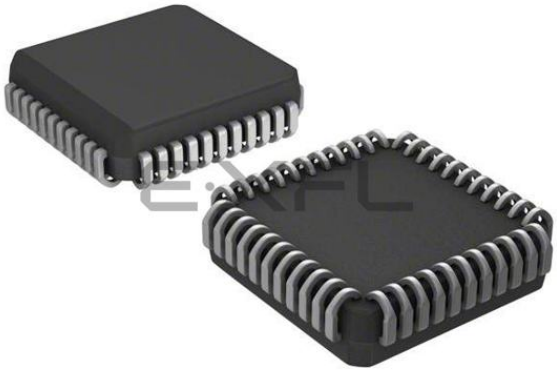


Welcome to E-XFL.COM

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	16MHz
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-16e-l

1.0 OVERVIEW

This data sheet covers the PIC17C4X group of the PIC17CXX family of microcontrollers. The following devices are discussed in this data sheet:

- PIC17C42
- PIC17CR42
- PIC17C42A
- PIC17C43
- PIC17CR43
- PIC17C44

The PIC17CR42, PIC17C42A, PIC17C43, PIC17CR43, and PIC17C44 devices include architectural enhancements over the PIC17C42. These enhancements will be discussed throughout this data sheet.

The PIC17C4X devices are 40/44-Pin, EPROM/ROM-based members of the versatile PIC17CXX family of low-cost, high-performance, CMOS, fully-static, 8-bit microcontrollers.

All PIC16/17 microcontrollers employ an advanced RISC architecture. The PIC17CXX has enhanced core features, 16-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 16-bit wide instruction word with a separate 8-bit wide data. The two stage instruction pipeline allows all instructions to execute in a single cycle, except for program branches (which require two cycles). A total of 55 instructions (reduced instruction set) are available in the PIC17C42 and 58 instructions in all the other devices. Additionally, a large register set gives some of the architectural innovations used to achieve a very high performance. For mathematical intensive applications all devices, except the PIC17C42, have a single cycle 8 x 8 Hardware Multiplier.

PIC17CXX microcontrollers typically achieve a 2:1 code compression and a 4:1 speed improvement over other 8-bit microcontrollers in their class.

PIC17C4X devices have up to 454 bytes of RAM and 33 I/O pins. In addition, the PIC17C4X adds several peripheral features useful in many high performance applications including:

- Four timer/counters
- Two capture inputs
- Two PWM outputs
- A Universal Synchronous Asynchronous Receiver Transmitter (USART)

These special features reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. There are four oscillator options, of which the single pin RC oscillator provides a low-cost solution, the LF oscillator is for low frequency crystals and minimizes power consumption, XT is a standard crystal, and the EC is for external clock input. The SLEEP (power-down) mode offers additional power saving. The user can wake-up the chip from SLEEP through several external and internal interrupts and device resets.

There are four configuration options for the device operational modes:

- Microprocessor
- Microcontroller
- Extended microcontroller
- Protected microcontroller

The microprocessor and extended microcontroller modes allow up to 64K-words of external program memory.

A highly reliable Watchdog Timer with its own on-chip RC oscillator provides protection against software malfunction.

Table 1-1 lists the features of the PIC17C4X devices.

A UV-erasable CERDIP-packaged version is ideal for code development while the cost-effective One-Time Programmable (OTP) version is suitable for production in any volume.

The PIC17C4X fits perfectly in applications ranging from precise motor control and industrial process control to automotive, instrumentation, and telecom applications. Other applications that require extremely fast execution of complex software programs or the flexibility of programming the software code as one of the last steps of the manufacturing process would also be well suited. The EPROM technology makes customization of application programs (with unique security codes, combinations, model numbers, parameter storage, etc.) fast and convenient. Small footprint package options make the PIC17C4X ideal for applications with space limitations that require high performance. High speed execution, powerful peripheral features, flexible I/O, and low power consumption all at low cost make the PIC17C4X ideal for a wide range of embedded control applications.

1.1 Family and Upward Compatibility

Those users familiar with the PIC16C5X and PIC16CXX families of microcontrollers will see the architectural enhancements that have been implemented. These enhancements allow the device to be more efficient in software and hardware requirements. Please refer to Appendix A for a detailed list of enhancements and modifications. Code written for PIC16C5X or PIC16CXX can be easily ported to PIC17CXX family of devices (Appendix B).

1.2 Development Support

The PIC17CXX family is supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a universal programmer, a “C” compiler, and fuzzy logic support tools.

2.0 PIC17C4X DEVICE VARIETIES

A variety of frequency ranges and packaging options are available. Depending on application and production requirements, the proper device option can be selected using the information in the PIC17C4X Product Selection System section at the end of this data sheet. When placing orders, please use the "PIC17C4X Product Identification System" at the back of this data sheet to specify the correct part number.

For the PIC17C4X family of devices, there are four device "types" as indicated in the device number:

1. **C**, as in PIC17**C**42. These devices have EPROM type memory and operate over the standard voltage range.
2. **LC**, as in PIC17**LC**42. These devices have EPROM type memory, operate over an extended voltage range, and reduced frequency range.
3. **CR**, as in PIC17**CR**42. These devices have ROM type memory and operate over the standard voltage range.
4. **LCR**, as in PIC17**LCR**42. These devices have ROM type memory, operate over an extended voltage range, and reduced frequency range.

2.1 UV Erasable Devices

The UV erasable version, offered in CERDIP package, is optimal for prototype development and pilot programs.

The UV erasable version can be erased and reprogrammed to any of the configuration modes. Microchip's PRO MATE™ programmer supports programming of the PIC17C4X. Third party programmers also are available; refer to the *Third Party Guide* for a list of sources.

2.2 One-Time-Programmable (OTP) Devices

The availability of OTP devices is especially useful for customers expecting frequent code changes and updates.

The OTP devices, packaged in plastic packages, permit the user to program them once. In addition to the program memory, the configuration bits must also be programmed.

2.3 Quick-Turnaround-Production (QTP) Devices

Microchip offers a QTP Programming Service for factory production orders. This service is made available for users who choose not to program a medium to high quantity of units and whose code patterns have stabilized. The devices are identical to the OTP devices but with all EPROM locations and configuration options already programmed by the factory. Certain code and prototype verification procedures apply before production shipments are available. Please contact your local Microchip Technology sales office for more details.

2.4 Serialized Quick-Turnaround Production (SQTP^{SM}) Devices

Microchip offers a unique programming service where a few user-defined locations in each device are programmed with different serial numbers. The serial numbers may be random, pseudo-random or sequential.

Serial programming allows each device to have a unique number which can serve as an entry-code, password or ID number.

ROM devices do not allow serialization information in the program memory space.

For information on submitting ROM code, please contact your regional sales office.

2.5 Read Only Memory (ROM) Devices

Microchip offers masked ROM versions of several of the highest volume parts, thus giving customers a low cost option for high volume, mature products.

For information on submitting ROM code, please contact your regional sales office.

TABLE 3-1: PINOUT DESCRIPTIONS

Name	DIP No.	PLCC No.	QFP No.	I/O/P Type	Buffer Type	Description
						PORTD is a bi-directional I/O Port.
RD0/AD8	40	43	15	I/O	TTL	This is also the upper byte of the 16-bit system bus in microprocessor mode or extended microprocessor mode or extended microcontroller mode. In multiplexed system bus configuration these pins are address output as well as data input or output.
RD1/AD9	39	42	14	I/O	TTL	
RD2/AD10	38	41	13	I/O	TTL	
RD3/AD11	37	40	12	I/O	TTL	
RD4/AD12	36	39	11	I/O	TTL	
RD5/AD13	35	38	10	I/O	TTL	
RD6/AD14	34	37	9	I/O	TTL	
RD7/AD15	33	36	8	I/O	TTL	
						PORTE is a bi-directional I/O Port.
RE0/ALE	30	32	4	I/O	TTL	In microprocessor mode or extended microcontroller mode, it is the Address Latch Enable (ALE) output. Address should be latched on the falling edge of ALE output.
RE1/$\overline{OE}$	29	31	3	I/O	TTL	In microprocessor or extended microcontroller mode, it is the Output Enable ($\overline{OE}$) control output (active low).
RE2/$\overline{WR}$	28	30	2	I/O	TTL	In microprocessor or extended microcontroller mode, it is the Write Enable ($\overline{WR}$) control output (active low).
TEST	27	29	1	I	ST	Test mode selection control input. Always tie to VSS for normal operation.
VSS	10, 31	11, 12, 33, 34	5, 6, 27, 28	P		Ground reference for logic and I/O pins.
VDD	1	1, 44	16, 17	P		Positive supply for logic and I/O pins.

Legend: I = Input only; O = Output only; I/O = Input/Output; P = Power; — = Not Used; TTL = TTL input; ST = Schmitt Trigger input.

5.1 Interrupt Status Register (INTSTA)

The Interrupt Status/Control register (INTSTA) records the individual interrupt requests in flag bits, and contains the individual interrupt enable bits (not for the peripherals).

The PEIF bit is a read only, bit wise OR of all the peripheral flag bits in the PIR register (Figure 5-4).

Note:	T0IF, INTF, T0CKIF, or PEIF will be set by the specified condition, even if the corresponding interrupt enable bit is clear (interrupt disabled) or the GLINTD bit is set (all interrupts disabled).

Care should be taken when clearing any of the INTSTA register enable bits when interrupts are enabled (GLINTD is clear). If any of the INTSTA flag bits (T0IF, INTF, T0CKIF, or PEIF) are set in the same instruction cycle as the corresponding interrupt enable bit is cleared, the device will vector to the reset address (0x00).

When disabling any of the INTSTA enable bits, the GLINTD bit should be set (disabled).

FIGURE 5-2: INTSTA REGISTER (ADDRESS: 07h, UNBANKED)

R - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0	R/W - 0
PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE
bit7							bit0

R = Readable bit
W = Writable bit
- n = Value at POR reset

bit 7: **PEIF**: Peripheral Interrupt Flag bit
This bit is the OR of all peripheral interrupt flag bits AND'ed with their corresponding enable bits.
1 = A peripheral interrupt is pending
0 = No peripheral interrupt is pending

bit 6: **T0CKIF**: External Interrupt on T0CKI Pin Flag bit
This bit is cleared by hardware, when the interrupt logic forces program execution to vector (18h).
1 = The software specified edge occurred on the RA1/T0CKI pin
0 = The software specified edge did not occur on the RA1/T0CKI pin

bit 5: **T0IF**: TMR0 Overflow Interrupt Flag bit
This bit is cleared by hardware, when the interrupt logic forces program execution to vector (10h).
1 = TMR0 overflowed
0 = TMR0 did not overflow

bit 4: **INTF**: External Interrupt on INT Pin Flag bit
This bit is cleared by hardware, when the interrupt logic forces program execution to vector (08h).
1 = The software specified edge occurred on the RA0/INT pin
0 = The software specified edge did not occur on the RA0/INT pin

bit 3: **PEIE**: Peripheral Interrupt Enable bit
This bit enables all peripheral interrupts that have their corresponding enable bits set.
1 = Enable peripheral interrupts
0 = Disable peripheral interrupts

bit 2: **T0CKIE**: External Interrupt on T0CKI Pin Enable bit
1 = Enable software specified edge interrupt on the RA1/T0CKI pin
0 = Disable interrupt on the RA1/T0CKI pin

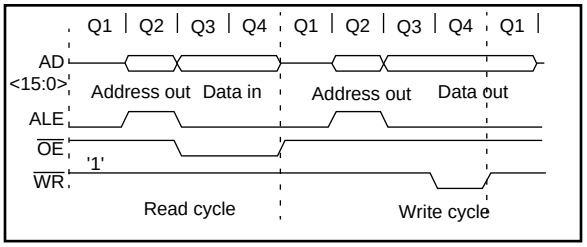
bit 1: **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enable TMR0 overflow interrupt
0 = Disable TMR0 overflow interrupt

bit 0: **INTE**: External Interrupt on RA0/INT Pin Enable bit
1 = Enable software specified edge interrupt on the RA0/INT pin
0 = Disable software specified edge interrupt on the RA0/INT pin

6.1.2 EXTERNAL MEMORY INTERFACE

When either microprocessor or extended microcontroller mode is selected, PORTC, PORTD and PORTE are configured as the system bus. PORTC and PORTD are the multiplexed address/data bus and PORTE is for the control signals. External components are needed to demultiplex the address and data. This can be done as shown in Figure 6-4. The waveforms of address and data are shown in Figure 6-3. For complete timings, please refer to the electrical specification section.

FIGURE 6-3: EXTERNAL PROGRAM MEMORY ACCESS WAVEFORMS



The system bus requires that there is no bus conflict (minimal leakage), so the output value (address) will be capacitively held at the desired value.

As the speed of the processor increases, external EPROM memory with faster access time must be used. Table 6-2 lists external memory speed requirements for a given PIC17C4X device frequency.

In extended microcontroller mode, when the device is executing out of internal memory, the control signals will continue to be active. That is, they indicate the action that is occurring in the internal memory. The external memory access is ignored.

This following selection is for use with Microchip EPROMs. For interfacing to other manufacturers memory, please refer to the electrical specifications of the desired PIC17C4X device, as well as the desired memory device to ensure compatibility.

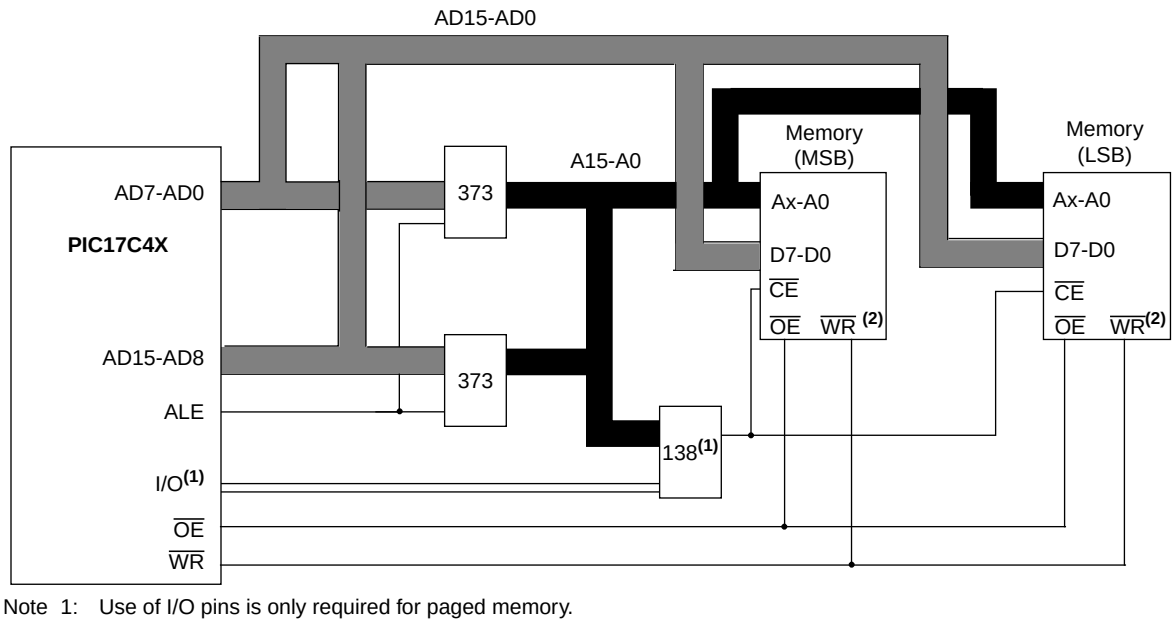
TABLE 6-2: EPROM MEMORY ACCESS TIME ORDERING SUFFIX

PIC17C4X Oscillator Frequency	Instruction Cycle Time (TCY)	EPROM Suffix	
		PIC17C42	PIC17C43 PIC17C44
8 MHz	500 ns	-25	-25
16 MHz	250 ns	-12	-15
20 MHz	200 ns	-90	-10
25 MHz	160 ns	N.A.	-70
33 MHz	121 ns	N.A.	(1)

Note 1: The access times for this requires the use of fast SRAMS.

Note: The external memory interface is not supported for the LC devices.

FIGURE 6-4: TYPICAL EXTERNAL PROGRAM MEMORY CONNECTION DIAGRAM



Note 1: Use of I/O pins is only required for paged memory.
2: This signal is unused for ROM and EPROM devices.

TABLE 6-3: SPECIAL FUNCTION REGISTERS

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 (3)
Unbanked											
00h	INDF0	Uses contents of FSR0 to address data memory (not a physical register)								---- ----	---- ----
01h	FSR0	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8-bits of PC								0000 0000	0000 0000
03h(1)	PCLATH	Holding register for upper 8-bits of PC								0000 0000	uuuu uuuu
04h	ALUSTA	FS3	FS2	FS1	FS0	OV	Z	DC	C	1111 xxxx	1111 uuuu
05h	T0STA	INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—	0000 000-	0000 000-
06h(2)	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	—	—	--11 11--	--11 qq--
07h	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
08h	INDF1	Uses contents of FSR1 to address data memory (not a physical register)								---- ----	---- ----
09h	FSR1	Indirect data memory address pointer 1								xxxx xxxx	uuuu uuuu
0Ah	WREG	Working register								xxxx xxxx	uuuu uuuu
0Bh	TMR0L	TMR0 register; low byte								xxxx xxxx	uuuu uuuu
0Ch	TMR0H	TMR0 register; high byte								xxxx xxxx	uuuu uuuu
0Dh	TBLPTRL	Low byte of program memory table pointer								(4)	(4)
0Eh	TBLPTRH	High byte of program memory table pointer								(4)	(4)
0Fh	BSR	Bank select register								0000 0000	0000 0000
Bank 0											
10h	PORTA	$\overline{RBPU}$	—	RA5	RA4	RA3	RA2	RA1/T0CKI	RA0/INT	0-xx xxxx	0-uu uuuu
11h	DDRB	Data direction register for PORTB								1111 1111	1111 1111
12h	PORTB	PORTB data latch								xxxx xxxx	uuuu uuuu
13h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h	RCREG	Serial port receive register								xxxx xxxx	uuuu uuuu
15h	TXSTA	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
16h	TXREG	Serial port transmit register								xxxx xxxx	uuuu uuuu
17h	SPBRG	Baud rate generator register								xxxx xxxx	uuuu uuuu
Bank 1											
10h	DDRC	Data direction register for PORTC								1111 1111	1111 1111
11h	PORTC	RC7/ AD7	RC6/ AD6	RC5/ AD5	RC4/ AD4	RC3/ AD3	RC2/ AD2	RC1/ AD1	RC0/ AD0	xxxx xxxx	uuuu uuuu
12h	DDRD	Data direction register for PORTD								1111 1111	1111 1111
13h	PORTD	RD7/ AD15	RD6/ AD14	RD5/ AD13	RD4/ AD12	RD3/ AD11	RD2/ AD10	RD1/ AD9	RD0/ AD8	xxxx xxxx	uuuu uuuu
14h	DDRE	Data direction register for PORTE								---- -111	---- -111
15h	PORTE	—	—	—	—	—	RE2/$\overline{WR}$	RE1/$\overline{OE}$	RE0/ALE	---- -xxx	---- -uuu
16h	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q - value depends on condition. Shaded cells are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for PC<15:8> whose contents are updated from or transferred to the upper byte of the program counter.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in CPUSTA are not affected by a MCLR reset.

3: Other (non power-up) resets include: external reset through MCLR and the Watchdog Timer Reset.

4: The following values are for both TBLPTRL and TBLPTRH:
All PIC17C4X devices (Power-on Reset 0000 0000) and (All other resets 0000 0000)
except the PIC17C42 (Power-on Reset xxxx xxxx) and (All other resets uuuu uuuu)

5: The PRODL and PRODH registers are not implemented on the PIC17C42.

6.3 Stack Operation

The PIC17C4X devices have a 16 x 16-bit wide hardware stack (Figure 6-1). The stack is not part of either the program or data memory space, and the stack pointer is neither readable nor writable. The PC is "PUSHed" onto the stack when a CALL instruction is executed or an interrupt is acknowledged. The stack is "POPed" in the event of a RETURN, RETLW, or a RETFIE instruction execution. PCLATH is not affected by a "PUSH" or a "POP" operation.

The stack operates as a circular buffer, with the stack pointer initialized to '0' after all resets. There is a stack available bit (STKAV) to allow software to ensure that the stack has not overflowed. The STKAV bit is set after a device reset. When the stack pointer equals Fh, STKAV is cleared. When the stack pointer rolls over from Fh to 0h, the STKAV bit will be held clear until a device reset.

Note 1: There is not a status bit for stack underflow. The STKAV bit can be used to detect the underflow which results in the stack pointer being at the top of stack.

Note 2: There are no instruction mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, RETURN, RETLW, and RETFIE instructions, or the vectoring to an interrupt vector.

Note 3: After a reset, if a "POP" operation occurs before a "PUSH" operation, the STKAV bit will be cleared. This will appear as if the stack is full (underflow has occurred). If a "PUSH" operation occurs next (before another "POP"), the STKAV bit will be locked clear. Only a device reset will cause this bit to set.

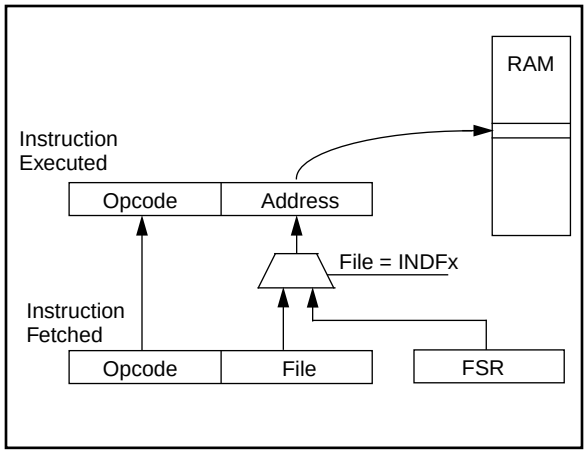
After the device is "PUSHed" sixteen times (without a "POP"), the seventeenth push overwrites the value from the first push. The eighteenth push overwrites the second push (and so on).

6.4 Indirect Addressing

Indirect addressing is a mode of addressing data memory where the data memory address in the instruction is not fixed. That is, the register that is to be read or written can be modified by the program. This can be useful for data tables in the data memory. Figure 6-10 shows the operation of indirect addressing. This shows the moving of the value to the data memory address specified by the value of the FSR register.

Example 6-1 shows the use of indirect addressing to clear RAM in a minimum number of instructions. A similar concept could be used to move a defined number of bytes (block) of data to the USART transmit register (TXREG). The starting address of the block of data to be transmitted could easily be modified by the program.

FIGURE 6-10: INDIRECT ADDRESSING



6.4.1 INDIRECT ADDRESSING REGISTERS

The PIC17C4X has four registers for indirect addressing. These registers are:

- INDF0 and FSR0
- INDF1 and FSR1

Registers INDF0 and INDF1 are not physically implemented. Reading or writing to these registers activates indirect addressing, with the value in the corresponding FSR register being the address of the data. The FSR is an 8-bit register and allows addressing anywhere in the 256-byte data memory address range. For banked memory, the bank of memory accessed is specified by the value in the BSR.

If file INDF0 (or INDF1) itself is read indirectly via an FSR, all '0's are read (Zero bit is set). Similarly, if INDF0 (or INDF1) is written to indirectly, the operation will be equivalent to a NOP, and the status bits are not affected.

6.4.2 INDIRECT ADDRESSING OPERATION

The indirect addressing capability has been enhanced over that of the PIC16CXX family. There are two control bits associated with each FSR register. These two bits configure the FSR register to:

- Auto-decrement the value (address) in the FSR after an indirect access
- Auto-increment the value (address) in the FSR after an indirect access
- No change to the value (address) in the FSR after an indirect access

These control bits are located in the ALUSTA register. The FSR1 register is controlled by the FS3:FS2 bits and FSR0 is controlled by the FS1:FS0 bits.

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the ALUSTA register. For example, if the indirect address causes the FSR to equal '0', the Z bit will not be set.

If the FSR register contains a value of 0h, an indirect read will read 0h (Zero bit is set) while an indirect write will be equivalent to a NOP (status bits are not affected).

Indirect addressing allows single cycle data transfers within the entire data space. This is possible with the use of the MOVPF and MOVFP instructions, where either 'p' or 'f' is specified as INDF0 (or INDF1).

If the source or destination of the indirect address is in banked memory, the location accessed will be determined by the value in the BSR.

A simple program to clear RAM from 20h - FFh is shown in Example 6-1.

EXAMPLE 6-1: INDIRECT ADDRESSING

```
    MOVLW    0x20          ;
    MOVWF    FSR0          ; FSR0 = 20h
    BCF      ALUSTA, FS1   ; Increment FSR
    BSF      ALUSTA, FS0   ; after access
    BCF      ALUSTA, C     ; C = 0
    MOVLW    END_RAM + 1   ;
LP  CLRF     INDF0         ; Addr(FSR) = 0
    CPFSEQ   FSR0          ; FSR0 = END_RAM+1?
    GOTO     LP            ; NO, clear next
    :                      ; YES, All RAM is
    :                      ; cleared
```

6.5 Table Pointer (TBLPTRL and TBLPTRH)

File registers TBLPTRL and TBLPTRH form a 16-bit pointer to address the 64K program memory space. The table pointer is used by instructions TABLWT and TABLRD.

The TABLRD and the TABLWT instructions allow transfer of data between program and data space. The table pointer serves as the 16-bit address of the data word within the program memory. For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 7.0.

6.6 Table Latch (TBLATH, TBLATL)

The table latch (TBLAT) is a 16-bit register, with TBLATH and TBLATL referring to the high and low bytes of the register. It is not mapped into data or program memory. The table latch is used as a temporary holding latch during data transfer between program and data memory (see descriptions of instructions TABLRD, TABLWT, TLRD and TLWT). For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 7.0.

6.7 Program Counter Module

The Program Counter (PC) is a 16-bit register. PCL, the low byte of the PC, is mapped in the data memory. PCL is readable and writable just as is any other register. PCH is the high byte of the PC and is not directly addressable. Since PCH is not mapped in data or program memory, an 8-bit register PCLATH (PC high latch) is used as a holding latch for the high byte of the PC. PCLATH is mapped into data memory. The user can read or write PCH through PCLATH.

The 16-bit wide PC is incremented after each instruction fetch during Q1 unless:

- Modified by GOTO, CALL, LCALL, RETURN, RETLW, or RETFIE instruction
- Modified by an interrupt response
- Due to destination write to PCL by an instruction

"Skips" are equivalent to a forced NOP cycle at the skipped address.

Figure 6-11 and Figure 6-12 show the operation of the program counter for various situations.

FIGURE 6-11: PROGRAM COUNTER OPERATION

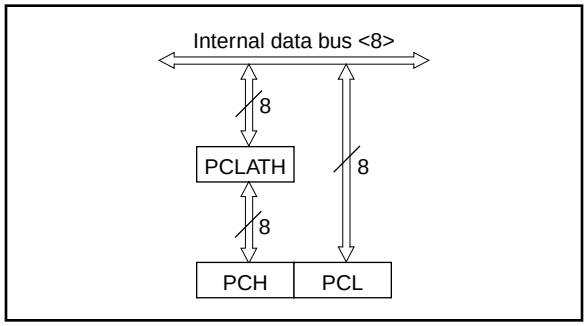
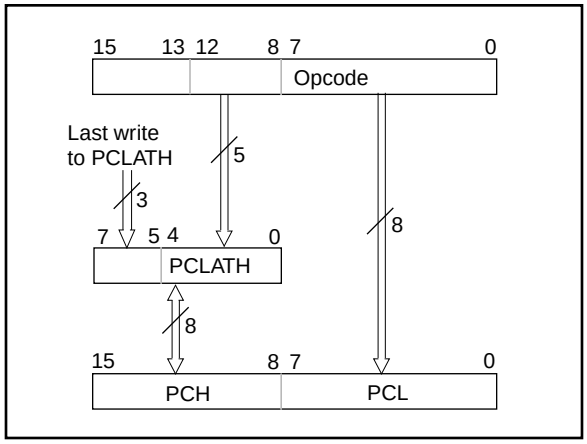


FIGURE 6-12: PROGRAM COUNTER USING THE CALL AND GOTO INSTRUCTIONS



Using Figure 6-11, the operations of the PC and PCLATH for different instructions are as follows:

a) LCALL instructions:
   An 8-bit destination address is provided in the instruction (opcode). PCLATH is unchanged.
   PCLATH → PCH
   Opcode<7:0> → PCL

b) Read instructions on PCL:
   Any instruction that reads PCL.
   PCL → data bus → ALU or destination
   PCH → PCLATH

c) Write instructions on PCL:
   Any instruction that writes to PCL.
   8-bit data → data bus → PCL
   PCLATH → PCH

d) Read-Modify-Write instructions on PCL:
   Any instruction that does a read-write-modify operation on PCL, such as ADDWF PCL.
   Read: PCL → data bus → ALU
   Write: 8-bit result → data bus → PCL
   PCLATH → PCH

e) RETURN instruction:
   PCH → PCLATH
   Stack<MRU> → PC<15:0>

Using Figure 6-12, the operation of the PC and PCLATH for GOTO and CALL instructions is a follows:

CALL, GOTO instructions:
A 13-bit destination address is provided in the instruction (opcode).
Opcode<12:0> → PC <12:0>
PC<15:13> → PCLATH<7:5>
Opcode<12:8> → PCLATH <4:0>

The read-modify-write only affects the PCL with the result. PCH is loaded with the value in the PCLATH. For example, ADDWF PCL will result in a jump within the current page. If PC = 03F0h, WREG = 30h and PCLATH = 03h before instruction, PC = 0320h after the instruction. To accomplish a true 16-bit computed jump, the user needs to compute the 16-bit destination address, write the high byte to PCLATH and then write the low value to PCL.

The following PC related operations do not change PCLATH:

a) LCALL, RETLW, and RETFIE instructions.
b) Interrupt vector is forced onto the PC.
c) Read-modify-write instructions on PCL (e.g. BSF PCL).

7.3 Table Reads

The table read allows the program memory to be read. This allows constant data to be stored in the program memory space, and retrieved into data memory when needed. Example 7-2 reads the 16-bit value at program memory address TBLPTR. After the dummy byte has been read from the TABLATH, the TABLATH is loaded with the 16-bit data from program memory address TBLPTR + 1. The first read loads the data into the latch, and can be considered a dummy read (unknown data loaded into 'f'). INDF0 should be configured for either auto-increment or auto-decrement.

EXAMPLE 7-2: TABLE READ

```
MOVLW   HIGH (TBL_ADDR) ; Load the Table
MOVWF   TBLPTRH         ;   address
MOVLW   LOW (TBL_ADDR)  ;
MOVWF   TBLPTRL         ;
TABLRD  0,0,DUMMY       ; Dummy read,
                        ;  Updates TABLATCH
TLRD    1, INDF0        ; Read HI byte
                        ;   of TABLATCH
TABLRD  0,1,INDF0       ; Read LO byte
                        ;   of TABLATCH and
                        ;   Update TABLATCH
```

FIGURE 7-7: TABLRD TIMING

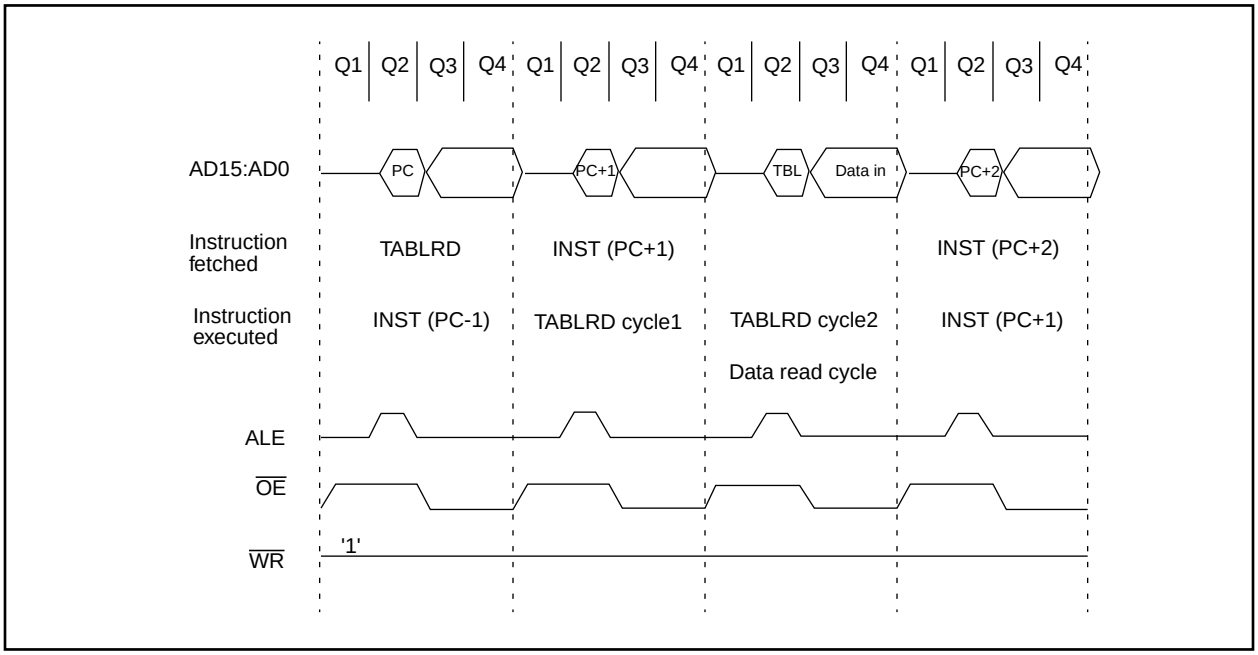
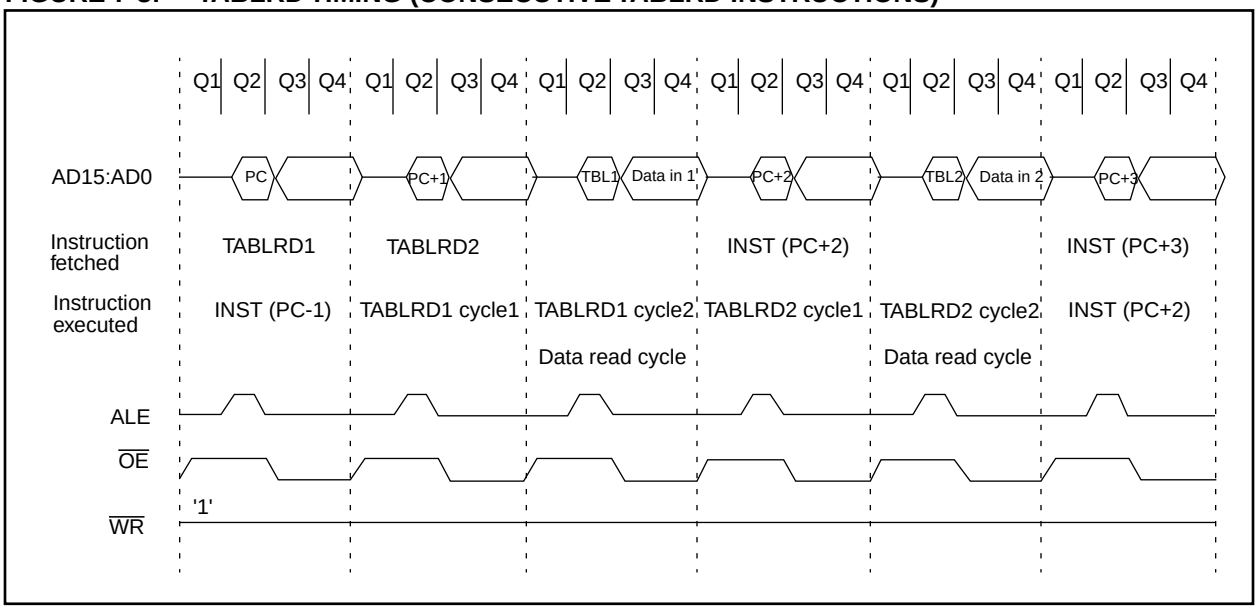


FIGURE 7-8: TABLRD TIMING (CONSECUTIVE TABLRD INSTRUCTIONS)



11.3 Read/Write Consideration for TMR0

Although TMR0 is a 16-bit timer/counter, only 8-bits at a time can be read or written during a single instruction cycle. Care must be taken during any read or write.

11.3.1 READING 16-BIT VALUE

The problem in reading the entire 16-bit value is that after reading the low (or high) byte, its value may change from FFh to 00h.

Example 11-1 shows a 16-bit read. To ensure a proper read, interrupts must be disabled during this routine.

EXAMPLE 11-1: 16-BIT READ

```
MOVPF   TMR0L, TMPLO    ;read low tmr0
MOVPF   TMR0H, TMPHI    ;read high tmr0
MOVFP   TMPLO, WREG     ;tmplo –> wreg
CPFSLT  TMR0L           ;tmr0l < wreg?
RETURN                  ;no then return
MOVPF   TMR0L, TMPLO    ;read low tmr0
MOVPF   TMR0H, TMPHI    ;read high tmr0
RETURN                  ;return
```

11.3.2 WRITING A 16-BIT VALUE TO TMR0

Since writing to either TMR0L or TMR0H will effectively inhibit increment of that half of the TMR0 in the next cycle (following write), but not inhibit increment of the other half, the user must write to TMR0L first and TMR0H next in two consecutive instructions, as shown in Example 11-2. The interrupt must be disabled. Any write to either TMR0L or TMR0H clears the prescaler.

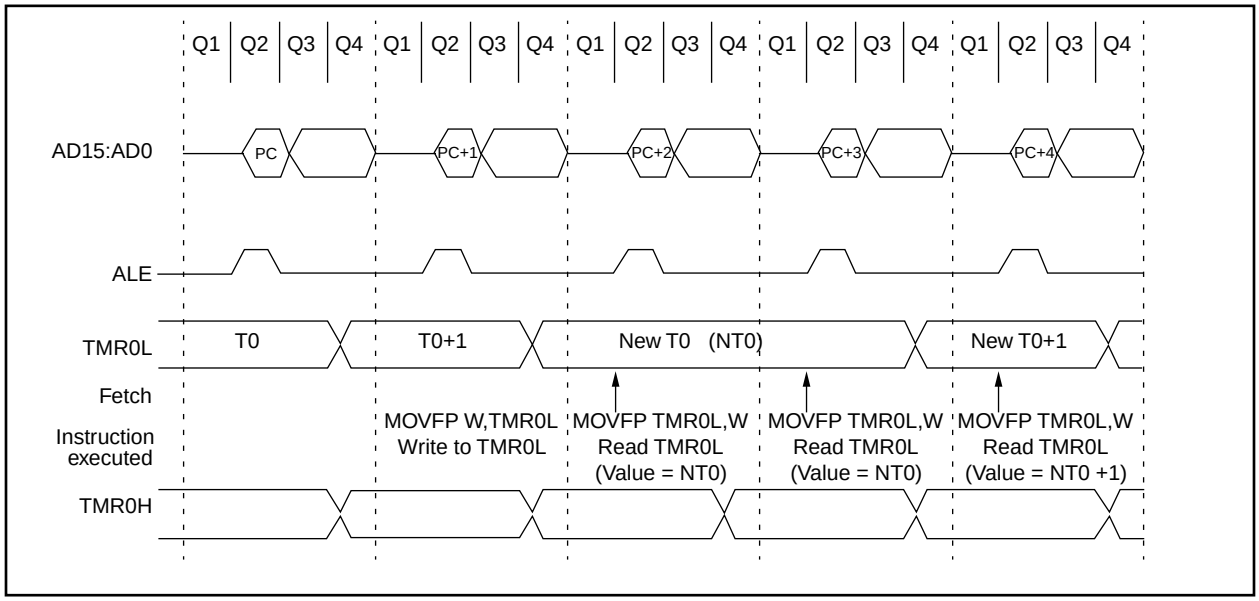
EXAMPLE 11-2: 16-BIT WRITE

```
BSF   CPUSTA, GLINTD ; Disable interrupt
MOVFP RAM_L, TMR0L   ;
MOVFP RAM_H, TMR0H   ;
BCF   CPUSTA, GLINTD ; Done, enable interrupt
```

11.4 Prescaler Assignments

Timer0 has an 8-bit prescaler. The prescaler assignment is fully under software control; i.e., it can be changed “on the fly” during program execution. When changing the prescaler assignment, clearing the prescaler is recommended before changing assignment. The value of the prescaler is “unknown,” and assigning a value that is less then the present value makes it difficult to take this unknown time into account.

FIGURE 11-4: TMR0 TIMING: WRITE HIGH OR LOW BYTE



13.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 13-4. The data comes in the RA4/RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter operating at 16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC.

Once asynchronous mode is selected, reception is enabled by setting bit CREN (RCSTA<4>).

The heart of the receiver is the receive (serial) shift register (RSR). After sampling the stop bit, the received data in the RSR is transferred to the RCREG (if it is empty). If the transfer is complete, the interrupt bit RCIF (PIR<0>) is set. The actual interrupt can be enabled/disabled by setting/clearing the RCIE (PIE<0>) bit. RCIF is a read only bit which is cleared by the hardware. It is cleared when RCREG has been read and is empty. RCREG is a double buffered register; (i.e. it is a two deep FIFO). It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte begin shifting to the RSR. On detection of the stop bit of the third byte, if the RCREG is still full, then the overrun error bit, OERR (RCSTA<1>) will be set. The word in the RSR will be lost. RCREG can be read twice to retrieve the two bytes in the FIFO. The OERR bit has to be cleared in software which is done by resetting the receive logic (CREN is set). If the OERR bit is set, transfers from the RSR to RCREG are inhibited, so it is essential to clear the OERR bit if it is set. The framing error bit FERR (RCSTA<2>) is set if a stop bit is not detected.

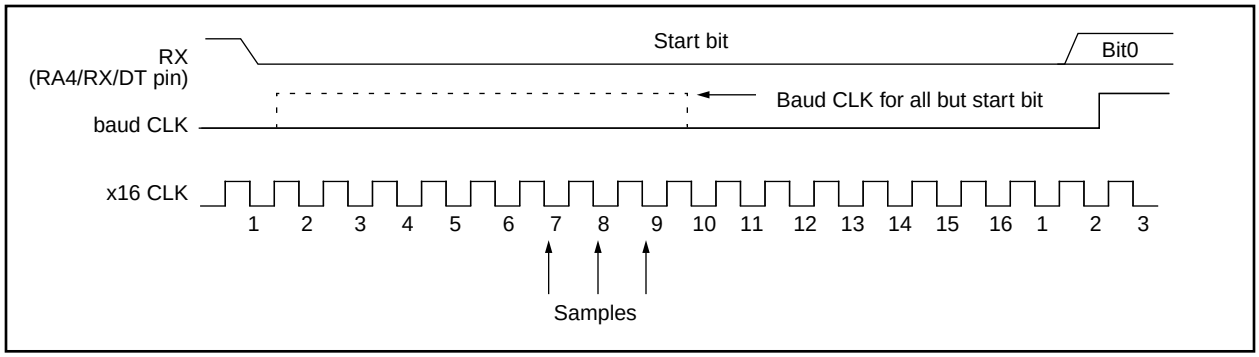
Note:	The FERR and the 9th receive bit are buffered the same way as the receive data. Reading the RCREG register will allow the RX9D and FERR bits to be loaded with values for the next received Received data; therefore, it is essential for the user to read the RCSTA register before reading RCREG in order not to lose the old FERR and RX9D information.

13.2.3 SAMPLING

The data on the RA4/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RA4/RX/DT pin. The sampling is done on the seventh, eighth and ninth falling edges of a x16 clock (Figure 11-3).

The x16 clock is a free running clock, and the three sample points occur at a frequency of every 16 falling edges.

FIGURE 13-7: RX PIN SAMPLING SCHEME

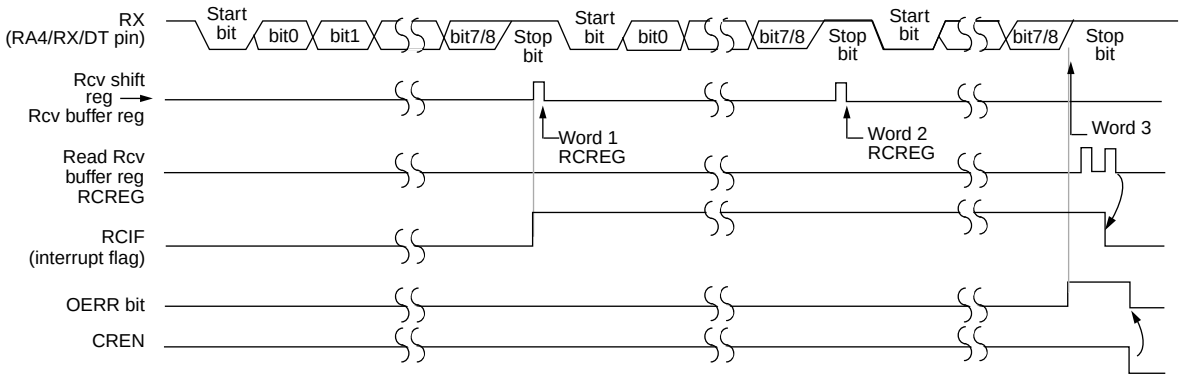


Steps to follow when setting up an Asynchronous Reception:

1. Initialize the SPBRG register for the appropriate baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are desired, then set the RCIE bit.
4. If 9-bit reception is desired, then set the RX9 bit.
5. Enable the reception by setting the CREN bit.
6. The RCIF bit will be set when reception completes and an interrupt will be generated if the RCIE bit was set.
7. Read RCSTA to get the ninth bit (if enabled) and FERR bit to determine if any error occurred during reception.
8. Read RCREG for the 8-bit received data.
9. If an overrun error occurred, clear the error by clearing the OERR bit.

Note: To terminate a reception, either clear the SREN and CREN bits, or the SPEN bit. This will reset the receive logic, so that it will be in the proper state when receive is re-enabled.

FIGURE 13-8: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

TABLE 13-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

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)
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
13h, Bank 0	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank 0	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
15h, Bank 0	TXSTA	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u
17h, Bank 0	SPBRG	Baud rate generator register								xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as a '0', shaded cells are not used for asynchronous reception.
Note 1: Other (non power-up) resets include: external reset through $\overline{\text{MCLR}}$ and Watchdog Timer Reset.

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

16.0 DEVELOPMENT SUPPORT

16.1 Development Tools

The PIC16/17 microcontrollers are supported with a full range of hardware and software development tools:

- PICMASTER/PICMASTER CE Real-Time In-Circuit Emulator
- ICEPIC Low-Cost PIC16C5X and PIC16CXXX In-Circuit Emulator
- PRO MATE® II Universal Programmer
- PICSTART® Plus Entry-Level Prototype Programmer
- PICDEM-1 Low-Cost Demonstration Board
- PICDEM-2 Low-Cost Demonstration Board
- PICDEM-3 Low-Cost Demonstration Board
- MPASM Assembler
- MPLAB-SIM Software Simulator
- MPLAB-C (C Compiler)
- Fuzzy logic development system (fuzzyTECH®–MP)

16.2 PICMASTER: High Performance Universal In-Circuit Emulator with MPLAB IDE

The PICMASTER Universal In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for all microcontrollers in the PIC12C5XX, PIC14000, PIC16C5X, PIC16CXXX and PIC17CXX families. PICMASTER is supplied with the MPLAB™ Integrated Development Environment (IDE), which allows editing, “make” and download, and source debugging from a single environment.

Interchangeable target probes allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the PICMASTER allows expansion to support all new Microchip microcontrollers.

The PICMASTER Emulator System has been designed as a real-time emulation system with advanced features that are generally found on more expensive development tools. The PC compatible 386 (and higher) machine platform and Microsoft Windows® 3.x environment were chosen to best make these features available to you, the end user.

A CE compliant version of PICMASTER is available for European Union (EU) countries.

16.3 ICEPIC: Low-cost PIC16CXXX In-Circuit Emulator

ICEPIC is a low-cost in-circuit emulator solution for the Microchip PIC16C5X and PIC16CXXX families of 8-bit OTP microcontrollers.

ICEPIC is designed to operate on PC-compatible machines ranging from 286-AT® through Pentium™ based machines under Windows 3.x environment. ICEPIC features real time, non-intrusive emulation.

16.4 PRO MATE II: Universal Programmer

The PRO MATE II Universal Programmer is a full-featured programmer capable of operating in stand-alone mode as well as PC-hosted mode.

The PRO MATE II has programmable V_{DD} and V_{PP} supplies which allows it to verify programmed memory at V_{DD} min and V_{DD} max for maximum reliability. It has an LCD display for displaying error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode the PRO MATE II can read, verify or program PIC16C5X, PIC16CXXX, PIC17CXX and PIC14000 devices. It can also set configuration and code-protect bits in this mode.

16.5 PICSTART Plus Entry Level Development System

The PICSTART programmer is an easy-to-use, low-cost 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 is not recommended for production programming.

PICSTART Plus supports all PIC12C5XX, PIC14000, PIC16C5X, PIC16CXXX and PIC17CXX devices with up to 40 pins. Larger pin count devices such as the PIC16C923 and PIC16C924 may be supported with an adapter socket.

Applicable Devices	42	R42	42A	43	R43	44

TABLE 17-1: CROSS REFERENCE OF DEVICE SPECS FOR OSCILLATOR CONFIGURATIONS AND FREQUENCIES OF OPERATION (COMMERCIAL DEVICES)

OSC	PIC17C42-16	PIC17C42-25
RC	VDD: 4.5V to 5.5V IDD: 6 mA max. IPD: 5 μA max. at 5.5V (WDT disabled) Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 6 mA max. IPD: 5 μA max. at 5.5V (WDT disabled) Freq: 4 MHz max.
XT	VDD: 4.5V to 5.5V IDD: 24 mA max. IPD: 5 μA max. at 5.5V (WDT disabled) Freq: 16 MHz max.	VDD: 4.5V to 5.5V IDD: 38 mA max. IPD: 5 μA max. at 5.5V (WDT disabled) Freq: 25 MHz max.
EC	VDD: 4.5V to 5.5V IDD: 24 mA max. IPD: 5 μA max. at 5.5V (WDT disabled) Freq: 16 MHz max.	VDD: 4.5V to 5.5V IDD: 38 mA max. IPD: 5 μA max. at 5.5V (WDT disabled) Freq: 25 MHz max.
LF	VDD: 4.5V to 5.5V IDD: 150 μA max. at 32 kHz (WDT enabled) IPD: 5 μA max. at 5.5V (WDT disabled) Freq: 2 MHz max.	VDD: 4.5V to 5.5V IDD: 150 μA max. at 32 kHz (WDT enabled) IPD: 5 μA max. at 5.5V (WDT disabled) Freq: 2 MHz max.

Applicable Devices	42	R42	42A	43	R43	44

18.0 PIC17C42 DC AND AC CHARACTERISTICS

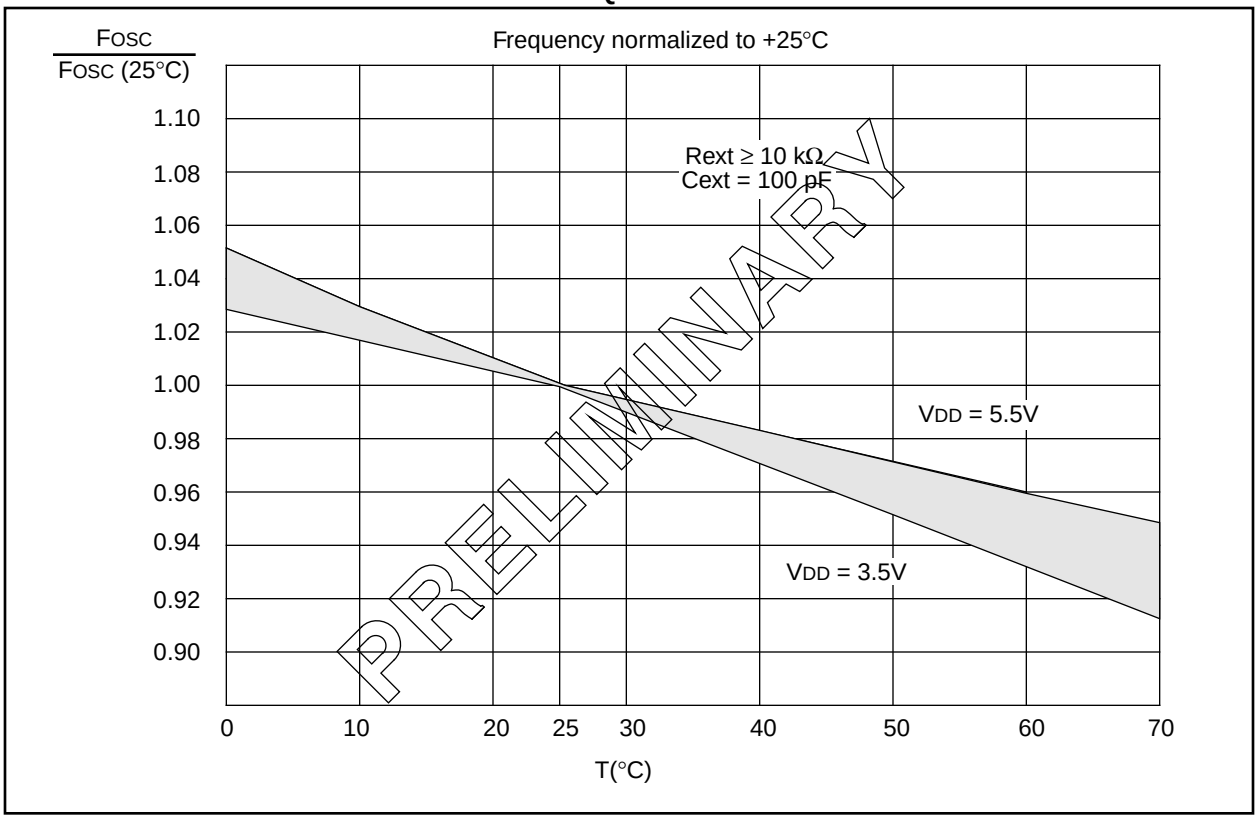
The graphs and tables provided in this section are for design guidance and are not tested or guaranteed. In some graphs or tables the data presented are outside specified operating range (e.g. outside specified VDD range). This is for information only and devices are ensured to operate properly only within the specified range.

The data presented in this section is a statistical summary of data collected on units from different lots over a period of time. "Typical" represents the mean of the distribution while "max" or "min" represents (mean + 3σ) and (mean - 3σ) respectively where σ is standard deviation.

TABLE 18-1: PIN CAPACITANCE PER PACKAGE TYPE

Pin Name	Typical Capacitance (pF)			
	40-pin DIP	44-pin PLCC	44-pin MQFP	44-pin TQFP
All pins, except $\overline{MCLR}$, VDD, and VSS	10	10	10	10
$\overline{MCLR}$ pin	20	20	20	20

FIGURE 18-1: TYPICAL RC OSCILLATOR FREQUENCY vs. TEMPERATURE



Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-9: TYPICAL IPD vs. VDD WATCHDOG DISABLED 25°C

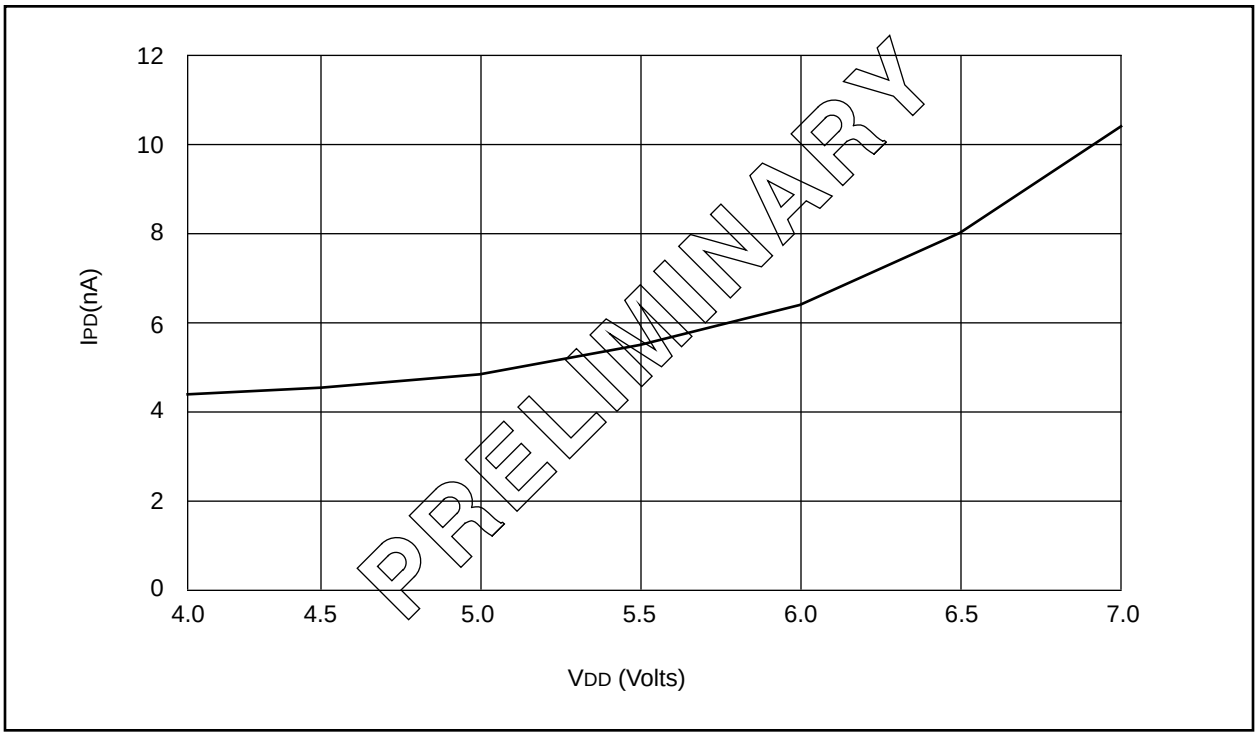
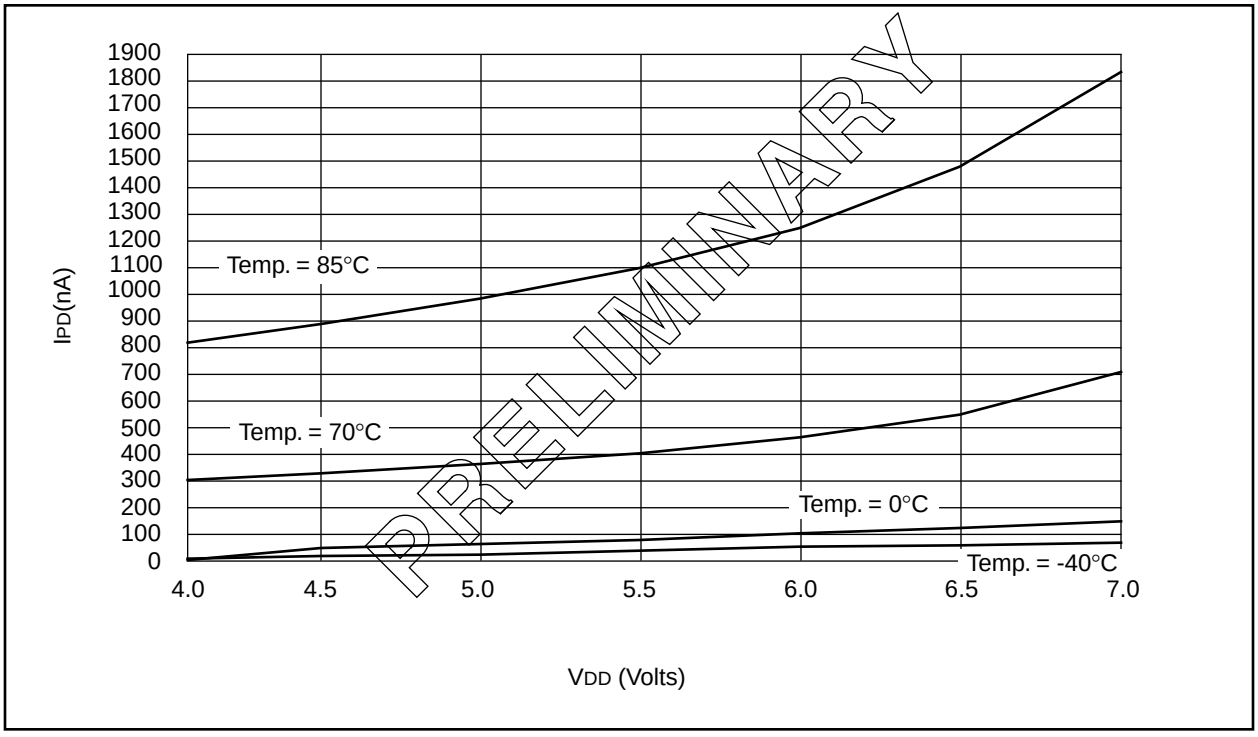


FIGURE 20-10: MAXIMUM IPD vs. VDD WATCHDOG DISABLED



NOTES:

LIST OF TABLES

