

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "<u>Embedded -</u> <u>Microcontrollers</u>"

Details

E·XF

Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	4MHz
Connectivity	-
Peripherals	Brown-out Detect/Reset, POR, WDT
Number of I/O	13
Program Memory Size	896B (512 x 14)
Program Memory Type	ОТР
EEPROM Size	128 x 8
RAM Size	96 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	18-SOIC (0.295", 7.50mm Width)
Supplier Device Package	18-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lce623t-04i-so

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16CE62X family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16CE62X uses a Harvard architecture in which program and data are accessed from separate memories using separate buses. This improves bandwidth over traditional von Neumann architecture where program and data are fetched from the same memory. Separating program and data memory further allows instructions to be sized differently than 8-bit wide data word. Instruction opcodes are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions (35) execute in a single-cycle (200 ns @ 20 MHz) except for program branches.

The table below lists program memory (EPROM), data memory (RAM) and non-volatile memory (EEPROM) for each PIC16CE62X device.

Device	Program Memory	RAM Data Memory	EEPROM Data Memory	
PIC16CE623	512x14	96x8	128x8	
PIC16CE624	1Kx14	96x8	128x8	
PIC16CE625	2Kx14	128x8	128x8	

The PIC16CE62X can directly or indirectly address its register files or data memory. All special function registers including the program counter are mapped in the data memory. The PIC16CE62X family has an orthogonal (symmetrical) instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC16CE62X simple yet efficient. In addition, the learning curve is reduced significantly.

The PIC16CE62X devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

The ALU is 8 bits wide and capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. In two-operand instructions, typically one operand is the working register (W register). The other operand is a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

Depending on the instruction executed, the ALU may affect the values of the Carry (C), Digit Carry (DC), and Zero (Z) bits in the STATUS register. The C and DC bits operate as a Borrow and Digit Borrow out bit respectively, bit in subtraction. See the SUBLW and SUBWF instructions for examples.

A simplified block diagram is shown in Figure 3-1, with a description of the device pins in Table 3-1.

3.1 Clocking Scheme/Instruction Cycle

The clock input (OSC1/CLKIN pin) is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow is shown in Figure 3-2.

3.2 Instruction Flow/Pipelining

An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle, while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (i.e., GOTO) then two cycles are required to complete the instruction (Example 3-1).

A fetch cycle begins with the program counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the "Instruction Register (IR)" in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).



FIGURE 3-2: CLOCK/INSTRUCTION CYCLE





All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline, while the new instruction is being fetched and then executed.

4.0 MEMORY ORGANIZATION

4.1 <u>Program Memory Organization</u>

The PIC16CE62X has a 13-bit program counter capable of addressing an 8K x 14 program memory space. Only the first 512 x 14 (0000h - 01FFh) for the PIC16CE623, 1K x 14 (0000h - 03FFh) for the PIC16CE624 and 2K x 14 (0000h - 07FFh) for the PIC16CE625 are physically implemented. Accessing a location above these boundaries will cause a wrap-around within the first 512 x 14 space (PIC16CE623) or 1K x 14 space (PIC16CE624) or 2K x 14 space (PIC16CE625). The reset vector is at 0000h and the interrupt vector is at 0004h (Figure 4-1, Figure 4-2, Figure 4-3).

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK FOR THE PIC16CE623



FIGURE 4-2: PROGRAM MEMORY MAP AND STACK FOR THE PIC16CE624



FIGURE 4-3: PROGRAM MEMORY MAP AND STACK FOR THE PIC16CE625



4.2 Data Memory Organization

The data memory (Figure 4-4 and Figure 4-5) is partitioned into two Banks which contain the General Purpose Registers and the Special Function Registers. Bank 0 is selected when the RP0 bit is cleared. Bank 1 is selected when the RP0 bit (STATUS <5>) is set. The Special Function Registers are located in the first 32 locations of each Bank. Register locations 20-7Fh (Bank0) on the PIC16CE623/624 and 20-7Fh (Bank0) and A0-BFh (Bank1) on the PIC16CE625 are General Purpose Registers implemented as static RAM. Some special purpose registers are mapped in Bank 1. In all three microcontrollers, address space F0h-FFh (Bank1) is mapped to 70-7Fh (Bank0) as common RAM.

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file is organized as 96×8 in the PIC16CE623/624 and 128 x 8 in the PIC16CE625. Each is accessed either directly or indirectly through the File Select Register FSR (Section 4.4).

4.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral functions for controlling the desired operation of the device (Table 4-1). These registers are static RAM. The special registers can be classified into two sets (core and peripheral). The Special Function Registers associated with the "core" functions are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR Reset	Value on all other resets ⁽¹⁾
Bank 0											
00h	INDF	Addressin register)	ig this locat	ion uses co	ontents of F	SR to addre	ess data me	emory (not a	a physical	xxxx xxxx	xxxx xxxx
01h	TMR0	Timer0 M	odule's Reg	jister						xxxx xxxx	uuuu uuuu
02h	PCL	Program (Counter's (F	PC) Least S	Significant B	yte				0000 0000	0000 0000
03h	STATUS	IRP ⁽²⁾	RP1 ⁽²⁾	RP0	TO	PD	Z	DC	С	0001 1xxx	000q quuu
04h	FSR	Indirect da	ata memory	address p	ointer					xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	RA4	RA3	RA2	RA1	RA0	x 0000	u 0000
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
07h	Unimplemented									_	-
08h	Unimplemented									-	-
09h	Unimplemented									-	-
0Ah	PCLATH	—	—	—	Write buff	er for upper	5 bits of pr	ogram cou	nter	0 0000	0 0000
0Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	CMIF	—	—	—	—	_	—	-0	-0
0Dh-1Eh	Unimplemented									-	-
1Fh	CMCON	C2OUT	C10UT		—	CIS	CM2	CM1	CM0	00 0000	00 0000
Bank 1											
80h	INDF	Addressin register)	ig this locat	ion uses co	ontents of F	SR to addre	ess data me	emory (not a	a physical	XXXX XXXX	XXXX XXXX
81h	OPTION	RBPU	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Program (Counter's (F	PC) Least S	Significant B	yte				0000 0000	0000 0000
83h	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	С	0001 1xxx	000q quuu
84h	FSR	Indirect da	ata memory	address p	ointer					xxxx xxxx	uuuu uuuu
85h	TRISA	—	—		TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1 1111	1 1111
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
87h	Unimplemented									_	-
88h	Unimplemented									_	_
89h	Unimplemented									_	_
8Ah	PCLATH	—	_	_	Write buff	er for upper	5 bits of pr	ogram cou	nter	0 0000	0 0000
8Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
8Ch	PIE1	—	CMIE	_	_	—	_	_	_	-0	- 0
8Dh	Unimplemented									_	_
8Eh	PCON	—	_	_	_	—	_	POR	BOD	0x	uq
8Fh-9Eh	Unimplemented									-	_
90h	EEINTF	_	—	—	—	_	EESCL	EESDA	EEVDD	111	111
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000

TABLE 4-1: SPECIAL REGISTERS FOR THE PIC16CE62X

Legend: — = Unimplemented locations read as '0', u = unchanged, x = unknown, q = value depends on condition, shaded = unimplemented

Note 1: Other (non power-up) resets include MCLR reset, Brown-out Reset and Watchdog Timer Reset during normal operation.

Note 2: IRP & RPI bits are reserved; always maintain these bits clear.

NOTES:

TABLE 5-1:PORTA FUNCTIONS

Name	Bit #	Buffer Type	Function
RA0/AN0	bit0	ST	Input/output or comparator input
RA1/AN1	bit1	ST	Input/output or comparator input
RA2/AN2/VREF	bit2	ST	Input/output or comparator input or VREF output
RA3/AN3	bit3	ST	Input/output or comparator input/output
RA4/T0CKI	bit4	ST	Input/output or external clock input for TMR0 or comparator output. Output is open drain type.

Legend: ST = Schmitt Trigger input

TABLE 5-2:	SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR	Value on All Other Resets
05h	PORTA	—	_	—	RA4	RA3	RA2	RA1	RA0	x 0000	u 0000
85h	TRISA	—	—		TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1 1111	1 1111
1Fh	CMCON	C2OUT	C1OUT		—	CIS	CM2	CM1	CM0	00 0000	00 0000
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000

Legend: — = Unimplemented locations, read as '0', x = unknown, u = unchanged

Note: Shaded bits are not used by PORTA.

7.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control (i.e., it can be changed "on-the-fly" during program execution). To avoid an unintended device RESET, the following instruction sequence (Example 7-1) must be executed when changing the prescaler assignment from Timer0 to WDT.

EXAMPLE 7-1: CHANGING PRESCALER (TIMER0 \rightarrow WDT)

1.BCF	STATUS, RPO	;Skip if already in
		; Bank 0
2.CLRWDT		;Clear WDT
3.CLRF	TMR0	;Clear TMR0 & Prescaler
4.BSF	STATUS, RPO	;Bank 1
5.MOVLW	'00101111'b	;These 3 lines (5, 6, 7)
6.MOVWF	OPTION	; are required only if
		; desired PS<2:0> are
7.CLRWDT		; 000 or 001
8.MOVLW	'00101xxx'b	;Set Postscaler to
9.MOVWF	OPTION	; desired WDT rate
10.BCF	STATUS, RPO	;Return to Bank 0

To change prescaler from the WDT to the TMR0 module, use the sequence shown in Example 7-2. This precaution must be taken even if the WDT is disabled.

EXAMPLE 7-2: CHANGING PRESCALER (WDT \rightarrow TIMER0)

CLRWDT		;Clear WDT and
		;prescaler
BSF	STATUS, RPO	
MOVLW	b'xxxx0xxx'	;Select TMR0, new
		;prescale value and
		;clock source
MOVWF	OPTION_REG	
BCF	STATUS, RPO	

TABLE 7-1: REGISTERS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR	Value on All Other Resets
01h	TMR0	Timer0	Fimer0 module register							xxxx xxxx	uuuu uuuu
0Bh/8Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION	RBPU	INTEDG	TOCS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	_	_	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1 1111	1 1111

Legend: — = Unimplemented locations, read as '0', x = unknown, u = unchanged.

Note: Shaded bits are not used by TMR0 module.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR	Value on All Other Resets
1Fh	CMCON	C2OUT	C1OUT	_	—	CIS	CM2	CM1	CM0	00 0000	00 0000
9Fh	VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000
0Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	—	CMIF	_	—	—	_	—	—	-0	-0
8Ch	PIE1	—	CMIE	_	—	—	_	—	—	-0	-0
85h	TRISA		_	_	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1 1111	1 1111
Logondi	_ Unimn	lomontor	h rood oo	"0"	llakaowa		hongod				

TABLE 8-1: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Legend: - = Unimplemented, read as "0", x = Unknown, u = unchanged

EXAMPLE 9-1: VOLTAGE REFERENCE CONFIGURATION

MOVLW	0x02	;	4 Inputs Muxed
MOVWF	CMCON	;	to 2 comps.
BSF	STATUS, RPO	;	go to Bank 1
MOVLW	0x07	;	RA3-RA0 are
MOVWF	TRISA	;	outputs
MOVLW	0xA6	;	enable VREF
MOVWF	VRCON	;	low range
		;	set VR<3:0>=6
BCF	STATUS, RPO	;	go to Bank 0
CALL	DELAY10	;	10µs delay

9.2 <u>Voltage Reference Accuracy/Error</u>

The full range of VSS to VDD cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 9-1) keep VREF from approaching VSS or VDD. The Voltage Reference is VDD derived and therefore, the VREF output changes with fluctuations in VDD. The absolute accuracy of the Voltage Reference can be found in Table 13-2.

9.3 Operation During Sleep

When the device wakes up from sleep through an interrupt or a Watchdog Timer time-out, the contents of the VRCON register are not affected. To minimize current consumption in SLEEP mode, the Voltage Reference should be disabled.

9.4 Effects of a Reset

A device reset disables the Voltage Reference by clearing bit VREN (VRCON<7>). This reset also disconnects the reference from the RA2 pin by clearing bit VROE (VRCON<6>) and selects the high voltage range by clearing bit VRR (VRCON<5>). The VREF value select bits, VRCON<3:0>, are also cleared.

9.5 <u>Connection Considerations</u>

The Voltage Reference Module operates independently of the comparator module. The output of the reference generator may be connected to the RA2 pin if the TRISA<2> bit is set and the VROE bit, VRCON<6>, is set. Enabling the Voltage Reference output onto the RA2 pin with an input signal present will increase current consumption. Connecting RA2 as a digital output with VREF enabled will also increase current consumption.

The RA2 pin can be used as a simple D/A output with limited drive capability. Due to the limited drive capability, a buffer must be used in conjunction with the Voltage Reference output for external connections to VREF. Figure 9-2 shows an example buffering technique.

VREF Module Voltage Reference Output Impedance

FIGURE 9-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE

Note 1: R is dependent upon the Voltage Reference Configuration VRCON<3:0> and VRCON<5>.

TABLE 9-1: REGISTERS ASSOCIATED WITH VOLTAGE REFERENCE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value On POR / BOD	Value On All Other Resets
9Fh	VRCON	VREN	VROE	VRR	_	VR3	VR2	VR1	VR0	000- 0000	000- 0000
1Fh	CMCON	C2OUT	C10UT	_	—	CIS	CM2	CM1	CM0	00 0000	00 0000
85h	TRISA	—	—	_	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1 1111	1 1111

Legend: - = Unimplemented, read as "0"

10.4 <u>Power-on Reset (POR), Power-up</u> <u>Timer (PWRT), Oscillator Start-up</u> <u>Timer (OST) and Brown-out Reset</u> (BOD)

10.4.1 POWER-ON RESET (POR)

The on-chip POR circuit holds the chip in reset until VDD has reached a high enough level for proper operation. To take advantage of the POR, just tie the $\overline{\text{MCLR}}$ pin through a resistor to VDD. This will eliminate external RC components usually needed to create Power-on Reset. A maximum rise time for VDD is required. See electrical specifications for details.

The POR circuit does not produce an internal reset when VDD declines.

When the device starts normal operation (exits the reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in reset until the operating conditions are met.

For additional information, refer to Application Note AN607, "Power-up Trouble Shooting".

10.4.2 POWER-UP TIMER (PWRT)

The Power-up Timer provides a fixed 72 ms (nominal) time-out on power-up only, from POR or Brown-out Reset. The Power-up Timer operates on an internal RC oscillator. The chip is kept in reset as long as PWRT is active. The PWRT delay allows the VDD to rise to an acceptable level. A configuration bit, PWRTE, can disable (if set) or enable (if cleared or programmed) the Power-up Timer. The Power-up Timer should always be enabled when Brown-out Reset is enabled.

The Power-Up Time delay will vary from chip-to-chip and due to VDD, temperature and process variation. See DC parameters for details.

10.4.3 OSCILLATOR START-UP TIMER (OST)

The Oscillator Start-Up Timer (OST) provides a 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over. This ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP and HS modes and only on power-on reset or wake-up from SLEEP.

10.4.4 BROWN-OUT RESET (BOD)

The PIC16CE62X members have on-chip Brown-out Reset circuitry. A configuration bit, BOREN, can disable (if clear/programmed) or enable (if set) the Brown-out Reset circuitry. If VDD falls below 4.0V (refer to BVDD parameter D005) for greater than parameter (TBOR) in Table 13-5, the brown-out situation will reset the chip. A reset won't occur if VDD falls below 4.0V for less than parameter (TBOR).

On any reset (Power-on, Brown-out, Watch-dog, etc.) the chip will remain in reset until VDD rises above BVDD. The Power-up Timer will then be invoked and will keep the chip in reset an additional 72 ms.

If VDD drops below BVDD while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be re-initialized. Once VDD rises above BVDD, the Power-Up Timer will execute a 72 ms reset. The Power-up Timer should always be enabled when Brown-out Reset is enabled. Figure 10-7 shows typical Brown-out situations.



FIGURE 10-7: BROWN-OUT SITUATIONS





FIGURE 10-9: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2



FIGURE 10-10: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD)



^{© 1998-2013} Microchip Technology Inc.

10.5 Interrupts

The PIC16CE62X has 4 sources of interrupt:

- External interrupt RB0/INT
- TMR0 overflow interrupt
- PortB change interrupts (pins RB<7:4>)
- · Comparator interrupt

The interrupt control register (INTCON) records individual interrupt requests in flag bits. It also has individual and global interrupt enable bits.

A global interrupt enable bit, GIE (INTCON<7>) enables (if set) all un-masked interrupts or disables (if cleared) all interrupts. Individual interrupts can be disabled through their corresponding enable bits in INTCON register. GIE is cleared on reset.

The "return from interrupt" instruction, RETFIE, exits interrupt routine, as well as sets the GIE bit, which re-enable RB0/INT interrupts.

The INT pin interrupt, the RB port change interrupt and the TMR0 overflow interrupt flags are contained in the INTCON register.

The peripheral interrupt flag is contained in the special register PIR1. The corresponding interrupt enable bit is contained in special registers PIE1.

When an interrupt is responded to, the GIE is cleared to disable any further interrupt, the return address is pushed into the stack and the PC is loaded with 0004h. Once in the interrupt service routine, the source(s) of

the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid RB0/INT recursive interrupts.

For external interrupt events, such as the INT pin or PORTB change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends on when the interrupt event occurs (Figure 10-16). The latency is the same for one or two cycle instructions. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid multiple interrupt requests.

- Note 1: Individual interrupt flag bits are set, regardless of the status of their corresponding mask bit or the GIE bit.
 - 2: When an instruction that clears the GIE bit is executed, any interrupts that were pending for execution in the next cycle are ignored. The CPU will execute a NOP in the cycle immediately following the instruction which clears the GIE bit. The interrupts which were ignored are still pending to be serviced when the GIE bit is set again.



FIGURE 10-15: INTERRUPT LOGIC

10.8 Power-Down Mode (SLEEP)

The Power-down mode is entered by executing a SLEEP instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the \overline{PD} bit in the STATUS register is cleared, the \overline{TO} bit is set and the oscillator driver is turned off. The I/O ports maintain the status they had before SLEEP was executed (driving high, low, or hi-impedance).

For lowest current consumption in this mode, all I/O pins should be either at VDD or VSS, with no external circuitry drawing current from the I/O pin, and the comparators and VREF should be disabled. I/O pins that are hi-impedance inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The TOCKI input should also be at VDD or VSS for lowest current consumption. The contribution from on chip pull-ups on PORTB should be considered.

The MCLR pin must be at a logic high level (VIHMC).

Note:	It should be noted that a RESET generated
	by a WDT time-out does not drive MCLR
	pin low.

10.8.1 WAKE-UP FROM SLEEP

The device can wake-up from SLEEP through one of the following events:

- 1. External reset input on MCLR pin
- 2. Watchdog Timer Wake-up (if WDT was enabled)
- 3. Interrupt from RB0/INT pin, RB Port change, or the Peripheral Interrupt (Comparator).

The first event will cause a device reset. The two latter events are considered a continuation of program execution. The \overline{TO} and \overline{PD} bits in the STATUS register can be used to determine the cause of device reset. \overline{PD} bit, which is set on power-up is cleared when SLEEP is invoked. \overline{TO} bit is cleared if WDT wake-up occurred.

When the SLEEP instruction is being executed, the next instruction (PC + 1) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is set (enabled), the device executes the instruction after the SLEEP instruction after the SLEEP instruction after the instruction and then branches to the interrupt address (0004h). In cases where the execution of the instruction following SLEEP is not desirable, the user should have an NOP after the SLEEP instruction.

If the global interrupts are disabled (GIE is
cleared), but any interrupt source has both
its interrupt enable bit and the correspond-
ing interrupt flag bits set, the device will
immediately wake-up from sleep. The
sleep instruction is completely executed.

The WDT is cleared when the device wakes-up from sleep, regardless of the source of wake-up.

; a1 a2 a3 a4 ; a1 a2 a3 a osc1////////////////////////////////////	4 Q1	a1 a2 a3 a4	a1 a2 a3 a4	; a1 a2 a3 a4 /~	; a1 a2 a3 a4; ////////////////////////////////////
CLKOUT(4)	Tost(2)		\/	<u>\</u> /	\ł
INT pin INTF flag (INTCON<1>) GIE bit (INTCON<7>)	Processor in SLEEP		Interrupt Latency		
INSTRUCTION FLOW			I I	I I	
PC PC PC+1	PC+2	X PC+2	X PC + 2	X 0004h	X 0005h
$\begin{array}{l} \mbox{Instruction} \\ \mbox{fetched} \end{array} \left\{ \begin{array}{l} \mbox{Inst(PC)} = \mbox{SLEEP} & \mbox{Inst(PC + 1)} \end{array} \right.$		Inst(PC + 2)	1 1 1	Inst(0004h)	Inst(0005h)
Instruction { Inst(PC - 1) SLEEP	1 1 1	Inst(PC + 1)	Dummy cycle	Dummy cycle	Inst(0004h)

FIGURE 10-19: WAKE-UP FROM SLEEP THROUGH INTERRUPT

Note 1: XT, HS or LP oscillator mode assumed.

2: TOST = 1024TOSC (drawing not to scale) This delay does not occur for RC osc mode.

3: GIE = '1' assumed. In this case after wake- up, the processor jumps to the interrupt routine. If GIE = '0', execution will continue in-line.

4: CLKOUT is not available in these osc modes, but shown here for timing reference.

CLRWDT	Clear Watchdog Timer
Syntax:	[label] CLRWDT
Operands:	None
Operation:	$\begin{array}{l} 00h \rightarrow WDT \\ 0 \rightarrow WDT \ prescaler, \\ 1 \rightarrow \overline{TO} \\ 1 \rightarrow \overline{PD} \end{array}$
Status Affected:	TO, PD
Encoding:	00 0000 0110 0100
	Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{\text{TO}}$ and $\overline{\text{PD}}$ are set.
Words:	1
Cycles:	1
Example	CLRWDT
	Before Instruction WDT counter = ? After Instruction WDT counter = $0x00$ WDT prescaler= 0 TO = 1 PD = 1
COMF	Complement f
Syntax:	[label] COME fd

COMI	oomplement
Syntax:	[label] COMF f,d
Operands:	$\begin{array}{l} 0\leq f\leq 127\\ d\in[0,1] \end{array}$
Operation:	$(\overline{f}) \rightarrow (dest)$
Status Affected:	Z
Encoding:	00 1001 dfff ffff
Description:	complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	COMF REG1,0
	Before Instruction REG1 = 0x13 After Instruction REG1 = 0x13 W = 0xEC

DECF	Decreme	ent f			
Syntax:	[label]	DECF f,	d		
Operands:	$\begin{array}{l} 0\leq f\leq 12\\ d\in [0,1] \end{array}$	7			
Operation:	(f) - 1 \rightarrow	(dest)			
Status Affected:	Z				
Encoding:	00	0011	df	ff	ffff
Description:	Decremen result is st is 1, the re ter 'f'.	t register ored in th sult is sto	'f'. If e W i ored b	'd' is (regist back i), the er. If 'd' n regis-
Words:	1				
Cycles:	1				
Example	DECF	CNT,	1		
	Before In After Inst	struction CNT Z ruction CNT Z	= =	0x01 0 0x00 1)

DECFSZ	Decrement f, Skip if 0
Syntax:	[label] DECFSZ f,d
Operands:	$\begin{array}{l} 0\leq f\leq 127\\ d\in [0,1] \end{array}$
Operation:	(f) - 1 \rightarrow (dest); skip if result = 0
Status Affected:	None
Encoding:	00 1011 dfff ffff
Description:	The contents of register 'f are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two-cycle instruction.
Words:	1
Cycles:	1(2)
Example	HERE DECFSZ CNT, 1 GOTO LOOP CONTINUE • •
	$\begin{array}{rcl} Before \ Instruction \\ PC &= & address \ {\tt HERE} \\ After \ Instruction \\ CNT &= & CNT - 1 \\ if \ CNT &= & 0, \\ PC &= & address \ {\tt CONTINUE} \\ if \ CNT \neq & 0, \\ PC &= & address \ {\tt HERE+1} \\ \end{array}$

GOTO	Unconditional Branch	INCFSZ	Increment f, Skip if 0
Syntax:	[<i>label</i>] GOTO k	Syntax:	[label] INCFSZ f,d
Operands:	$0 \le k \le 2047$	Operands:	$0 \leq f \leq 127$
Operation:	$k \rightarrow PC < 10:0 >$		d ∈ [0,1]
	$PCLATH<4:3> \rightarrow PC<12:11>$	Operation:	(f) + 1 \rightarrow (dest), skip if result = 0
Status Affected:	None	Status Affected:	None
Encoding:	10 1kkk kkkk kkkk	Encoding:	00 1111 dfff ffff
Description: Words: Cycles:	GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two-cycle instruction. 1	Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two curls instruction
Example	GOTO THERE	Words:	1
	After Instruction	Cycles:	1(2)
	PC = Address THERE	Example	HERE INCFSZ CNT, 1 GOTO LOOP CONTINUE •

 $\begin{array}{rcl} Before \ Instruction \\ PC & = & address \ HERE \\ After \ Instruction \\ CNT & = & CNT + 1 \\ if \ CNT = & 0, \\ PC & = & address \ CONTINUE \\ if \ CNT \neq & 0, \\ PC & = & address \ HERE \ +1 \\ \end{array}$

INCF	Increment f
Syntax:	[label] INCF f,d
Operands:	$\begin{array}{l} 0 \leq f \leq 127 \\ d \in [0,1] \end{array}$
Operation:	(f) + 1 \rightarrow (dest)
Status Affected:	Z
Encoding:	00 1010 dfff ffff
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.
Words:	1
Cycles:	1
Example	INCF CNT, 1
	$\begin{array}{rrrr} \text{Before Instruction} \\ & \text{CNT} & = & 0 \text{xFF} \\ & Z & = & 0 \end{array}$ After Instruction $\begin{array}{rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr$

IORLW	Inclusive OR Literal with W
Syntax:	[<i>label</i>] IORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .OR. $k \rightarrow$ (W)
Status Affected:	Z
Encoding:	11 1000 kkkk kkkk
Description:	The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1
Example	IORLW 0x35
	Before Instruction W = 0x9A After Instruction W = 0xBF Z = 1

SWAPF	Swap Nib	bles in	f		XORLW	Exclusiv	ve OR L	iteral wit	th W
Syntax:	[<i>label</i>] SWAPF f,d				Syntax:	[label]	[label] XORLW k		
Operands:	$0 \leq f \leq 127$ d $\in [0,1]$				Operands:	$0 \le k \le 255$			
Operation:	$(f<3:0>) \rightarrow (dest<7:4>),$ $(f<7:4>) \rightarrow (dest<3:0>)$			Status Affected:	(W) .XOR. $K \rightarrow (W)$				
Status Affected:	None				Encoding:	11	1010	kkkk	kkkk
Encoding:	00 1110 dfff ffff				Description:	The contents of the W register are			er are
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in W register. If 'd'			Words:	The result is placed in the W register.				
Marda	is 1, the res	sult is pla	ced in reg	ister 't'.	Cycles:	1			
Cycles:	1				Example:	XORLW	0xAF		
Example	SWAPF R	EG,	0			Before I	nstructio	n	
	Before Ins	truction					W =	0xB5	
		REG1	= 0x/	45		After Ins	truction		
	After Instru	uction					W =	0x1A	
		REG1 W	= 0x/ = 0x5	45 5A					

TRIS	Load TRIS Register
Syntax:	[label] TRIS f
Operands:	$5 \le f \le 7$
Operation:	(W) \rightarrow TRIS register f;
Status Affected:	None
Encoding:	00 0000 0110 0fff
Description: Words: Cycles: Example	The instruction is supported for code compatibility with the PIC16C5X products. Since TRIS registers are readable and writable, the user can directly address them. 1
	To maintain upward compatibility with future PIC [®] MCU products, do not use this instruction.

XORWF	Exclusive OR W with f					
Syntax:	[label]	XORWF	f,d			
Operands:	$\begin{array}{l} 0\leq f\leq 127\\ d\in [0,1] \end{array}$					
Operation:	(W) .XOR. (f) \rightarrow (dest)					
Status Affected:	Z					
Encoding:	0 0	0110	dff	Ē	ffff	
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in reg- ister 'f'.					
Words:	1					
Cycles:	1					
Example	XORWF	REG 3	1			
	Before Instruction					
		REG W	= =	0xA 0xE	AF 35	
	After Instruction					
		REG W	= =	Ox1 OxE	I A 35	

stand-alone mode the PRO MATE II can read, verify or program PIC devices. It can also set code-protect bits in this mode.

12.11 <u>PICSTART Plus Entry Level</u> <u>Development System</u>

The PICSTART programmer is an easy-to-use, lowcost prototype programmer. It connects to the PC via one of the COM (RS-232) ports. MPLAB Integrated Development Environment software makes using the programmer simple and efficient.

PICSTART Plus supports all PIC devices with up to 40 pins. Larger pin count devices such as the PIC16C92X, and PIC17C76X may be supported with an adapter socket. PICSTART Plus is CE compliant.

12.12 <u>SIMICE Entry-Level</u> <u>Hardware Simulator</u>

SIMICE is an entry-level hardware development system designed to operate in a PC-based environment with Microchip's simulator MPLAB-SIM. Both SIMICE and MPLAB-SIM run under Microchip Technology's MPLAB Integrated Development Environment (IDE) software. Specifically, SIMICE provides hardware simulation for Microchip's PIC12C5XX, PIC12CE5XX, and PIC16C5X families of PIC 8-bit microcontrollers. SIM-ICE works in conjunction with MPLAB-SIM to provide non-real-time I/O port emulation. SIMICE enables a developer to run simulator code for driving the target system. In addition, the target system can provide input to the simulator code. This capability allows for simple and interactive debugging without having to manually generate MPLAB-SIM stimulus files. SIMICE is a valuable debugging tool for entry-level system development.

12.13 <u>PICDEM-1 Low-Cost PIC MCU</u> <u>Demonstration Board</u>

The PICDEM-1 is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The users can program the sample microcontrollers provided with the PICDEM-1 board, on a PRO MATE II or PICSTART-Plus programmer, and easily test firmware. The user can also connect the PICDEM-1 board to the MPLAB-ICE emulator and download the firmware to the emulator for testing. Additional prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push-button switches and eight LEDs connected to PORTB.

12.14 <u>PICDEM-2 Low-Cost PIC16CXX</u> <u>Demonstration Board</u>

The PICDEM-2 is a simple demonstration board that supports the PIC16C62, PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-2 board, on a PRO MATE II programmer or PICSTART-Plus, and easily test firmware. The MPLAB-ICE emulator may also be used with the PICDEM-2 board to test firmware. Additional prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push-button switches, a potentiometer for simulated analog input, a Serial EEPROM to demonstrate usage of the I²C bus and separate headers for connection to an LCD module and a keypad.

12.15 <u>PICDEM-3 Low-Cost PIC16CXXX</u> <u>Demonstration Board</u>

The PICDEM-3 is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers with a LCD Module. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-3 board, on a PRO MATE II programmer or PICSTART Plus with an adapter socket, and easily test firmware. The MPLAB-ICE emulator may also be used with the PICDEM-3 board to test firmware. Additional prototype area has been provided to the user for adding hardware and connecting it to the microcontroller socket(s). Some of the features include an RS-232 interface, push-button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM-3 board is an LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM-3 provides an additional RS-232 interface and Windows 3.1 software for showing the demultiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware demultiplexer for the LCD signals.

12.16 PICDEM-17

The PICDEM-17 is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756, PIC17C762, and PIC17C766. All necessary hardware is included to run basic demo programs, which are supplied on a 3.5-inch disk. A programmed sample is included, and the user may erase it and program it with the other sample programs using the PRO MATE II or PICSTART Plus device programmers and easily debug

PIC16XXXXX FAMILY

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

TO: RE:	Technical Publications Manager Reader Response	Total Pages Sent			
Eroi	n: Nome				
FIU					
	Address				
	City / State / ZIP / Country				
	Telephone: ()	FAX: ()			
Арр	lication (optional):				
Wo	uld you like a reply? Y N				
Dev	ice: PIC16xxxxx family	Literature Number: DS40182D			
Que	estions:				
1.	What are the best features of this document?				
2.	How does this document meet your hardware and so	oftware development needs?			
3.	. Do you find the organization of this document easy to follow? If not, why?				
4.	What additions to the document do you think would enhance the structure and subject?				
5.	What deletions from the document could be made without affecting the overall usefulness?				
6.	Is there any incorrect or misleading information (what	at and where)?			
7.	How would you improve this document?				

NOTES: