 1 overflow has been detected. 1: Timer 1 has overflowed its maximum count
TR1	Timer 1 Run Control. This bit enables/disables the operation of Timer 1. Halting this timer will preserve the current count in TH1 and TL1. 0: Timer 1 is halted. 1: Timer 1 is enabled.
TF0	Timer 0 Overflow Flag. This bit indicates when Timer 0 overflows its maximum count as defined by the current mode. This bit can be cleared by software and is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine. 0: No Timer 0 overflow has been detected. 1: Timer 0 has overflowed its maximum count
TR0	Timer 0 Run Control. This bit enables/disables the operation of Timer 0. Halting this timer will preserve the current count in TH0 and TL0. 0: Timer 0 is halted. 1: Timer 0 is enabled.
IE1	Interrupt 1 Edge Detect. This bit is set when an edge/level of the type defined by IT1 is detected. If IT1=1, this bit will remain set until cleared in software or the start of the External Interrupt 1 service routine. If IT1=0, this bit will inversely reflect the state of the /INT1 pin.
IT1	Interrupt 1 Type Select. This bit selects whether the /INT1 pin will detect edge or level triggered interrupts. 0: /INT1 is level triggered. 1: /INT1 is edge triggered.
IE0	Interrupt 0 Edge Detect. This bit is set when an edge/level of the type defined by IT0 is detected. If IT0=1, this bit will remain set until cleared in software or the start of the

	External Interrupt 0 service routine. If IT0=0, this bit will inversely reflect the state of the /INT0 pin.
IT0	Interrupt 0 Type Select. This bit selects whether the /INT0 pin will detect edge or level triggered interrupts. 0: /INT0 is level triggered. 1: /INT0 is edge triggered.

15.7 Timer Mode Control Register (TMOD)

Bit No.	7	6	5	4	3	2	1	0	
89h	GATE	C/T	M1	M0	GATE	C/T	M1	M0	TMOD
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function		
GATE	Timer 1 Gate Control. This bit enables/disables the ability of Timer 1 to increment. 0: Timer 1 will clock when TR1=1, regardless of the state of /INT1. 1: Timer 1 will clock only when TR1=1 and /INT1=1.		
C/T	Timer 1 Counter/Timer Select. 0: Timer 1 is incremented by internal clocks. 1: Timer 1 is incremented by pulses on T1 when TR1 (TCON.6) is 1.		
M1, M0	Timer 1 Mode Select. These bits select the operating mode of Timer 1.		
	M1	M0	Mode
	0	0	Mode 0: 8 bits with 5-bit prescale
	0	1	Mode 1: 16 bits
	1	0	Mode 2: 8 bits with auto-reload
	1	1	Mode 3: Timer 1 is halted, but holds its count.
GATE	Timer 0 Gate Control. This bit enables/disables the ability of Timer 0 to increment. 0: Timer 0 will clock when TR0=1, regardless of the states of /INT0. 1: Timer 0 will clock only when TR0=1 and /INT0=1.		
C/T	Timer 0 Counter/Timer Select. 0: Timer 0 is incremented by internal clocks. 1: Timer 0 is incremented by pulses on T0 when TR0 (TCON.4) is 1.		
M1, M0	Timer 0 Mode Select. These bits select the operating mode of Timer 0. When Timer 0 is in mode 3, TL0 is started/stopped by TR0 and TH0 is started/stopped by TR1. Run control for Timer 1 is then provided via the Timer 1 mode selection.		

	M1	M0	Mode
	0	0	Mode 0: 8 bits with 5-bit prescale
	0	1	Mode 1: 16 bits
	1	0	Mode 2: 8 bits with auto-reload
	1	1	Mode 3: Timer 1 is halted, but holds its count.

15.8 Timer 0 Low Byte Register (TL0)

Bit No.	7	6	5	4	3	2	1	0	
8Ah	TL0.7	TL0.6	TL0.5	TL0.4	TL0.3	TL0.2	TL0.1	TL0.0	TL0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register contains the least significant byte of Timer 0.

15.9 Timer 1 Low Byte Register (TL1)

Bit No.	7	6	5	4	3	2	1	0	
8Bh	TL1.7	TL1.6	TL1.5	TL1.4	TL1.3	TL1.2	TL1.1	TL1.0	TL1
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register contains the least significant byte of Timer 1.

15.10 Timer 0 High Byte Register(TH0)

Bit No.	7	6	5	4	3	2	1	0	
8Ch	TH0.7	TH0.6	TH0.5	TH0.4	TH0.3	TH0.2	TH0.1	TH0.0	TH0
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register contains the most significant byte of Timer 0.

15.11 Timer 1 High Byte Register (TH1)

Bit No.	7	6	5	4	3	2	1	0	
8Dh	TH1.7	TH1.6	TH1.5	TH1.4	TH1.3	TH1.2	TH1.1	TH1.0	TH1
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register contains the most significant byte of Timer 1.

15.12 Clock Control Register (CKCON)

Bit No.	7	6	5	4	3	2	1	0	
8Eh	WD1	WD0	T2M	T1M	T0M	-	U1T2DIS	U0T2DIS	CKCON
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	-	R/W(0)	R/W(0)	

Symbol	Function			
WD1, WD0	Watchdog Timer Mode Select. These bits determine the Watchdog timer time-out period. The timer divides the crystal frequency by a programmable value as shown below. The divider value is expressed in clock (crystal) cycles. Note that the reset time-out is 512 clocks longer than the interrupt, regardless of whether the interrupt is enabled.			
	WD1	WD0	Interrupt Divider	Reset Divider
	0	0	2^{17}	2^{17} + 512
	0	1	2^{20}	2^{20} + 512
	1	0	2^{23}	2^{23} + 512
	1	1	2^{26}	2^{26} + 512
T2M	Timer 2 Clock Select. This bit controls the division of the system clock that drives Timer 2. This bit has no effect when the timer is in baud rate generator or clock output modes. Clearing this bit to 0 maintains 80C52 compatibility. This bit has no effect on instruction cycle timing. 0: Timer 2 uses a divide by 12 of the crystal frequency. 1: Timer 2 uses a divide by 4 of the crystal frequency.			
T1M	Timer 1 Clock Select. This bit controls the division of the system clock that drives Timer 1. Clearing this bit to 0 maintains 80C52 compatibility. This bit has no effect on instruction cycle timing. 0: Timer 1 uses a divide by 12 of the crystal frequency. 1: Timer 1 uses a divide by 4 of the crystal frequency.			
T0M	Timer 0 Clock Select. This bit controls the division of the system clock that drives Timer 0. Clearing this bit to 0 maintains 80C52 compatibility. This bit has no effect on instruction cycle timing. 0: Timer 0 uses a divide by 12 of the crystal frequency.			

	1: Timer 0 uses a divide by 4 of the crystal frequency.
-	Reserved. These bits will read 0
U1T2DIS	This bit disables RCLK/TCLK control of UART1 to generate its baud rate with T1 overflow.
U0T2DIS	This bit disables RCLK/TCLK control of UART0 to generate its baud rate with T1 overflow.

15.13 RING Control Configuration Register (RINGCON)

Bit No.	7	6	5	4	3	2	1	0	
87h	S7	S6	S5	S4	S3	S2	S1	S0	RINGCON
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

S[7:0]: The internal ring oscillator can be tuned.

15.14 Port 1 Register (P1)

Bit No.	7	6	5	4	3	2	1	0	
90h	P1.7 /INT5 ADC1.7	P1.6 INT4 ADC1.6	P1.5 /INT3 ADC1.5	P1.4 INT2 ADC1.4	P1.3 ADC1.3	P1.2 ADC1.2	P1.1 T2EX ADC1.1	P1.0 T2 ADC1.0	P1
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

This register functions as a general purpose I/O port. In addition, all the pins have an alternate function listed below. Each of the functions is controlled by several other SFRs. The associated Port 1 latch bit must contain a logic one before the pin can be used in its alternate function capacity.

Symbol	Function
/INT5	External Interrupt 5. A falling edge on this pin will cause an external interrupt 5 if enabled.
INT4	External Interrupt 4. A rising edge on this pin will cause an external interrupt 4 if enabled.
/INT3	External Interrupt 3. A falling edge on this pin will cause an external interrupt 3 if enabled.
INT2	External Interrupt 2. A rising edge on this pin will cause an external interrupt 2 if enabled.

ADC1.x	ADC Analog Input (8+x).
T2EX	Timer 2 Capture/Reload Trigger. A 1-to-0 transition on this pin will cause the value in the T2 registers to be transferred into the capture registers if enabled by EXEN2 (T2CON.3). When in auto-reload mode, a 1-to-0 transition on this pin will reload the Timer 2 register with the value in RCAP2L and RCAP2H if enabled by EXEN2.
T2	Timer 2 External Input. A 1-to-0 transition on this pin will cause Timer 2 to increment or decrement depending on the timer configuration.

15.15 External Interrupt Flag Register (EXIF)

Bit No.	7	6	5	4	3	2	1	0	
91h	IE5	IE4	IE3	IE2	XT/RG	RGMD	RGSL	BGS	EXIF
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(1)	R(0)	R/W(0)	R/W(1)	

Symbol	Function
IE5	External Interrupt 5 Flag. This bit will be set when a falling edge is detected on /INT5. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.
IE4	External Interrupt 4 Flag. This bit will be set when a rising edge is detected on INT4. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.
IE3	External Interrupt 3 Flag. This bit will be set when a falling edge is detected on /INT3. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.
IE2	External Interrupt 2 Flag. This bit will be set when a rising edge is detected on INT2. This bit must be cleared manually by software. Setting this bit in software will cause an interrupt if enabled.
XT/RG	System clock selection 0: The internal ring oscillator is selected as the clock source. 1: The external clock is selected as the clock source.
RGMD	Ring mode. Generally RGMD is the inversion of XT/RG except when the ring oscillator provides clock during waking-up from the power down mode.
RGSL	1: Even if the crystal clock is selected as the system clock, the ring oscillator is used for waking-up from the power down mode during 65,536 cycles of the crystal clock.

BGS	Band-Gap Select. This bit enables/disables the band-gap reference during Stop mode. Disabling the band-gap reference provides significant power savings in Stop mode, but sacrifices the ability to perform a power fail interrupt or power-fail reset while stopped. The state of this bit will be undefined on devices which do not incorporate a band-gap reference. 0: The band-gap reference is disabled in Stop mode but will function during normal operation. 1: The band-gap reference will operate in Stop mode.

15.16 Low Capture/Compare Register (C0CAP0L)

Bit No.	7	6	5	4	3	2	1	0	
92h	C0CAP0L.7	C0CAP0L.6	C0CAP0L.5	C0CAP0L.4	C0CAP0L.3	C0CAP0L.2	C0CAP0L.1	C0CAP0L.0	C0CAP0L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.17 Low Capture/Compare Register (C0CAP1L)

Bit No.	7	6	5	4	3	2	1	0	
93h	C0CAP1L.7	C0CAP1L.6	C0CAP1L.5	C0CAP1L.4	C0CAP1L.3	C0CAP1L.2	C0CAP1L.1	C0CAP1L.0	C0CAP1L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.18 Low Capture/Compare Register (C0CAP2L)

Bit No.	7	6	5	4	3	2	1	0	
94h	C0CAP2L.7	C0CAP2L.6	C0CAP2L.5	C0CAP2L.4	C0CAP2L.3	C0CAP2L.2	C0CAP2L.1	C0CAP2L.0	C0CAP2L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.19 Low Capture/Compare Register (C0CAP3L)

Bit No.	7	6	5	4	3	2	1	0	
95h	C0CAP3L.7	C0CAP3L.6	C0CAP3L.5	C0CAP3L.4	C0CAP3L.3	C0CAP3L.2	C0CAP3L.1	C0CAP3L.0	C0CAP3L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.20 Low Capture/Compare Register (C0CAP4L)

Bit No.	7	6	5	4	3	2	1	0	
96h	C0CAP4L.7	C0CAP4L.6	C0CAP4L.5	C0CAP4L.4	C0CAP4L.3	C0CAP4L.2	C0CAP4L.1	C0CAP4L.0	C0CAP4L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.21 Low Capture/Compare Register (C0CAP5L)

Bit No.	7	6	5	4	3	2	1	0	
97h	C0CAP5L.7	C0CAP5L.6	C0CAP5L.5	C0CAP5L.4	C0CAP5L.3	C0CAP5L.2	C0CAP5L.1	C0CAP5L.0	C0CAP5L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.22 Serial Port Control Register (SCON)

Bit No.	7	6	5	4	3	2	1	0	
98h	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	SCON
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

<table>
<tr><th>Symbol</th><th colspan="4">Function</th></tr>
<tr><td rowspan="6">SM0, SM1</td><td colspan="4">Serial Port Mode. These bits control the mode of serial port.</td></tr>
<tr><td>SM0</td><td>SM1</td><td>Mode</td><td>Period</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>8-bit shift register (OSC/4)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>8-bit UART (Variable)</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>9-bit UART (OSC/32 or OSC/16)</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>9-bit UART (Variable)</td></tr>
<tr><td>SM2</td><td colspan="4">Automatic Address Recognition. The function of this bit is dependent on the serial port mode.
Mode 1: When set, reception is ignored if invalid stop bit received.
Mode 2/3: When this bit is set, automatic address recognition is enabled in modes 2 and 3. This will prevent the RI bit from being set, and an interrupt being asserted, if the 9th bit received is not 1.</td></tr>
<tr><td>REN</td><td colspan="4">Receive Enable. This bit enables/disables the serial port receiver shift register.
0: serial port reception disabled.
1: serial port receiver enabled (modes 1, 2 and 3). Initiate synchronous reception (mode</td></tr>
</table>

	0).
TB8	9th Transmission Bit State. This bit defines the state of the 9th transmission bit in serial port modes 2 and 3.
RB8	9th Received Bit State. This bit identifies the state of the 9th reception bit of received data in serial port modes 2 and 3. In serial port mode 1, when SM2=0, RB8 is the state of the stop bit. RB8 is not used in mode 0.
TI	Transmitter Interrupt Flag. This bit indicates that data in the serial port buffer has been completely shifted out. In serial port mode 0, TI is set at the end of the 8th data bit. In all other modes, this bit is set at the end of the last data bit. This bit must be manually cleared by software.
RI	Receiver Interrupt Flag. This bit indicates that a byte of data has been received in the serial port buffer. In serial port mode 0, RI is set at the end of the 8th bit. In serial port mode 1, RI is set after the last sample of the incoming stop bit subject to the state of SM2. In modes 2 and 3, RI is set after the last sample of RB8. This bit must be manually cleared by software.

15.23 Serial Data Buffer Register (SBUF)

Bit No.	7	6	5	4	3	2	1	0	
99h	SBUF.7	SBUF.6	SBUF.5	SBUF.4	SBUF.3	SBUF.2	SBUF.1	SBUF.0	SBUF
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Data for serial port is read from or written to this location. The serial transmit and receive buffers are separate registers, but both are addressed at the same location.

15.24 High Capture/Compare Register (C0CAP0H)

Bit No.	7	6	5	4	3	2	1	0	
9Ah	C0CAP0H.7	C0CAP0H.6	C0CAP0H.5	C0CAP0H.4	C0CAP0H.3	C0CAP0H.2	C0CAP0H.1	C0CAP0H.0	C0CAP0H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.25 High Capture/Compare Register (C0CAP1H)

Bit No.	7	6	5	4	3	2	1	0	
9Bh	C0CAP1H.7	C0CAP1H.6	C0CAP1H.5	C0CAP1H.4	C0CAP1H.3	C0CAP1H.2	C0CAP1H.1	C0CAP1H.0	C0CAP1H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.26 High Capture/Compare Register (C0CAP2H)

Bit No.	7	6	5	4	3	2	1	0	
9Ch	C0CAP2H.7	C0CAP2H.6	C0CAP2H.5	C0CAP2H.4	C0CAP2H.3	C0CAP2H.2	C0CAP2H.1	C0CAP2H.0	C0CAP2H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.27 High Capture/Compare Register (C0CAP3H)

Bit No.	7	6	5	4	3	2	1	0	
9Dh	C0CAP3H.7	C0CAP3H.6	C0CAP3H.5	C0CAP3H.4	C0CAP3H.3	C0CAP3H.2	C0CAP3H.1	C0CAP3H.0	C0CAP3H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.28 High Capture/Compare Register (C0CAP4H)

Bit No.	7	6	5	4	3	2	1	0	
9Eh	C0CAP4H.7	C0CAP4H.6	C0CAP4H.5	C0CAP4H.4	C0CAP4H.3	C0CAP4H.2	C0CAP4H.1	C0CAP4H.0	C0CAP4H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.29 High Capture/Compare Register (C0CAP5H)

Bit No.	7	6	5	4	3	2	1	0	
9Fh	C0CAP5H.7	C0CAP5H.6	C0CAP5H.5	C0CAP5H.4	C0CAP5H.3	C0CAP5H.2	C0CAP5H.1	C0CAP5H.0	C0CAP5H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.30 Port 2 Register (P2)

Bit No.	7	6	5	4	3	2	1	0	
A0h	P2.7 ADC2.7	P2.6 ADC2.6	P2.5 ADC2.5	P2.4 ADC2.4	P2.3 ADC2.3	P2.2 ADC2.2	P2.1 ADC2.1	P2.0 ADC2.0	P2
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

This port functions as an address bus during external memory access, and as a general purpose I/O port on devices which incorporate internal program memory. During external memory cycles, this port will contain the MSB of the address. Port 2 is also used for analog inputs. ADC2.x means the analog input (16+x).

15.31 Serial Data Buffer Register of UART1 (SBUF1)

Bit No.	7	6	5	4	3	2	1	0	
A1h	SBUF1.7	SBUF1.6	SBUF1.5	SBUF1.4	SBUF1.3	SBUF1.2	SBUF1.1	SBUF1.0	SBUF1
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.32 Mode Control register (C0CAPM0)

Bit No.	7	6	5	4	3	2	1	0	
A2h	IPWM0	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	COCAPM0
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.33 Mode Control register (C0CAPM1)

Bit No.	7	6	5	4	3	2	1	0	
A3h	IPWM1	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	COCAPM1
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.34 Mode Control register (C0CAPM2)

Bit No.	7	6	5	4	3	2	1	0	
A4h	IPWM2	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	COCAPM2
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.35 Mode Control register (C0CAPM3)

Bit No.	7	6	5	4	3	2	1	0	
A5h	IPWM3	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	COCAPM3
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.36 Mode Control register (C0CAPM4)

Bit No.	7	6	5	4	3	2	1	0	
A6h	IPWM4	ECOM4	CAPP4	CAPN4	MAT4	TOG4	PWM4	ECCF4	COCAPM4
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.37 Mode Control register (C0CAPM5)

Bit No.	7	6	5	4	3	2	1	0	
A7h	IPWM5	ECOM5	CAPP5	CAPN5	MAT5	TOG5	PWM5	ECCF5	COCAPM5
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.38 Interrupt Enable Register (IE)

Bit No.	7	6	5	4	3	2	1	0	
A8h	EA	EADC	ET2	ES	ET1	EX1	ET0	EX0	IE
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
EA	Global Interrupt Enable. This bit controls the global masking of all interrupts except Power-Fail interrupt, which is enabled by the EPFI bit (WDCON.5). 0: Disable all interrupt sources. This bit overrides individual interrupt mask settings. 1: Enable all individual interrupt masks. Individual interrupts will occur if enabled.
EADC	Enable ADC Interrupt. This bit controls the masking of the ADC interrupt. 0: Disable all ADC interrupts. 1: Enable interrupt requests generated by the ADCF flag (ADCON.4)
ET2	Enable Timer 2 Interrupt. This bit controls the masking of the Timer 2 interrupt.

	0: Disable all Timer 2 interrupts. 1: Enable interrupt requests generated by the TF2 flag (T2CON.7)
ES0	Enable Serial port Interrupt. This bit controls the masking of the serial port interrupt. 0: Disable all serial port interrupts. 1: Enable interrupt requests generated by the RI (SCON.0) or TI (SCON.1) flags
ET1	Enable Timer 1 Interrupt. This bit controls the masking of the Timer 1 interrupt. 0: Disable all Timer 1 interrupts. 1: Enable interrupt requests generated by the TF1 flag (TCON.7).
EX1	Enable External Interrupt 1. This bit controls the masking of external interrupt 1. 0: Disable external interrupt 1. 1: Enable interrupt requests generated by the /INT1 pin.
ET0	Enable Timer 0 Interrupt. This bit controls the masking of the Timer 0 interrupt. 0: Disable all Timer 0 interrupts. 1: Enable interrupt requests generated by the TF0 flag (TCON.5).
EX0	Enable External Interrupt 0. This bit controls the masking of external interrupt 0. 0: Disable external interrupt 0. 1: Enable interrupt requests generated by the /INT0 pin.

15.39 Slave Address Register (SADDR)

Bit No.	7	6	5	4	3	2	1	0	
A9h	SADDR.7	SADDR.6	SADDR.5	SADDR.4	SADDR.3	SADDR.2	SADDR.1	SADDR.0	SADDR
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

The register is programmed with the given or broadcast address assigned to serial port.

15.40 Slave Address Register of UART1 (SADDR1)

Bit No.	7	6	5	4	3	2	1	0	
AAh	SADDR1.7	SADDR1.6	SADDR1.5	SADDR1.4	SADDR1.3	SADDR1.2	SADDR1.1	SADDR1.0	SADDR1
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.41 Slave Address Mask Enable Register of UART1 (SADEN1)

Bit No.	7	6	5	4	3	2	1	0	
ABh	SADEN1.7	SADEN1.6	SADEN1.5	SADEN1.4	SADEN1.3	SADEN1.2	SADEN1.1	SADEN1.0	SADEN1
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.42 PCA0 Counter Control Register(C0CON)

Bit No.	7	6	5	4	3	2	1	0	
ACh	CF	CR	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0	COCON
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.43 PCA0 Counter Mode Register(C0MOD)

Bit No.	7	6	5	4	3	2	1	0	
ADh	CIDL	PWMDYN	PWM16	CPS3	CPS2	CPS1	CPS0	ECF	COMOD
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.44 Low Byte Register of PCA0 Counter (C0L)

Bit No.	7	6	5	4	3	2	1	0	
AEh	C0L.7	C0L.6	C0L.5	C0L.4	C0L.3	C0L.2	C0L.1	C0L.0	C0L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.45 High Byte Register of PCA0 Counter (C0H)

Bit No.	7	6	5	4	3	2	1	0	
AFh	C0H.7	C0H.6	C0H.5	C0H.4	C0H.3	C0H.2	C0H.1	C0H.0	C0H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.46 Port 3 Register (P3)

Bit No.	7	6	5	4	3	2	1	0	
B0h	P3.7 /RD ADC3.7	P3.6 /WR ADC3.6	P3.5 T1 ADC3.5	P3.4 T0 ADC3.4	P3.3 /INT1 ADC3.3	P3.2 /INT0 ADC3.2	P3.1 TXD ADC3.1	P3.0 RXD ADC3.0	P3
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

This register functions as a general purpose I/O port. In addition, all the pins have an ***alternate*** function listed below. Each of the functions is controlled by other SFRs. The associated Port 3 latch bit must contain in a logic one before the pin can be used in its ***alternate*** function capacity.

Symbol	Function
/RD	External Data Memory Read Strobe. This pin provides an active low read strobe to an external memory device.
/WR	External Data Memory Write Strobe. This pin provides an active low write strobe to an external memory device.
T1	Timer/Counter 1 External Input. A 1-to-0 transition on this pin will increment Timer 1.
T0	Timer/Counter 0 External Input. A 1-to-0 transition on this pin will increment Timer 0.
/INT1	External Interrupt 1. A falling edge/low level on this pin will cause an external interrupt 1 if enabled.
/INT0	External Interrupt 0. A falling edge/low level on this pin will cause an external interrupt 0 if enabled.
TXD	Serial Port Transmit. This pin transmits the serial port data in serial port modes 1,2,3 and emits the synchronizing clock in serial port mode 0.
RXD	Serial Port Receive. This pin receives the serial port data in serial port modes 1,2,3 and is a bi-directional data transfer pin in serial port mode 0.
ADC3.x	Analog input (24+x)

15.47 Serial Port Control Register of UART1 (SCON1)

Bit No.	7	6	5	4	3	2	1	0	
B1h	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	SCON1
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.48 Interrupt Type Selection Register (IT)

Bit No.	7	6	5	4	3	2	1	0	
B2h	EIC2	FIC2	PIC2	IC2_EN	IT5	IT4	IT3	IT2	IT
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
EI2C	I2C Interrupt Enable Flag
FI2C	I2C Interrupt Flag
PI2C	I2C Interrupt Priority
ITx	External Interrupt x Type selection flag 0: Level detect 1: Edge detect

15.49 Port 0 Type Register (P0TYPE)

Bit No.	7	6	5	4	3	2	1	0	
B3h	P0TYPE.7	P0TYPE.6	P0TYPE.5	P0TYPE.4	P0TYPE.3	P0TYPE.2	P0TYPE.1	P0TYPE.0	P0TYPE
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: Push-pull Output 1: Open-drain Output (Default)

15.50 Port 1 Type Register (P1TYPE)

Bit No.	7	6	5	4	3	2	1	0	
B4h	P1TYPE.7	P1TYPE.6	P1TYPE.5	P1TYPE.4	P1TYPE.3	P1TYPE.2	P1TYPE.1	P1TYPE.0	P1TYPE
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: Push-pull Output 1: Quasi-bidirectional Output (Default)

15.51 Port 2 Type Register (P2TYPE)

Bit No.	7	6	5	4	3	2	1	0	
B5h	P2TYPE.7	P2TYPE.6	P2TYPE.5	P2TYPE.4	P2TYPE.3	P2TYPE.2	P2TYPE.1	P2TYPE.0	P2TYPE
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: Push-pull Output 1: Quasi-bidirectional Output (Default)

15.52 Port 3 Type Register (P3TYPE)

Bit No.	7	6	5	4	3	2	1	0	
B5h	P3TYPE.7	P3TYPE.6	P3TYPE.5	P3TYPE.4	P3TYPE.3	P3TYPE.2	P3TYPE.1	P3TYPE.0	P3TYPE
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: Push-pull Output 1: Quasi-bidirectional Output (Default)

15.53 Interrupt Priority High (IPH) & Interrupt Priority Register (IP)

Bit No.	7	6	5	4	3	2	1	0	
B7h	-	PADCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	IPH
	R(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Bit No.	7	6	5	4	3	2	1	0	
B8h	-	PADC	PT2	PS	PT1	PX1	PT0	PX0	IP
	R(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

<table>
<tr><th>Symbol</th><th colspan="3">Function</th></tr>
<tr><td>-</td><td colspan="3">Reserved.</td></tr>
<tr><td rowspan="6">PADCH
PADC</td><td colspan="3">ADC Interrupt Priority. These bits control the priority of the ADC interrupt up to four levels.</td></tr>
<tr><td>PADCH</td><td>PADC</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>Priority level 0</td></tr>
<tr><td>0</td><td>1</td><td>Priority level 1</td></tr>
<tr><td>1</td><td>0</td><td>Priority level 2</td></tr>
<tr><td>1</td><td>1</td><td>Priority level 3</td></tr>
<tr><td rowspan="6">PT2H
PT2</td><td colspan="3">Timer 2 Interrupt Priority. These bits control the priority of the Timer 2 interrupt up to four levels.</td></tr>
<tr><td>PT2H</td><td>PT2</td><td>Description</td></tr>
<tr><td>0</td><td>0</td><td>Priority level 0</td></tr>
<tr><td>0</td><td>1</td><td>Priority level 1</td></tr>
<tr><td>1</td><td>0</td><td>Priority level 2</td></tr>
<tr><td>1</td><td>1</td><td>Priority level 3</td></tr>
<tr><td>PSH
PS</td><td colspan="3">Serial port Interrupt Priority. These bits control the priority of the serial port interrupt up to four levels.</td></tr>
</table>

	PSH	PS	Description
	0	0	Priority level 0
	0	1	Priority level 1
	1	0	Priority level 2
	1	1	Priority level 3
PT1H PT1	Timer 1 Interrupt Priority. These bits control the priority of the Timer 1 interrupt up to four levels.		
	PT1H	PT1	Description
	0	0	Priority level 0
	0	1	Priority level 1
	1	0	Priority level 2
	1	1	Priority level 3
PX1H PX1	External Interrupt 1 Priority. These bits control the priority of the external interrupt 1 up to four levels.		
	PX1H	PX1	Description
	0	0	Priority level 0
	0	1	Priority level 1
	1	0	Priority level 2
	1	1	Priority level 3
PT0H PT0	Timer 0 Interrupt Priority. These bits control the priority of the Timer 0 interrupt up to four levels.		
	PT0H	PT0	Description
	0	0	Priority level 0
	0	1	Priority level 1
	1	0	Priority level 2
	1	1	Priority level 3
PX0H PX0	External Interrupt 0 Priority. These bits control the priority of the external interrupt 0 up to four levels.		
	PX0H	PX0	Description
	0	0	Priority level 0
	0	1	Priority level 1
	1	0	Priority level 2
	1	1	Priority level 3

15.54 Slave Address Mask Enable Register (SADEN)

Bit No.	7	6	5	4	3	2	1	0	
B9h	SADEN.7	SADEN.6	SADEN.5	SADEN.4	SADEN.3	SADEN.2	SADEN.1	SADEN.0	SADEN
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register functions as a mask when comparing serial port addresses for automatic address recognition. When a bit in this register is set, the corresponding bit location in the SADDR will be exactly compared with the incoming serial port data to determine if a receiver interrupt should be generated. When a bit in this register is cleared, the corresponding bit in the SADDR becomes a don't care and is not compared against the incoming data. All incoming data will generate a receiver interrupt when this register is cleared.

15.55 Interrupt Polarity Selection Register (ITSEL)

Bit No.	7	6	5	4	3	2	1	0	
BAh	-	-	ITSEL.5	ITSEL.4	ITSEL.3	ITSEL.2	ITSEL.1	ITSEL.0	ITSEL
			R/W(0)	R/W(1)	R/W(0)	R/W(1)	R/W(0)	R/W(0)	

Symbol	Function
ITSEL.x	External interrupt x polarity selection flag 0: Low level or negative edge triggered 1: High level or positive edge triggered

15.56 Port 0 Input/Output Control Register (P0DIR)

Bit No.	7	6	5	4	3	2	1	0	
BBh	P0DIR.7	P0DIR.6	P0DIR.5	P0DIR.4	P0DIR.3	P0DIR.2	P0DIR.1	P0DIR.0	P0DIR
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

1: Input (default) 0: Output

15.57 Port 1 Input/Output Control Register (P1DIR)

Bit No.	7	6	5	4	3	2	1	0	
BCh	P1DIR.7	P1DIR.6	P1DIR.5	P1DIR.4	P1DIR.3	P1DIR.2	P1DIR.1	P1DIR.0	P1DIR
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

1: Input (default) 0: Output

15.58 Port 2 Input/Output Control Register (P2DIR)

Bit No.	7	6	5	4	3	2	1	0	
BDh	P2DIR.7	P2DIR.6	P2DIR.5	P2DIR.4	P2DIR.3	P2DIR.2	P2DIR.1	P2DIR.0	P2DIR
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

1: Input (default) 0: Output

15.59 Port 3 Input/Output Control Register (P3DIR)

Bit No.	7	6	5	4	3	2	1	0	
BEh	P3DIR.7	P3DIR.6	P3DIR.5	P3DIR.4	P3DIR.3	P3DIR.2	P3DIR.1	P3DIR.0	P3DIR
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

1: Input (default) 0: Output

15.60 High Address Register for MOVX with Ri (AUXAD)

Bit No.	7	6	5	4	3	2	1	0	
BFh	AUXAD.7	AUXAD.6	AUXAD.5	AUXAD.4	AUXAD.3	AUXAD.2	AUXAD.1	AUXAD.0	AUXAD
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.61 PLL Control Register (PLLCON)

Bit No.	7	6	5	4	3	2	1	0	
C1h	Lock	-	icp1	icp0	Dly_ctr	Ph_sel	PLLPD	PLLBP	PLLCON
	R(0)		R/W(0)	R/W(1)	R/W(1)	R/W(0)	R/W(1)	R/W(0)	

Symbol	Function
PLLBP	1: PLL Bypass Mode 0: PLL Normal Mode
PLLPD	1: PLL Power Down 0: PLL Active
Ph_sel	PFD phase control
Dly_ctr	PFD delay control
icp[1:0]	CP current control
LOCK	1: PLL Lock 0: PLL Unlock

15.62 PLL Input Divider Register (PLLNR)

Bit No.	7	6	5	4	3	2	1	0	
C2h	-	-	-	-	Odiv1	Odiv0	Rdiv1	Rdiv0	PLLNR
					R/W(1)	R/W(0)	R/W(1)	R/W(0)	

Symbol	Function
Rdiv[1:0]	Input 2-bit divider
Odiv[1:0]	Output 2-bit divider

15.63 PLL Feedback Divider Register (PLLFR)

Bit No.	7	6	5	4	3	2	1	0	
C3h	F7	F6	F5	F4	F3	F2	F1	F0	PLLFR
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
F[1:0]	Feedback 8-bit divider

15.64 Power Management Register (PMR)

Bit No.	7	6	5	4	3	2	1	0	
C4h	-	-	-	-	XTOFF	ALEOFF	WCLKE	WIOE	PMR
	-	-	-	-	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
-	Reserved.
XTOFF	1: External crystal oscillator disable. 0: External crystal oscillator restart (default)
ALEOFF	ALE Disable. This bit disables the expression of the ALE signal on the device pin during all on-board program and data memory accesses. External memory accesses will automatically enable ALE independent of ALEOFF. 0: ALE expression is enabled. 1: ALE expression is disabled.
WCLKE	1: Wake-up CLK enable 0: Wake-up CLK disable (default)
WIOE	1: Wake-up IO enable 0: Wake-up IO disable (default)

15.65 Status Register (STATUS)

Bit No.	7	6	5	4	3	2	1	0	
C5h	-	-	-	XTUP	-	-	-	-	STATUS
	-	-	-	R(1)	-	-	-	-	

Symbol	Function
-	Reserved.
XTUP	Crystal Oscillator Warm-up Status. This bit indicates whether the CPU crystal oscillator has completed the 65,536 cycle warm-up and is ready to operate from the external crystal or oscillator. This bit is cleared by H/W when Power-on Reset, during Power Down wake-up. This bit is set to 1 following a XTAL stabilization by H/W.

15.66 Internal Ring Oscillator Control Register (OSCICN)

Bit No.	7	6	5	4	3	2	1	0	
C6h	-	-	-	-	DIV2	RINGON	DIV1	DIV0	OSCICN
	-	-	-	-	R/W(0)	R/W(1)	R/W(0)	R/W(0)	

Symbol	Function
-	Reserved.
DIV[2:0]	Ring Oscillator divider. [0,0,0] = 4MHz [0,0,1] = 2MHz [0,1,0] = 1MHz [0,1,1] = 0.5MHz [1,0,0] = 12MHz [1,0,1] = 6MHz [1,1,0] = 3MHz [1,1,1] = 1.5MHz
RINGON	1 = Internal ring Oscillator is running. 0 = Internal ring Oscillator is killed. Don't clear RINGON bit when XTRG = 0.

15.67 I/O Configuration Register (IOCFG)

Bit No.	7	6	5	4	3	2	1	0	
C7h	-	-	-	-	ENAUX	-	-	-	IOCFG
	-	-	-	-	R/W(0)	-	-	-	

15.68 Timer 2 Control Register (T2CON)

Bit No.	7	6	5	4	3	2	1	0	
C8h	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	T2CON
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function			
TF2	Timer 2 Overflow Flag. This flag will be set when Timer 2 overflow from FFFFh to 0000h or the counter equal to the capture register in down count mode. It must be cleared by software. TF2 will only be set if RCLK and TCLK are both cleared to 0.			
EXF2	Timer 2 External Flag. A negative transition on the T2EX pin (P1.1) or Timer 2 underflow/overflow will cause this flag to set based on the CP/RL2 (T2CON.0), EXEN2 (T2CON.3), and DCEN (T2MOD.0) bits. If set by a negative transition, this flag must be cleared to 0 by software. Setting this bit in software or detection of a negative transition on the T2EX pin will force a timer 2 interrupt if enabled.			
	CP/RL2	EXEN2	DCEN	Result
	1	0	X	Negative transition on P1.1 will not affect EXF2 bit.

	1	1	X	Negative transition on P1.1 will set EXF2 bit.
	0	0	0	Negative transition on P1.1 will not affect EXF2 bit.
	0	1	0	Negative transition on P1.1 will set EXF2 bit.
	0	X	1	EXF2 toggles whenever Timer 2 underflow/overflows and can be used as a 17th bit of resolution. In this mode, EXF2 will not cause an interrupt.
RCLK	Receive Clock Flag. This bit determines the serial port timebase when receiving data in serial modes 1 or 3. 0: Timer 1 overflow is used to determine receiver baud rate for serial port. 1: Timer 2 overflow is used to determine receiver baud rate for serial port. Setting this bit will force Timer 2 into baud rate generation mode. The timer will operate from a divide by 2 of the external clock.			
TCLK	Transmit Clock Flag. This bit determines the serial port timebase when transmitting data in serial modes 1 or 3. 0: Timer 1 overflow is used to determine transmitter baud rate for serial port. 1: Timer 2 overflow is used to determine transmitter baud rate for serial port. Setting this bit will force Timer 2 into baud rate generation mode. The timer will operate from a divide by 2 of the external clock.			
EXEN2	Timer 2 External Enable. This bit enables the capture/reload function on the T2EX pin if Timer 2 is not generating baud rates for the serial port. 0: Timer 2 will ignore all external events at T2EX. 1: Timer 2 will capture or reload a value if a negative transition is detected on the T2EX pin.			
TR2	Timer 2 Run Control. This bit enables/disables the operation of Timer 2. Halting this timer will preserve the current count in TH2 and TL2. 0: Timer 2 is halted. 1: Timer 2 is enabled.			
C/T2	Counter/Timer Select. This bit determines whether Timer 2 will function as a timer or counter. Independent of this bit, Timer 2 runs at 2 clocks per tick when used in either baud rate generator or clock output mode. 0: Timer 2 functions as a timer. The speed of Timer 2 is determined by the T2M bit (CKCON.5). 1: Timer 2 will count negative transitions on the T2 pin (P1.0).			

CP/RL2	Capture/Reload Select. This bit determines whether the capture or reload function will be used for Timer 2. If either RCLK or TCLK is set, this bit will not function and the timer will function in an auto-reload mode following each overflow. 0: Auto-reloads will occur when Timer 2 overflows or a falling edge is detected on T2EX if EXEN2=1. 1: Timer 2 captures will occur when a falling edge is detected on T2EX if EXEN2=1.

15.69 Timer 2 Mode Register (T2MOD)

Bit No.	7	6	5	4	3	2	1	0	
C9h	-	-	-	-	-	-	T2OE	DCEN	T2MOD
	-	-	-	-	-	-	R/W(0)	R/W(0)	

Symbol	Function
-	Reserved. Read data will be indeterminate.
T2OE	Timer 2 Output Enable. This bit enables/disables the clock output function of the T2 pin (P1.0). 0: The T2 pin functions as either a standard port pin or as a counter input for Timer 2. 1: Timer 2 will drive the T2 pin with a clock output if C/T2=0. Also, Timer 2 roll-overs will not cause interrupts.
DCEN	Down Count Enable. This bit, in conjunction with the T2EX pin, controls the direction that Timer 2 counts in 16-bit auto-reload mode.

15.70 Timer 2 Capture/Reload Low Byte Register (RCAP2L)

Bit No.	7	6	5	4	3	2	1	0	
CAh	RCAP2L.7	RCAP2L.6	RCAP2L.5	RCAP2L.4	RCAP2L.3	RCAP2L.2	RCAP2L.1	RCAP2L.0	RCAP2L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register is used to capture the TL2 value when Timer 2 is configured in capture mode. RCAP2L is also used as the low byte of a 16-bit reload value when Timer 2 is configured in auto-reload mode.

15.71 Timer 2 Capture/Reload High Byte Register (RCAP2H)

Bit No.	7	6	5	4	3	2	1	0	
CBh	RCAP2H.7	RCAP2H.6	RCAP2H.5	RCAP2H.4	RCAP2H.3	RCAP2H.2	RCAP2H.1	RCAP2H.0	RCAP2H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register is used to capture the TH2 value when Timer 2 is configured in capture mode. RCAP2H is also used as the high byte of a 16-bit reload value when Timer 2 is configured in auto-reload mode.

15.72 Timer 2 Low Byte Register(TL2)

Bit No.	7	6	5	4	3	2	1	0	
CCh	TL2.7	TL2.6	TL2.5	TL2.4	TL2.3	TL2.2	TL2.1	TL2.0	TL2
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register contains the low byte of Timer 2.

15.73 Timer 2 High Byte Register (TH2)

Bit No.	7	6	5	4	3	2	1	0	
CDh	TH2.7	TH2.6	TH2.5	TH2.4	TH2.3	TH2.2	TH2.1	TH2.0	TH2
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register contains the high byte of Timer 2.

15.74 PCA1 Counter Control Register (C1CON)

Bit No.	7	6	5	4	3	2	1	0	
CEh	CF	CR	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0	C1CON
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.75 PCA1 Counter Counter Mode Register (C1MOD)

Bit No.	7	6	5	4	3	2	1	0	
CFh	CIDL	PWMDYN	PWM16	CPS3	CPS2	CPS1	CPS0	ECF	C1MOD
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.76 Program Status Word Register (PSW)

Bit No.	7	6	5	4	3	2	1	0	
D0h	CY	AC	F0	RS1	RS0	OV	F1	P	PSW
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R(0)	

Symbol	Function			
CY	Carry Flag. This bit is set when if the last arithmetic operation resulted in a carry (during addition) or a borrow (during subtraction). Otherwise it is cleared to 0 by all arithmetic operations.			
AC	Auxiliary Carry Flag. This bit is set to 1 if the last arithmetic operation resulted in a carry into (during addition), or a borrow (during subtraction) from the high order nibble. Otherwise it is cleared to 0 by all arithmetic operations.			
F0	User Flag 0. This is a general purpose flag for software control.			
RS1, RS0	Register Bank Select. These bits select which register bank is addressed during register accesses.			
	RS1	RS0	Register Bank	Address
	0	0	0	00H ~ 07H
	0	1	1	08H ~ 0FH
	1	0	2	10H ~ 17H
	1	1	3	18H ~ 1FH
OV	Overflow Flag. This bit is set to 1 if the last arithmetic operation resulted in a carry (addition), borrow (subtraction), or overflow (multiply or divide). Otherwise it is cleared to 0 by all arithmetic operations.			
F1	User Flag 1. This is a general purpose flag for software control.			
P	Parity Flag. This bit is set to 1 if the modulo-2 sum of the eight bits of the accumulator is 1 (odd parity); and cleared to 0 on even parity.			

15.77 Port 0 Pull-up Control Register (P0SEL)

Bit No.	7	6	5	4	3	2	1	0	
D1h	P0SEL.7	P0SEL.6	P0SEL.5	P0SEL.4	P0SEL.3	P0SEL.2	P0SEL.1	P0SEL.0	P0SEL
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: Pull-up resistor ON 1: Pull-up resistor OFF (Default)

15.78 Low Capture/Compare Register (C1CAP0L)

Bit No.	7	6	5	4	3	2	1	0	
D2h	C1CAP0L.7	C1CAP0L.6	C1CAP0L.5	C1CAP0L.4	C1CAP0L.3	C1CAP0L.2	C1CAP0L.1	C1CAP0L.0	C1CAP0L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.79 Low Capture/Compare Register (C1CAP1L)

Bit No.	7	6	5	4	3	2	1	0	
D3h	C1CAP1L.7	C1CAP1L.6	C1CAP1L.5	C1CAP1L.4	C1CAP1L.3	C1CAP1L.2	C1CAP1L.1	C1CAP1L.0	C1CAP1L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.80 Low Capture/Compare Register (C1CAP2L)

Bit No.	7	6	5	4	3	2	1	0	
D4h	C1CAP2L.7	C1CAP2L.6	C1CAP2L.5	C1CAP2L.4	C1CAP2L.3	C1CAP2L.2	C1CAP2L.1	C1CAP2L.0	C1CAP2L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.81 Low Capture/Compare Register (C1CAP3L)

Bit No.	7	6	5	4	3	2	1	0	
D5h	C1CAP3L.7	C1CAP3L.6	C1CAP3L.5	C1CAP3L.4	C1CAP3L.3	C1CAP3L.2	C1CAP3L.1	C1CAP3L.0	C1CAP3L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.82 Low Capture/Compare Register (C1CAP4L)

Bit No.	7	6	5	4	3	2	1	0	
D6h	C1CAP4L.7	C1CAP4L.6	C1CAP4L.5	C1CAP4L.4	C1CAP4L.3	C1CAP4L.2	C1CAP4L.1	C1CAP4L.0	C1CAP4L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.83 Low Capture/Compare Register (C1CAP5L)

Bit No.	7	6	5	4	3	2	1	0	
D7h	C1CAP5L.7	C1CAP5L.6	C1CAP5L.5	C1CAP5L.4	C1CAP5L.3	C1CAP5L.2	C1CAP5L.1	C1CAP5L.0	C1CAP5L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.84 Watchdog Control & Power Status Register (WDCON)

Bit No.	7	6	5	4	3	2	1	0	
D8h	-	POR	EPFI	PFI	WDIF	WTRF	EWT	RWT	WDCON
		R/W(1)	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
-	Reserved.
POR	Power-On-Reset Flag. This bit indicates whether the last reset was a power-on-reset. This bit is set following a power-on-reset and unaffected by all other resets. 0: Last reset was from a source other than a power-on-reset. 1: Last reset was a power-on-reset.
EPFI	Enable Power fail Interrupt. This bit enables/disables the ability of the internal band-gap reference to generate a power-fail interrupt when V_{CC} falls below approximately 4.0V. While in Stop mode, both this bit and the band-gap select bit BGS (EXIF.0) must be set to enable the power-fail interrupt. 0: Power-fail interrupt disabled. 1: Power-fail interrupt enabled during normal operation. Power-fail interrupt enabled in Stop mode if BGS is set.

<table>
<tr><td>PFI</td><td>Power Fail Interrupt Flag. When set, this bit indicates that a power-fail interrupt has occurred. This bit must be cleared in software before exiting the interrupt service routine, or another interrupt will be generated. Setting this bit in software will generate a power-fail interrupt, if enabled.</td></tr>
<tr><td>WDIF</td><td>Watchdog Interrupt Flag. This bit, in conjunction with the Watchdog timer interrupt enable bit EWDI (EIE.4) and Enable Watchdog Timer Reset bit EWT (WDCON.1) indicates if a watchdog timer event has occurred and what action will be taken. This bit must be cleared in software before exiting the interrupt service routine, or another interrupt will be generated. Setting this bit in software will generate a watchdog interrupt if enabled.
<table>
<tr><th>EWT</th><th>EWDI</th><th>WDIF</th><th>Result</th></tr>
<tr><td>X</td><td>X</td><td>0</td><td>No watchdog event has occurred.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Watchdog time-out has expired. No interrupt has been generated.</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Watchdog interrupt has occurred.</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Watchdog time-out has expired. No interrupt has been generated. Watchdog timer reset will occur in 512 cycles if RWT is not strobed.</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Watchdog interrupt has occurred. Watchdog timer reset will occur in 512 cycles if RWT is not set.</td></tr>
</table></td></tr>
<tr><td>WTRF</td><td>Watchdog Timer Reset Flag. When set, this bit indicates that a watchdog timer reset has occurred. It is typically interrogated to determine if a reset was caused by watchdog timer reset. It is cleared by a power-on-reset, but otherwise must be cleared by software before the next reset of any kind or software may erroneously determine that a watchdog timer reset has occurred. Setting this bit in software will not generate a watchdog timer reset. If the EWT bit is cleared, the watchdog timer will have no effect on this bit.</td></tr>
<tr><td>EWT</td><td>Enable Watchdog Timer Reset. This bit enables/disables the ability of the watchdog timer to reset the device. This bit has no effect on the ability of the watchdog timer to generate a watchdog interrupt. The time-out period of the watchdog timer is controlled by the Watchdog Timer Mode Select bits (CKCON.7-6). Clearing these bits will disable the ability of the watchdog timer to generate a reset, but have no affect on the timer itself, or its ability to generate a watchdog timer interrupt.
0: A time-out of the watchdog timer will not cause the device to reset.
1: A time-out of the watchdog timer will cause the device to reset.</td></tr>
</table>

RWT	Reset Watchdog Timer. This bit serves as the strobe for the Watchdog function. During the time-out period, software must set the RWT bit if the Watchdog is enabled. Failing to set the RWT will cause a reset when the time-out has elapsed. There is no need to set the RWT bit to a 0 because it is self-clearing.

15.85 Port 1 Pull-up Control Register (P1SEL)

Bit No.	7	6	5	4	3	2	1	0	
D9h	P1SEL.7	P1SEL.6	P1SEL.5	P1SEL.4	P1SEL.3	P1SEL.2	P1SEL.1	P1SEL.0	P1SEL
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: Pull-up resistor ON (Default) 1: Pull-up resistor OFF

15.86 High Capture/Compare Register (C1CAP0H)

Bit No.	7	6	5	4	3	2	1	0	
DAh	C1CAP0H.7	C1CAP0H.6	C1CAP0H.5	C1CAP0H.4	C1CAP0H.3	C1CAP0H.2	C1CAP0H.1	C1CAP0H.0	C1CAP0H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.87 High Capture/Compare Register (C1CAP1H)

Bit No.	7	6	5	4	3	2	1	0	
DBh	C1CAP1H.7	C1CAP1H.6	C1CAP1H.5	C1CAP1H.4	C1CAP1H.3	C1CAP1H.2	C1CAP1H.1	C1CAP1H.0	C1CAP1H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.88 High Capture/Compare Register (C1CAP2H)

Bit No.	7	6	5	4	3	2	1	0	
DCh	C1CAP2H.7	C1CAP2H.6	C1CAP2H.5	C1CAP2H.4	C1CAP2H.3	C1CAP2H.2	C1CAP2H.1	C1CAP2H.0	C1CAP2H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.89 High Capture/Compare Register (C1CAP3H)

Bit No.	7	6	5	4	3	2	1	0	
DDh	C1CAP3H.7	C1CAP3H.6	C1CAP3H.5	C1CAP3H.4	C1CAP3H.3	C1CAP3H.2	C1CAP3H.1	C1CAP3H.0	C1CAP3H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.90 High Capture/Compare Register (C1CAP4H)

Bit No.	7	6	5	4	3	2	1	0	
DEh	C1CAP4H.7	C1CAP4H.6	C1CAP4H.5	C1CAP4H.4	C1CAP4H.3	C1CAP4H.2	C1CAP4H.1	C1CAP4H.0	C1CAP4H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.91 High Capture/Compare Register (C1CAP5H)

Bit No.	7	6	5	4	3	2	1	0	
DFh	C1CAP5H.7	C1CAP5H.6	C1CAP5H.5	C1CAP5H.4	C1CAP5H.3	C1CAP5H.2	C1CAP5H.1	C1CAP5H.0	C1CAP5H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.92 Accumulator (ACC/A)

Bit No.	7	6	5	4	3	2	1	0	
E0h	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0	ACC/A
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register serves as the accumulator for arithmetic operations.

15.93 Port 2 Pull-up Control Register (P2SEL)

Bit No.	7	6	5	4	3	2	1	0	
E1h	P2SEL.7	P2SEL.6	P2SEL.5	P2SEL.4	P2SEL.3	P2SEL.2	P2SEL.1	P2SEL.0	P2SEL
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: Pull-up resistor ON (Default) 1: Pull-up resistor OFF

15.94 Mode Control Register of PCA1 MODULE0 (C1CAPM0)

Bit No.	7	6	5	4	3	2	1	0	
E2h	IPWM0	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	C1CAPM0
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.95 Mode Control Register of PCA1 MODULE0 (C1CAPM1)

Bit No.	7	6	5	4	3	2	1	0	
E3h	IPWM1	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	C1CAPM1
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.96 Mode Control Register of PCA1 MODULE0 (C1CAPM2)

Bit No.	7	6	5	4	3	2	1	0	
E4h	IPWM2	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	C1CAPM2
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.97 Mode Control Register of PCA1 MODULE0 (C1CAPM3)

Bit No.	7	6	5	4	3	2	1	0	
E5h	IPWM3	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	C1CAPM3
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.98 Mode Control Register of PCA1 MODULE0 (C1CAPM4)

Bit No.	7	6	5	4	3	2	1	0	
E6h	IPWM4	ECOM4	CAPP4	CAPN4	MAT4	TOG4	PWM4	ECCF4	C1CAPM4
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.99 Mode Control Register of PCA1 MODULE0 (C1CAPM5)

Bit No.	7	6	5	4	3	2	1	0	
E7h	IPWM5	ECOM5	CAPP5	CAPN5	MAT5	TOG5	PWM5	ECCF5	C1CAPM5
	R/W(0)	R/W(1)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.100 External Interrupt Enable Register (EIE)

Bit No.	7	6	5	4	3	2	1	0	
E8h	EPCA1	EPCA0	ES1	EWDT	EX5	EX4	EX3	EX2	EIE
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
EPCA1	PCA1 interrupt enable
EPCA0	PCA0 interrupt enable
ES1	UART1 interrupt enable
EWDT	Watchdog timer interrupt Enable. This bit enables/disables the watchdog timer interrupt. 0: Disable the watchdog timer interrupt. 1: Enable the interrupt requests generated by the watchdog timer.
EX5	External Interrupt 5 Enable. This bit enables/disables external interrupt 5. 0: Disable external interrupt 5. 1: Enable the interrupt requests generated by the /INT5 pin.
EX4	External Interrupt 4 Enable. This bit enables/disables external interrupt 4. 0: Disable external interrupt 4. 1: Enable the interrupt requests generated by the INT4 pin.
EX3	External Interrupt 3 Enable. This bit enables/disables external interrupt 3. 0: Disable external interrupt 3. 1: Enable the interrupt requests generated by the /INT3 pin.
EX2	External Interrupt 2 Enable. This bit enables/disables external interrupt 2. 0: Disable external interrupt 2. 1: Enable the interrupt requests generated by the INT2 pin.

15.101 Port 3 Pull-up Control Register (P3SEL)

Bit No.	7	6	5	4	3	2	1	0	
E9h	P3SEL.7	P3SEL.6	P3SEL.5	P3SEL.4	P3SEL.3	P3SEL.2	P3SEL.1	P3SEL.0	P3SEL
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: Pull-up resistor ON (Default) 1: Pull-up resistor OFF

15.102 Low Byte Register of PCA1 Counter (C1L)

Bit No.	7	6	5	4	3	2	1	0	
EAh	C1L.7	C1L.6	C1L.5	C1L.4	C1L.3	C1L.2	C1L.1	C1L.0	C1L
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.103 High Byte Register of PCA1 Counter (C1H)

Bit No.	7	6	5	4	3	2	1	0	
EBh	C1H.7	C1H.6	C1H.5	C1H.4	C1H.3	C1H.2	C1H.1	C1H.0	C1H
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

15.104 ADC Input Channel Enable Bar Register (ADCHENB0)

Bit No.	7	6	5	4	3	2	1	0	
ECh	ADCHENB0.7	ADCHENB0.6	ADCHENB0.5	ADCHEN.B04	ADCHENB0.3	ADCHENB0.2	ADCHENB0.1	ADCHENB0.0	ADCHEN
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: ADC0 channel ON 1: ADC0 channel OFF (default)

15.105 ADC Input Channel Enable Bar Register (ADCHENB1)

Bit No.	7	6	5	4	3	2	1	0	
EDh	ADCHENB1.7	ADCHENB1.6	ADCHENB1.5	ADCHEN.B14	ADCHENB0.3	ADCHENB0.2	ADCHENB0.1	ADCHENB0.0	ADCHEN
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: ADC1 channel ON 1: ADC1 channel OFF (default)

15.106 ADC Input Channel Enable Bar Register (ADCHENB2)

Bit No.	7	6	5	4	3	2	1	0	
ECh	ADCHENB2.7	ADCHENB2.6	ADCHENB2.5	ADCHEN.B24	ADCHENB2.3	ADCHENB2.2	ADCHENB2.1	ADCHENB2.0	ADCHEN
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: ADC2 channel ON 1: ADC2 channel OFF (default)

15.107 ADC Input Channel Enable Bar Register (ADCHENB3)

Bit No.	7	6	5	4	3	2	1	0	
ECh	ADCHENB3.7	ADCHENB3.6	ADCHENB3.5	ADCHENB3.4	ADCHENB3.3	ADCHENB3.2	ADCHENB3.1	ADCHENB3.0	ADCHEN
	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	

0: ADC3 channel ON 1: ADC3 channel OFF (default)

15.108 ADC Clock and MUX Selection Register (ADCSEL)

Bit No.	7	6	5	4	3	2	1	0	
EDh	ADIV2	ADIV1	ADIV0	ADC4	ADC3	ADCS2	ADCS1	ADCS0	ADCSEL
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
ADIV[2:0]	ADC clock selection. [000] : FSYS / 2. [001] : FSYS/ 4. [010] : FSYS / 8. [011] : FSYS / 16. [100] : FSYS / 32. [101] : FSYS / 64. [110] : FSYS / 128. [111] : FSYS / 256.
ADCS[4:0]	ADC channel selection [00000] : ADC0.0 channel selection. [00001] : ADC0.1 channel selection. [00010] : ADC0.2 channel selection. [00011] : ADC0.3 channel selection. [00100] : ADC0.4 channel selection. [00101] : ADC0.5 channel selection. [00110] : ADC0.6 channel selection. [00111] : ADC0.7 channel selection. [01000] : ADC1.0 channel selection. [10111] : ADC2.7 channel selection. [11000] : ADC3.0 channel selection. [11001] : ADC3.1 channel selection. [11010] : ADC3.2 channel selection. [11011] : ADC3.3 channel selection. [11100] : ADC3.4 channel selection. [11101] : ADC3.5 channel selection. [11110] : ADC3.6 channel selection. [11111] : ADC3.7 channel selection.

15.109 ADC Result Value Register (ADCR)

Bit No.	7	6	5	4	3	2	1	0	
EEh	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2	SAR1	ADCR
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

[9:2] of ADC conversion result will be stored into ADCR. LSB of conversion result will be stored into ADCON[0].

15.110 ADC Control Register (ADCON)

Bit No.	7	6	5	4	3	2	1	0	
EFh	AD_EN	AD_REQ	AD_END	ADCF	-	-	SAR1	SAR0	ADCON
	R/W(0)	R/W(0)	R(1)	R/W(0)			R/W(0)	R/W(0)	

Symbol	Function
AD_EN	ADC Ready Enable.
AD_REQ	ADC Start Enable. Cleared by H/W when AD_END goes to 1 from 0.
AD_END	Cleared by H/W when ADC conversion start. Set by H/W when ADC conversion has been finished. 0: Data conversion is now running. 1: ADC is idle state.
ADCF	ADC Interrupt Flag. This bit must be cleared by S/W.
-	Reserved.
SAR1, SAR0	LSB of ADC result value

15.111 B Register (B)

Bit No.	7	6	5	4	3	2	1	0	
F0h	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0	B
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

This register serves as a second accumulator for certain arithmetic operations.

15.112 IAP Routine Access Enable Register (FAEN)

Bit No.	7	6	5	4	3	2	1	0	
F7h	-	-	-	-	-	-	-	FLASH_AEN	FAEN
	-	-	-	-	-	-	-	R/W(0)	

15.113 Extended Interrupt Priority Register (EIP)

Bit No.	7	6	5	4	3	2	1	0	
F8h	PPCA1	PPCA0	PS1	PWDT	PX5	PX4	PX3	PX2	EIP
	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
PPCA1	PCA1 interrupt priority bit.
PPCA0	PCA0 interrupt priority bit
PS1	UART1 interrupt priority bit
PWDT	Watchdog timer Interrupt Priority. This bit controls the priority of the watchdog timer interrupt. 0: The Watchdog timer interrupt is a low priority interrupt. 1: The Watchdog timer interrupt is a high priority interrupt.
PX5	External Interrupt 5 Priority. This bit controls the priority of the external interrupt 5. 0: The external interrupt 5 is a low priority interrupt. 1: The external interrupt 5 is a high priority interrupt.
PX4	External Interrupt 4 Priority. This bit controls the priority of the external interrupt 4. 0: The external interrupt 4 is a low priority interrupt. 1: The external interrupt 4 is a high priority interrupt.
PX3	External Interrupt 3 Priority. This bit controls the priority of the external interrupt 3. 0: The external interrupt 3 is a low priority interrupt. 1: The external interrupt 3 is a high priority interrupt.
PX2	External Interrupt 2 Priority. This bit controls the priority of the external interrupt 2. 0: The external interrupt 2 is a low priority interrupt. 1: The external interrupt 2 is a high priority interrupt.

15.114 Wakeup/I2C Register (UINDEX)

Bit No.	7	6	5	4	3	2	1	0	
F9h	I2C_BS	I2C_RXP	-	UINDX4	UINDX3	UINDX2	UINDX1	UINDX0	UINDEX
	R(0)	R/W(1)		R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	

Symbol	Function
I2C_BS	I2C Busy Flag 0: Idle 1: Busy
I2C_RXP	I2C RX FIFO pop 0: Idle 1: Pop FIFO, and move data to UDATA SFR (cleared automatically by H/W)
UINDX[4:0]	Wakeup Index Register [10000] : I2C RX FIFO read indirect address [10001] : I2C TX FIFO write indirect address [10010] : I2C RX FIFO pointer indirect address [10011] : I2C TX FIFO pointer indirect address [0XXXX] : Wakeup Register

15.115 Wakeup/I2C Data Register (UDATA)

Bit No.	7	6	5	4	3	2	1	0	
FAh	UDAT7	UDAT6	UDAT5	UDAT4	UDAT3	UDAT2	UDAT1	UDAT0	UDATA
	R/W(0)	R/W(1)	R/W(1)	R/W(0)	R/W(0)	R/W(1)	R/W(1)	R/W(1)	

15.116 Clock Selection Register (CLKSEL)

Bit No.	7	6	5	4	3	2	1	0	
FBh	-	-	-	XT/HF	WDEM	XR/PL	RG/PR	OSC32EB	CLKSEL
				R/W(0)	R/W(0)	R/W(1)	R/W(1)	R/W(0)	

Symbol	Function
XT/HF	XTAL division flag ([1] : XTAL/2 division)
WDEM	Watchdog timer extension mode select
XR/PL	PLL clock / XTRG clock selection 0: PLL clock 1: XTAL / RING MUX clock

RG/PR	Ring clock selection. 0: 32KHz ring clock for WDT power down. 1: 4MHz ring clock for normal operation.
OSC32EB	32KHz RING Enable Bar 0: 32KHz RING Enable 1: 32KHz RING DISABLE