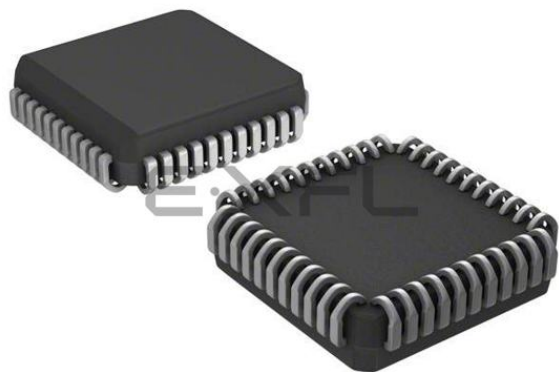




Welcome to [E-XFL.COM](https://www.e-xfl.com)

What is "[Embedded - Microcontrollers](#)"?

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

Applications of "[Embedded - Microcontrollers](#)"

Details

Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	33MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	8KB (4K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	454 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 6V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°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/pic17c43t-33-l

Table of Contents

1.0 Overview 5

2.0 PIC17C4X Device Varieties 7

3.0 Architectural Overview 9

4.0 Reset 15

5.0 Interrupts 21

6.0 Memory Organization 29

7.0 Table Reads and Table Writes 43

8.0 Hardware Multiplier 49

9.0 I/O Ports 53

10.0 Overview of Timer Resources 65

11.0 Timer0 67

12.0 Timer1, Timer2, Timer3, PWMs and Captures 71

13.0 Universal Synchronous Asynchronous Receiver Transmitter (USART) Module 83

14.0 Special Features of the CPU 99

15.0 Instruction Set Summary 107

16.0 Development Support 143

17.0 PIC17C42 Electrical Characteristics 147

18.0 PIC17C42 DC and AC Characteristics 163

19.0 PIC17CR42/42A/43/R43/44 Electrical Characteristics 175

20.0 PIC17CR42/42A/43/R43/44 DC and AC Characteristics 193

21.0 Packaging Information 205

Appendix A: Modifications 211

Appendix B: Compatibility 211

Appendix C: What's New 212

Appendix D: What's Changed 212

Appendix E: PIC16/17 Microcontrollers 213

Appendix F: Errata for PIC17C42 Silicon 223

Index 226

PIC17C4X Product Identification System 237

For register and module descriptions in this data sheet, device legends show which devices apply to those sections. For example, the legend below shows that some features of only the PIC17C43, PIC17CR43, PIC17C44 are described in this section.

Applicable Devices					
42	R42	42A	43	R43	44

To Our Valued Customers

We constantly strive to improve the quality of all our products and documentation. We have spent an exceptional amount of time to ensure that these documents are correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears in error from the previous version of the PIC17C4X Data Sheet (Literature Number DS30412B), please use the reader response form in the back of this data sheet to inform us. We appreciate your assistance in making this a better document.

To assist you in the use of this document, Appendix C contains a list of new information in this data sheet, while Appendix D contains information that has changed

PIC17C4X

NOTES:

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: TOIF, 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 (TOIF, 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	TOIF	INTF	PEIE	T0CKIE	TOIE	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: **TOIF:** 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: **TOIE:** 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

6.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC17C4X; program memory and data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into General Purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

6.1 Program Memory Organization

PIC17C4X devices have a 16-bit program counter capable of addressing a 64K x 16 program memory space. The reset vector is at 0000h and the interrupt vectors are at 0008h, 0010h, 0018h, and 0020h (Figure 6-1).

6.1.1 PROGRAM MEMORY OPERATION

The PIC17C4X can operate in one of four possible program memory configurations. The configuration is selected by two configuration bits. The possible modes are:

- Microprocessor
- Microcontroller
- Extended Microcontroller
- Protected Microcontroller

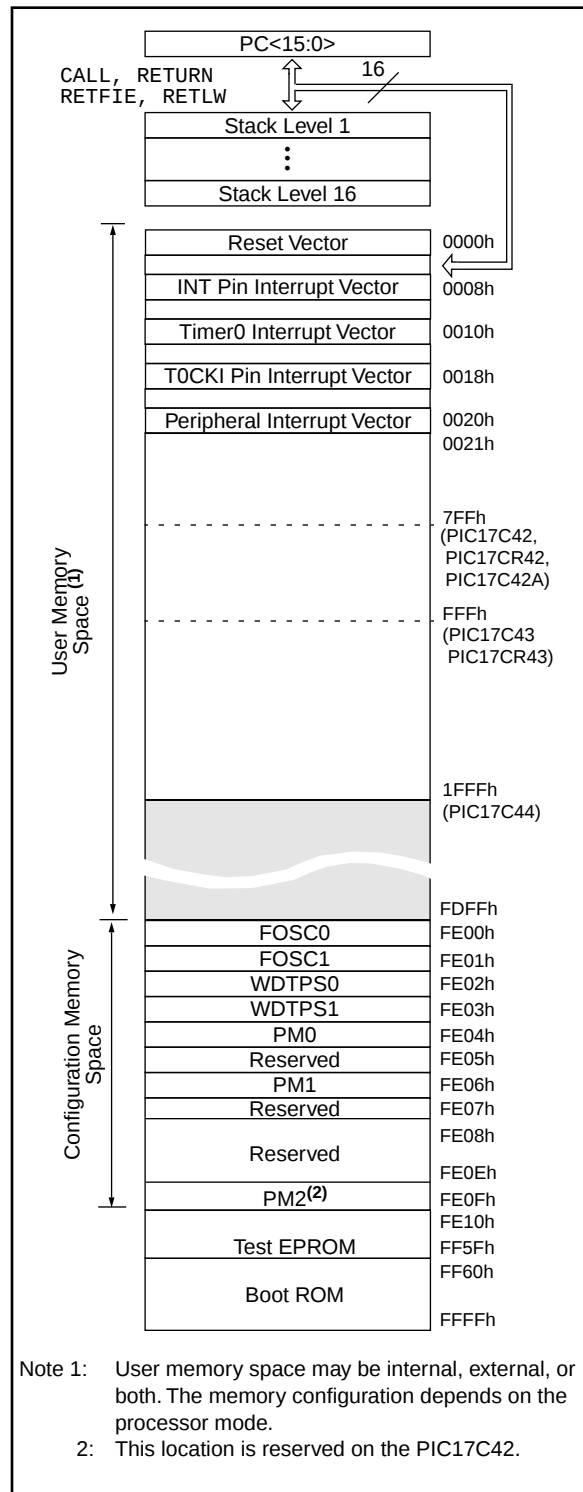
The microcontroller and protected microcontroller modes only allow internal execution. Any access beyond the program memory reads unknown data. The protected microcontroller mode also enables the code protection feature.

The extended microcontroller mode accesses both the internal program memory as well as external program memory. Execution automatically switches between internal and external memory. The 16-bits of address allow a program memory range of 64K-words.

The microprocessor mode only accesses the external program memory. The on-chip program memory is ignored. The 16-bits of address allow a program memory range of 64K-words. Microprocessor mode is the default mode of an unprogrammed device.

The different modes allow different access to the configuration bits, test memory, and boot ROM. Table 6-1 lists which modes can access which areas in memory. Test Memory and Boot Memory are not required for normal operation of the device. Care should be taken to ensure that no unintended branches occur to these areas.

FIGURE 6-1: PROGRAM MEMORY MAP AND STACK



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.

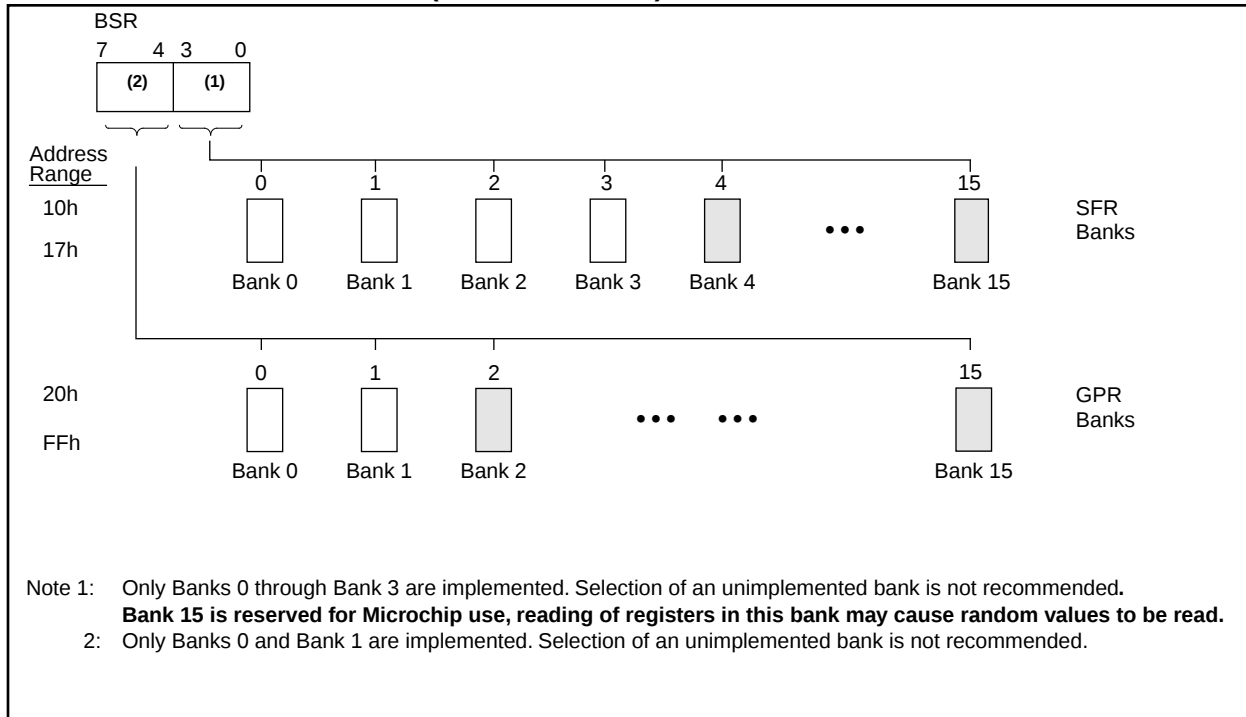
All the Special Function Registers (SFRs) are mapped into the data memory space. In order to accommodate the large number of registers, a banking scheme has been used. A segment of the SFRs, from address 10h to address 17h, is banked. The lower nibble of the bank select register (BSR) selects the currently 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 MOVLB 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)



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

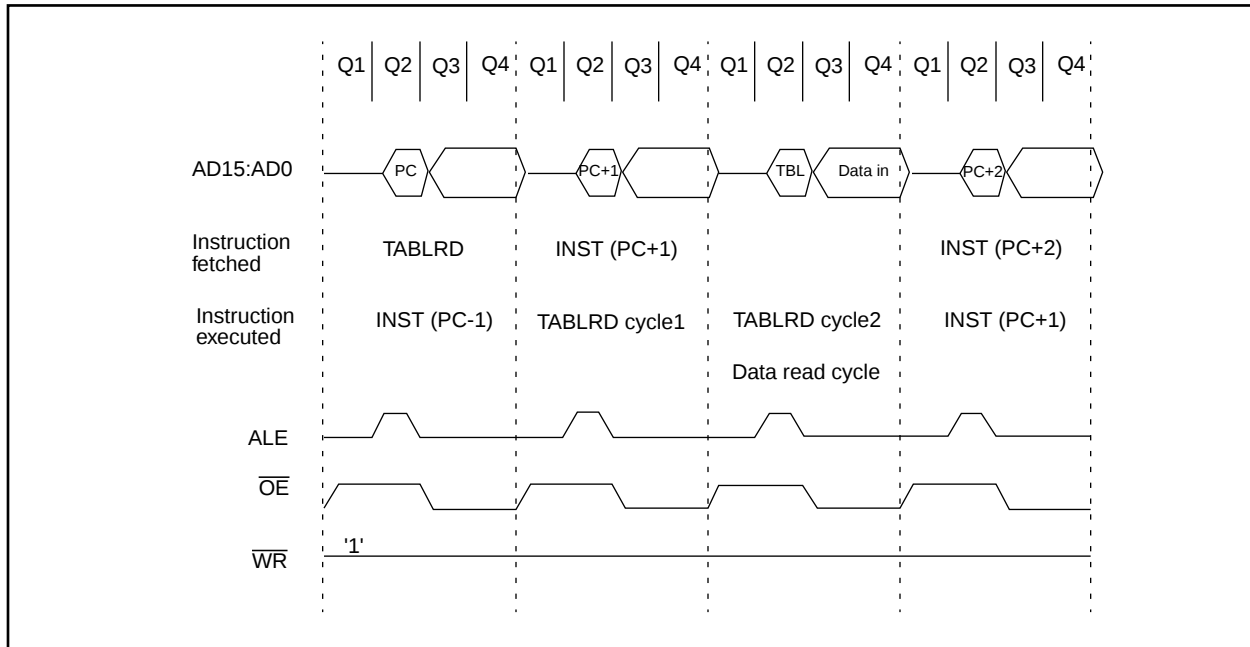
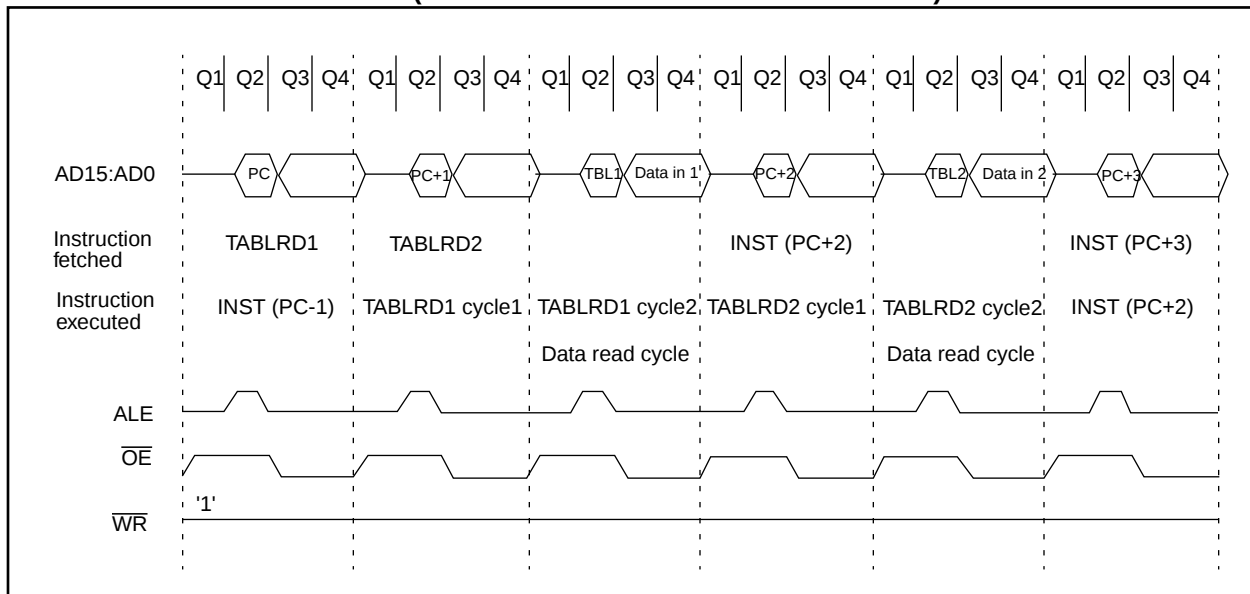


FIGURE 7-8: TABLRD TIMING (CONSECUTIVE TABLRD INSTRUCTIONS)



8.0 HARDWARE MULTIPLIER

All PIC17C4X devices except the PIC17C42, have an 8 x 8 hardware multiplier included in the ALU of the device. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored into the 16-bit PRODUct register (PRODH:PRODL). The multiplier does not affect any flags in the ALUSTA register.

Making the 8 x 8 multiplier execute in a single cycle gives the following advantages:

- Higher computational throughput
- Reduces code size requirements for multiply algorithms

The performance increase allows the device to be used in applications previously reserved for Digital Signal Processors.

Table 8-1 shows a performance comparison between the PIC17C42 and all other PIC17CXX devices, which have the single cycle hardware multiply.

Example 8-1 shows the sequence to do an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 8-2 shows the sequence to do an 8 x 8 signed multiply. To account for the sign bits of the arguments, each argument's most significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 8-1: 8 x 8 MULTIPLY ROUTINE

```
MOVFP    ARG1, WREG
MULWF    ARG2          ; ARG1 * ARG2 ->
                        ; PRODH:PRODL
```

EXAMPLE 8-2: 8 x 8 SIGNED MULTIPLY ROUTINE

```
MOVFP    ARG1, WREG
MULWF    ARG2          ; ARG1 * ARG2 ->
                        ; PRODH:PRODL

BTFSC    ARG2, SB      ; Test Sign Bit
SUBWF    PRODH, F       ; PRODH = PRODH
                        ; - ARG1

MOVFP    ARG2, WREG
BTFSC    ARG1, SB      ; Test Sign Bit
SUBWF    PRODH, F       ; PRODH = PRODH
                        ; - ARG2
```

TABLE 8-1: PERFORMANCE COMPARISON

Routine	Device	Program Memory (Words)	Cycles (Max)	Time	
				@ 25 MHz	@ 33 MHz
8 x 8 unsigned	PIC17C42	13	69	11.04 µs	N/A
	All other PIC17CXX devices	1	1	160 ns	121 ns
8 x 8 signed	PIC17C42	—	—	—	N/A
	All other PIC17CXX devices	6	6	960 ns	727 ns
16 x 16 unsigned	PIC17C42	21	242	38.72 µs	N/A
	All other PIC17CXX devices	24	24	3.84 µs	2.91 µs
16 x 16 signed	PIC17C42	52	254	40.64 µs	N/A
	All other PIC17CXX devices	36	36	5.76 µs	4.36 µs

TABLE 9-5: PORTC FUNCTIONS

Name	Bit	Buffer Type	Function
RC0/AD0	bit0	TTL	Input/Output or system bus address/data pin.
RC1/AD1	bit1	TTL	Input/Output or system bus address/data pin.
RC2/AD2	bit2	TTL	Input/Output or system bus address/data pin.
RC3/AD3	bit3	TTL	Input/Output or system bus address/data pin.
RC4/AD4	bit4	TTL	Input/Output or system bus address/data pin.
RC5/AD5	bit5	TTL	Input/Output or system bus address/data pin.
RC6/AD6	bit6	TTL	Input/Output or system bus address/data pin.
RC7/AD7	bit7	TTL	Input/Output or system bus address/data pin.

Legend: TTL = TTL input.

TABLE 9-6: REGISTERS/BITS ASSOCIATED WITH PORTC

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)
11h, Bank 1	PORTC	RC7/ AD7	RC6/ AD6	RC5/ AD5	RC4/ AD4	RC3/ AD3	RC2/ AD2	RC1/ AD1	RC0/ AD0	xxxx xxxx	uuuu uuuu
10h, Bank 1	DDRC	Data direction register for PORTC								1111 1111	1111 1111

Legend: x = unknown, u = unchanged.

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

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

```

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

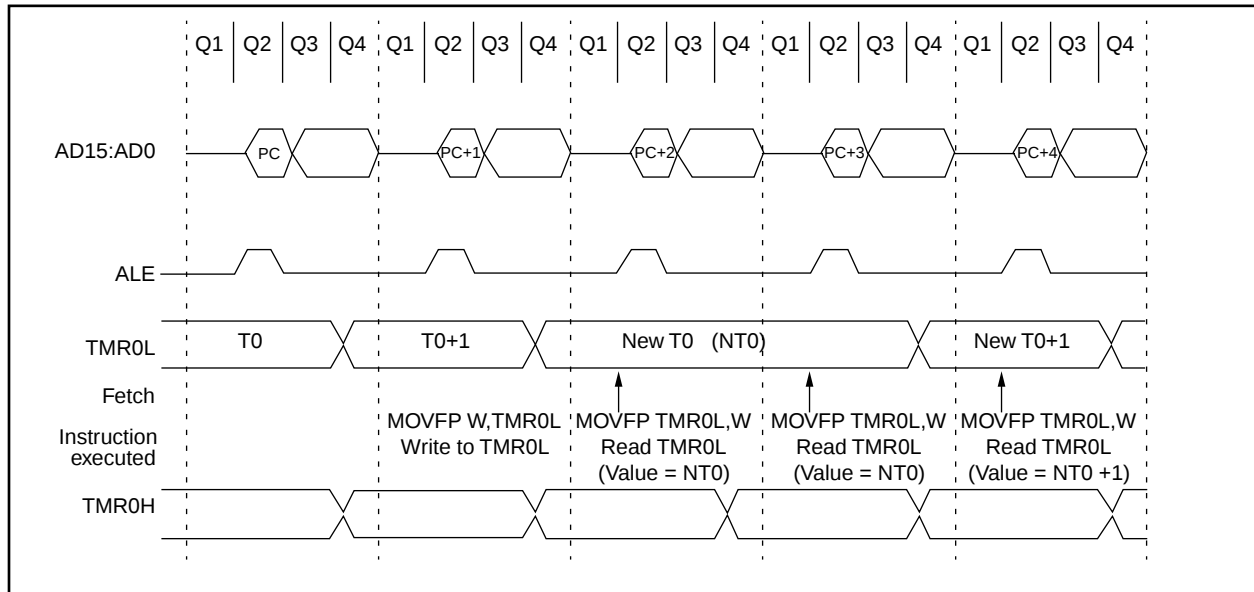
```

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 than the present value makes it difficult to take this unknown time into account.

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



13.3 USART Synchronous Master Mode

In Master Synchronous mode, the data is transmitted in a half-duplex manner; i.e. transmission and reception do not occur at the same time: when transmitting data, the reception is inhibited and vice versa. The synchronous mode is entered by setting the SYNC (TXSTA<4>) bit. In addition, the SPEN (RCSTA<7>) bit is set in order to configure the RA5 and RA4 I/O ports to CK (clock) and DT (data) lines respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting the CSRC (TXSTA<7>) bit.

13.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 13-3. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer TXREG. TXREG is loaded with data in software. The TSR is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from TXREG (if available). Once TXREG transfers the data to the TSR (occurs in one Tcy at the end of the current BRG cycle), TXREG is empty and the TXIF (PIR<1>) bit is set. This interrupt can be enabled/disabled by setting/clearing the TXIE bit (PIE<1>). TXIF will be set regardless of the state of bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into TXREG. While TXIF indicates the status of TXREG, TRMT (TXSTA<1>) shows the status of the TSR. TRMT is a read only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR is empty. The TSR is not mapped in data memory, so it is not available to the user.

Transmission is enabled by setting the TXEN (TXSTA<5>) bit. The actual transmission will not occur until TXREG has been loaded with data. The first data bit will be shifted out on the next available rising edge of the clock on the RA5/TX/CK pin. Data out is stable around the falling edge of the synchronous clock (Figure 13-10). The transmission can also be started by first loading TXREG and then setting TXEN. This is advantageous when slow baud rates are selected, since BRG is kept in RESET when the TXEN, CREN, and SREN bits are clear. Setting the TXEN bit will start the BRG, creating a shift clock immediately. Normally when transmission is first started, the TSR is empty, so a transfer to TXREG will result in an immediate transfer to the TSR, resulting in an empty TXREG. Back-to-back transfers are possible.

Clearing TXEN during a transmission will cause the transmission to be aborted and will reset the transmitter. The RA4/RX/DT and RA5/TX/CK pins will revert to hi-impedance. If either CREN or SREN are set during a transmission, the transmission is aborted and the

RA4/RX/DT pin reverts to a hi-impedance state (for a reception). The RA5/TX/CK pin will remain an output if the CSRC bit is set (internal clock). The transmitter logic is not reset, although it is disconnected from the pins. In order to reset the transmitter, the user has to clear the TXEN bit. If the SREN bit is set (to interrupt an ongoing transmission and receive a single word), then after the single word is received, SREN will be cleared and the serial port will revert back to transmitting, since the TXEN bit is still set. The DT line will immediately switch from hi-impedance receive mode to transmit and start driving. To avoid this, TXEN should be cleared.

In order to select 9-bit transmission, the TX9 (TXSTA<6>) bit should be set and the ninth bit should be written to TX9D (TXSTA<0>). The ninth bit must be written before writing the 8-bit data to TXREG. This is because a data write to TXREG can result in an immediate transfer of the data to the TSR (if the TSR is empty). If the TSR was empty and TXREG was written before writing the "new" TX9D, the "present" value of TX9D is loaded.

Steps to follow when setting up a Synchronous Master Transmission:

1. Initialize the SPBRG register for the appropriate baud rate (see Baud Rate Generator Section for details).
2. Enable the synchronous master serial port by setting the SYNC, SPEN, and CSRC bits.
3. Ensure that the CREN and SREN bits are clear (these bits override transmission when set).
4. If interrupts are desired, then set the TXIE bit (the GLINTD bit must be clear and the PEIE bit must be set).
5. If 9-bit transmission is desired, then set the TX9 bit.
6. Start transmission by loading data to the TXREG register.
7. If 9-bit transmission is selected, the ninth bit should be loaded in TX9D.
8. Enable the transmission by setting TXEN.

Writing the transmit data to the TXREG, then enabling the transmit (setting TXEN) allows transmission to start sooner than doing these two events in the reverse order.

Note: To terminate a transmission, either clear the SPEN bit, or the TXEN bit. This will reset the transmit logic, so that it will be in the proper state when transmit is re-enabled.

TABLE 13-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE 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 slave transmission.

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

TABLE 13-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE 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, Bank1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
13h, Bank0	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00u
14h, Bank0	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	xxxx xxxx	uuuu uuuu
17h, Bank1	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, Bank0	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 slave reception.

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

Table 15-2 lists the instructions recognized by the MPASM assembler.

Note 1: Any unused opcode is Reserved. Use of any reserved opcode may cause unexpected operation.

Note 2: The shaded instructions are not available in the PIC17C42

All instruction examples use the following format to represent a hexadecimal number:

0xhh

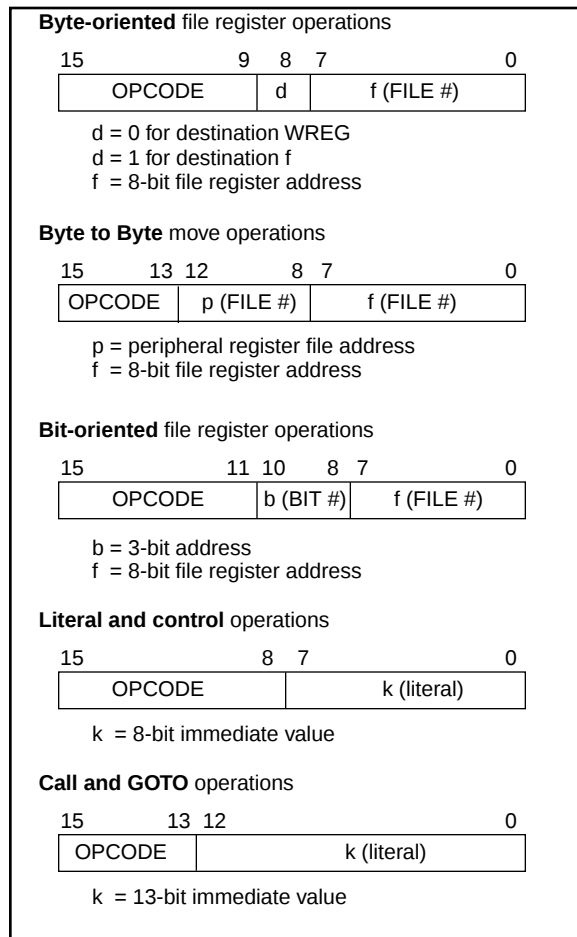
where h signifies a hexadecimal digit.

To represent a binary number:

0000 0100b

where b signifies a binary string.

FIGURE 15-1: GENERAL FORMAT FOR INSTRUCTIONS



15.1 Special Function Registers as Source/Destination

The PIC17C4X's orthogonal instruction set allows read and write of all file registers, including special function registers. There are some special situations the user should be aware of:

15.1.1 ALUSTA AS DESTINATION

If an instruction writes to ALUSTA, the Z, C, DC and OV bits may be set or cleared as a result of the instruction and overwrite the original data bits written. For example, executing CLRF ALUSTA will clear register ALUSTA, and then set the Z bit leaving 0000 0100b in the register.

15.1.2 PCL AS SOURCE OR DESTINATION

Read, write or read-modify-write on PCL may have the following results:

Read PC: PCH → PCLATH; PCL → dest

Write PCL: PCLATH → PCH;
8-bit destination value → PCL

Read-Modify-Write: PCL → ALU operand
PCLATH → PCH;
8-bit result → PCL

Where PCH = program counter high byte (not an addressable register), PCLATH = Program counter high holding latch, dest = destination, WREG or f.

15.1.3 BIT MANIPULATION

All bit manipulation instructions are done by first reading the entire register, operating on the selected bit and writing the result back (read-modify-write). The user should keep this in mind when operating on special function registers, such as ports.

CLRWDT Clear Watchdog Timer

Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT
 0 → WDT postscaler,
 1 → $\overline{TO}$
 1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0100
------	------	------	------

Description: CLRWDT instruction resets the watchdog timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ALUSTA	Execute	NOP

Example: CLRWDT

Before Instruction

 WDT counter = ?

After Instruction

 WDT counter = 0x00

 WDT Postscaler = 0

$\overline{TO}$ = 1

$\overline{PD}$ = 1

COMF Complement f

Syntax: [*label*] COMF f,d

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

Operation: $(\overline{f}) \rightarrow (\text{dest})$

Status Affected: Z

Encoding:

0001	001d	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are complemented. If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: COMF REG1, 0

Before Instruction

 REG1 = 0x13

After Instruction

 REG1 = 0x13

 WREG = 0xEC

MULLW		Multiply Literal with WREG			
Syntax:	[<i>label</i>] MULLW k				
Operands:	0 ≤ k ≤ 255				
Operation:	(k x WREG) → PRODH:PRODL				
Status Affected:	None				
Encoding:	1011	1100	kkkk	kkkk	
Description:	An unsigned multiplication is carried out between the contents of WREG and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte. WREG is unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.				
Words:	1				
Cycles:	1				

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Execute	Write registers PRODH: PRODL

Example: MULLW 0xC4

Before Instruction

WREG = 0xE2
 PRODH = ?
 PRODL = ?

After Instruction

WREG = 0xC4
 PRODH = 0xAD
 PRODL = 0x08

Note: This instruction is not available in the PIC17C42 device.

MULWF		Multiply WREG with f							
Syntax:	[<i>label</i>] MULWF f								
Operands:	$0 \leq f \leq 255$								
Operation:	$(\text{WREG} \times f) \rightarrow \text{PRODH:PRODL}$								
Status Affected:	None								
Encoding:	<table><tr><td>0011</td><td>0100</td><td>ffff</td><td>ffff</td></tr></table>					0011	0100	ffff	ffff
0011	0100	ffff	ffff						
Description:	An unsigned multiplication is carried out between the contents of WREG and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both WREG and 'f' are unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.								
Words:	1								
Cycles:	1								

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write registers PRODH: PRODL

Example: MULWF REG

Before Instruction

WREG = 0xC4
 REG = 0xB5
 PRODH = ?
 PRODL = ?

After Instruction

WREG = 0xC4
 REG = 0xB5
 PRODH = 0x8A
 PRODL = 0x94

Note: This instruction is not available in the PIC17C42 device.

PIC17C4X

SUBWF Subtract WREG from f

Syntax: [label] SUBWF f,d

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

Operation: $(f) - (W) \rightarrow (\text{dest})$

Status Affected: OV, C, DC, Z

Encoding:

0000	010d	ffff	ffff
------	------	------	------

Description: Subtract WREG from register 'f' (2's complement method). If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: SUBWF REG1, 1

Before Instruction

REG1 = 3
WREG = 2
C = ?

After Instruction

REG1 = 1
WREG = 2
C = 1 ; result is positive
Z = 0

Example 2:

Before Instruction

REG1 = 2
WREG = 2
C = ?

After Instruction

REG1 = 0
WREG = 2
C = 1 ; result is zero
Z = 1

Example 3:

Before Instruction

REG1 = 1
WREG = 2
C = ?

After Instruction

REG1 = FF
WREG = 2
C = 0 ; result is negative
Z = 0

SUBWFB Subtract WREG from f with Borrow

Syntax: [label] SUBWFB f,d

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

Operation: $(f) - (W) - \overline{C} \rightarrow (\text{dest})$

Status Affected: OV, C, DC, Z

Encoding:

0000	001d	ffff	ffff
------	------	------	------

Description: Subtract WREG and the carry flag (borrow) from register 'f' (2's complement method). If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: SUBWFB REG1, 1

Before Instruction

REG1 = 0x19 (0001 1001)
WREG = 0x0D (0000 1101)
C = 1

After Instruction

REG1 = 0x0C (0000 1011)
WREG = 0x0D (0000 1101)
C = 1 ; result is positive
Z = 0

Example2: SUBWFB REG1,0

Before Instruction

REG1 = 0x1B (0001 1011)
WREG = 0x1A (0001 1010)
C = 0

After Instruction

REG1 = 0x1B (0001 1011)
WREG = 0x00
C = 1 ; result is zero
Z = 1

Example3: SUBWFB REG1,1

Before Instruction

REG1 = 0x03 (0000 0011)
WREG = 0x0E (0000 1101)
C = 1

After Instruction

REG1 = 0xF5 (1111 0100) [2's comp]
WREG = 0x0E (0000 1101)
C = 0 ; result is negative
Z = 0

PIC17C4X

TABLRD Table Read

Example1: TABLRD 1, 1, REG ;

Before Instruction

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

After Instruction (table write completion)

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

Example2: TABLRD 0, 0, REG ;

Before Instruction

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

After Instruction (table write completion)

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

TABLWT Table Write

Syntax: [label] TABLWT t,i,f

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

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

Status Affected: None

Encoding:

1010	11ti	ffff	ffff
------	------	------	------

Description:

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

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

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

Words: 1

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

Q Cycle Activity:

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

PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 17-12: MEMORY INTERFACE READ TIMING

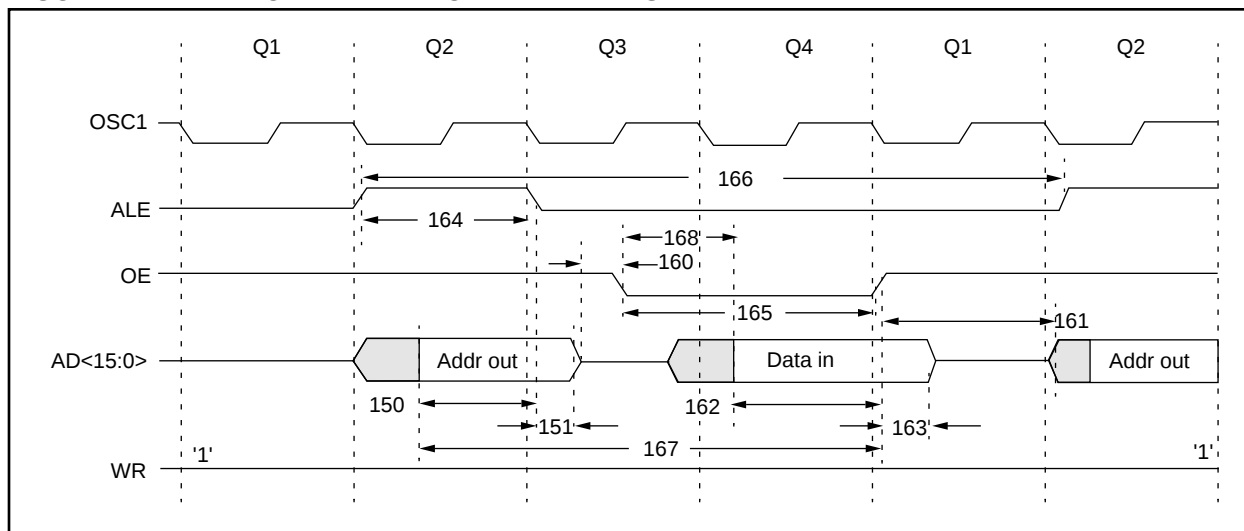


TABLE 17-12: MEMORY INTERFACE READ REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
150	TadV2aL	AD<15:0> (address) valid to ALE↓ (address setup time)	0.25Tcy - 30	—	—	ns	
151	TaL2adI	ALE↓ to address out invalid (address hold time)	5*	—	—	ns	
160	TadZ2oeL	AD<15:0> high impedance to OE↓	0*	—	—	ns	
161	ToeH2adD	OE↑ to AD<15:0> driven	0.25Tcy - 15	—	—	ns	
162	TadV2oeH	Data in valid before OE↑ (data setup time)	35	—	—	ns	
163	ToeH2adI	OE↑ to data in invalid (data hold time)	0	—	—	ns	
164	TaLH	ALE pulse width	—	0.25Tcy §	—	ns	
165	ToeL	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 (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.

PIC17C4X

Applicable Devices 42 R42 42A 43 R43 44

FIGURE 20-13: WDT TIMER TIME-OUT PERIOD vs. VDD

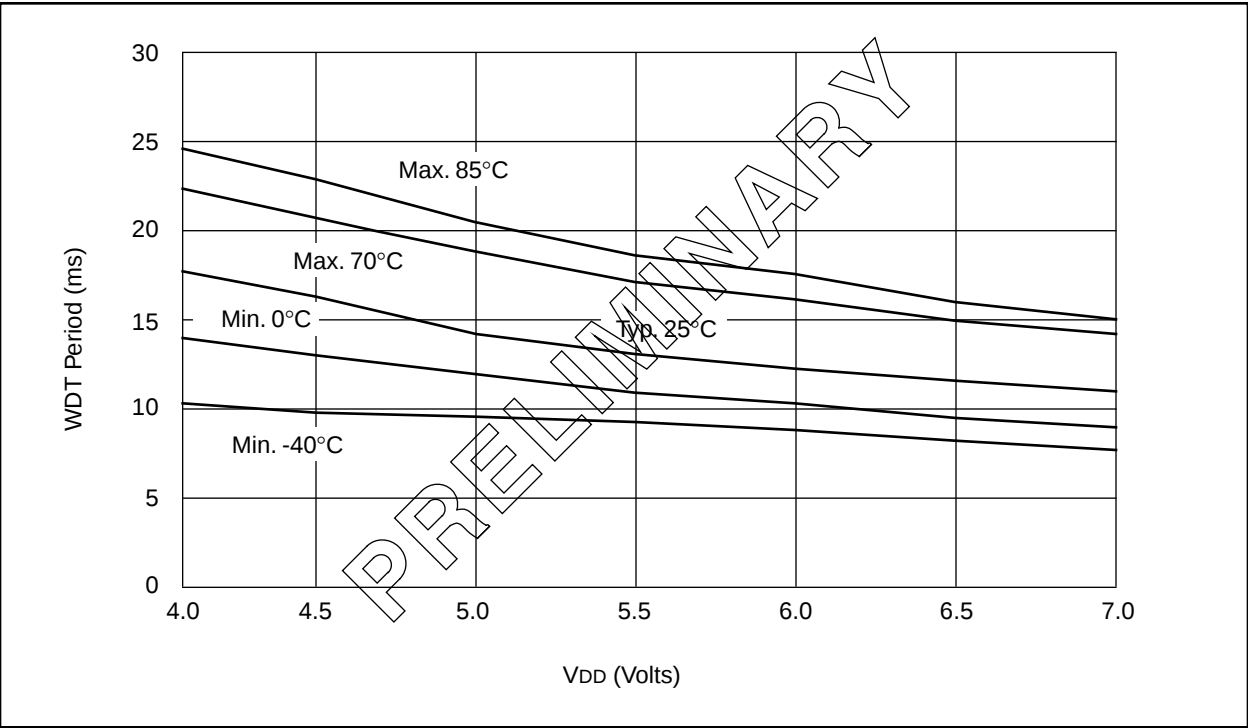
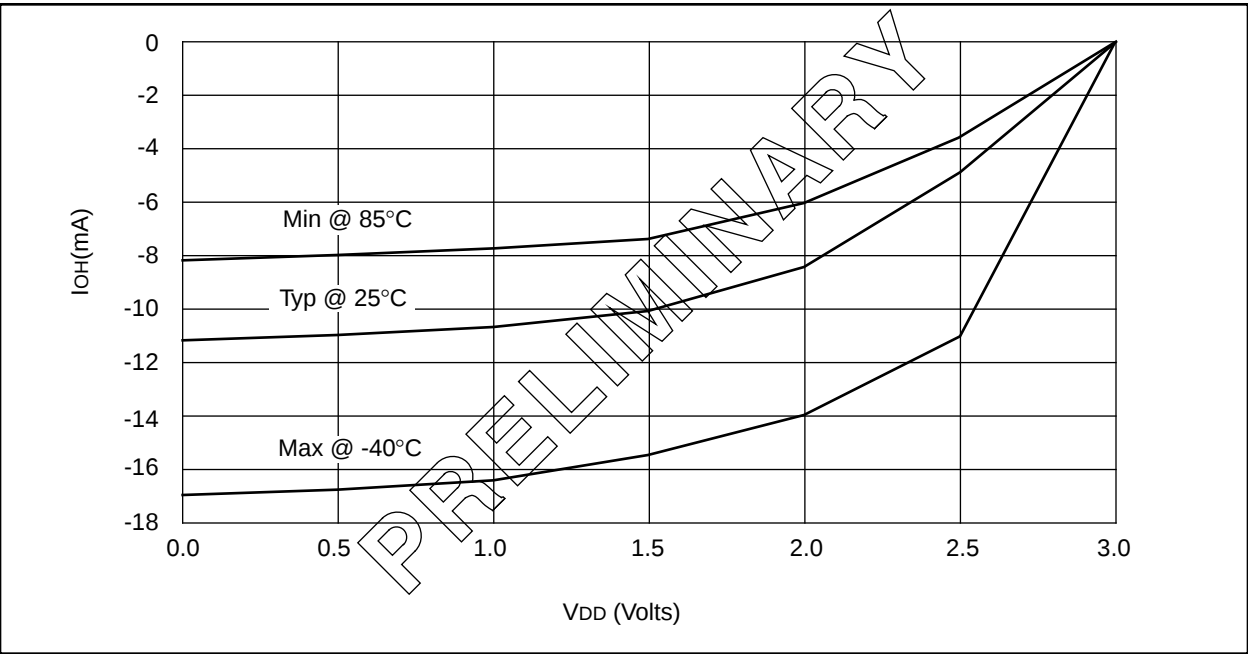


FIGURE 20-14: IOH vs. VOH, VDD = 3V



INDEX

A

ADDLW	112
ADDWF	112
ADDWFC	113
ALU	9
ALU STATUS Register (ALUSTA)	36
ALUSTA	34, 36, 108
ALUSTA Register	36
ANDLW	113
ANDWF	114
Application Notes	
AN552	55
Assembler	144
Asynchronous Master Transmission	90
Asynchronous Transmitter	89

B

Bank Select Register (BSR)	42
Banking	42
Baud Rate Formula	86
Baud Rate Generator (BRG)	86
Baud Rates	
Asynchronous Mode	88
Synchronous Mode	87
BCF	114
Bit Manipulation	108
Block Diagrams	
On-chip Reset Circuit	15
PIC17C42	10
PORTD	60
PORTE	62
PWM	75
RA0 and RA1	53
RA2 and RA3	54
RA4 and RA5	54
RB3:RB2 Port Pins	56
RB7:RB4 and RB1:RB0 Port Pins	55
RC7:RC0 Port Pins	58
Timer3 with One Capture and One Period Register ..	78
TMR1 and TMR2 in 16-bit Timer/Counter Mode	74
TMR1 and TMR2 in Two 8-bit Timer/Counter Mode ..	73
TMR3 with Two Capture Registers	79
WDT	104
BORROW	9
BRG	86
Brown-out Protection	18
BSF	115
BSR	34, 42
BSR Operation	42
BTFSC	115
BTFSS	116
BTG	116

C

C	9, 36
C Compiler (MP-C)	145
CA1/PR3	72
CA1ED0	71
CA1ED1	71

CA1IE	23
CA1IF	24
CA1OVF	72
CA2ED0	71
CA2ED1	71
CA2H	20, 35
CA2IE	23, 78
CA2IF	24, 78
CA2L	20, 35
CA2OVF	72
Calculating Baud Rate Error	86
CALL	39, 117
Capacitor Selection	
Ceramic Resonators	101
Crystal Oscillator	101
Capture	71, 78
Capture Sequence to Read Example	78
Capture1	
Mode	71
Overflow	72
Capture2	
Mode	71
Overflow	72
Carry (C)	9
Ceramic Resonators	100
Circular Buffer	39
Clearing the Prescaler	103
Clock/Instruction Cycle (Figure)	14
Clocking Scheme/Instruction Cycle (Section)	14
CLRF	117
CLRWDT	118
Code Protection	99, 106
COMF	118
Configuration	
Bits	100
Locations	100
Oscillator	100
Word	99
CPFSEQ	119
CPFSGT	119
CPFSLT	120
CPU STATUS Register (CPUTA)	37
CPUTA	34, 37, 105
CREN	84
Crystal Operation, Overtone Crystals	101
Crystal or Ceramic Resonator Operation	100
Crystal Oscillator	100
CSRC	83

D

Data Memory	
GPR	29, 32
Indirect Addressing	39
Organization	32
SFR	29, 32
Transfer to Program Memory	43
DAW	120
DC	9, 36
DDRB	19, 34, 55
DDRC	19, 34, 58
DDRD	19, 34, 60
DDRE	19, 34, 62
DECF	121
DECFSNZ	122
DECFSZ	121

WDT	99, 103
Clearing the WDT	103
Normal Timer	103
Period	103
Programming Considerations	103
WDTPS0	99
WDTPS1	99
WREG	34

X

XORLW	141
XORWF	141

Z

Z	9, 36
Zero (Z)	9

LIST OF EXAMPLES

Example 3-1: Signed Math	9
Example 3-2: Instruction Pipeline Flow	14
Example 5-1: Saving STATUS and WREG in RAM	27
Example 6-1: Indirect Addressing	40
Example 7-1: Table Write	46
Example 7-2: Table Read	48
Example 8-1: 8 x 8 Multiply Routine	49
Example 8-2: 8 x 8 Signed Multiply Routine	49
Example 8-3: 16 x 16 Multiply Routine	50
Example 8-4: 16 x 16 Signed Multiply Routine	51
Example 9-1: Initializing PORTB	57
Example 9-2: Initializing PORTC	58
Example 9-3: Initializing PORTD	60
Example 9-4: Initializing PORTE	62
Example 9-5: Read Modify Write Instructions on an I/O Port	64
Example 11-1: 16-Bit Read	69
Example 11-2: 16-Bit Write	69
Example 12-1: Sequence to Read Capture Registers	78
Example 12-2: Writing to TMR3	80
Example 12-3: Reading from TMR3	80
Example 13-1: Calculating Baud Rate Error	86
Example F-1: PIC17C42 to Sleep	223

LIST OF FIGURES

Figure 3-1: PIC17C42 Block Diagram	10
Figure 3-2: PIC17CR42/42A/43/R43/44 Block Diagram	11
Figure 3-3: Clock/Instruction Cycle	14
Figure 4-1: Simplified Block Diagram of On-chip Reset Circuit	15
Figure 4-2: Time-Out Sequence on Power-Up (MCLR Tied to VDD)	17
Figure 4-3: Time-Out Sequence on Power-Up (MCLR NOT Tied to VDD)	17
Figure 4-4: Slow Rise Time (MCLR Tied to VDD)	17
Figure 4-5: Oscillator Start-Up Time	18
Figure 4-6: Using On-Chip POR	18
Figure 4-7: Brown-out Protection Circuit 1	18
Figure 4-8: PIC17C42 External Power-On Reset Circuit (For Slow VDD Power-Up)	18
Figure 4-9: Brown-out Protection Circuit 2	18
Figure 5-1: Interrupt Logic	21
Figure 5-2: INTSTA Register (Address: 07h, Unbanked)	22
Figure 5-3: PIE Register (Address: 17h, Bank 1)	23
Figure 5-4: PIR Register (Address: 16h, Bank 1)	24
Figure 5-5: INT Pin / T0CKI Pin Interrupt Timing	26
Figure 6-1: Program Memory Map and Stack	29
Figure 6-2: Memory Map in Different Modes	30
Figure 6-3: External Program Memory Access Waveforms	31
Figure 6-4: Typical External Program Memory Connection Diagram	31
Figure 6-5: PIC17C42 Register File Map	33
Figure 6-6: PIC17CR42/42A/43/R43/44 Register File Map	33
Figure 6-7: ALUSTA Register (Address: 04h, Unbanked)	36
Figure 6-8: CPUSTA Register (Address: 06h, Unbanked)	37
Figure 6-9: TOSTA Register (Address: 05h, Unbanked)	38
Figure 6-10: Indirect Addressing	39
Figure 6-11: Program Counter Operation	41