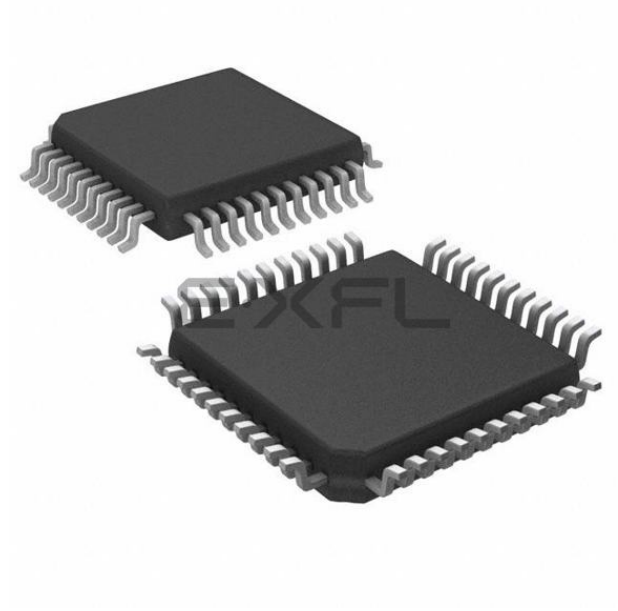


Welcome to E-XFL.COM

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	Active
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 ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-QFP
Supplier Device Package	44-MQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c42a-16i-pq

3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC17C4X can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC17C4X uses a modified Harvard architecture. This architecture has the program and data accessed from separate memories. So the device has a program memory bus and a data memory bus. This improves bandwidth over traditional von Neumann architecture, where program and data are fetched from the same memory (accesses over the same bus). Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word. PIC17C4X opcodes are 16-bits wide, enabling single word instructions. The full 16-bit wide program memory bus fetches a 16-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions execute in a single cycle (121 ns @ 33 MHz), except for program branches and two special instructions that transfer data between program and data memory.

The PIC17C4X can address up to 64K x 16 of program memory space.

The **PIC17C42** and **PIC17C42A** integrate 2K x 16 of EPROM program memory on-chip, while the **PIC17CR42** has 2K x 16 of ROM program memory on-chip.

The **PIC17C43** integrates 4K x 16 of EPROM program memory, while the **PIC17CR43** has 4K x 16 of ROM program memory.

The **PIC17C44** integrates 8K x 16 EPROM program memory.

Program execution can be internal only (microcontroller or protected microcontroller mode), external only (microprocessor mode) or both (extended microcontroller mode). Extended microcontroller mode does not allow code protection.

The PIC17CXX can directly or indirectly address its register files or data memory. All special function registers, including the Program Counter (PC) and Working Register (WREG), are mapped in the data memory. The PIC17CXX 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 PIC17CXX simple yet efficient. In addition, the learning curve is reduced significantly.

One of the PIC17CXX family architectural enhancements from the PIC16CXX family allows two file registers to be used in some two operand instructions. This allows data to be moved directly between two registers without going through the WREG register. This increases performance and decreases program memory usage.

The PIC17CXX 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.

The WREG register is an 8-bit working register used for ALU operations.

All PIC17C4X devices (except the PIC17C42) have an 8 x 8 hardware multiplier. This multiplier generates a 16-bit result in a single cycle.

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 $\overline{\text{borrow}}$ and $\overline{\text{digit}}$ $\overline{\text{borrow}}$ out bit, respectively, in subtraction. See the SUBLW and SUBWF instructions for examples.

Although the ALU does not perform signed arithmetic, the Overflow bit (OV) can be used to implement signed math. Signed arithmetic is comprised of a magnitude and a sign bit. The overflow bit indicates if the magnitude overflows and causes the sign bit to change state. Signed math can have greater than 7-bit values (magnitude), if more than one byte is used. The use of the overflow bit only operates on bit6 (MSb of magnitude) and bit7 (sign bit) of the value in the ALU. That is, the overflow bit is not useful if trying to implement signed math where the magnitude, for example, is 11-bits. If the signed math values are greater than 7-bits (15-, 24- or 31-bit), the algorithm must ensure that the low order bytes ignore the overflow status bit.

Care should be taken when adding and subtracting signed numbers to ensure that the correct operation is executed. Example 3-1 shows an item that must be taken into account when doing signed arithmetic on an ALU which operates as an unsigned machine.

EXAMPLE 3-1: SIGNED MATH

Hex Value	Signed Value Math	Unsigned Value Math
FFh	-127	255
+ 01h	+ 1	+ 1
= ?	= -126 (FEh)	= 0 (00h); Carry bit = 1

Signed math requires the result in REG to be FEh (-126). This would be accomplished by subtracting one as opposed to adding one.

Simplified block diagrams are shown in Figure 3-1 and Figure 3-2. The descriptions of the device pins are listed in Table 3-1.

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.

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.

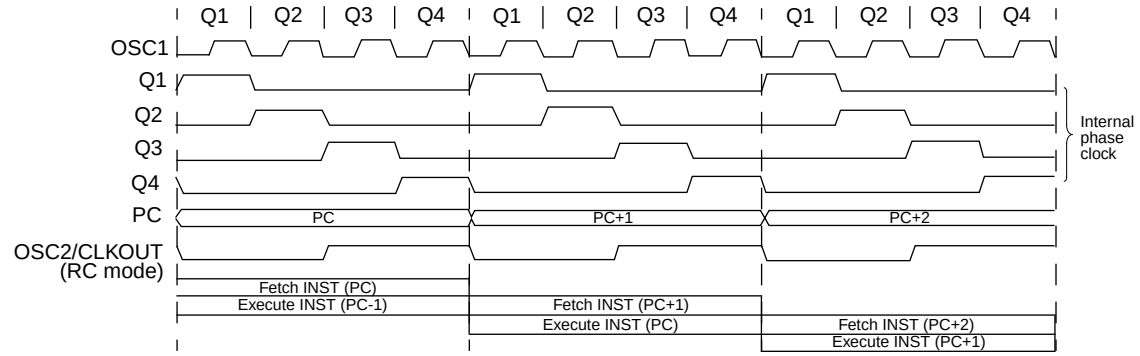
3.2 Instruction Flow/Pipelining

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).

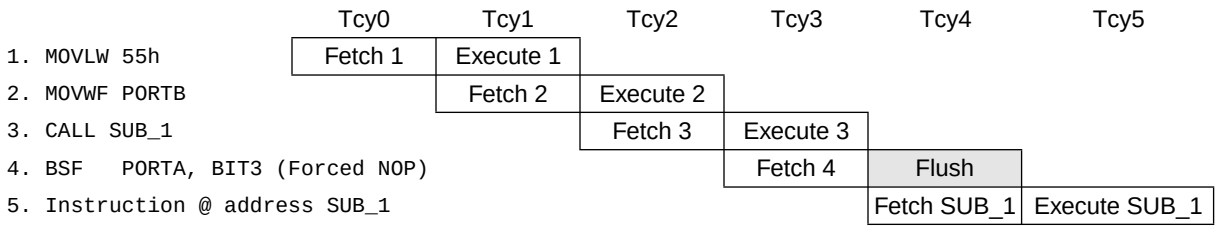
A fetch cycle begins with the program counter incrementing in Q1.

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



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.

TABLE 4-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS (Cont.'d)

Register	Address	Power-on Reset	$\overline{\text{MCLR}}$ Reset WDT Reset	Wake-up from SLEEP through interrupt
Bank 2				
TMR1	10h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR2	11h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3H	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR1	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR2	15h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR3/CA1L	16h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR3/CA1H	17h	xxxx xxxx	uuuu uuuu	uuuu uuuu
Bank 3				
PW1DCL	10h	xx-- ----	uu-- ----	uu-- ----
PW2DCL	11h	xx-- ----	uu-- ----	uu-- ----
PW1DCH	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PW2DCH	13h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA2L	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
CA2H	15h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TCON1	16h	0000 0000	0000 0000	uuuu uuuu
TCON2	17h	0000 0000	0000 0000	uuuu uuuu
Unbanked				
PRODL (5)	18h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODH (5)	19h	xxxx xxxx	uuuu uuuu	uuuu uuuu

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

Note 1: One or more bits in INTSTA, PIR will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GLINTD bit is cleared, the PC is loaded with the interrupt vector.

3: See Table 4-3 for reset value of specific condition.

4: Only applies to the PIC17C42.

5: Does not apply to the PIC17C42.

5.0 INTERRUPTS

The PIC17C4X devices have 11 sources of interrupt:

- External interrupt from the RA0/INT pin
- Change on RB7:RB0 pins
- TMR0 Overflow
- TMR1 Overflow
- TMR2 Overflow
- TMR3 Overflow
- USART Transmit buffer empty
- USART Receive buffer full
- Capture1
- Capture2
- T0CKI edge occurred

There are four registers used in the control and status of interrupts. These are:

- CPUSTA
- INTSTA
- PIE
- PIR

The CPUSTA register contains the GLINTD bit. This is the Global Interrupt Disable bit. When this bit is set, all interrupts are disabled. This bit is part of the controller core functionality and is described in the Memory Organization section.

When an interrupt is responded to, the GLINTD bit is automatically set to disable any further interrupt, the return address is pushed onto the stack and the PC is loaded with the interrupt vector address. There are four interrupt vectors. Each vector address is for a specific interrupt source (except the peripheral interrupts which have the same vector address). These sources are:

- External interrupt from the RA0/INT pin
- TMR0 Overflow
- T0CKI edge occurred
- Any peripheral interrupt

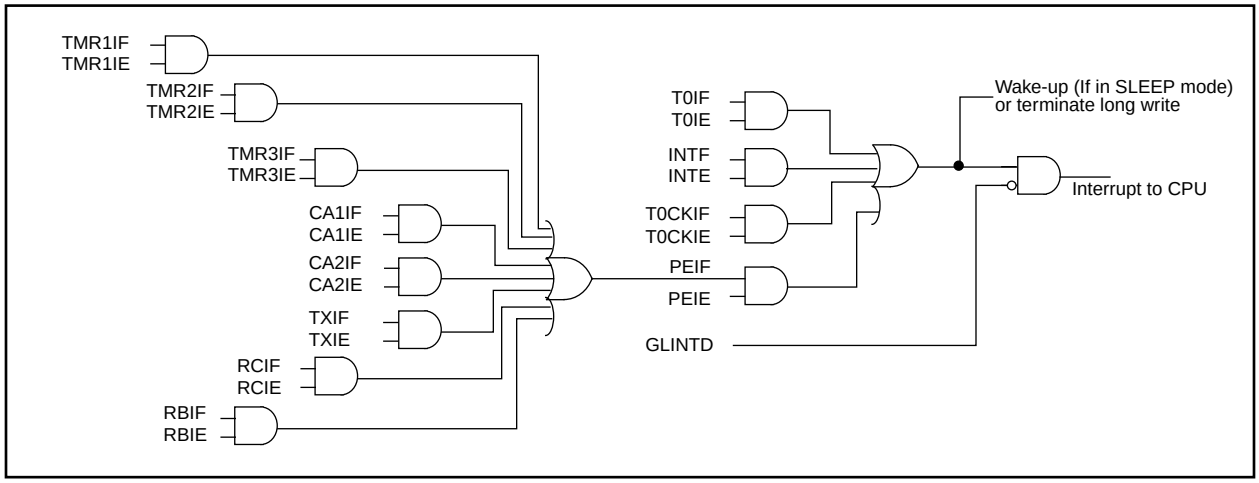
When program execution vectors to one of these interrupt vector addresses (except for the peripheral interrupt address), the interrupt flag bit is automatically cleared. Vectoring to the peripheral interrupt vector address does not automatically clear the source of the interrupt. In the peripheral interrupt service routine, the source(s) of the interrupt can be determined by testing the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid infinite interrupt requests.

All of the individual interrupt flag bits will be set regardless of the status of their corresponding mask bit or the GLINTD bit.

For external interrupt events, there will be an interrupt latency. For two cycle instructions, the latency could be one instruction cycle longer.

The “return from interrupt” instruction, `RETFIE`, can be used to mark the end of the interrupt service routine. When this instruction is executed, the stack is “POPed”, and the GLINTD bit is cleared (to re-enable interrupts).

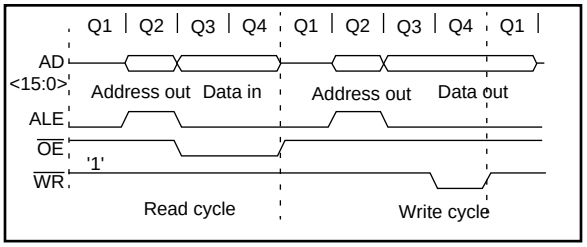
FIGURE 5-1: INTERRUPT LOGIC



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.

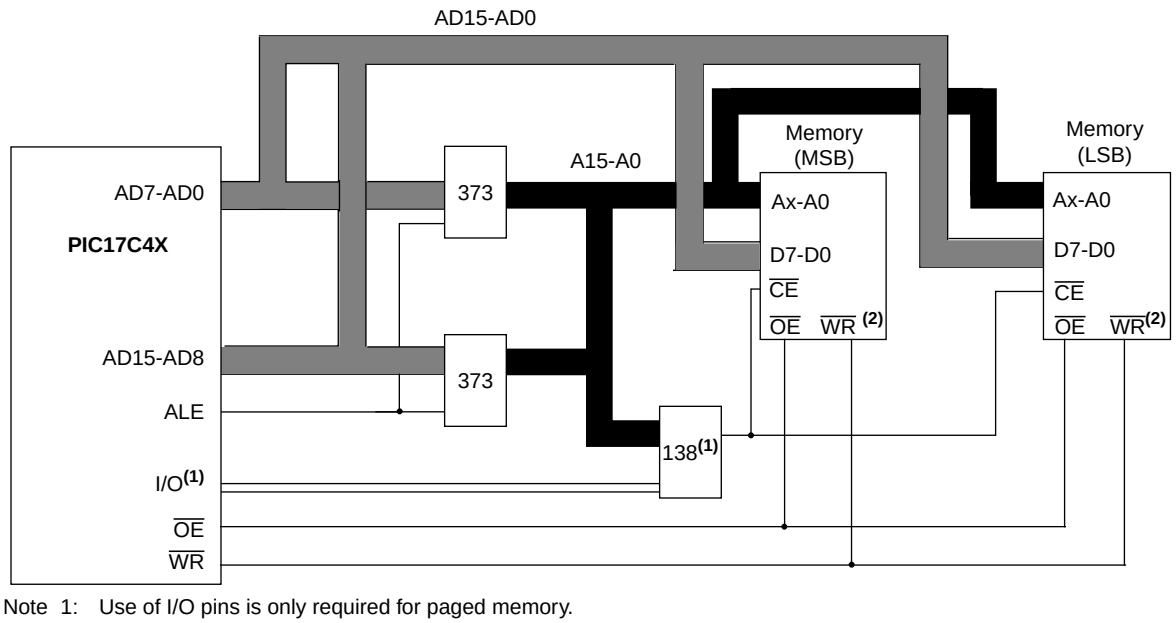
TABLE 6-2: EPROM MEMORY ACCESS TIME ORDERING SUFFIX

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.

6.2.2.3 TMR0 STATUS/CONTROL REGISTER (T0STA)

This register contains various control bits. Bit7 (INTEDG) is used to control the edge upon which a signal on the RA0/INT pin will set the RB0/INT interrupt flag. The other bits configure the Timer0 prescaler and clock source. (Figure 11-1).

FIGURE 6-9: 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, reads 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 Timer0.
1 = Internal instruction clock cycle (T_{CY})
0 = T0CKI pin

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

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'

13.3.2 USART SYNCHRONOUS MASTER RECEPTION

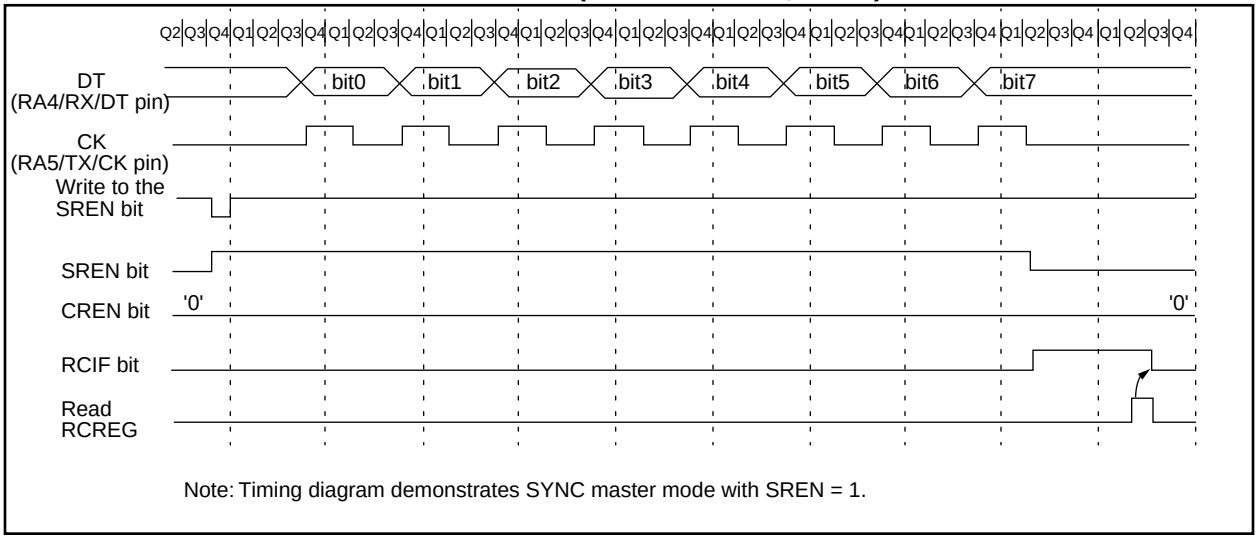
Once synchronous mode is selected, reception is enabled by setting either the SREN (RCSTA<5>) bit or the CREN (RCSTA<4>) bit. Data is sampled on the RA4/RX/DT pin on the falling edge of the clock. If SREN is set, then only a single word is received. If CREN is set, the reception is continuous until CREN is reset. If both bits are set, then CREN takes precedence. After clocking the last bit, the received data in the Receive Shift Register (RSR) is transferred to 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 RESET by the hardware. In this case it is reset 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 to begin shifting into the RSR. On the clocking of the last bit of the third byte, if RCREG is still full, then the overrun error bit OERR (RCSTA<1>) is 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. This is done by clearing the CREN bit. If OERR bit is set, transfers from RSR to RCREG are inhibited, so it is essential to clear OERR bit if it is set. The 9th receive bit is 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 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.

Steps to follow when setting up a Synchronous Master Reception:

1. Initialize the SPBRG register for the appropriate baud rate. See Section 13.1 for details.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN, and CSRC.
3. If interrupts are desired, then set the RCIE bit.
4. If 9-bit reception is desired, then set the RX9 bit.
5. If a single reception is required, set bit SREN. For continuous reception set bit CREN.
6. The RCIF bit will be set when reception is complete and an interrupt will be generated if the RCIE bit was set.
7. Read RCSTA to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading RCREG.
9. If any error occurred, clear the error by clearing CREN.

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 13-11: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



Note: Timing diagram demonstrates SYNC master mode with SREN = 1.

CALL **Subroutine Call**

Syntax: [*label*] CALL k

Operands: 0 ≤ k ≤ 4095

Operation: PC+ 1→ TOS, k → PC<12:0>, k<12:8> → PCLATH<4:0>; PC<15:13> → PCLATH<7:5>

Status Affected: None

Encoding:

111k	kkkk	kkkk	kkkk

Description: Subroutine call within 8K page. First, return address (PC+1) is pushed onto the stack. The 13-bit value is loaded into PC bits<12:0>. Then the upper-eight bits of the PC are copied into PCLATH. `Call` is a two-cycle instruction.

See `LCALL` for calls outside 8K memory space.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>	Execute	NOP
Forced NOP	NOP	Execute	NOP

Example: `HERE CALL THERE`

Before Instruction
PC = Address(`HERE`)

After Instruction
PC = Address(`THERE`)
TOS = Address (`HERE + 1`)

CLRF **Clear f**

Syntax: [*label*] CLRF f,s

Operands: 0 ≤ f ≤ 255

Operation: 00h → f, s ∈ [0,1]
00h → dest

Status Affected: None

Encoding:

0010	100s	ffff	ffff

Description: Clears the contents of the specified register(s).
s = 0: Data memory location 'f' and WREG are cleared.
s = 1: Data memory location 'f' is cleared.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register 'f' and other specified register

Example: `CLRF FLAG_REG`

Before Instruction
FLAG_REG = 0x5A

After Instruction
FLAG_REG = 0x00

RRNCF **Rotate Right f (no carry)**

Syntax: [*label*] RRNCF f,d

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

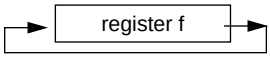
Operation: f<n> → d<n-1>;
f<0> → d<7>

Status Affected: None

Encoding:

0010	000d	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is 0 the result is placed in WREG. If 'd' is 1 the result is placed back in register 'f'.



Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1:
```
RRNCF   REG, 1
```
Before Instruction
```
WREG  =  ?
REG   =  1101 0111
```
After Instruction
```
WREG  =  0
REG   =  1110 1011
```

Example 2:
```
RRNCF   REG, 0
```
Before Instruction
```
WREG  =  ?
REG   =  1101 0111
```
After Instruction
```
WREG  =  1110 1011
REG   =  1101 0111
```

SETF **Set f**

Syntax: [*label*] SETF f,s

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

Operation: FFh → f;
FFh → d

Status Affected: None

Encoding:

0010	101s	ffff	ffff

Description: If 's' is 0, both the data memory location 'f' and WREG are set to FFh. If 's' is 1 only the data memory location 'f' is set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register 'f' and other specified register

Example1:
```
SETF    REG, 0
```
Before Instruction
```
REG   =  0xDA
WREG  =  0x05
```
After Instruction
```
REG   =  0xFF
WREG  =  0xFF
```
Example2:
```
SETF    REG, 1
```
Before Instruction
```
REG   =  0xDA
WREG  =  0x05
```
After Instruction
```
REG   =  0xFF
WREG  =  0x05
```

TABLWT **Table Write**

Example1: `TABLWT 0, 1, REG`

Before Instruction

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

After Instruction (table write completion)

REG	=	0x53
TBLATH	=	0x53
TBLATL	=	0x55
TBLPTR	=	0xA357
MEMORY(TBLPTR - 1)	=	0x5355

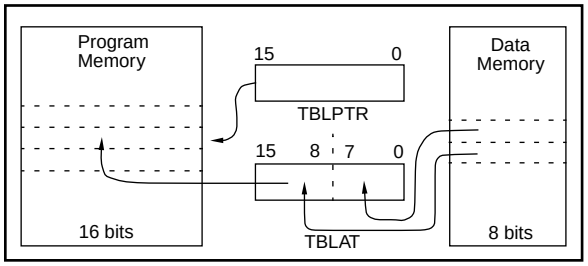
Example 2: `TABLWT 1, 0, REG`

Before Instruction

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

After Instruction (table write completion)

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x53
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0xAA53



TLRD **Table Latch Read**

Syntax: [*label*] TLRD t,f

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

Operation: If t = 0,
TBLATL → f;
If t = 1,
TBLATH → f

Status Affected: None

Encoding:

1010	00tx	ffff	ffff

Description: Read data from 16-bit table latch (TBLAT) into file register 'f'. Table Latch is unaffected.
If t = 1; high byte is read
If t = 0; low byte is read
This instruction is used in conjunction with TABLRD to transfer data from program memory to data memory.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `TLRD t, RAM`

Before Instruction

t	=	0	
RAM	=	?	
TBLAT	=	0x00AF	(TBLATH = 0x00) (TBLATL = 0xAF)

After Instruction

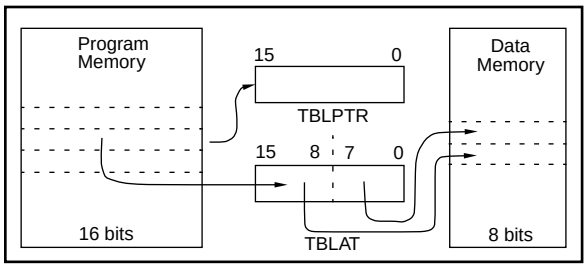
RAM	=	0xAF	
TBLAT	=	0x00AF	(TBLATH = 0x00) (TBLATL = 0xAF)

Before Instruction

t	=	1	
RAM	=	?	
TBLAT	=	0x00AF	(TBLATH = 0x00) (TBLATL = 0xAF)

After Instruction

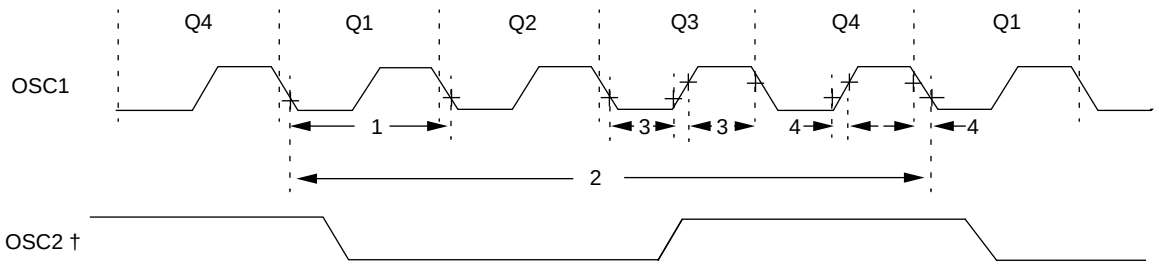
RAM	=	0x00	
TBLAT	=	0x00AF	(TBLATH = 0x00) (TBLATL = 0xAF)



Applicable Devices	42	R42	42A	43	R43	44

17.4 Timing Diagrams and Specifications

FIGURE 17-2: EXTERNAL CLOCK TIMING



† In EC and RC modes only.

TABLE 17-2: EXTERNAL CLOCK TIMING REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fosc	**External CLKIN Frequency**	DC	—	16	MHz	EC osc mode - PIC17C42-16
		(Note 1)	DC	—	25	MHz	- PIC17C42-25
		Oscillator Frequency	DC	—	4	MHz	RC osc mode
		(Note 1)	1	—	16	MHz	XT osc mode - PIC17C42-16
			1	—	25	MHz	- PIC17C42-25
			DC	—	2	MHz	LF osc mode
1	Tosc	**External CLKIN Period**	62.5	—	—	ns	EC osc mode - PIC17C42-16
		(Note 1)	40	—	—	ns	- PIC17C42-25
		Oscillator Period	250	—	—	ns	RC osc mode
		(Note 1)	62.5	—	1,000	ns	XT osc mode - PIC17C42-16
			40	—	1,000	ns	- PIC17C42-25
			500	—	—	ns	LF osc mode
2	TCY	**Instruction Cycle Time (Note 1)**	160	4/Fosc	DC	ns	
3	TosL, TosH	**Clock in (OSC1) High or Low Time**	10 ‡	—	—	ns	EC oscillator
4	TosR, TosF	**Clock in (OSC1) Rise or Fall Time**	—	—	5 ‡	ns	EC oscillator

† 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.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1 pin.
When an external clock input is used, the "Max." cycle time limit is "DC" (no clock) for all devices.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 17-12: MEMORY INTERFACE READ TIMING

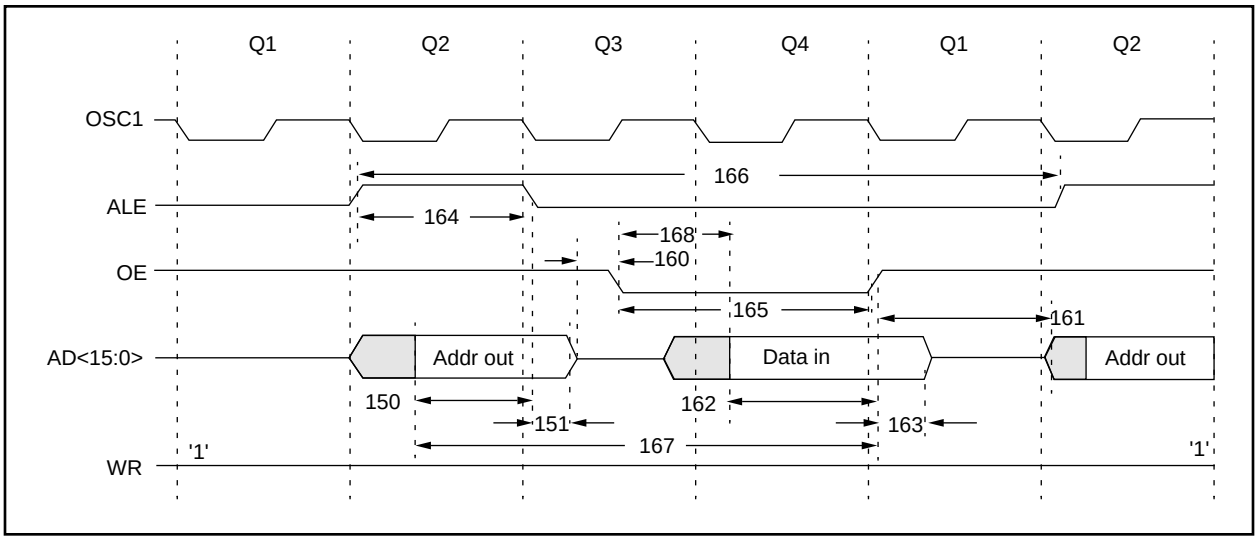


TABLE 17-12: MEMORY INTERFACE READ REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2alL	AD<15:0> (address) valid to ALE↓ (address setup time)	0.25Tcy - 30	—	—	ns	
151	TalL2adI	ALE↓ to address out invalid (address hold time)	5*	—	—	ns	
160	TadZ2oeL	AD<15:0> high impedance to $\overline{OE}$↓	0*	—	—	ns	
161	ToeH2adD	$\overline{OE}$↑ to AD<15:0> driven	0.25Tcy - 15	—	—	ns	
162	TadV2oeH	Data in valid before $\overline{OE}$↑ (data setup time)	35	—	—	ns	
163	ToeH2adI	$\overline{OE}$↑to data in invalid (data hold time)	0	—	—	ns	
164	TalH	ALE pulse width	—	0.25TCY §	—	ns	
165	ToeL	$\overline{OE}$ pulse width	0.5Tcy - 35 §	—	—	ns	
166	TalH2alH	ALE↑ to ALE↑ (cycle time)	—	TCY §	—	ns	
167	Tacc	Address access time	—	—	0.75 TCY-40	ns	
168	Toe	Output enable access time ($\overline{OE}$ low to Data Valid)	—	—	0.5 TCY - 60	ns	

* These parameters are characterized but not tested.

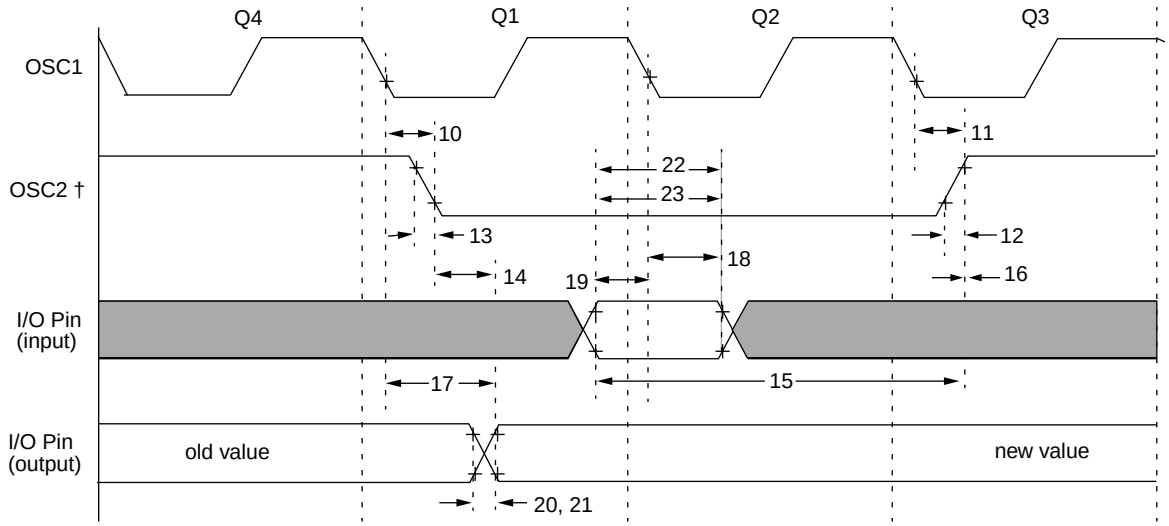
† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification guaranteed by design.

NOTES:

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 19-3: CLKOUT AND I/O TIMING



† In EC and RC modes only.

TABLE 19-3: CLKOUT AND I/O TIMING REQUIREMENTS

Parameter No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
10	TosH2ckL	OSC1↓ to CLKOUT↓		—	15 ‡	30 ‡	ns	Note 1
11	TosH2ckH	OSC1↓ to CLKOUT↑		—	15 ‡	30 ‡	ns	Note 1
12	TckR	CLKOUT rise time		—	5 ‡	15 ‡	ns	Note 1
13	TckF	CLKOUT fall time		—	5 ‡	15 ‡	ns	Note 1
14	TckH2ioV	CLKOUT ↑ to Port out valid	PIC17CR42/42A/43/ R43/44	—	—	0.5TCY + 20 ‡	ns	Note 1
			PIC17LCR42/42A/43/ R43/44	—	—	0.5TCY + 50 ‡	ns	Note 1
15	TioV2ckH	Port in valid before CLKOUT↑	PIC17CR42/42A/43/ R43/44	0.25TCY + 25 ‡	—	—	ns	Note 1
			PIC17LCR42/42A/43/ R43/44	0.25TCY + 50 ‡	—	—	ns	Note 1
16	TckH2ioI	Port in hold after CLKOUT↑		0 ‡	—	—	ns	Note 1
17	TosH2ioV	OSC1↓ (Q1 cycle) to Port out valid		—	—	100 ‡	ns	
18	TosH2ioI	OSC1↓ (Q2 cycle) to Port input invalid (I/O in hold time)		0 ‡	—	—	ns	
19	TioV2osH	Port input valid to OSC1↓ (I/O in setup time)		30 ‡	—	—	ns	
20	TioR	Port output rise time		—	10 ‡	35 ‡	ns	
21	TioF	Port output fall time		—	10 ‡	35 ‡	ns	
22	TinHL	INT pin high or low time		25 *	—	—	ns	
23	TrbHL	RB7:RB0 change INT high or low time		25 *	—	—	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.

Note 1: Measurements are taken in EC Mode where CLKOUT output is 4 x TOSC.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-17: I_{OL} vs. V_{OL}, V_{DD} = 5V

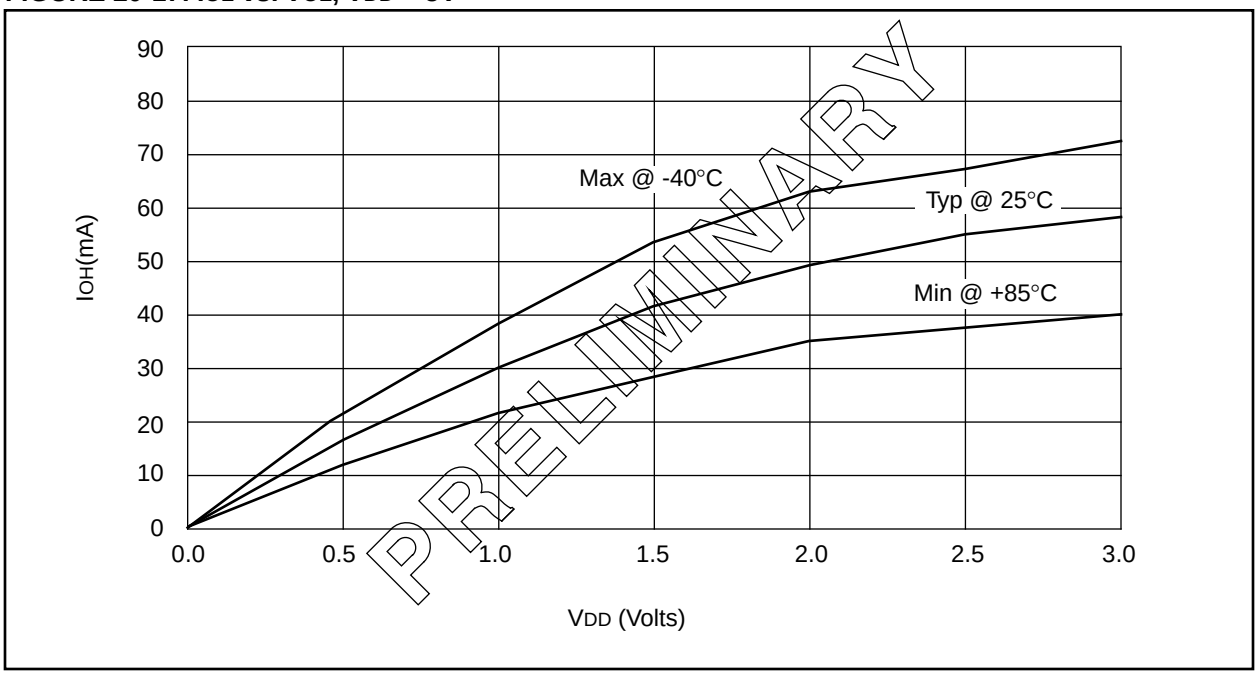
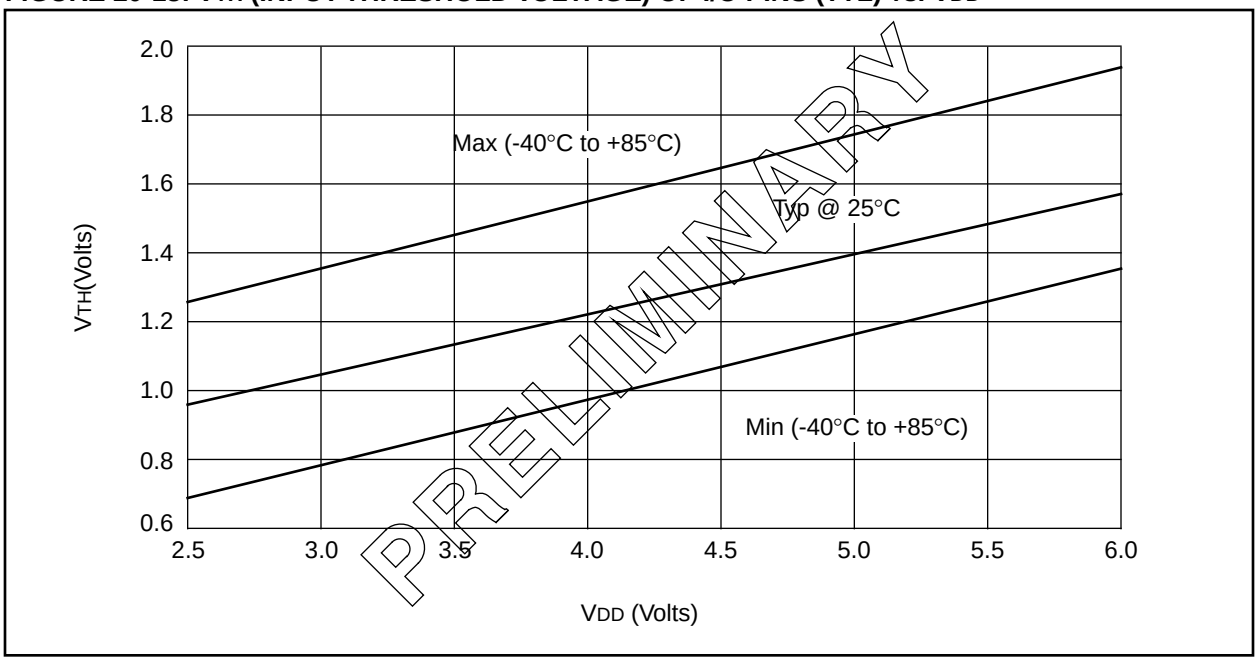
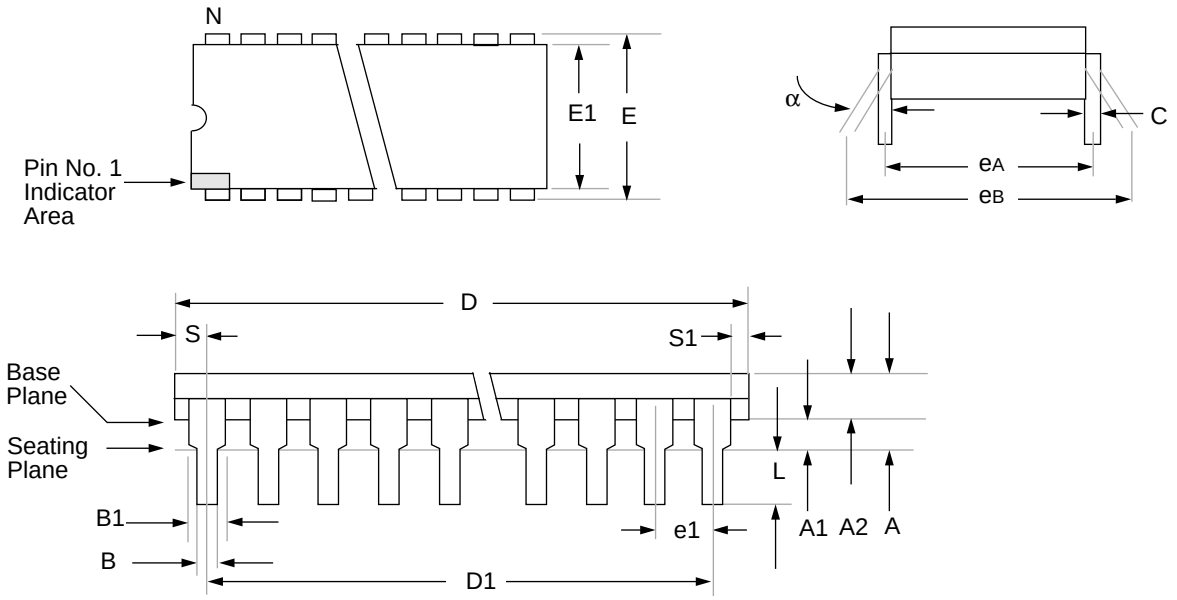


FIGURE 20-18: V_{TH} (INPUT THRESHOLD VOLTAGE) OF I/O PINS (TTL) vs. V_{DD}



NOTES:

21.2 40-Lead Plastic Dual In-line (600 mil)



Package Group: Plastic Dual In-Line (PLA)						
	Millimeters			Inches		
Symbol	Min	Max	Notes	Min	Max	Notes
α	0°	10°		0°	10°	
A	–	5.080		–	0.200	
A1	0.381	–		0.015	–	
A2	3.175	4.064		0.125	0.160	
B	0.355	0.559		0.014	0.022	
B1	1.270	1.778	**Typical**	0.050	0.070	**Typical**
C	0.203	0.381	**Typical**	0.008	0.015	**Typical**
D	51.181	52.197		2.015	2.055	
D1	48.260	48.260	**Reference**	1.900	1.900	**Reference**
E	15.240	15.875		0.600	0.625	
E1	13.462	13.970		0.530	0.550	
e1	2.489	2.591	**Typical**	0.098	0.102	**Typical**
eA	15.240	15.240	**Reference**	0.600	0.600	**Reference**
eB	15.240	17.272		0.600	0.680	
L	2.921	3.683		0.115	0.145	
N	40	40		40	40	
S	1.270	–		0.050	–	
S1	0.508	–		0.020	–	

PIN COMPATIBILITY

Devices that have the same package type and VDD, VSS and $\overline{MCLR}$ pin locations are said to be pin compatible. This allows these different devices to operate in the same socket. Compatible devices may only requires minor software modification to allow proper operation in the application socket (ex., PIC16C56 and PIC16C61 devices). Not all devices in the same package size are pin compatible; for example, the PIC16C62 is compatible with the PIC16C63, but not the PIC16C55.

Pin compatibility does not mean that the devices offer the same features. As an example, the PIC16C54 is pin compatible with the PIC16C71, but does not have an A/D converter, weak pull-ups on PORTB, or interrupts.

TABLE E-1: PIN COMPATIBLE DEVICES

Pin Compatible Devices	Package
PIC12C508, PIC12C509	8-pin
PIC16C54, PIC16C54A, PIC16CR54A, PIC16C56, PIC16C58A, PIC16CR58A, PIC16C61, PIC16C554, PIC16C556, PIC16C558 PIC16C620, PIC16C621, PIC16C622, PIC16C710, PIC16C71, PIC16C711, PIC16F83, PIC16CR83, PIC16C84, PIC16F84A, PIC16CR84	18-pin 20-pin
PIC16C55, PIC16C57, PIC16CR57B	28-pin
PIC16C62, PIC16CR62, PIC16C62A, PIC16C63, PIC16C72, PIC16C73, PIC16C73A	28-pin
PIC16C64, PIC16CR64, PIC16C64A, PIC16C65, PIC16C65A, PIC16C74, PIC16C74A	40-pin
PIC17C42, PIC17CR42, PIC17C42A, PIC17C43, PIC17CR43, PIC17C44	40-pin
PIC16C923, PIC16C924	64/68-pin