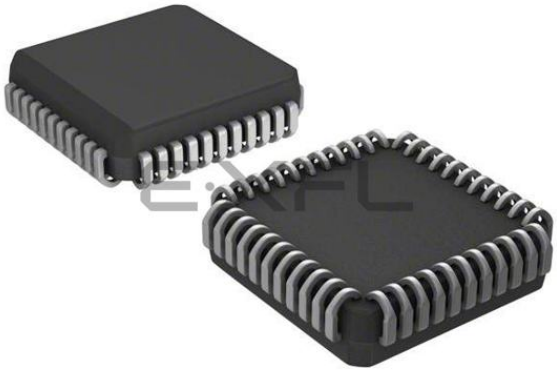


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	33MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	4KB (2K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	232 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	44-PLCC (16.59x16.59)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c42at-33e-l

NOTES:

3.1 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) 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, and 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 are shown in Figure 3-3.

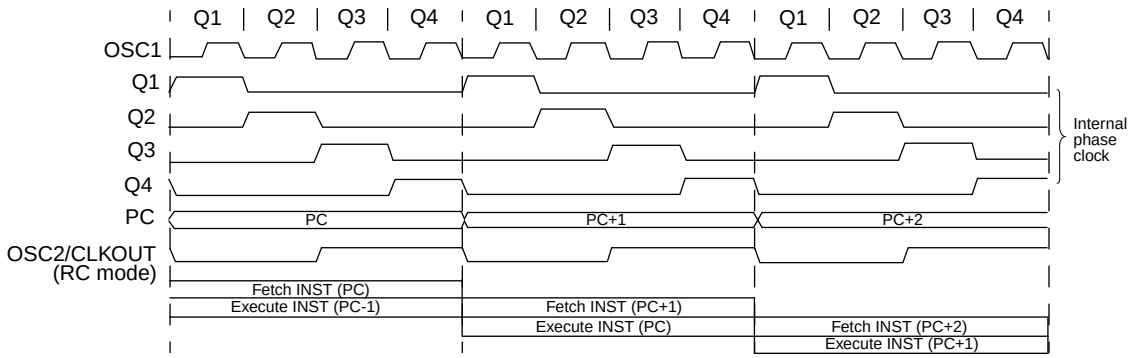
3.2 Instruction Flow/Pipelining

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 (e.g. `GOTO`) then two cycles are required to complete the instruction (Example 3-2).

A fetch cycle begins with the program counter 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 3-3: CLOCK/INSTRUCTION CYCLE



EXAMPLE 3-2: INSTRUCTION PIPELINE FLOW

	Tcy0	Tcy1	Tcy2	Tcy3	Tcy4	Tcy5
1. `MOVLW 55h`	Fetch 1	Execute 1				
2. `MOVWF PORTB`		Fetch 2	Execute 2			
3. `CALL SUB_1`			Fetch 3	Execute 3		
4. `BSF PORTA, BIT3 (Forced NOP)`				Fetch 4	Flush	
5. `Instruction @ address SUB_1`					Fetch SUB_1	Execute SUB_1

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.

Example 8-4 shows the sequence to do an 16 x 16 signed multiply. Equation 8-2 shows the algorithm that used. The 32-bit result is stored in four registers RES3:RES0. To account for the sign bits of the arguments, each argument pairs most significant bit (MSb) is tested and the appropriate subtractions are done.

EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

RES3:RES0

= ARG1H:ARG1L * ARG2H:ARG2L

= (ARG1H * ARG2H * 2^{16}) +

(ARG1H * ARG2L * 2^{8}) +

(ARG1L * ARG2H * 2^{8}) +

(ARG1L * ARG2L) +

(-1 * ARG2H<7> * ARG1H:ARG1L * 2^{16}) +

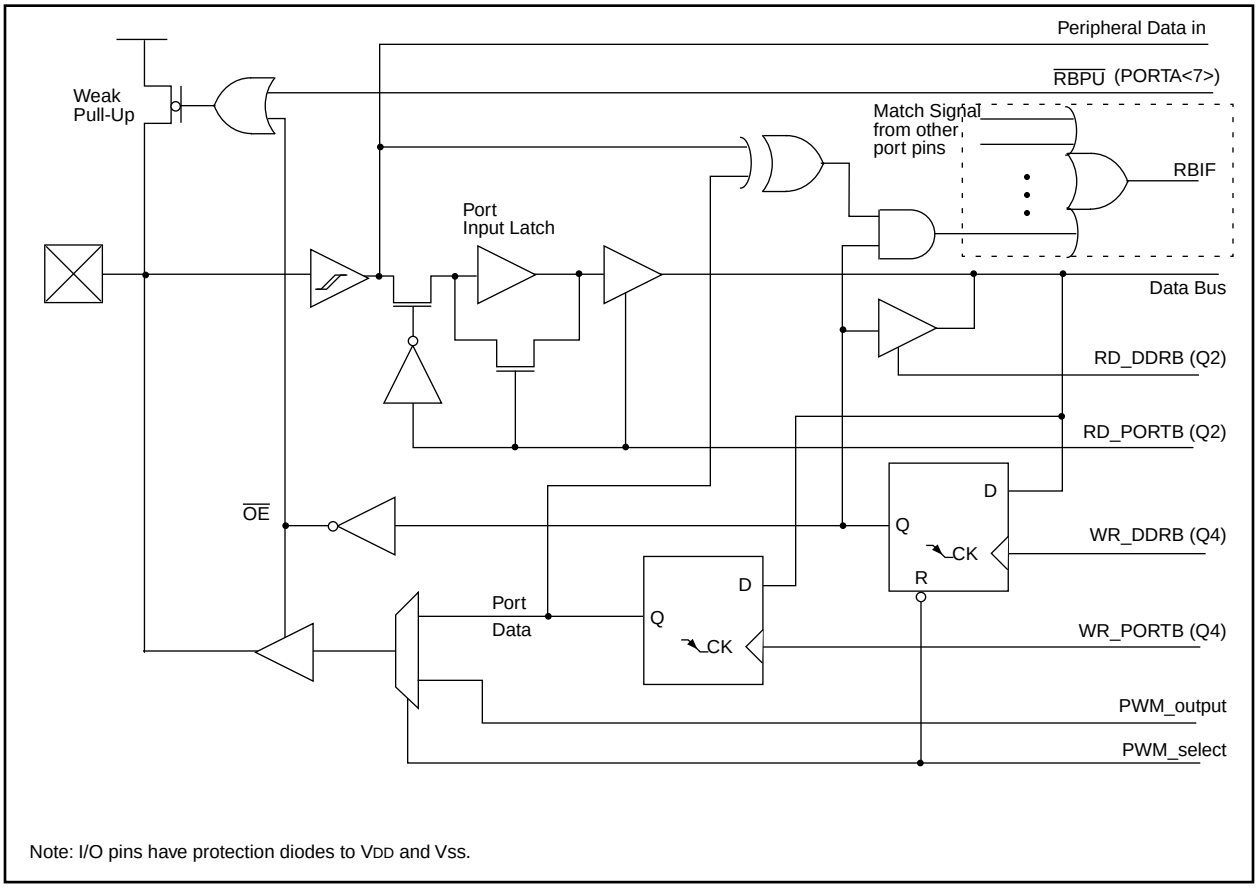
(-1 * ARG1H<7> * ARG2H:ARG2L * 2^{16})

EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
   MOVFP    ARG1L, WREG
   MULWF    ARG2L       ; ARG1L * ARG2L ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES1 ;
   MOVPF    PRODL, RES0 ;
;
   MOVFP    ARG1H, WREG
   MULWF    ARG2H       ; ARG1H * ARG2H ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES3 ;
   MOVPF    PRODL, RES2 ;
;
   MOVFP    ARG1L, WREG
   MULWF    ARG2H       ; ARG1L * ARG2H ->
                        ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   MOVFP    ARG1H, WREG ;
   MULWF    ARG2L       ; ARG1H * ARG2L ->
                        ;   PRODH:PRODL

   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   BTFSS    ARG2H, 7    ; ARG2H:ARG2L neg?
   GOTO     SIGN_ARG1   ; no, check ARG1
   MOVFP    ARG1L, WREG ;
   SUBWF    RES2        ;
   MOVFP    ARG1H, WREG ;
   SUBWFB   RES3
;
SIGN_ARG1
   BTFSS    ARG1H, 7    ; ARG1H:ARG1L neg?
   GOTO     CONT_CODE   ; no, done
   MOVFP    ARG2L, WREG ;
   SUBWF    RES2        ;
   MOVFP    ARG2H, WREG ;
   SUBWFB   RES3
;
CONT_CODE
     :
```

FIGURE 9-5: BLOCK DIAGRAM OF RB3 AND RB2 PORT PINS



Note: I/O pins have protection diodes to VDD and Vss.

9.5 I/O Programming Considerations

9.5.1 BI-DIRECTIONAL I/O PORTS

Any instruction which writes, operates internally as a read followed by a write operation. For example, the BCF and BSF instructions read the register into the CPU, execute the bit operation, and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit5 of PORTB will cause all eight bits of PORTB to be read into the CPU. Then the BSF operation takes place on bit5 and PORTB is written to the output latches. If another bit of PORTB is used as a bi-directional I/O pin (e.g. bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and re-written to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit0 is switched into output mode later on, the content of the data latch may now be unknown.

Reading a port reads the values of the port pins. Writing to the port register writes the value to the port latch. When using read-modify-write instructions (BCF, BSF, BTG, etc.) on a port, the value of the port pins is read, the desired operation is performed with this value, and the value is then written to the port latch.

Example 9-5 shows the effect of two sequential read-modify-write instructions on an I/O port.

EXAMPLE 9-5: READ MODIFY WRITE INSTRUCTIONS ON AN I/O PORT

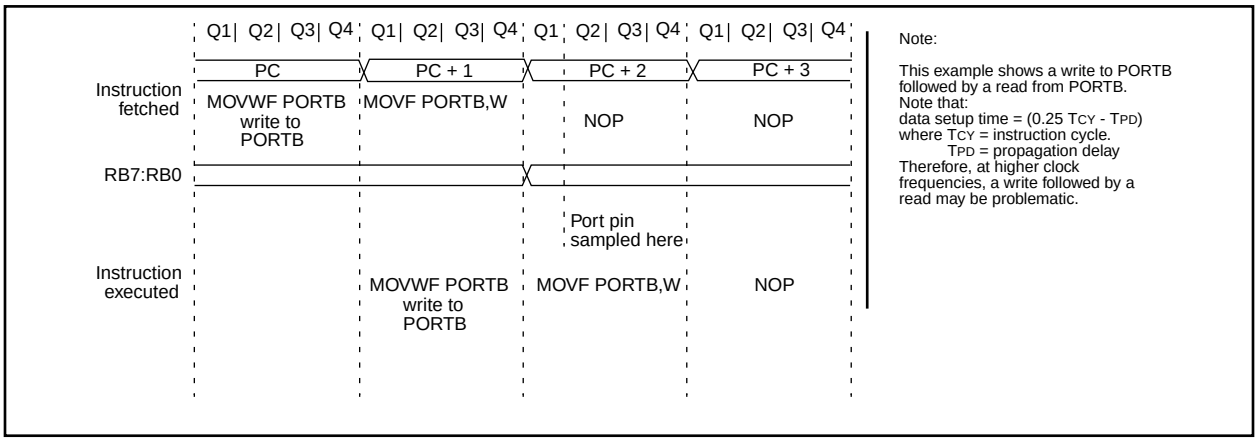
```
; Initial PORT settings: PORTB<7:4> Inputs
;                        PORTB<3:0> Outputs
; PORTB<7:6> have pull-ups and are
; not connected to other circuitry
;
;                       PORT latch  PORT pins
;                       ----------  ---------
;
   BCF   PORTB, 7       01pp pppp   11pp pppp
   BCF   PORTB, 6       10pp pppp   11pp pppp
;
   BCF   DDRB, 7        10pp pppp   11pp pppp
   BCF   DDRB, 6        10pp pppp   10pp pppp
;
; Note that the user may have expected the
; pin values to be 00pp pppp. The 2nd BCF
; caused RB7 to be latched as the pin value
; (High).
```

> **Note:** A pin actively outputting a Low or High should not be driven from external devices in order to change the level on this pin (i.e. “wired-or”, “wired-and”). The resulting high output currents may damage the device.

9.5.2 SUCCESSIVE OPERATIONS ON I/O PORTS

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (Figure 9-9). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such to allow the pin voltage to stabilize (load dependent) before executing the instruction that reads the values on that I/O port. Otherwise, the previous state of that pin may be read into the CPU rather than the “new” state. When in doubt, it is better to separate these instructions with a NOP or another instruction not accessing this I/O port.

FIGURE 9-9: SUCCESSIVE I/O OPERATION



11.0 TIMER0

The Timer0 module consists of a 16-bit timer/counter, TMR0. The high byte is TMR0H and the low byte is TMR0L. A software programmable 8-bit prescaler makes an effective 24-bit overflow timer. The clock source is also software programmable as either the internal instruction clock or the RA1/T0CKI pin. The control bits for this module are in register T0STA (Figure 11-1).

FIGURE 11-1: T0STA REGISTER (ADDRESS: 05h, UNBANKED)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0
INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented, Read as '0'
-n = Value at POR reset

bit 7: **INTEDG**: RA0/INT Pin Interrupt Edge Select bit
This bit selects the edge upon which the interrupt is detected
1 = Rising edge of RA0/INT pin generates interrupt
0 = Falling edge of RA0/INT pin generates interrupt

bit 6: **T0SE**: Timer0 Clock Input Edge Select bit
This bit selects the edge upon which TMR0 will increment
When T0CS = 0
1 = Rising edge of RA1/T0CKI pin increments TMR0 and/or generates a T0CKIF interrupt
0 = Falling edge of RA1/T0CKI pin increments TMR0 and/or generates a T0CKIF interrupt
When T0CS = 1
Don't care

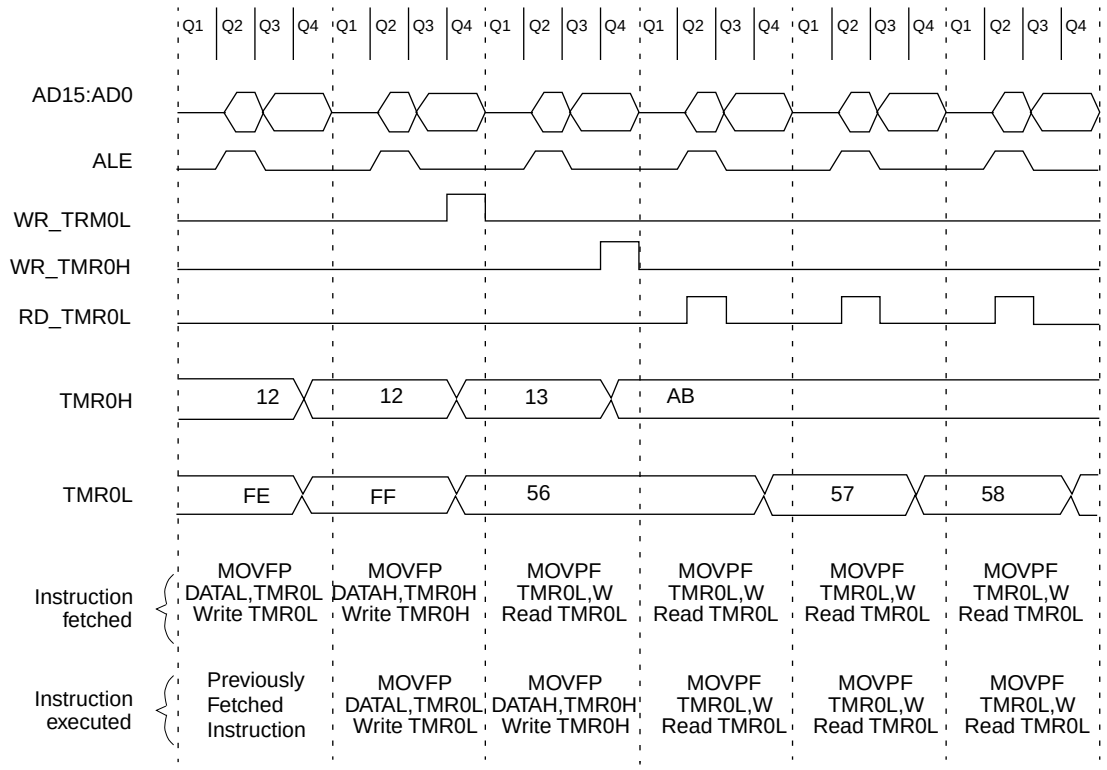
bit 5: **T0CS**: Timer0 Clock Source Select bit
This bit selects the clock source for TMR0.
1 = Internal instruction clock cycle (TCY)
0 = T0CKI pin

bit 4-1: **PS3:PS0**: Timer0 Prescale Selection bits
These bits select the prescale value for TMR0.

PS3:PS0	Prescale Value
0000	1:1
0001	1:2
0010	1:4
0011	1:8
0100	1:16
0101	1:32
0110	1:64
0111	1:128
1xxx	1:256

bit 0: **Unimplemented**: Read as '0'

FIGURE 11-5: TMR0 READ/WRITE IN TIMER MODE



In this example, old TMR0 value is 12FEh, new value of AB56h is written.

TABLE 11-1: REGISTERS/BITS ASSOCIATED WITH TIMER0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note1)
05h, Unbanked	T0STA	INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—	0000 000-	0000 000-
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	—	—	--11 11--	--11 qq--
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
0Bh, Unbanked	TMR0L	TMR0 register; low byte								xxxx xxxx	uuuu uuuu
0Ch, Unbanked	TMR0H	TMR0 register; high byte								xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as a '0', q - value depends on condition, Shaded cells are not used by Timer0.
Note 1: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

12.1.3 USING PULSE WIDTH MODULATION (PWM) OUTPUTS WITH TMR1 AND TMR2

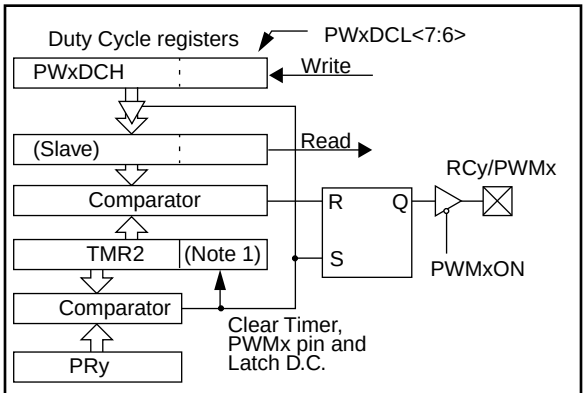
Two high speed pulse width modulation (PWM) outputs are provided. The PWM1 output uses Timer1 as its time-base, while PWM2 may be software configured to use either Timer1 or Timer2 as the time-base. The PWM outputs are on the RB2/PWM1 and RB3/PWM2 pins.

Each PWM output has a maximum resolution of 10-bits. At 10-bit resolution, the PWM output frequency is 24.4 kHz (@ 25 MHz clock) and at 8-bit resolution the PWM output frequency is 97.7 kHz. The duty cycle of the output can vary from 0% to 100%.

Figure 12-5 shows a simplified block diagram of the PWM module. The duty cycle register is double buffered for glitch free operation. Figure 12-6 shows how a glitch could occur if the duty cycle registers were not double buffered.

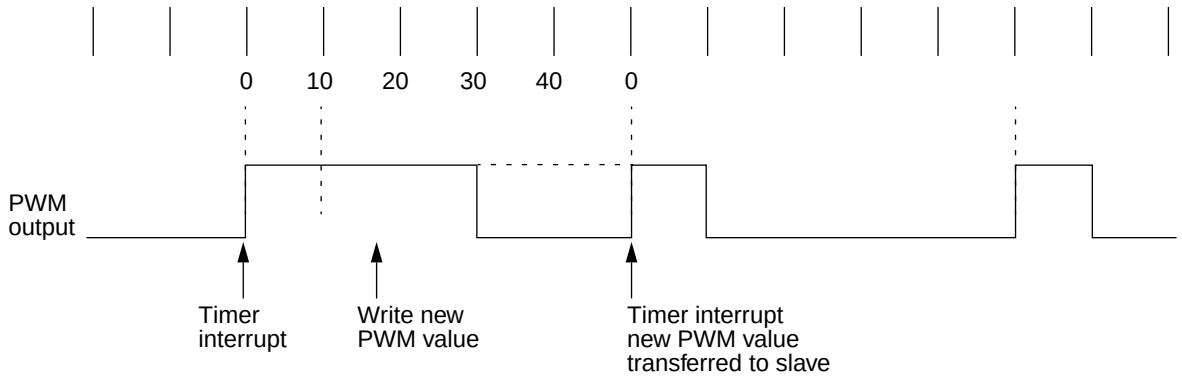
The user needs to set the PWM1ON bit (TCON2<4>) to enable the PWM1 output. When the PWM1ON bit is set, the RB2/PWM1 pin is configured as PWM1 output and forced as an output irrespective of the data direction bit (DDRB<2>). When the PWM1ON bit is clear, the pin behaves as a port pin and its direction is controlled by its data direction bit (DDRB<2>). Similarly, the PWM2ON (TCON2<5>) bit controls the configuration of the RB3/PWM2 pin.

FIGURE 12-5: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time-base.

FIGURE 12-6: PWM OUTPUT



Note The dotted line shows PWM output if duty cycle registers were not double buffered. If the new duty cycle is written after the timer has passed that value, then the PWM does not reset at all during the current cycle causing a “glitch”.

In this example, PWM period = 50. Old duty cycle is 30. New duty cycle value is 10.

14.4 Power-down Mode (SLEEP)

The Power-down mode is entered by executing a SLEEP instruction. This clears the Watchdog Timer and postscaler (if enabled). The $\overline{PD}$ bit is cleared and the $\overline{TO}$ bit is set (in the CPUSTA register). In SLEEP mode, the oscillator driver is turned off. The I/O ports maintain their status (driving high, low, or hi-impedance).

The $\overline{MCLR}$/VPP pin must be at a logic high level (VIHMC). A WDT time-out RESET does not drive the $\overline{MCLR}$/VPP pin low.

14.4.1 WAKE-UP FROM SLEEP

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

- A POR reset
- External reset input on $\overline{MCLR}$/VPP pin
- WDT Reset (if WDT was enabled)
- Interrupt from RA0/INT pin, RB port change, T0CKI interrupt, or some Peripheral Interrupts

The following peripheral interrupts can wake-up from SLEEP:

- Capture1 interrupt
- Capture2 interrupt
- USART synchronous slave transmit interrupt
- USART synchronous slave receive interrupt

Other peripherals can not generate interrupts since during SLEEP, no on-chip Q clocks are present.

Any reset event will cause a device reset. Any interrupt event is considered a continuation of program execution. The $\overline{TO}$ and $\overline{PD}$ bits in the CPUSTA register can be used to determine the cause of device reset. The $\overline{PD}$ bit, which is set on power-up, is cleared when SLEEP is invoked. The $\overline{TO}$ bit is cleared if WDT time-out occurred (and caused wake-up).

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 GLINTD bit. If the GLINTD bit is set (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GLINTD bit is clear (enabled), the device executes the instruction after the SLEEP instruction and then branches to the interrupt vector address. In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

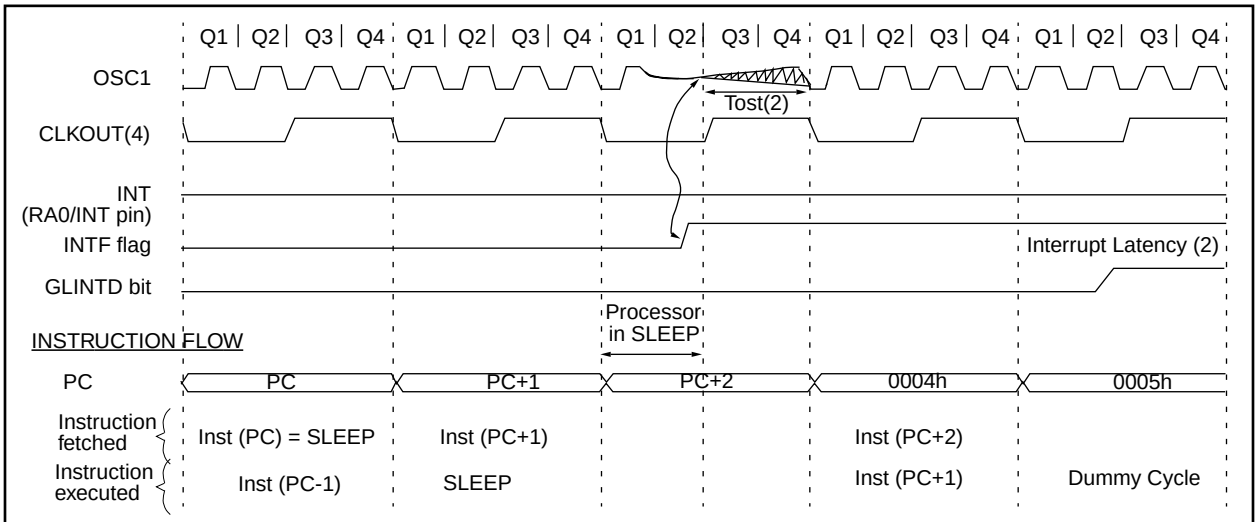
> **Note:** If the global interrupts are disabled (GLINTD is set), but any interrupt source has both its interrupt enable bit and the corresponding interrupt flag bits set, the device will immediately wake-up from sleep. The $\overline{TO}$ bit is set, and the $\overline{PD}$ bit is cleared.

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

14.4.1.1 WAKE-UP DELAY

When the oscillator type is configured in XT or LF mode, the Oscillator Start-up Timer (OST) is activated on wake-up. The OST will keep the device in reset for 1024TOSC. This needs to be taken into account when considering the interrupt response time when coming out of SLEEP.

FIGURE 14-9: WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note 1: XT or LF oscillator mode assumed.
2: Tost = 1024Tosc (drawing not to scale). This delay will not be there for RC osc mode.
3: When GLINTD = 0 processor jumps to interrupt routine after wake-up. If GLINTD = 1, execution will continue in line.
4: CLKOUT is not available in these osc modes, but shown here for timing reference.

ADDWFC — ADD WREG and Carry bit to f

Syntax: [*label*] ADDWFC f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

Operation: (WREG) + (f) + C → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0001	000d	ffff	ffff

Description: Add WREG, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed in data memory location 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write to destination

Example: `ADDWFC REG 0`

Before Instruction
Carry bit = 1
REG = 0x02
WREG = 0x4D

After Instruction
Carry bit = 0
REG = 0x02
WREG = 0x50

ANDLW — And Literal with WREG

Syntax: [*label*] ANDLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .AND. (k) → (WREG)

Status Affected: Z

Encoding:

1011	0101	kkkk	kkkk

Description: The contents of WREG are AND'ed with the 8-bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Execute	Write to WREG

Example: `ANDLW 0x5F`

Before Instruction
WREG = 0xA3

After Instruction
WREG = 0x03

DECF — Decrement f

Syntax: [*label*] DECF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

Operation: (f) – 1 → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	011d	ffff	ffff

Description: Decrement register 'f'. If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write to destination

Example: `DECF CNT, 1`

Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

DECFSZ — Decrement f, skip if 0

Syntax: [*label*] DECFSZ f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

Operation: (f) – 1 → (dest);
skip if result = 0

Status Affected: None

Encoding:

0001	011d	ffff	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0 the result is placed in WREG. 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, and an NOP is executed instead making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write to destination

Example:

```
HERE        DECFSZ   CNT, 1
            GOTO     LOOP
CONTINUE
```

Before Instruction
PC = Address (HERE)

After Instruction
CNT = CNT - 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE+1)

TABLRD **Table Read**

Example1: `TABLRD 1, 1, REG ;`

Before Instruction

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x55
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0x1234

After Instruction (table write completion)

REG	=	0xAA
TBLATH	=	0x12
TBLATL	=	0x34
TBLPTR	=	0xA357
MEMORY(TBLPTR)	=	0x5678

Example2: `TABLRD 0, 0, REG ;`

Before Instruction

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x55
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0x1234

After Instruction (table write completion)

REG	=	0x55
TBLATH	=	0x12
TBLATL	=	0x34
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0x1234

TABLWT **Table Write**

Syntax: [*label*] TABLWT t,i,f

Operands: $0 \le f \le 255$
$i \in [0,1]$
$t \in [0,1]$

Operation: If t = 0,
f → TBLATL;
If t = 1,
f → TBLATH;
TBLAT → Prog Mem (TBLPTR);
If i = 1,
TBLPTR + 1 → TBLPTR

Status Affected: None

Encoding:

1010	11ti	ffff	ffff

Description:

1. Load value in 'f' into 16-bit table latch (TBLAT)
   If t = 0: load into low byte;
   If t = 1: load into high byte
2. The contents of TBLAT is written to the program memory location pointed to by TBLPTR
   If TBLPTR points to external program memory location, then the instruction takes two-cycle
   If TBLPTR points to an internal EPROM location, then the instruction is terminated when an interrupt is received.

Note: The $\overline{MCLR}$/VPP pin must be at the programming voltage for successful programming of internal memory.
If $\overline{MCLR}$/VPP = VDD
the programming sequence of internal memory will be executed, but will not be successful (although the internal memory location may be disturbed)

3. The TBLPTR can be automatically incremented
   If i = 0; TBLPTR is not incremented
   If i = 1; TBLPTR is incremented

Words: 1

Cycles: 2 (many if write is to on-chip EPROM program memory)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register TBLATH or TBLATL

NOTES:

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 17-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

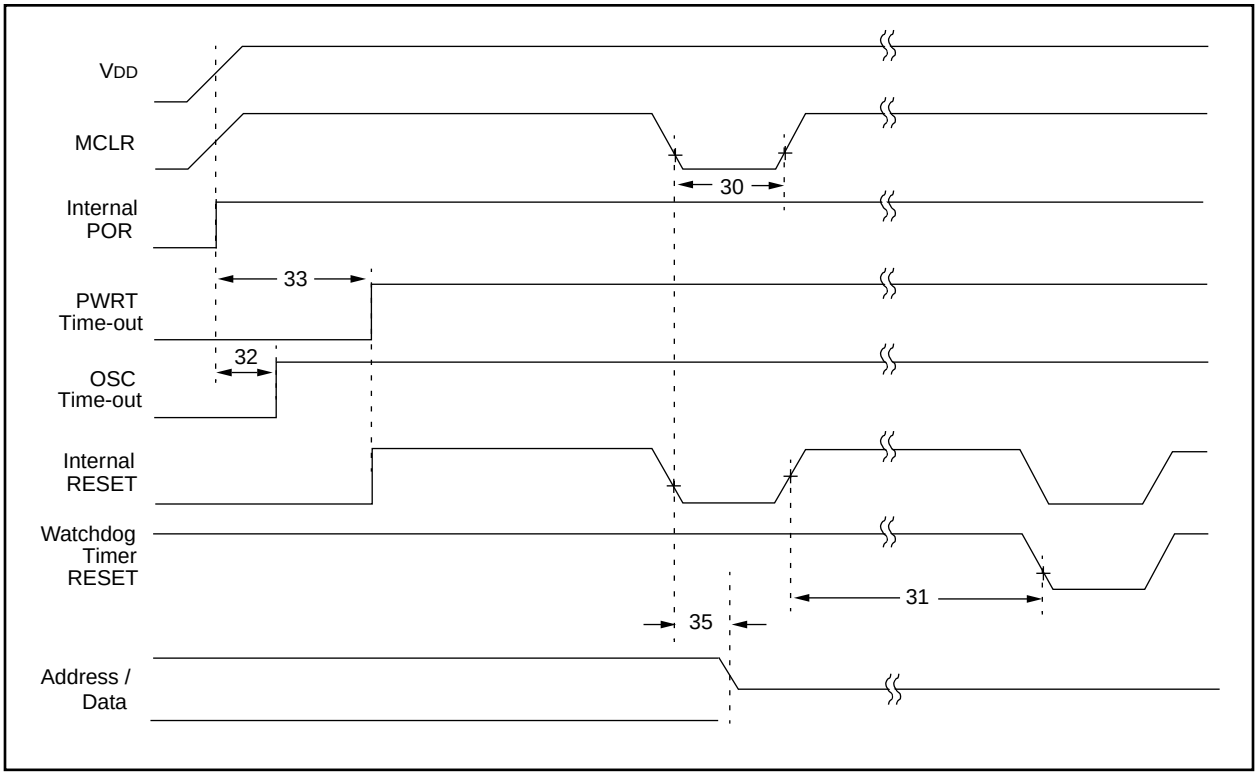


TABLE 17-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
30	TmcL	$\overline{\text{MCLR}}$ Pulse Width (low)	100 *	—	—	ns	
31	Twdt	Watchdog Timer Time-out Period (Prescale = 1)	5 *	12	25 *	ms	
32	Tost	Oscillation Start-up Timer Period		1024 TOSC §		ms	TOSC = OSC1 period
33	Tpwrt	Power-up Timer Period	40 *	96	200 *	ms	
35	TmcL2adl	$\overline{\text{MCLR}}$ to System Interface bus (AD15:AD0) invalid	—	—	100 *	ns	

* These parameters are characterized but not tested.

† Data in “Typ” column is at 5V, 25˚C unless otherwise stated. These parameters are for design guidance only and are not tested.

‡ These parameters are for design guidance only and are not tested, nor characterized.

§ This specification ensured by design.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 17-9: USART MODULE: SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

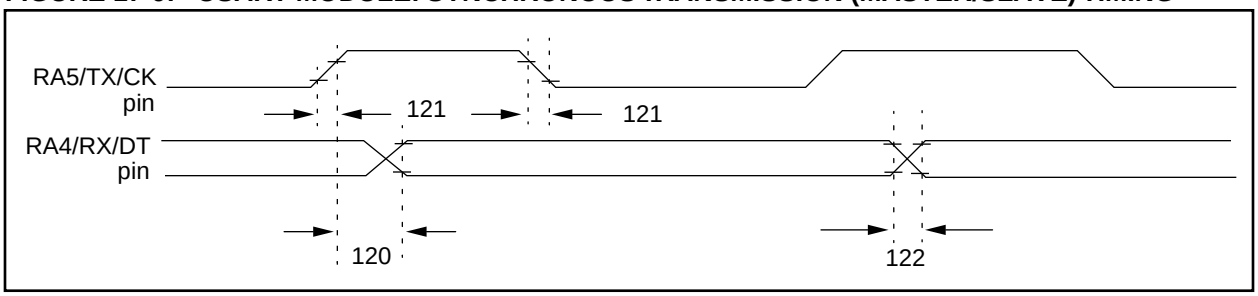


TABLE 17-9: SERIAL PORT SYNCHRONOUS TRANSMISSION REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
120	TckH2dtV	SYNC XMIT (MASTER & SLAVE) Clock high to data out valid	—	—	65	ns	
121	TckRF	Clock out rise time and fall time (Master Mode)	—	10	35	ns	
122	TdtRF	Data out rise time and fall time	—	10	35	ns	

† Data in “Typ” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 17-10: USART MODULE: SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

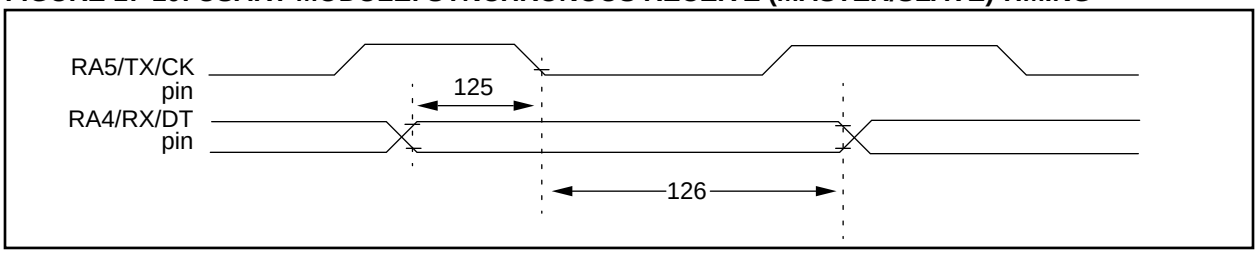


TABLE 17-10: SERIAL PORT SYNCHRONOUS RECEIVE REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
125	TdtV2ckL	SYNC RCV (MASTER & SLAVE) Data hold before CK↓ (DT hold time)	15	—	—	ns	
126	TckL2dtl	Data hold after CK↓ (DT hold time)	15	—	—	ns	

† Data in “Typ” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 19-11: MEMORY INTERFACE WRITE TIMING (NOT SUPPORTED IN PIC17LC4X DEVICES)

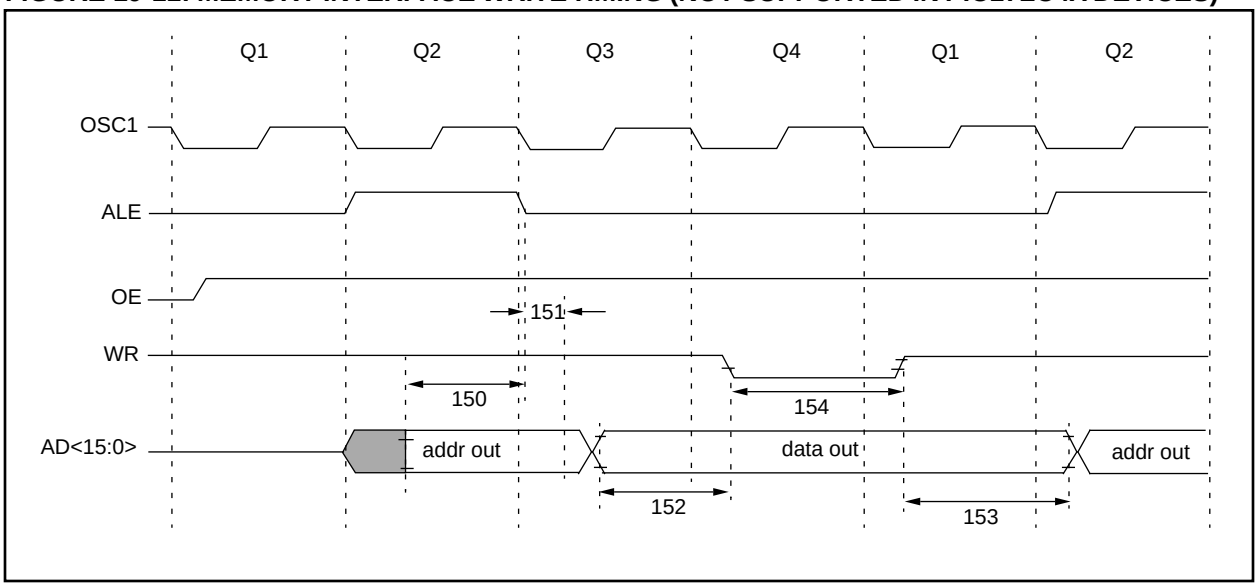


TABLE 19-11: MEMORY INTERFACE WRITE REQUIREMENTS (NOT SUPPORTED IN PIC17LC4X DEVICES)

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2alL	AD<15:0> (address) valid to ALE↓ (address setup time)	0.25Tcy - 10	—	—	ns	
151	TalL2adI	ALE↓ to address out invalid (address hold time)	0	—	—	ns	
152	TadV2wrL	Data out valid to $\overline{WR}$↓ (data setup time)	0.25Tcy - 40	—	—	ns	
153	TwrH2adI	$\overline{WR}$↑ to data out invalid (data hold time)	—	0.25TCY §	—	ns	
154	TwrL	$\overline{WR}$ pulse width	—	0.25TCY §	—	ns	

* These parameters are characterized but not tested.

† Data in “Typ” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

E.3 PIC16CXXX Family of Devices

	Clock	Memory			Peripherals				Features			
	Maximum Frequency of Operation (MHz)	Program Memory (X14 words)	EPROM	Data Memory (bytes)	Timer Module(s)	Comparator(s)	Internal Reference Voltage	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Brown-out Reset	Packages
PIC16C554	20	512	80	TMR0	—	—	3	13	2.5-6.0	—	18-pin DIP, SOIC; 20-pin SSOP	
PIC16C556	20	1K	80	TMR0	—	—	3	13	2.5-6.0	—	18-pin DIP, SOIC; 20-pin SSOP	
PIC16C558	20	2K	128	TMR0	—	—	3	13	2.5-6.0	—	18-pin DIP, SOIC; 20-pin SSOP	
PIC16C620	20	512	80	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP	
PIC16C621	20	1K	80	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP	
PIC16C622	20	2K	128	TMR0	2	Yes	4	13	2.5-6.0	Yes	18-pin DIP, SOIC; 20-pin SSOP	

All PIC16/17 Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

All PIC16C6XXX Family devices use serial programming with clock pin RB6 and data pin RB7.

Delay From External Clock Edge 68
Development Support 143
Development Tools 143
Device Drawings
44-Lead Plastic Surface Mount (MQFP
10x10 mm Body 1.6/0.15 mm Lead Form) 209
DIGIT BORROW 9
Digit Carry (DC) 9
Duty Cycle 75

E

Electrical Characteristics
PIC17C42
Absolute Maximum Ratings 147
Capture Timing 159
CLKOUT and I/O Timing 156
DC Characteristics 149
External Clock Timing 155
Memory Interface Read Timing 162
Memory Interface Write Timing 161
PWM Timing 159
RESET, Watchdog Timer, Oscillator Start-up
Timer and Power-up Timer 157
Timer0 Clock Timings 158
Timer1, Timer2 and Timer3 Clock Timing 158
USART Module, Synchronous Receive 160
USART Module, Synchronous Transmission ... 160
PIC17C43/44
Absolute Maximum Ratings 175
Capture Timing 188
CLKOUT and I/O Timing 185
DC Characteristics 177
External Clock Timing 184
Memory Interface Read Timing 191
Memory Interface Write Timing 190
Parameter Measurement Information 183
RESET, Watchdog Timer, Oscillator Start-up
Timer and Power-up Timer Timing 186
Timer0 Clock Timing 187
Timer1, Timer2 and Timer3 Clock Timing 187
Timing Parameter Symbology 182
USART Module Synchronous Receive
Timing 189
USART Module Synchronous Transmission
Timing 189
EPROM Memory Access Time Order Suffix 31
Extended Microcontroller 29
Extended Microcontroller Mode 31
External Memory Interface 31
External Program Memory Waveforms 31

F

Family of Devices 6
PIC14000 213
PIC16C5X 214
PIC16CXXX 215
PIC16C6X 216
PIC16C7X 217
PIC16C8X 218
PIC16C9XX 219
PIC17CXX 220
FERR 84, 91
FOSC0 99
FOSC1 99
FS0 36
FS1 36
FS2 36
FS3 36
FSR0 ..... 34, 40
FSR1 ..... 34, 40
Fuzzy Logic Dev. System (*fuzzy*TECH®-MP) 143, 145

G

General Format for Instructions 108
General Purpose RAM 29
General Purpose RAM Bank 42
General Purpose Register (GPR) 32
GLINTD 25, 37, 78, 105
GOTO 122
GPR (General Purpose Register) 32
Graphs
I_{OH} vs. V_{OH}, V_{DD} = 3V 170, 200
I_{OH} vs. V_{OH}, V_{DD} = 5V 171, 201
I_{OL} vs. V_{OL}, V_{DD} = 3V 171, 201
I_{OL} vs. V_{OL}, V_{DD} = 5V 172, 202
Maximum I_{DD} vs. Frequency
(External Clock 125°C to -40°C) 167, 197
Maximum I_{PD} vs. V_{DD} Watchdog Disabled 168, 198
Maximum I_{PD} vs. V_{DD} Watchdog Enabled 169, 199
RC Oscillator Frequency vs.
V_{DD} (Cext = 100 pF) 164, 194
RC Oscillator Frequency vs.
V_{DD} (Cext = 22 pF) 164, 194
RC Oscillator Frequency vs.
V_{DD} (Cext = 300 pF) 165, 195
Transconductance of LF Oscillator vs.V_{DD} 166, 196
Transconductance of XT Oscillator vs. V_{DD} 166, 196
Typical I_{DD} vs. Frequency
(External Clock 25°C) 167, 197
Typical I_{PD} vs. V_{DD} Watchdog Disabled 25°C . 168, 198
Typical I_{PD} vs. V_{DD} Watchdog Enabled 25°C .. 169, 199
Typical RC Oscillator vs. Temperature 163, 193
V_{TH} (Input Threshold Voltage) of I/O Pins vs.
V_{DD} 172, 202
V_{TH} (Input Threshold Voltage) of OSC1 Input
(In XT, HS, and LP Modes) vs. V_{DD} 173, 203
V_{TH}, V_{IL} of $\overline{MCLR}$, T0CKI and OSC1
(In RC Mode) vs. V_{DD} 173, 203
WDT Timer Time-Out Period vs. V_{DD} 170, 200

H

Hardware Multiplier 49

I

I/O Ports
Bi-directional 64
I/O Ports 53
Programming Considerations 64
Read-Modify-Write Instructions 64
Successive Operations 64
INCF 123
INCFSNZ 124
INCFSZ 123
INDF0 ..... 34, 40
INDF1 ..... 34, 40

MP-C C Compiler 145
MPSIM Software Simulator 143, 145
MULLW 129
Multiply Examples
  16 x 16 Routine 50
  16 x 16 Signed Routine 51
  8 x 8 Routine 49
  8 x 8 Signed Routine 49
MULWF 129

N

NEGW 130
NOP 130

O

OERR 84
Opcode Field Descriptions 107
OSC Selection 99
Oscillator
  Configuration 100
  Crystal 100
  External Clock 101
  External Crystal Circuit 102
  External Parallel Resonant Crystal Circuit 102
  External Series Resonant Crystal Circuit 102
  RC 102
  RC Frequencies 165, 195
Oscillator Start-up Time (Figure) 18
Oscillator Start-up Timer (OST) 15, 99
OST 15, 99
OV 9, 36
Overflow (OV) 9

P

Package Marking Information 210
Packaging Information 205
Parameter Measurement Information 154
PC (Program Counter) 41
PCH 41
PCL 34, 41, 108
PCLATH 34, 41
$\overline{\text{PD}}$ 37, 105
PEIE 22, 78
PEIF 22
Peripheral Bank 42
Peripheral Interrupt Enable 23
Peripheral Interrupt Request (PIR) 24
PICDEM-1 Low-Cost PIC16/17 Demo Board 143, 144
PICDEM-2 Low-Cost PIC16CXX Demo Board 143, 144
PICDEM-3 Low-Cost PIC16C9XXX Demo Board 144
PICMASTER® RT In-Circuit Emulator 143
PICSTART® Low-Cost Development System 143
PIE 19, 34, 92, 96, 98
Pin Compatible Devices 221
PIR 19, 34, 92, 96, 98
PM0 ..... 99, 106
PM1 ..... 99, 106
POP 27, 39
POR 15, 99
PORTA 19, 34, 53
PORTB 19, 34, 55
PORTC 19, 34, 58
PORTD 19, 34, 60
PORTE 19, 34, 62
Power-down Mode 105
Power-on Reset (POR) 15, 99
Power-up Timer (PWRT) 15, 99
PR1 ..... 20, 35
PR2 ..... 20, 35
PR3/CA1H 20
PR3/CA1L 20
PR3H/CA1H 35
PR3L/CA1L 35
Prescaler Assignments 69
PRO MATE® Universal Programmer 143
PRODH 20
PRODL 20
Program Counter (PC) 41
Program Memory
  External Access Waveforms 31
  External Connection Diagram 31
  Map 29
  Modes
    Extended Microcontroller 29
    Microcontroller 29
    Microprocessor 29
    Protected Microcontroller 29
  Operation 29
  Organization 29
  Transfers from Data Memory 43
Protected Microcontroller 29
PS0 ..... 38, 67
PS1 ..... 38, 67
PS2 ..... 38, 67
PS3 ..... 38, 67
PUSH 27, 39
PW1DCH 20, 35
PW1DCL 20, 35
PW2DCH 20, 35
PW2DCL 20, 35
PWM 71, 75
  Duty Cycle 76
  External Clock Source 76
  Frequency vs. Resolution 76
  Interrupts 76
  Max Resolution/Frequency for External
  Clock Input 77
  Output 75
  Periods 76
PWM1 72
PWM1ON 72, 75
PWM2 72
PWM2ON 72, 75
PWRT 15, 99

R

RA1/T0CKI pin 67
RBIE 23
RBIF 24
$\overline{\text{RBPU}}$ 55
RC Oscillator 102
RC Oscillator Frequencies 165, 195
RCIE 23
RCIF 24
RCREG 19, 34, 91, 92, 96, 97
RCSTA 19, 34, 92, 96, 98
Reading 16-bit Value 69

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Trademarks

Printed on recycled paper.

DNV Certification, Inc. USA
ANSI • RAB QMS ACCREDITED
DNV MSC The Netherlands Accredited by the RvA
DNV
ISO 9001 / QS-9000 REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.