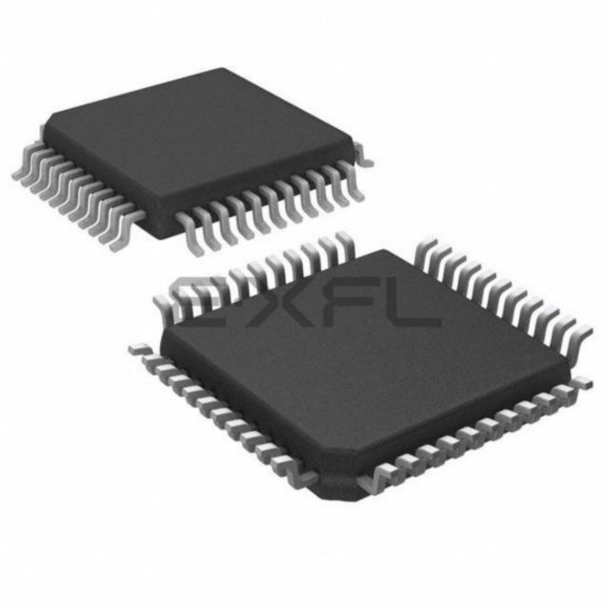


Welcome to E-XFL.COM

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.

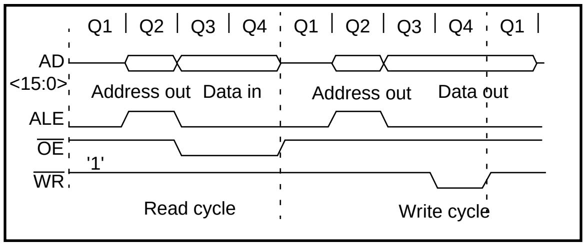
Applications of "Embedded - Microcontrollers"

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-QFP
Supplier Device Package	44-MQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c42at-16e-pq

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.

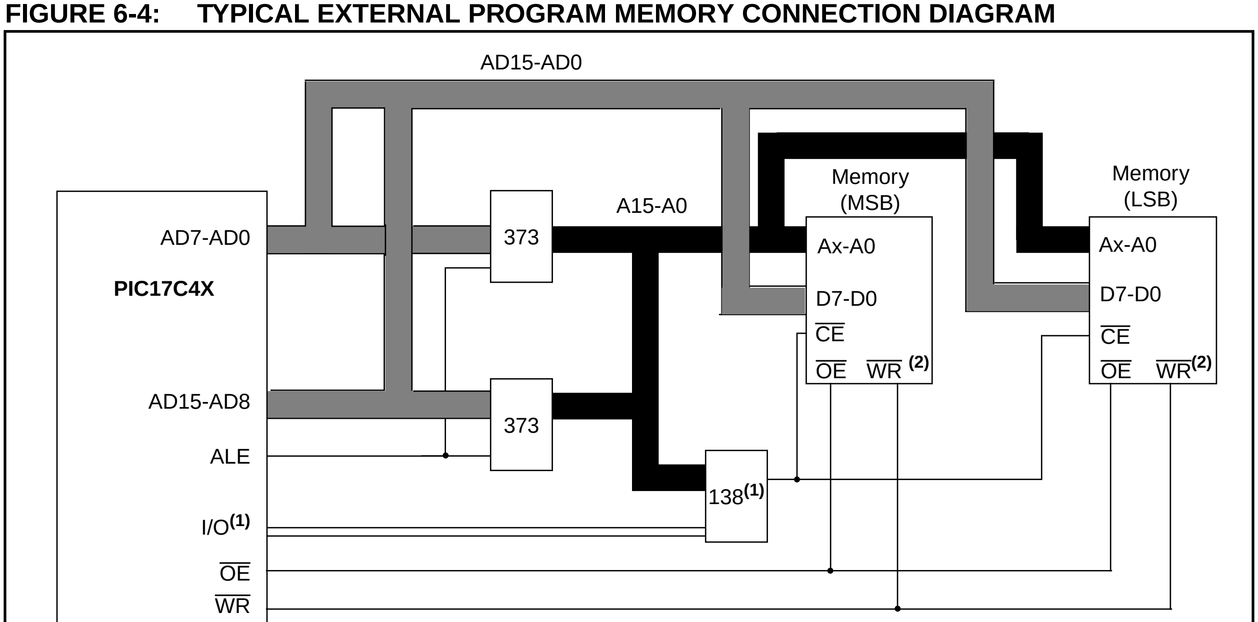
TABLE 6-2: EPROM MEMORY ACCESS TIME ORDERING SUFFIX

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.

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.
<u>When T0CS = 0</u>
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
<u>When T0CS = 1</u>
Don’t care

bit 5: **T0CS**: Timer0 Clock Source Select bit
This bit selects the clock source for Timer0.
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 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'

6.8 Bank Select Register (BSR)

The BSR is used to switch between banks in the data memory area (Figure 6-13). In the PIC17C42, PIC17CR42, and PIC17C42A only the lower nibble is implemented. While in the PIC17C43, PIC17CR43, and PIC17C44 devices, the entire byte is implemented. The lower nibble is used to select the peripheral register bank. The upper nibble is used to select the general purpose memory bank.

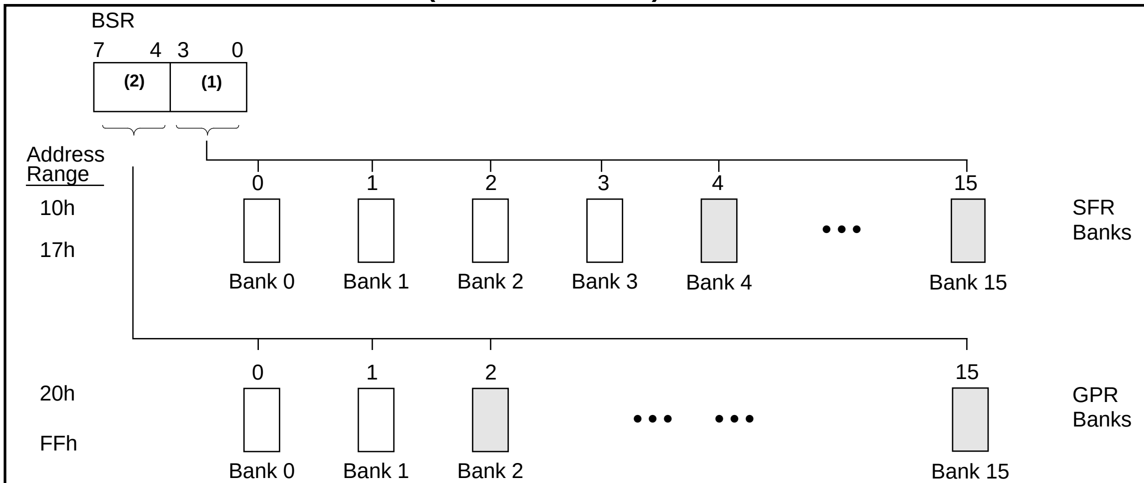
All the Special Function Registers (SFRs) are mapped into the data memory space. In order to accommodate the large number of registers, a banking scheme has been used. A segment of the SFRs, from address 10h to address 17h, is banked. The lower nibble of the bank select register (BSR) selects the currently active "peripheral bank." Effort has been made to group the peripheral registers of related functionality in one bank. However, it will still be necessary to switch from bank to bank in order to address all peripherals related to a single task. To assist this, a `MOVLB bank` instruction is in the instruction set.

For the PIC17C43, PIC17CR43, and PIC17C44 devices, the need for a large general purpose memory space dictated a general purpose RAM banking scheme. The upper nibble of the BSR selects the currently active general purpose RAM bank. To assist this, a `MOVLR bank` instruction has been provided in the instruction set.

If the currently selected bank is not implemented (such as Bank 13), any read will read all '0's. Any write is completed to the bit bucket and the ALU status bits will be set/cleared as appropriate.

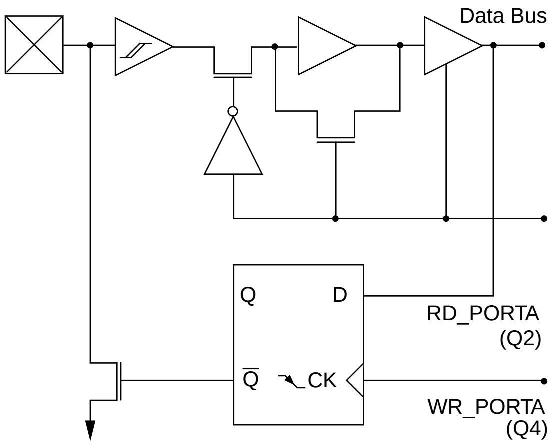
Note:	Registers in Bank 15 in the Special Function Register area, are reserved for Microchip use. Reading of registers in this bank may cause random values to be read.

FIGURE 6-13: BSR OPERATION (PIC17C43/R43/44)



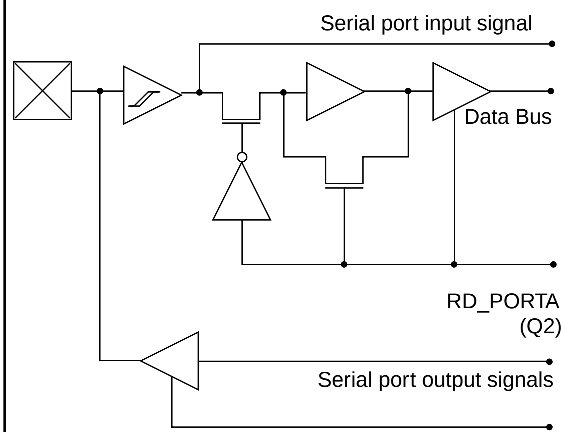
Note 1: Only Banks 0 through Bank 3 are implemented. Selection of an unimplemented bank is not recommended. **Bank 15 is reserved for Microchip use, reading of registers in this bank may cause random values to be read.**
2: Only Banks 0 and Bank 1 are implemented. Selection of an unimplemented bank is not recommended.

FIGURE 9-2: RA2 AND RA3 BLOCK DIAGRAM



Note: I/O pins have protection diodes to VSS.

FIGURE 9-3: RA4 AND RA5 BLOCK DIAGRAM



$\overline{OE}$ = SPEN,SYNC,TXEN, $\overline{CREN}$, $\overline{SREN}$ for RA4

$\overline{OE}$ = SPEN ($\overline{SYNC}$+SYNC,$\overline{CSRC}$) for RA5

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

TABLE 9-1: PORTA FUNCTIONS

Name	Bit0	Buffer Type	Function
RA0/INT	bit0	ST	Input or external interrupt input.
RA1/T0CKI	bit1	ST	Input or clock input to the TMR0 timer/counter, and/or an external interrupt input.
RA2	bit2	ST	Input/Output. Output is open drain type.
RA3	bit3	ST	Input/Output. Output is open drain type.
RA4/RX/DT	bit4	ST	Input or USART Asynchronous Receive or USART Synchronous Data.
RA5/TX/CK	bit5	ST	Input or USART Asynchronous Transmit or USART Synchronous Clock.
$\overline{RBPU}$	bit7	—	Control bit for PORTB weak pull-ups.

Legend: ST = Schmitt Trigger input.

TABLE 9-2: REGISTERS/BITS ASSOCIATED WITH PORTA

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)
10h, Bank 0	PORTA	$\overline{RBPU}$	—	RA5	RA4	RA3	RA2	RA1/T0CKI	RA0/INT	0-xx xxxx	0-uu uuuu
05h, Unbanked	T0STA	INTEDG	T0SE	T0CS	PS3	PS2	PS1	PS0	—	0000 000-	0000 000-
13h, Bank 0	RCSTA	SPEN	RC9	SREN	CREN	—	FERR	OERR	RC9D	0000 -00x	0000 -00u
15h, Bank 0	TXSTA	CSRC	TX9	TXEN	SYNC	—	—	TRMT	TX9D	0000 --1x	0000 --1u

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

9.4.1 PORTE AND DDRE REGISTER

PORTE is a 3-bit bi-directional port. The corresponding data direction register is DDRE. A '1' in DDRE configures the corresponding port pin as an input. A '0' in the DDRE register configures the corresponding port pin as an output. Reading PORTE reads the status of the pins, whereas writing to it will write to the port latch. PORTE is multiplexed with the system bus. When operating as the system bus, PORTE contains the control signals for the address/data bus (AD15:AD0). These control signals are Address Latch Enable (ALE), Output Enable ($\overline{OE}$), and Write ($\overline{WR}$). The control signals $\overline{OE}$ and $\overline{WR}$ are active low signals. The timing for the system bus is shown in the Electrical Characteristics section.

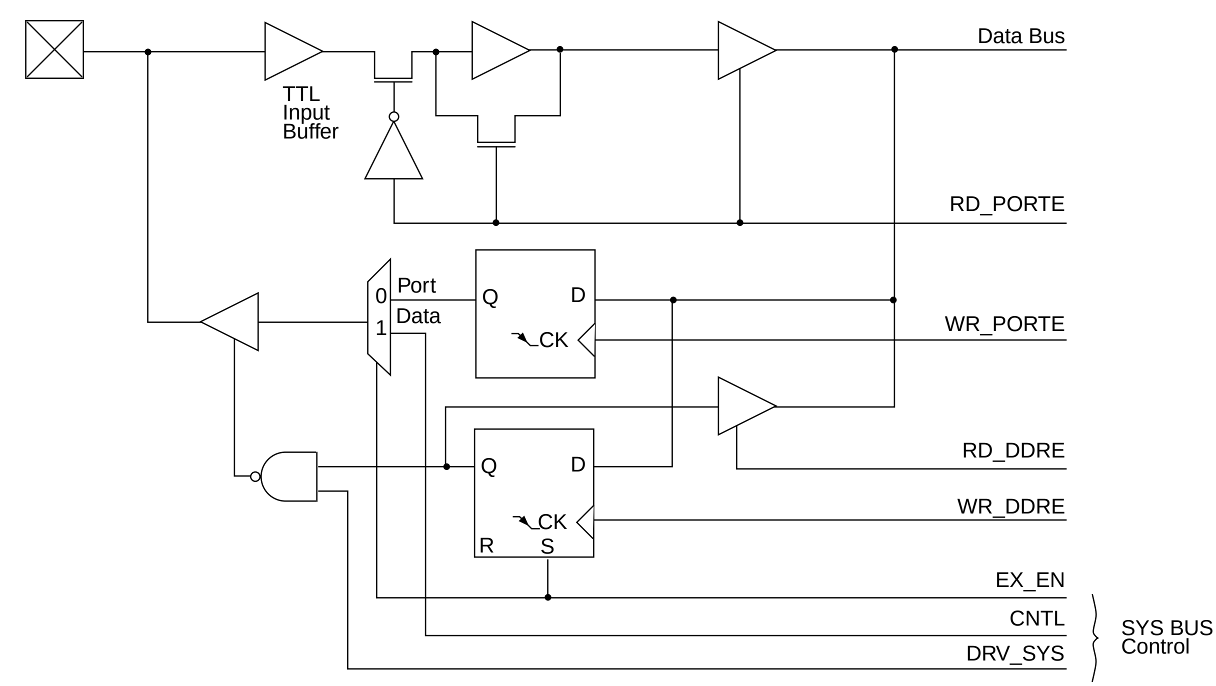
Note:	This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

Example 9-4 shows the instruction sequence to initialize PORTE. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized.

EXAMPLE 9-4: INITIALIZING PORTE

```
MOVLB  1          ; Select Bank 1
CLRF   PORTE      ; Initialize PORTE data
                  ;  latches before setting
                  ;  the data direction
                  ;  register
MOVLW  0x03       ; Value used to initialize
                  ;  data direction
MOVWF  DDRE       ; Set RE<1:0> as inputs
                  ;  RE<2> as outputs
                  ;  RE<7:3> are always
                  ;  read as '0'
```

FIGURE 9-8: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)



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

12.1.2 TIMER1 & TIMER2 IN 16-BIT MODE

To select 16-bit mode, the T16 bit must be set. In this mode TMR1 and TMR2 are concatenated to form a 16-bit timer (TMR2:TMR1). The 16-bit timer increments until it matches the 16-bit period register (PR2:PR1). On the following timer clock, the timer value is reset to 0h, and the TMR1IF bit is set.

When selecting the clock source for the16-bit timer, the TMR1CS bit controls the entire 16-bit timer and TMR2CS is a “don’t care.” When TMR1CS is clear, the timer increments once every instruction cycle (Fosc/4). When TMR1CS is set, the timer increments on every falling edge of the RB4/TCLK12 pin. For the 16-bit timer to increment, both TMR1ON and TMR2ON bits must be set (Table 12-1).

12.1.2.1 EXTERNAL CLOCK INPUT FOR TMR1:TMR2

When TMR1CS is set, the 16-bit TMR2:TMR1 increments on the falling edge of clock input TCLK12. The input on the RB4/TCLK12 pin is sampled and synchronized by the internal phase clocks twice every instruction cycle. This causes a delay from the time a falling edge appears on RB4/TCLK12 to the time TMR2:TMR1 is actually incremented. For the external clock input timing requirements, see the Electrical Specification section.

TABLE 12-1: TURNING ON 16-BIT TIMER

TMR2ON	TMR1ON	Result
1	1	16-bit timer (TMR2:TMR1) ON
0	1	Only TMR1 increments
x	0	16-bit timer OFF

FIGURE 12-4: TMR1 AND TMR2 IN 16-BIT TIMER/COUNTER MODE

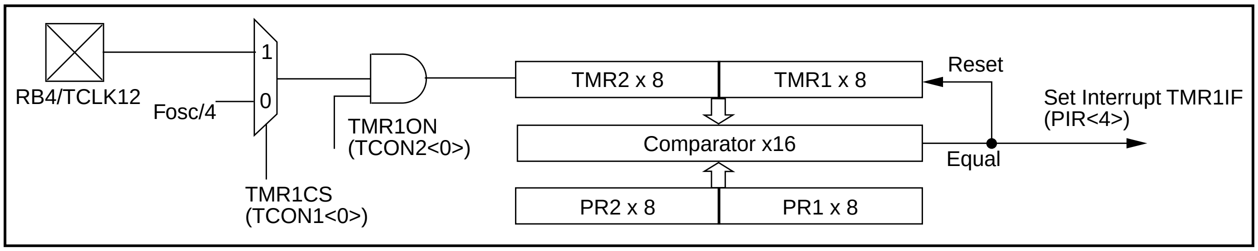


TABLE 12-2: SUMMARY OF TIMER1 AND TIMER2 REGISTERS

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 (Note1)
16h, Bank 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/$\overline{PR3}$	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
10h, Bank 2	TMR1	Timer1 register								xxxx xxxx	uuuu uuuu
11h, Bank 2	TMR2	Timer2 register								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	—	—	--11 11--	--11 qq--
14h, Bank 2	PR1	Timer1 period register								xxxx xxxx	uuuu uuuu
15h, Bank 2	PR2	Timer2 period register								xxxx xxxx	uuuu uuuu
10h, Bank 3	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h, Bank 3	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
12h, Bank 3	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h, Bank 3	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	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 Timer1 or Timer2.

Note 1: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and WDT Timer Reset.

12.2.2 DUAL CAPTURE REGISTER MODE

This mode is selected by setting CA1/$\overline{\text{PR3}}$. A block diagram is shown in Figure 12-8. In this mode, TMR3 runs without a period register and increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 interrupt Flag (TMR3IF) is set on this roll over. The TMR3IF bit must be cleared in software.

Registers PR3H/CA1H and PR3L/CA1L make a 16-bit capture register (Capture1). It captures events on pin RB0/CAP1. Capture mode is configured by the CA1ED1 and CA1ED0 bits. Capture1 Interrupt Flag bit (CA1IF) is set on the capture event. The corresponding interrupt mask bit is CA1IE. The Capture1 Overflow Status bit is CA1OVF.

The Capture2 overflow status flag bit is double buffered. The master bit is set if one captured word is already residing in the Capture2 register and another “event” has occurred on the RB1/CA2 pin. The new event will not transfer the TMR3 value to the capture register which protects the previous unread capture value. When the user reads both the high and the low bytes (in any order) of the Capture2 register, the master overflow bit is transferred to the slave overflow bit (CA2OVF) and then the master bit is reset. The user can then read TCON2 to determine the value of CA2OVF.

The operation of the Capture1 feature is identical to Capture2 (as described in Section 12.2.1).

FIGURE 12-8: TIMER3 WITH TWO CAPTURE REGISTERS BLOCK DIAGRAM

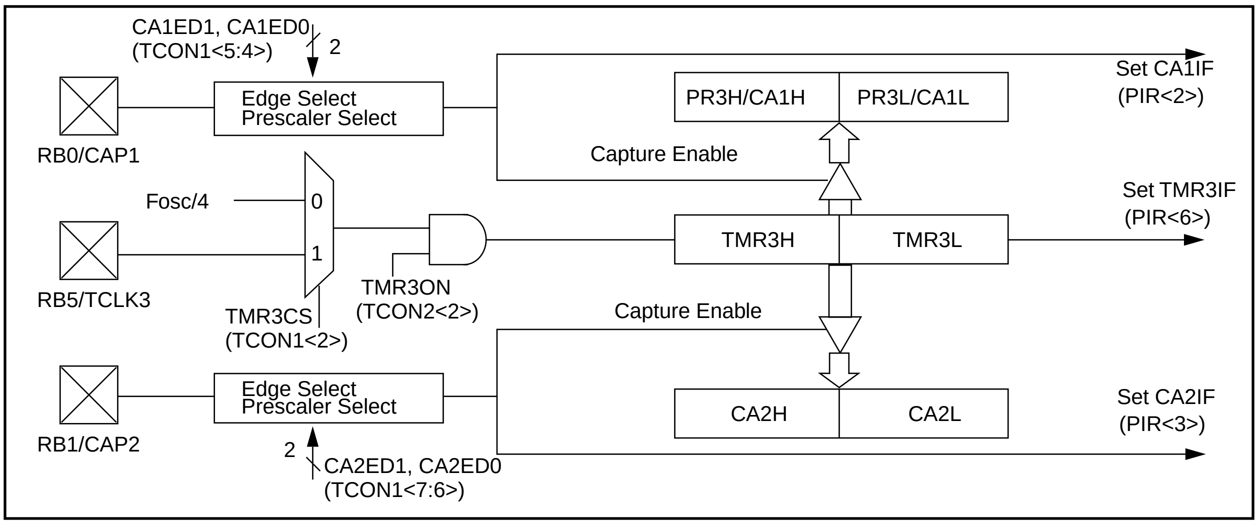


TABLE 12-5: REGISTERS ASSOCIATED WITH CAPTURE

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 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/$\overline{\text{PR3}}$	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
12h, Bank 2	TMR3L	TMR3 register; low byte								xxxx xxxx	uuuu uuuu
13h, Bank 2	TMR3H	TMR3 register; high byte								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{\text{TO}}$	$\overline{\text{PD}}$	—	—	--11 11--	--11 qq--
16h, Bank 2	PR3L/CA1L	Timer3 period register, low byte/capture1 register, low byte								xxxx xxxx	uuuu uuuu
17h, Bank 2	PR3H/CA1H	Timer3 period register, high byte/capture1 register, high byte								xxxx xxxx	uuuu uuuu
14h, Bank 3	CA2L	Capture2 low byte								xxxx xxxx	uuuu uuuu
15h, Bank 3	CA2H	Capture2 high byte								xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q - value depends on condition, shaded cells are not used by Capture.

Note 1: Other (non power-up) resets include: external reset through $\overline{\text{MCLR}}$ and WDT Timer Reset.

FIGURE 13-2: RCSTA REGISTER (ADDRESS: 13h, BANK 0)

R/W - 0	R/W - 0	R/W - 0	R/W - 0	U - 0	R - 0	R - 0	R - x
SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D
bit7							bit 0

R = Readable bit
W = Writable bit
-n = Value at POR reset
(x = unknown)

bit 7: **SPEN**: Serial Port Enable bit
1 = Configures RA5/RX/DT and RA4/TX/CK pins as serial port pins
0 = Serial port disabled

bit 6: **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5: **SREN**: Single Receive Enable bit
This bit enables the reception of a single byte. After receiving the byte, this bit is automatically cleared.
Synchronous mode:
1 = Enable reception
0 = Disable reception
Note: This bit is ignored in synchronous slave reception.
Asynchronous mode:
Don't care

bit 4: **CREN**: Continuous Receive Enable bit
This bit enables the continuous reception of serial data.
Asynchronous mode:
1 = Enable reception
0 = Disables reception
Synchronous mode:
1 = Enables continuous reception until CREN is cleared (CREN overrides SREN)
0 = Disables continuous reception

bit 3: **Unimplemented**: Read as '0'

bit 2: **FERR**: Framing Error bit
1 = Framing error (Updated by reading RCREG)
0 = No framing error

bit 1: **OERR**: Overrun Error bit
1 = Overrun (Cleared by clearing CREN)
0 = No overrun error

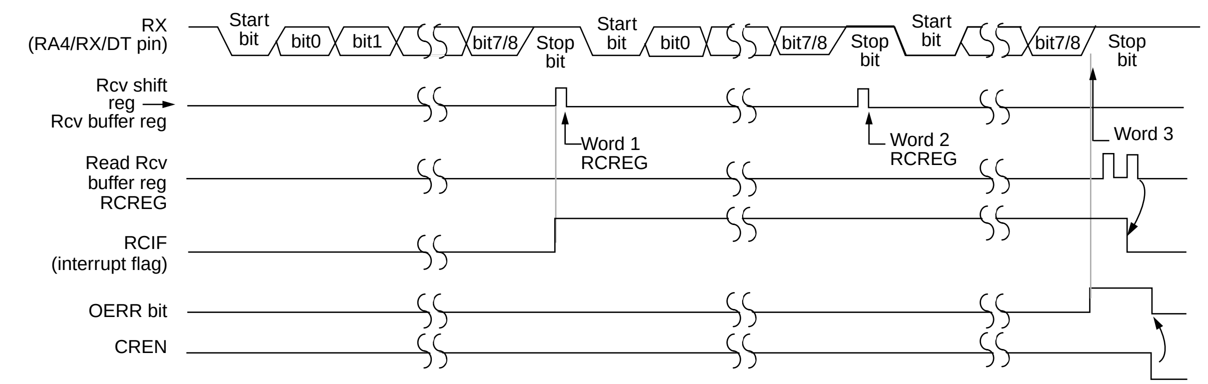
bit 0: **RX9D**: 9th bit of receive data (can be the software calculated parity bit)

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.

TABLE 13-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

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
16h, Bank 0	TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	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 synchronous master transmission.

Note 1: Other (non power-up) resets include: external reset through $\overline{\text{MCLR}}$ and Watchdog Timer Reset.

FIGURE 13-9: SYNCHRONOUS TRANSMISSION

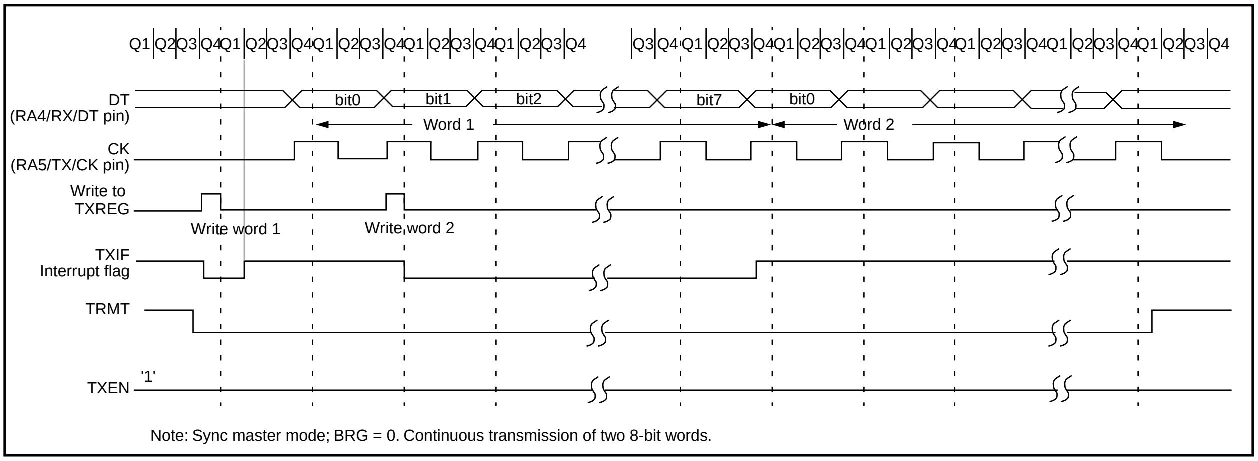


FIGURE 13-10: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

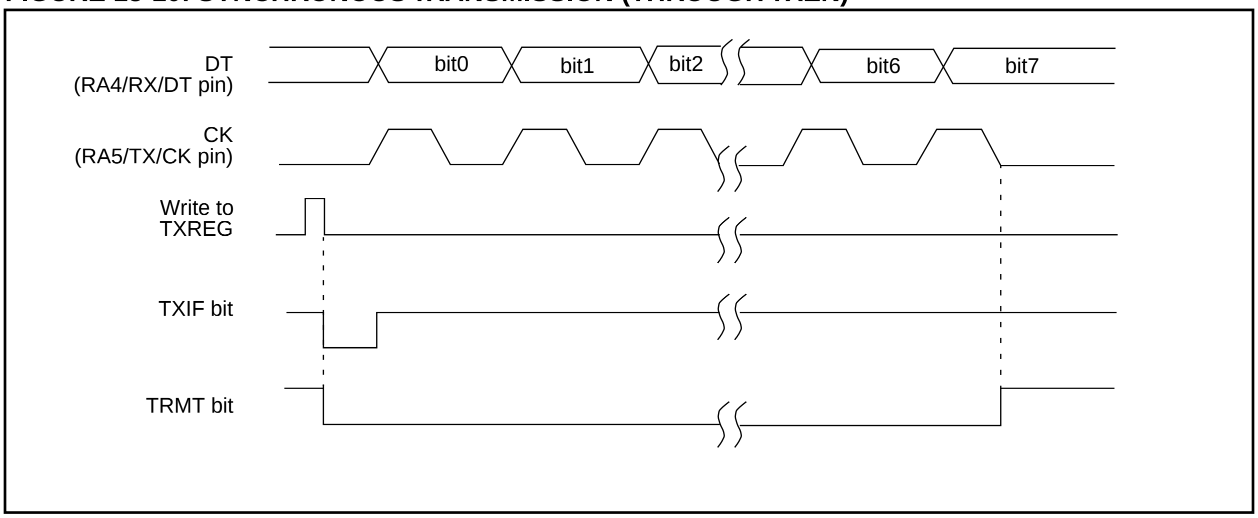


TABLE 15-2: PIC17CXX INSTRUCTION SET

Mnemonic,	Operands	Description	Cycles	16-bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	**f,d**	ADD WREG to f	1	0000	111d	ffff	ffff	OV,C,DC,Z	
ADDWFC	**f,d**	ADD WREG and Carry bit to f	1	0001	000d	ffff	ffff	OV,C,DC,Z	
ANDWF	**f,d**	AND WREG with f	1	0000	101d	ffff	ffff	Z	
CLRF	**f,s**	Clear f, or Clear f and Clear WREG	1	0010	100s	ffff	ffff	None	3
COMF	**f,d**	Complement f	1	0001	001d	ffff	ffff	Z	
CPFSEQ	**f**	Compare f with WREG, skip if f = WREG	1 (2)	0011	0001	ffff	ffff	None	6,8
CPFSGT	**f**	Compare f with WREG, skip if f > WREG	1 (2)	0011	0010	ffff	ffff	None	2,6,8
CPFSLT	**f**	Compare f with WREG, skip if f < WREG	1 (2)	0011	0000	ffff	ffff	None	2,6,8
DAW	**f,s**	Decimal Adjust WREG Register	1	0010	111s	ffff	ffff	C	3
DECF	**f,d**	Decrement f	1	0000	011d	ffff	ffff	OV,C,DC,Z	
DECFSZ	**f,d**	Decrement f, skip if 0	1 (2)	0001	011d	ffff	ffff	None	6,8
DCFSNZ	**f,d**	Decrement f, skip if not 0	1 (2)	0010	011d	ffff	ffff	None	6,8
INCF	**f,d**	Increment f	1	0001	010d	ffff	ffff	OV,C,DC,Z	
INCFSZ	**f,d**	Increment f, skip if 0	1 (2)	0001	111d	ffff	ffff	None	6,8
INFSNZ	**f,d**	Increment f, skip if not 0	1 (2)	0010	010d	ffff	ffff	None	6,8
IORWF	**f,d**	Inclusive OR WREG with f	1	0000	100d	ffff	ffff	Z	
MOVFP	**f,p**	Move f to p	1	011p	pppp	ffff	ffff	None	
MOVPF	**p,f**	Move p to f	1	010p	pppp	ffff	ffff	Z	
MOVWF	**f**	Move WREG to f	1	0000	0001	ffff	ffff	None	
MULWF	**f**	Multiply WREG with f	1	0011	0100	ffff	ffff	None	9
NEGW	**f,s**	Negate WREG	1	0010	110s	ffff	ffff	OV,C,DC,Z	1,3
NOP	**—**	No Operation	1	0000	0000	0000	0000	None	
RLCF	**f,d**	Rotate left f through Carry	1	0001	101d	ffff	ffff	C	
RLNCF	**f,d**	Rotate left f (no carry)	1	0010	001d	ffff	ffff	None	
RRCF	**f,d**	Rotate right f through Carry	1	0001	100d	ffff	ffff	C	
RRNCF	**f,d**	Rotate right f (no carry)	1	0010	000d	ffff	ffff	None	
SETF	**f,s**	Set f	1	0010	101s	ffff	ffff	None	3
SUBWF	**f,d**	Subtract WREG from f	1	0000	010d	ffff	ffff	OV,C,DC,Z	1
SUBWFB	**f,d**	Subtract WREG from f with Borrow	1	0000	001d	ffff	ffff	OV,C,DC,Z	1
SWAPF	**f,d**	Swap f	1	0001	110d	ffff	ffff	None	
TABLRD	**t,i,f**	Table Read	2 (3)	1010	10ti	ffff	ffff	None	7

Legend: Refer to Table 15-1 for opcode field descriptions.

Note 1: 2’s Complement method.

2: Unsigned arithmetic.

3: If s = '1', only the file is affected: If s = '0', both the WREG register and the file are affected; If only the Working register (WREG) is required to be affected, then f = WREG must be specified.

4: During an LCALL, the contents of PCLATH are loaded into the MSB of the PC and kkkk kkkk is loaded into the LSB of the PC (PCL)

5: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. When writing to external program memory, it is a two-cycle instruction.

6: Two-cycle instruction when condition is true, else single cycle instruction.

7: Two-cycle instruction except for TABLRD to PCL (program counter low byte) in which case it takes 3 cycles.

8: A “skip” means that instruction fetched during execution of current instruction is not executed, instead an NOP is executed.

9: These instructions are not available on the PIC17C42.

TABLE 15-2: PIC17CXX INSTRUCTION SET (Cont.'d)

Mnemonic, Operands		Description	Cycles	16-bit Opcode MSb			16-bit Opcode LSb	Status Affected	Notes
TABLWT	**t,i,f**	Table Write	2	1010	11ti	ffff	ffff	None	5
TLRD	**t,f**	Table Latch Read	1	1010	00tx	ffff	ffff	None	
TLWT	**t,f**	Table Latch Write	1	1010	01tx	ffff	ffff	None	
TSTFSZ	**f**	Test f, skip if 0	1 (2)	0011	0011	ffff	ffff	None	6,8
XORWF	**f,d**	Exclusive OR WREG with f	1	0000	110d	ffff	ffff	Z	
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	**f,b**	Bit Clear f	1	1000	1bbb	ffff	ffff	None	
BSF	**f,b**	Bit Set f	1	1000	0bbb	ffff	ffff	None	
BTFSC	**f,b**	Bit test, skip if clear	1 (2)	1001	1bbb	ffff	ffff	None	6,8
BTFSS	**f,b**	Bit test, skip if set	1 (2)	1001	0bbb	ffff	ffff	None	6,8
BTG	**f,b**	Bit Toggle f	1	0011	1bbb	ffff	ffff	None	
LITERAL AND CONTROL OPERATIONS									
ADDLW	**k**	ADD literal to WREG	1	1011	0001	kkkk	kkkk	OV,C,DC,Z	
ANDLW	**k**	AND literal with WREG	1	1011	0101	kkkk	kkkk	Z	
CALL	**k**	Subroutine Call	2	111k	kkkk	kkkk	kkkk	None	7
CLRWDT	**—**	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	**k**	Unconditional Branch	2	110k	kkkk	kkkk	kkkk	None	7
IORLW	**k**	Inclusive OR literal with WREG	1	1011	0011	kkkk	kkkk	Z	
LCALL	**k**	Long Call	2	1011	0111	kkkk	kkkk	None	4,7
MOVLB	**k**	Move literal to low nibble in BSR	1	1011	1000	uuuu	kkkk	None	
MOVLR	**k**	Move literal to high nibble in BSR	1	1011	101x	kkkk	uuuu	None	9
MOVLW	**k**	Move literal to WREG	1	1011	0000	kkkk	kkkk	None	
MULLW	**k**	Multiply literal with WREG	1	1011	1100	kkkk	kkkk	None	9
RETFIE	**—**	Return from interrupt (and enable interrupts)	2	0000	0000	0000	0101	GLINTD	7
RETLW	**k**	Return literal to WREG	2	1011	0110	kkkk	kkkk	None	7
RETURN	**—**	Return from subroutine	2	0000	0000	0000	0010	None	7
SLEEP	**—**	Enter SLEEP Mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	
SUBLW	**k**	Subtract WREG from literal	1	1011	0010	kkkk	kkkk	OV,C,DC,Z	
XORLW	**k**	Exclusive OR literal with WREG	1	1011	0100	kkkk	kkkk	Z	

Legend: Refer to Table 15-1 for opcode field descriptions.

Note 1: 2's Complement method.

2: Unsigned arithmetic.

3: If s = '1', only the file is affected: If s = '0', both the WREG register and the file are affected; If only the Working register (WREG) is required to be affected, then f = WREG must be specified.

4: During an `LCALL`, the contents of PCLATH are loaded into the MSB of the PC and `kkkk kkkk` is loaded into the LSB of the PC (PCL)

5: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. When writing to external program memory, it is a two-cycle instruction.

6: Two-cycle instruction when condition is true, else single cycle instruction.

7: Two-cycle instruction except for `TABLRD` to PCL (program counter low byte) in which case it takes 3 cycles.

8: A "skip" means that instruction fetched during execution of current instruction is not executed, instead an NOP is executed.

9: These instructions are not available on the PIC17C42.

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 $\rightarrow$ (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) $\rightarrow$ (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

CPFSEQ — Compare f with WREG, skip if f = WREG

Syntax:	[*label*] CPFSEQ f
Operands:	$0 \le f \le 255$
Operation:	(f) – (WREG), skip if (f) = (WREG) (unsigned comparison)
Status Affected:	None
Encoding:	`0011` `0001` `ffff` `ffff`
Description:	Compares the contents of data memory location 'f' to the contents of WREG by performing an unsigned subtraction. If 'f' = WREG then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction.
Words:	1
Cycles:	1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	NOP

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example:

```
HERE     CPFSEQ REG
NEQUAL   :
EQUAL    :
```

Before Instruction
- PC Address = `HERE`
- WREG = ?
- REG = ?

After Instruction
- If REG = WREG;
  - PC = Address (`EQUAL`)
- If REG ≠ WREG;
  - PC = Address (`NEQUAL`)

CPFSGT — Compare f with WREG, skip if f > WREG

Syntax:	[*label*] CPFSGT f
Operands:	$0 \le f \le 255$
Operation:	(f) – (WREG), skip if (f) > (WREG) (unsigned comparison)
Status Affected:	None
Encoding:	`0011` `0010` `ffff` `ffff`
Description:	Compares the contents of data memory location 'f' to the contents of the WREG by performing an unsigned subtraction. If the contents of 'f' > the contents of WREG then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction.
Words:	1
Cycles:	1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	NOP

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example:

```
HERE       CPFSGT REG
NGREATER   :
GREATER    :
```

Before Instruction
- PC = Address (`HERE`)
- WREG = ?

After Instruction
- If REG > WREG;
  - PC = Address (`GREATER`)
- If REG ≤ WREG;
  - PC = Address (`NGREATER`)

INFSNZ — Increment f, skip if not 0

Syntax:	[*label*] INFSNZ f,d
Operands:	$0 \le f \le 255$ $d \in [0,1]$
Operation:	(f) + 1 → (dest), skip if not 0
Status Affected:	None
Encoding:	`0010` `010d` `ffff` `ffff`
Description:	The contents of register 'f' are incremented. 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 not 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

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example:

```
HERE    INFSNZ  REG, 1
ZERO
NZERO
```

Before Instruction
REG = REG

After Instruction
REG = REG + 1
If REG = 1;
PC = Address (ZERO)
If REG = 0;
PC = Address (NZERO)

IORLW — Inclusive OR Literal with WREG

Syntax:	[*label*] IORLW k
Operands:	$0 \le k \le 255$
Operation:	(WREG) .OR. (k) → (WREG)
Status Affected:	Z
Encoding:	`1011` `0011` `kkkk` `kkkk`
Description:	The contents of WREG are OR'ed with the eight 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: `IORLW 0x35`

Before Instruction
WREG = 0x9A

After Instruction
WREG = 0xBF

Applicable Devices	42	R42	42A	43	R43	44

17.3 Timing Parameter Symbology

The timing parameter symbols have been created using one of the following formats:

1. TppS2ppS
2. TppS

T			
F	Frequency	T	Time

Lowercase symbols (pp) and their meanings:

pp			
ad	Address/Data	ost	Oscillator Start-up Timer
al	ALE	pwrt	Power-up Timer
cc	Capture1 and Capture2	rb	PORTB
ck	CLKOUT or clock	rd	$\overline{\text{RD}}$
dt	Data in	rw	$\overline{\text{RD}}$ or $\overline{\text{WR}}$
in	INT pin	t0	T0CKI
io	I/O port	t123	TCLK12 and TCLK3
mc	$\overline{\text{MCLR}}$	wdt	Watchdog Timer
oe	$\overline{\text{OE}}$	wr	$\overline{\text{WR}}$
os	OSC1		

Uppercase symbols and their meanings:

S			
D	Driven	L	Low
E	Edge	P	Period
F	Fall	R	Rise
H	High	V	Valid
I	Invalid (Hi-impedance)	Z	Hi-impedance

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 19-12: MEMORY INTERFACE READ TIMING (NOT SUPPORTED IN PIC17LC4X DEVICES)

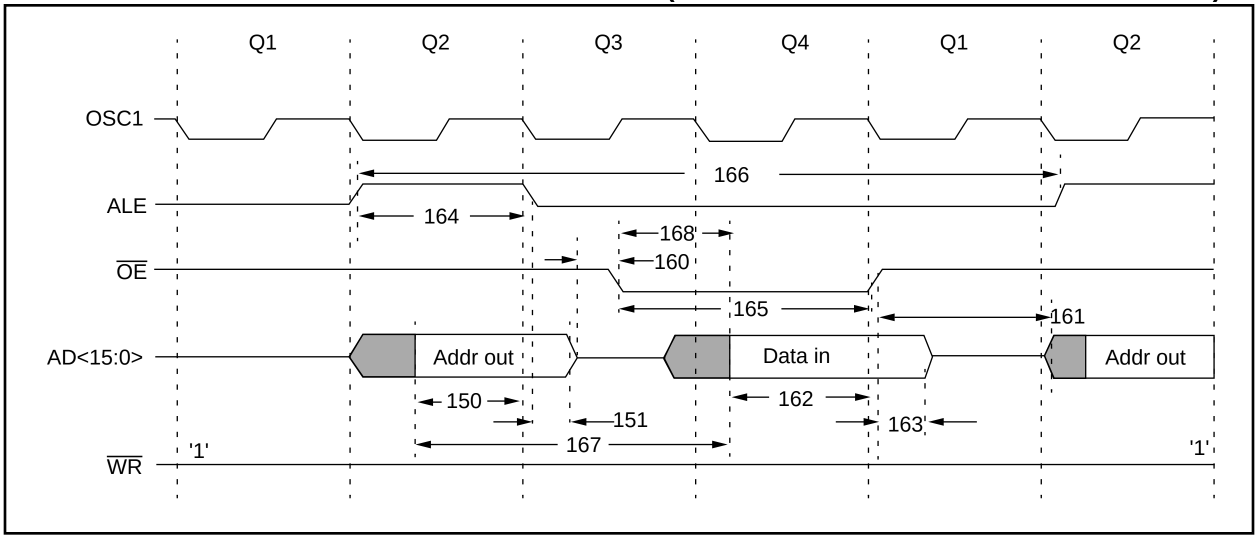


TABLE 19-12: MEMORY INTERFACE READ REQUIREMENTS (NOT SUPPORTED IN PIC17LC4X DEVICES)

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2alL	AD15:AD0 (address) valid to ALE↓ (address setup time)	0.25Tcy - 10	—	—	ns	
151	TalL2adI	ALE↓ to address out invalid (address hold time)	5*	—	—	ns	
160	TadZ2oeL	AD15:AD0 hi-impedance to $\overline{\text{OE}}$↓	0*	—	—	ns	
161	ToeH2adD	$\overline{\text{OE}}$↑ to AD15:AD0 driven	0.25Tcy - 15	—	—	ns	
162	TadV2oeH	Data in valid before $\overline{\text{OE}}$↑ (data setup time)	35	—	—	ns	
163	ToeH2adI	$\overline{\text{OE}}$↑to data in invalid (data hold time)	0	—	—	ns	
164	TalH	ALE pulse width	—	0.25TCY §	—	ns	
165	ToeL	$\overline{\text{OE}}$ pulse width	0.5Tcy - 35 §	—	—	ns	
166	TalH2alH	ALE↑ to ALE↑(cycle time)	—	TCY §	—	ns	
167	Tacc	Address access time	—	—	0.75TCY - 30	ns	
168	Toe	Output enable access time ($\overline{\text{OE}}$ low to Data Valid)	—	—	0.5TCY - 45	ns	

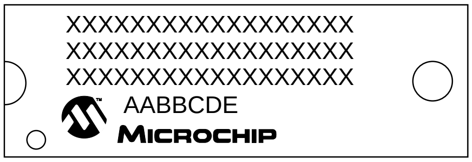
* These parameters are characterized but not tested.

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

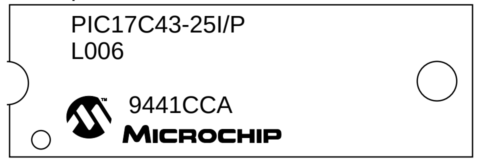
§ This specification ensured by design.

21.6 Package Marking Information

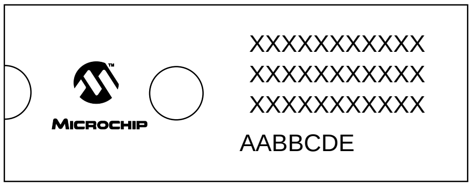
40-Lead PDIP/CERDIP



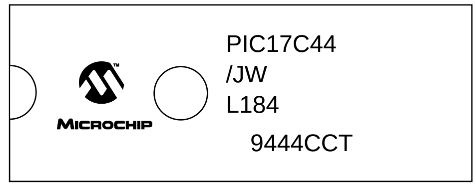
Example



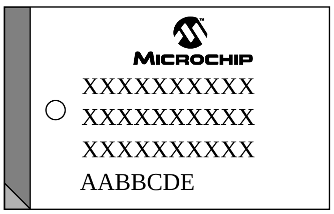
40 Lead CERDIP Windowed



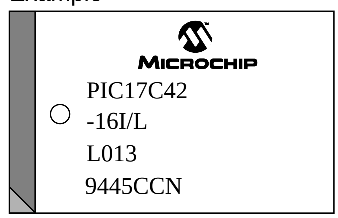
Example



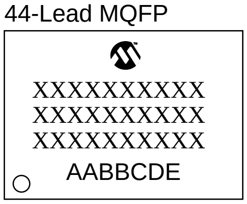
44-Lead PLCC



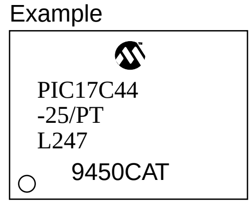
Example



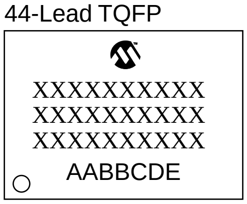
44-Lead MQFP



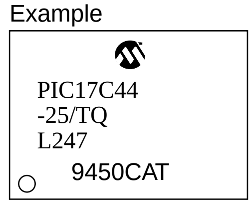
Example



44-Lead TQFP



Example



Legend:	MM...M	Microchip part number information
	XX...X	Customer specific information*
	AA	Year code (last 2 digits of calendar year)
	BB	Week code (week of January 1 is week '01')
	C	Facility code of the plant at which wafer is manufactured C = Chandler, Arizona, U.S.A., S = Tempe, Arizona, U.S.A.
	D	Mask revision number
	E	Assembly code of the plant or country of origin in which part was assembled

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

* Standard OTP marking consists of Microchip part number, year code, week code, facility code, mask rev#, and assembly code. For OTP marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

APPENDIX A: MODIFICATIONS

The following is the list of modifications over the PIC16CXX microcontroller family:

1. Instruction word length is increased to 16-bit. This allows larger page sizes both in program memory (8 Kwords verses 2 Kwords) and register file (256 bytes versus 128 bytes).
2. Four modes of operation: microcontroller, protected microcontroller, extended microcontroller, and microprocessor.
3. 22 new instructions.
   The `MOVF`, `TRIS` and `OPTION` instructions have been removed.
4. 4 new instructions for transferring data between data memory and program memory. This can be used to “self program” the EPROM program memory.
5. Single cycle data memory to data memory transfers possible (`MOVPF` and `MOVFP` instructions). These instructions do not affect the Working register (WREG).
6. W register (WREG) is now directly addressable.
7. A PC high latch register (PCLATH) is extended to 8-bits. The PCLATCH register is now both readable and writable.
8. Data memory paging is redefined slightly.
9. DDR registers replaces function of TRIS registers.
10. Multiple Interrupt vectors added. This can decrease the latency for servicing the interrupt.
11. Stack size is increased to 16 deep.
12. BSR register for data memory paging.
13. Wake up from SLEEP operates slightly differently.
14. The Oscillator Start-Up Timer (OST) and Power-Up Timer (PWRT) operate in parallel and not in series.
15. PORTB interrupt on change feature works on all eight port pins.
16. TMR0 is 16-bit plus 8-bit prescaler.
17. Second indirect addressing register added (FSR1 and FSR2). Configuration bits can select the FSR registers to auto-increment, auto-decrement, remain unchanged after an indirect address.
18. Hardware multiplier added (8 x 8 $\rightarrow$ 16-bit) (PIC17C43 and PIC17C44 only).
19. Peripheral modules operate slightly differently.
20. Oscillator modes slightly redefined.
21. Control/Status bits and registers have been placed in different registers and the control bit for globally enabling interrupts has inverse polarity.
22. Addition of a test mode pin.
23. In-circuit serial programming is not implemented.

APPENDIX B: COMPATIBILITY

To convert code written for PIC16CXX to PIC17CXX, the user should take the following steps:

1. Remove any `TRIS` and `OPTION` instructions, and implement the equivalent code.
2. Separate the interrupt service routine into its four vectors.
3. Replace:
   ```
   MOVF    REG1, W
   ```
   with:
   ```
   MOVFP   REG1, WREG
   ```
4. Replace:
   ```
   MOVF    REG1, W
   MOVWF   REG2
   ```
   with:
   ```
   MOVPF   REG1, REG2 ; Addr(REG1)<20h
   ```
   or
   ```
   MOVFP   REG1, REG2 ; Addr(REG2)<20h
   ```

> **Note:** If REG1 and REG2 are both at addresses greater then 20h, two instructions are required.
> ```
> MOVFP   REG1, WREG ;
> MOVPF   WREG, REG2 ;
> ```

5. Ensure that all bit names and register names are updated to new data memory map location.
6. Verify data memory banking.
7. Verify mode of operation for indirect addressing.
8. Verify peripheral routines for compatibility.
9. Weak pull-ups are enabled on reset.

To convert code from the PIC17C42 to all the other PIC17C4X devices, the user should take the following steps:

1. If the hardware multiply is to be used, ensure that any variables at address 18h and 19h are moved to another address.
2. Ensure that the upper nibble of the BSR was not written with a non-zero value. This may cause unexpected operation since the RAM bank is no longer 0.
3. The disabling of global interrupts has been enhanced so there is no additional testing of the GLINTD bit after a `BSF CPUSTA, GLINTD` instruction.

INDEX

A

B

C

D