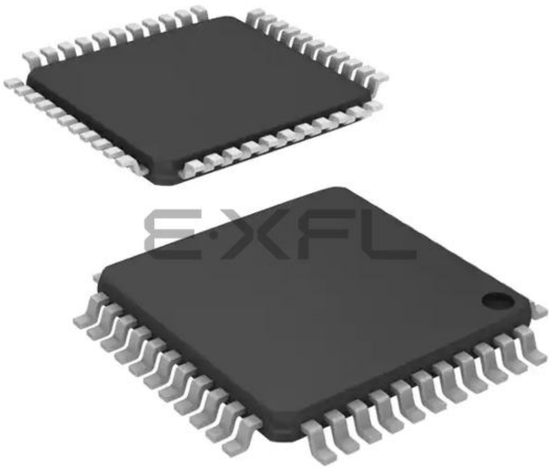


Welcome to E-XFL.COM

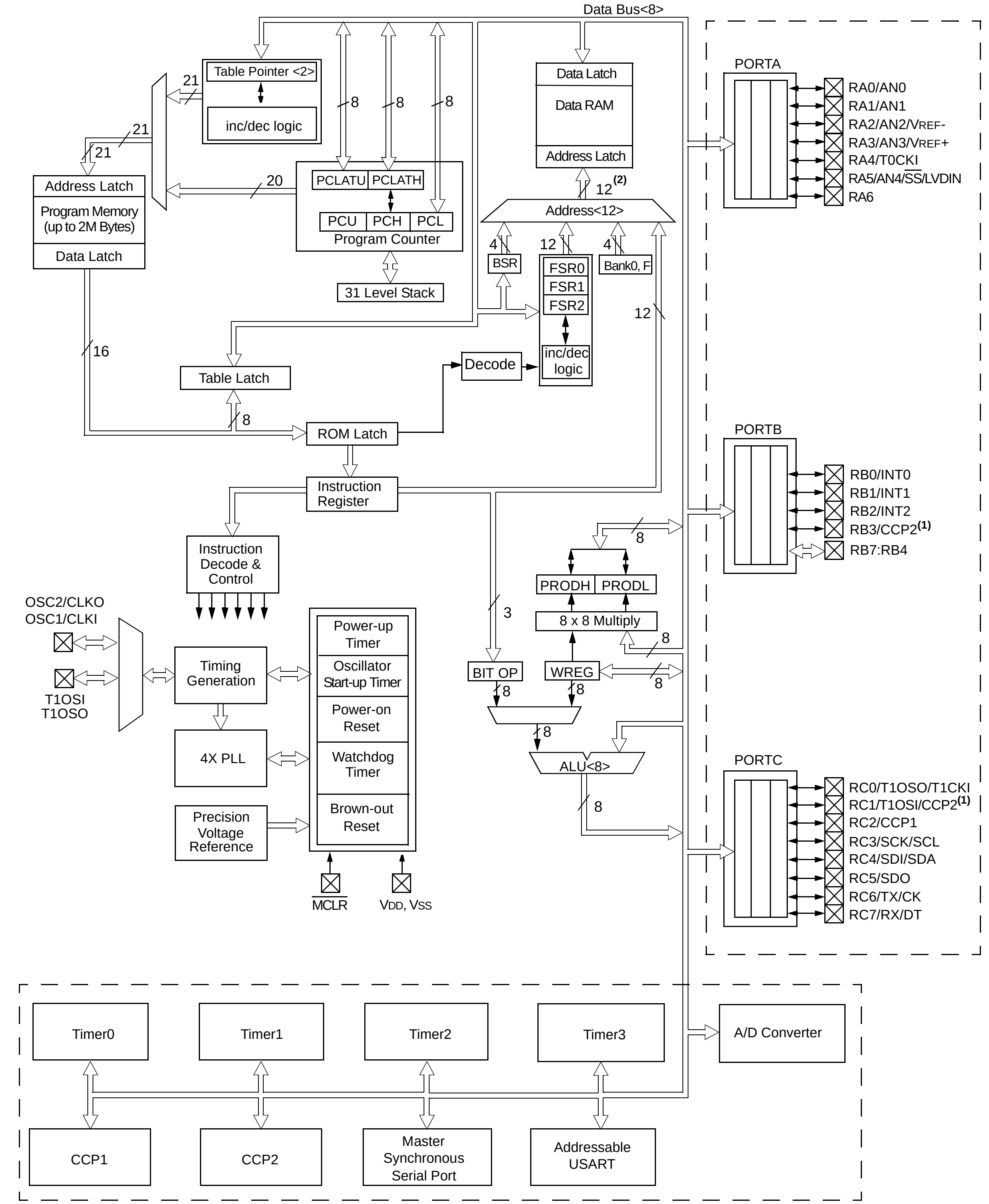
## What is "Embedded - Microcontrollers"?

"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 | 40MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, LVD, POR, PWM, WDT |
| Number of I/O | 33 |
| Program Memory Size | 32KB (16K x 16) |
| Program Memory Type | OTP |
| EEPROM Size | - |
| RAM Size | 1.5K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.5V ~ 5.5V |
| Data Converters | A/D 8x10b |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-TQFP |
| Supplier Device Package | 44-TQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic18lc452-i-pt |

**FIGURE 1-1: PIC18C2X2 BLOCK DIAGRAM**




**Note 1:** Optional multiplexing of CCP2 input/output with RB3 is enabled by selection of configuration bit.

**2:** The high order bits of the Direct Address for the RAM are from the BSR register (except for the `MOVFF` instruction).

**3:** Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

**TABLE 3-1: TIME-OUT IN VARIOUS SITUATIONS**

| Oscillator Configuration | Power-up(2) | | Brown-out(2) | Wake-up from SLEEP or Oscillator Switch |
|---|---|---|---|---|
| | $\overline{PWRTE}$ = 0 | $\overline{PWRTE}$ = 1 | | |
| HS with PLL enabled(1) | 72 ms + 1024TOSC + 2ms | 1024TOSC + 2 ms | 72 ms + 1024TOSC + 2ms | 1024TOSC + 2 ms |
| HS, XT, LP | 72 ms + 1024TOSC | 1024TOSC | 72 ms + 1024TOSC | 1024TOSC |
| EC | 72 ms | — | 72 ms | — |
| External RC | 72 ms | — | 72 ms | — |

**Note 1:** 2 ms is the nominal time required for the 4x PLL to lock.
**2:** 72 ms is the nominal Power-up Timer delay.

**REGISTER 3-1: RCON REGISTER BITS AND POSITIONS**

| R/W-0 | R/W-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | LWRT | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

**Note:** See Register 4-3 on page 53 for bit definitions.

**TABLE 3-2: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER**

| Condition | Program Counter | RCON Register | RI | TO | PD | POR | BOR | STKFUL | STKUNF |
|---|---|---|---|---|---|---|---|---|---|
| Power-on Reset | 0000h | 00-1 1100 | 1 | 1 | 1 | 0 | 0 | u | u |
| $\overline{MCLR}$ Reset during normal operation | 0000h | 00-u uuuu | u | u | u | u | u | u | u |
| Software Reset during normal operation | 0000h | 0u-0 uuuu | 0 | u | u | u | u | u | u |
| Stack Full Reset during normal operation | 0000h | 0u-u uu11 | u | u | u | u | u | u | 1 |
| Stack Underflow Reset during normal operation | 0000h | 0u-u uu11 | u | u | u | u | u | 1 | u |
| $\overline{MCLR}$ Reset during SLEEP | 0000h | 00-u 10uu | u | 1 | 0 | u | u | u | u |
| WDT Reset | 0000h | 0u-u 01uu | 1 | 0 | 1 | u | u | u | u |
| WDT Wake-up | PC + 2 | uu-u 00uu | u | 0 | 0 | u | u | u | u |
| Brown-out Reset | 0000h | 0u-1 11u0 | 1 | 1 | 1 | 1 | 0 | u | u |
| Interrupt wake-up from SLEEP | PC + 2(1) | uu-u 00uu | u | 1 | 0 | u | u | u | u |

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0'.
**Note 1:** When the wake-up is due to an interrupt and the GIEH or GIEL bits are set, the PC is loaded with the interrupt vector (0x000008h or 0x000018h).

**TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS**

| Register | Applicable Devices | | | | Power-on Reset, Brown-out Reset | MCLR Resets WDT Reset RESET Instruction Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|---|---|
| TOSU | 242 | 442 | 252 | 452 | ---0 0000 | ---0 0000 | ---0 uuuu(3) |
| TOSH | 242 | 442 | 252 | 452 | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| TOSL | 242 | 442 | 252 | 452 | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| STKPTR | 242 | 442 | 252 | 452 | 00-0 0000 | 00-0 0000 | uu-u uuuu(3) |
| PCLATU | 242 | 442 | 252 | 452 | ---0 0000 | ---0 0000 | ---u uuuu |
| PCLATH | 242 | 442 | 252 | 452 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCL | 242 | 442 | 252 | 452 | 0000 0000 | 0000 0000 | PC + 2(2) |
| TBLPTRU | 242 | 442 | 252 | 452 | --00 0000 | --00 0000 | --uu uuuu |
| TBLPTRH | 242 | 442 | 252 | 452 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TBLPTRL | 242 | 442 | 252 | 452 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TABLAT | 242 | 442 | 252 | 452 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PRODH | 242 | 442 | 252 | 452 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PRODL | 242 | 442 | 252 | 452 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INTCON | 242 | 442 | 252 | 452 | 0000 000x | 0000 000u | uuuu uuuu(1) |
| INTCON2 | 242 | 442 | 252 | 452 | 1111 -1-1 | 1111 -1-1 | uuuu -u-u(1) |
| INTCON3 | 242 | 442 | 252 | 452 | 11-0 0-00 | 11-0 0-00 | uu-u u-uu(1) |
| INDF0 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |
| POSTINC0 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |
| POSTDEC0 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |
| PREINC0 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |
| PLUSW0 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |
| FSR0H | 242 | 442 | 252 | 452 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR0L | 242 | 442 | 252 | 452 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| WREG | 242 | 442 | 252 | 452 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INDF1 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |
| POSTINC1 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |
| POSTDEC1 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |
| PREINC1 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |
| PLUSW1 | 242 | 442 | 252 | 452 | N/A | N/A | N/A |

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

**Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

**2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

**3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

**4:** See Table 3-2 for RESET value for specific condition.

**5:** Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

**6:** The long write enable is only reset on a POR or MCLR Reset.

**7:** Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they are read as '0'.

## 4.3 Fast Register Stack

A "fast interrupt return" option is available for interrupts. A Fast Register Stack is provided for the STATUS, WREG and BSR registers and are only one in depth. The stack is not readable or writable and is loaded with the current value of the corresponding register when the processor vectors for an interrupt. The values in the registers are then loaded back into the working registers, if the `FAST RETURN` instruction is used to return from the interrupt.

A low or high priority interrupt source will push values into the stack registers. If both low and high priority interrupts are enabled, the stack registers cannot be used reliably for low priority interrupts. If a high priority interrupt occurs while servicing a low priority interrupt, the stack register values stored by the low priority interrupt will be overwritten.

If high priority interrupts are not disabled during low priority interrupts, users must save the key registers in software during a low priority interrupt.

If no interrupts are used, the fast register stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the fast register stack for a subroutine call, a `FAST CALL` instruction must be executed.

Example 4-1 shows a source code example that uses the fast register stack.

**EXAMPLE 4-1: FAST REGISTER STACK CODE EXAMPLE**

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
        •
        •

SUB1    •
        •
        •
    RETURN FAST    ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

## 4.4 PCL, PCLATH and PCLATU

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21-bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<15:8> bits and is not directly readable or writable. Updates to the PCH register may be performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits and is not directly readable or writable. Updates to the PCU register may be performed through the PCLATU register.

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the LSB of PCL is fixed to a value of '0'. The PC increments by 2 to address sequential instructions in the program memory.

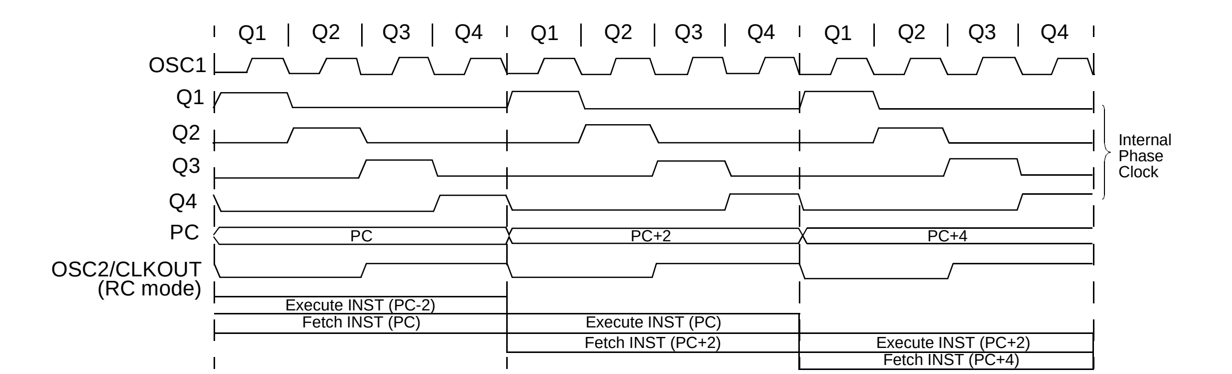
The `CALL`, `RCALL`, `GOTO` and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

The contents of PCLATH and PCLATU will be transferred to the program counter by an operation that writes PCL. Similarly, the upper two bytes of the program counter will be transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see Section 4.8.1).

## 4.5 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, 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 is shown in Figure 4-4.

**FIGURE 4-4: CLOCK/INSTRUCTION CYCLE**



#### 4.7.1 TWO-WORD INSTRUCTIONS

The PIC18CXX2 devices have four two-word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the 4 MSBs set to 1's and is a special kind of NOP instruction. The lower 12-bits of the second word contain data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two-word instruction is preceded by a conditional instruction that changes the PC. A program example that demonstrates this concept is shown in Example 4-3. Refer to Section 19.0 for further details of the instruction set.

**EXAMPLE 4-3: TWO-WORD INSTRUCTIONS**

| **CASE 1:** | | | |
|---|---|---|---|
| **Object Code** | **Source Code** | | |
| 0110 0110 0000 0000 | TSTFSZ | REG1 | ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF | REG1, REG2 | ; No, execute 2-word instruction |
| 1111 0100 0101 0110 | | | ; 2nd operand holds address of REG2 |
| 0010 0100 0000 0000 | ADDWF | REG3 | ; continue code |
| **CASE 2:** | | | |
| **Object Code** | **Source Code** | | |
| 0110 0110 0000 0000 | TSTFSZ | REG1 | ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF | REG1, REG2 | ; Yes |
| 1111 0100 0101 0110 | | | ; 2nd operand becomes NOP |
| 0010 0100 0000 0000 | ADDWF | REG3 | ; continue code |

## 4.8 Lookup Tables

Lookup tables are implemented two ways. These are:

- Computed GOTO
- Table Reads

#### 4.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

A lookup table can be formed with an ADDWF PCL instruction and a group of RETLW 0xnn instructions. WREG is loaded with an offset into the table, before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW 0xnn instructions that returns the value 0xnn to the calling function.

The offset value (value in WREG) specifies the number of bytes that the program counter should advance.

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

#### 4.8.2 TABLE READS/TABLE WRITES

A better method of storing data in program memory allows 2 bytes of data to be stored in each instruction location.

Lookup table data may be stored 2 bytes per program word by using table reads and writes. The table pointer (TBLPTR) specifies the byte address and the table latch (TABLAT) contains the data that is read from, or written to program memory. Data is transferred to/from program memory one byte at a time.

A description of the Table Read/Table Write operation is shown in Section 5.0.

## 5.1 Control Registers

Several control registers are used in conjunction with the TBLRD and TBLWT instructions. These include the:

- TBLPTR registers
- TABLAT register
- RCON register

### 5.1.1 RCON REGISTER

The LWRT bit specifies the operation of Table Writes to internal memory when the VPP voltage is applied to the MCLR pin. When the LWRT bit is set, the controller continues to execute user code, but long Table Writes are allowed (for programming internal program memory) from user mode. The LWRT bit can be cleared only by performing either a POR or MCLR Reset.

**REGISTER 5-1: RCON REGISTER (ADDRESS: FD0h)**

| R/W-0 | R/W-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | LWRT | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (16CXXX compatibility mode)

bit 6 **LWRT:** Long Write Enable bit
1 = Enable TBLWT to internal program memory
0 = Disable TBLWT to internal program memory.

**Note:** Only cleared on a POR or MCLR Reset.
This bit has no effect on TBLWTs to external program memory.

bit 5 **Unimplemented**: Read as '0'

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
1 = No RESET instruction occurred
0 = A RESET instruction occurred

bit 3 **$\overline{TO}$:** Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{PD}$:** Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
1 = No Brown-out Reset or POR Reset occurred
0 = A Brown-out Reset or POR Reset occurred
(must be set in software after a Brown-out Reset occurs)

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| - n = Value at POR reset | ’1’ = Bit is set | ’0’ = Bit is cleared | x = Bit is unknown |

## 8.0 I/O PORTS

Depending on the device selected, there are either five ports, or three ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Each port has three registers for its operation. These registers are:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)

The data latch (LAT register) is useful for read-modify-write operations on the value that the I/O pins are driving.

## 8.1 PORTA, TRISA and LATA Registers

PORTA is a 6-bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

| **Note:** | On a Power-on Reset, these pins are configured as digital inputs. |
|---|---|

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch.

The Data Latch register (LATA) is also memory mapped. Read-modify-write operations on the LATA register reads and writes the latched output value for PORTA.

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

The other PORTA pins are multiplexed with analog inputs and the analog VREF+ and VREF- inputs. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

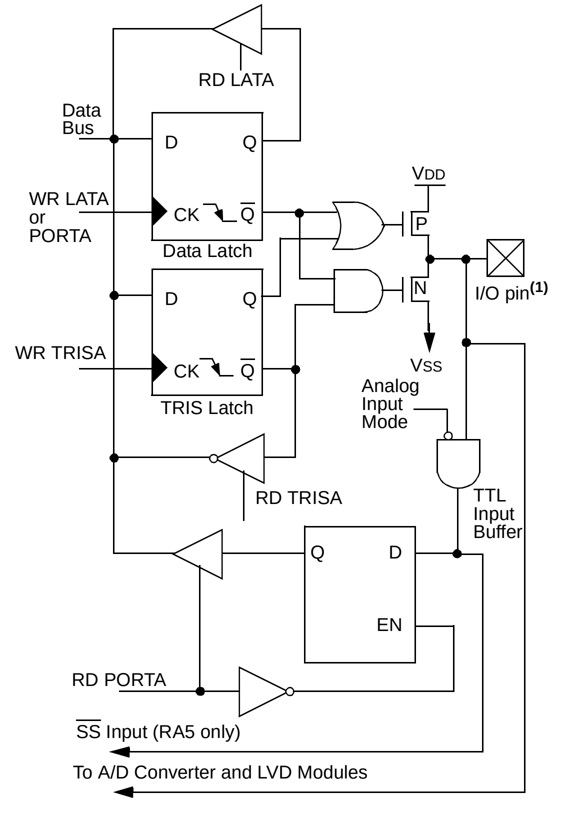
| **Note:** | On a Power-on Reset, these pins are configured as analog inputs and read as '0'. |
|---|---|

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

**EXAMPLE 8-1: INITIALIZING PORTA**

```
CLRF PORTA      ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF LATA       ; Alternate method
                ; to clear output
                ; data latches
MOVLW 0x07      ; Configure A/D
MOVWF ADCON1    ; for digital inputs
MOVLW 0xCF      ;Value used to
                ; initialize data
                ; direction
MOVWF TRISA     ; Set RA<3:0> as inputs
                ; RA<5:4> as outputs
```

**FIGURE 8-1: BLOCK DIAGRAM OF RA3:RA0 AND RA5 PINS**




Note 1: I/O pins have protection diodes to VDD and VSS.

**TABLE 8-1: PORTA FUNCTIONS**

| Name | Bit# | Buffer | Function |
|---|---|---|---|
| RA0/AN0 | bit0 | TTL | Input/output or analog input. |
| RA1/AN1 | bit1 | TTL | Input/output or analog input. |
| RA2/AN2/VREF- | bit2 | TTL | Input/output or analog input or VREF-. |
| RA3/AN3/VREF+ | bit3 | TTL | Input/output or analog input or VREF+. |
| RA4/T0CKI | bit4 | ST | Input/output or external clock input for Timer0.<br>Output is open drain type. |
| RA5/$\overline{SS}$/AN4/LVDIN | bit5 | TTL | Input/output or slave select input for synchronous serial port or analog input, or low voltage detect input. |
| OSC2/CLKO/RA6 | bit6 | TTL | OSC2 or clock output or I/O pin. |

Legend: TTL = TTL input, ST = Schmitt Trigger input

**TABLE 8-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|
| PORTA | — | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | --0x 0000 | --0u 0000 |
| LATA | — | Latch A Data Output Register | | | | | | | --xx xxxx | --uu uuuu |
| TRISA | — | PORTA Data Direction Register | | | | | | | --11 1111 | --11 1111 |
| ADCON1 | ADFM | ADCS2 | — | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | --0- 0000 | --0- 0000 |

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.
Shaded cells are not used by PORTA.

## 13.3 Capture Mode

In Capture mode, CCPR1H:CCPR1L captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on pin RC2/CCP1. An event is defined as:

- every falling edge
- every rising edge
- every 4th rising edge
- every 16th rising edge

An event is selected by control bits CCP1M3:CCP1M0 (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit CCP1IF (PIR1<2>) is set. It must be cleared in software. If another capture occurs before the value in register CCPR1 is read, the old captured value will be lost.

### 13.3.1 CCP PIN CONFIGURATION

In Capture mode, the RC2/CCP1 pin should be configured as an input by setting the TRISC<2> bit.

> **Note:** If the RC2/CCP1 is configured as an output, a write to the port can cause a capture condition.

### 13.3.2 TIMER1/TIMER3 MODE SELECTION

The timers that are to be used with the capture feature (either Timer1 and/or Timer3) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work. The timer to be used with each CCP module is selected in the T3CON register.

### 13.3.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit CCP1IE (PIE1<2>) clear to avoid false interrupts and should clear the flag bit, CCP1IF, following any such change in operating mode.

### 13.3.4 CCP PRESCALER

There are four prescaler settings, specified by bits CCP1M3:CCP1M0. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. This means that any RESET will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared, therefore, the first capture may be from a non-zero prescaler. Example 13-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

**EXAMPLE 13-1: CHANGING BETWEEN CAPTURE PRESCALERS**

```
CLRF    CCP1CON, F  ; Turn CCP module off
MOVLW   NEW_CAPT_PS ; Load WREG with the
                    ; new prescaler mode
                    ; value and CCP ON
MOVWF   CCP1CON     ; Load CCP1CON with
                    ; this value
```

**FIGURE 13-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**

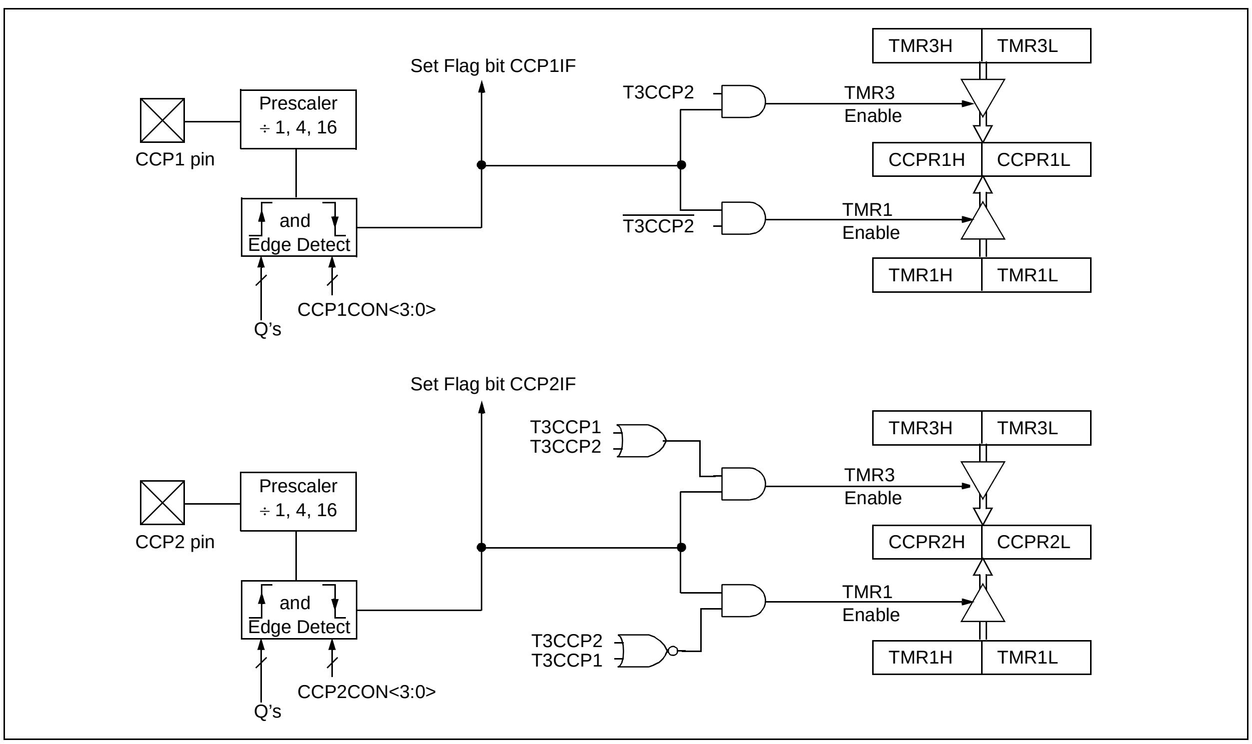


**TABLE 15-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)**

| BAUD RATE (K) | FOSC = 40 MHz | | | FOSC = 20 MHz | | | FOSC = 16 MHz | | | FOSC = 10 MHz | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | NA | — | — | NA | — | — | NA | — | — | NA | — | — |
| 1.2 | NA | — | — | 1.221 | +1.73 | 255 | 1.202 | +0.16 | 207 | 1.202 | +0.16 | 129 |
| 2.4 | 2.44 | -1.70 | 255 | 2.404 | +0.16 | 129 | 2.404 | +0.16 | 103 | 2.404 | +0.16 | 64 |
| 9.6 | 9.62 | -0.16 | 64 | 9.469 | -1.36 | 32 | 9.615 | +0.16 | 25 | 9.766 | +1.73 | 15 |
| 19.2 | 18.94 | +1.38 | 32 | 19.53 | +1.73 | 15 | 19.23 | +0.16 | 12 | 19.53 | +1.73 | 7 |
| 76.8 | 78.13 | -1.70 | 7 | 78.13 | +1.73 | 3 | 83.33 | +8.51 | 2 | 78.13 | +1.73 | 1 |
| 96 | 89.29 | +7.52 | 6 | 104.2 | +8.51 | 2 | NA | — | — | NA | — | — |
| 300 | 312.50 | -4.00 | 1 | 312.5 | +4.17 | 0 | NA | — | — | NA | — | — |
| 500 | 625.00 | -20.00 | 0 | NA | — | — | NA | — | — | NA | — | — |
| HIGH | 2.44 | — | 255 | 312.5 | — | 0 | 250 | — | 0 | 156.3 | — | 0 |
| LOW | 625.00 | — | 0 | 1.221 | — | 255 | 0.977 | — | 255 | 0.6104 | — | 255 |

| BAUD RATE (K) | FOSC = 7.15909 MHz | | | FOSC = 5.0688 MHz | | | FOSC = 4 MHz | | | FOSC = 3.579545 MHz | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | NA | — | — | 0.31 | +3.13 | 255 | 0.3005 | -0.17 | 207 | 0.301 | +0.23 | 185 |
| 1.2 | 1.203 | +0.23 | 92 | 1.2 | 0 | 65 | 1.202 | +1.67 | 51 | 1.190 | -0.83 | 46 |
| 2.4 | 2.380 | -0.83 | 46 | 2.4 | 0 | 32 | 2.404 | +1.67 | 25 | 2.432 | +1.32 | 22 |
| 9.6 | 9.322 | -2.90 | 11 | 9.9 | +3.13 | 7 | NA | — | — | 9.322 | -2.90 | 5 |
| 19.2 | 18.64 | -2.90 | 5 | 19.8 | +3.13 | 3 | NA | — | — | 18.64 | -2.90 | 2 |
| 76.8 | NA | — | — | 79.2 | +3.13 | 0 | NA | — | — | NA | — | — |
| 96 | NA | — | — | NA | — | — | NA | — | — | NA | — | — |
| 300 | NA | — | — | NA | — | — | NA | — | — | NA | — | — |
| 500 | NA | — | — | NA | — | — | NA | — | — | NA | — | — |
| HIGH | 111.9 | — | 0 | 79.2 | — | 0 | 62.500 | — | 0 | 55.93 | — | 0 |
| LOW | 0.437 | — | 255 | 0.3094 | — | 255 | 3.906 | — | 255 | 0.2185 | — | 255 |

| BAUD RATE (K) | FOSC = 1 MHz | | | FOSC = 32.768 kHz | | |
|---|---|---|---|---|---|---|
| | Actual Rate (K) | % Error | SPBRG value (decimal) | Actual Rate (K) | % Error | SPBRG value (decimal) |
| 0.3 | 0.300 | +0.16 | 51 | 0.256 | -14.67 | 1 |
| 1.2 | 1.202 | +0.16 | 12 | NA | — | — |
| 2.4 | 2.232 | -6.99 | 6 | NA | — | — |
| 9.6 | NA | — | — | NA | — | — |
| 19.2 | NA | — | — | NA | — | — |
| 76.8 | NA | — | — | NA | — | — |
| 96 | NA | — | — | NA | — | — |
| 300 | NA | — | — | NA | — | — |
| 500 | NA | — | — | NA | — | — |
| HIGH | 15.63 | — | 0 | 0.512 | — | 0 |
| LOW | 0.0610 | — | 255 | 0.0020 | — | 255 |

**NOTES:**

## BTG — Bit Toggle f

| | |
|---|---|
| Syntax: | [ *label* ] BTG f,b[,a] |
| Operands: | $0 \le f \le 255$<br>$0 \le b < 7$<br>$a \in [0,1]$ |
| Operation: | $(\overline{f<b>}) \rightarrow f<b>$ |
| Status Affected: | None |
| Encoding: | `0111` `bbba` `ffff` `ffff` |
| Description: | Bit 'b' in data memory location 'f' is inverted. If ‘a’ is 0, the Access Bank will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value (default). |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example:

```
BTG     PORTC,  4, 0
```

Before Instruction:

```
PORTC   =   0111 0101 [0x75]
```

After Instruction:

```
PORTC   =   0110 0101 [0x65]
```

## BOV — Branch if Overflow

| | |
|---|---|
| Syntax: | [ *label* ] BOV n |
| Operands: | $-128 \le n \le 127$ |
| Operation: | if overflow bit is ’1’<br>$(PC) + 2 + 2n \rightarrow PC$ |
| Status Affected: | None |
| Encoding: | `1110` `0100` `nnnn` `nnnn` |
| Description: | If the Overflow bit is ’1’, then the program will branch.<br>The 2’s complement number ’2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction. |
| Words: | 1 |
| Cycles: | 1(2) |

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:

```
HERE      BOV  Jump
```

Before Instruction

```
PC          =   address (HERE)
```

After Instruction

```
If Overflow =   1;
   PC       =   address (Jump)
If Overflow =   0;
   PC       =   address (HERE+2)
```

### DECFSZ — Decrement f, skip if 0

Syntax: [ *label* ] DECFSZ f [,d [,a]]

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

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

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 11da | ffff | ffff |

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default).
If the result is 0, the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

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

Before Instruction

```
PC      =   Address (HERE)
```

After Instruction

```
CNT     =   CNT - 1
If CNT  =   0;
   PC   =   Address (CONTINUE)
If CNT  ≠   0;
   PC   =   Address (HERE+2)
```

### DCFSNZ — Decrement f, skip if not 0

Syntax: [*label*] DCFSNZ f [,d [,a]

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

Operation: (f) – 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 11da | ffff | ffff |

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default).
If the result is not 0, the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE     DCFSNZ  TEMP, 1, 0
ZERO     :
NZERO    :
```

Before Instruction

```
TEMP          =   ?
```

After Instruction

```
TEMP          =   TEMP - 1,
If TEMP       =   0;
   PC         =   Address (ZERO)
If TEMP       ≠   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: | N,Z |
| Encoding: | `0000` `1001` `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' | Process Data | Write to WREG |

Example: `IORLW 0x35`

Before Instruction

```
WREG   =   0x9A
```

After Instruction

```
WREG   =   0xBF
```

**IORWF** **Inclusive OR WREG with f**

| | |
|---|---|
| Syntax: | [ *label* ] IORWF f [,d [,a] |
| Operands: | $0 \le f \le 255$<br>$d \in [0,1]$<br>$a \in [0,1]$ |
| Operation: | (WREG) .OR. (f) → dest |
| Status Affected: | N,Z |
| Encoding: | `0001` `00da` `ffff` `ffff` |
| Description: | Inclusive OR WREG with register 'f'. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `IORWF RESULT, 0, 1`

Before Instruction

```
RESULT  =   0x13
WREG    =   0x91
```

After Instruction

```
RESULT  =   0x13
WREG    =   0x93
```

## TBLRD — Table Read

Syntax: [ *label* ] TBLRD ( *; *+; *-; +*)

Operands: None

Operation:
if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT;
TBLPTR - No Change;
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) +1 → TBLPTR;
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) -1 → TBLPTR;
if TBLRD +*,
(TBLPTR) +1 → TBLPTR;
(Prog Mem (TBLPTR)) → TABLAT;

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 10nn<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2 Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The `TBLRD` instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation (Read Program Memory) | No operation | No operation (Write TABLAT) |

## TBLRD — Table Read (cont'd)

Example 1: `TBLRD *+ ;`

Before Instruction

```
TABLAT              =   0x55
TBLPTR              =   0x00A356
MEMORY(0x00A356)    =   0x34
```

After Instruction

```
TABLAT              =   0x34
TBLPTR              =   0x00A357
```

Example 2: `TBLRD +* ;`

Before Instruction

```
TABLAT              =   0xAA
TBLPTR              =   0x01A357
MEMORY(0x01A357)    =   0x12
MEMORY(0x01A358)    =   0x34
```

After Instruction

```
TABLAT              =   0x34
TBLPTR              =   0x01A358
```

## 20.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/
    MPLIB™ Object Librarian
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD for PIC16F87X
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Entry-Level Development Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM 2 Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 17 Demonstration Board
  - KEELOQ® Demonstration Board

### 20.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. The MPLAB IDE is a Windows®-based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or ‘C’)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

### 20.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PIC MCUs.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. The MPASM assembler generates relocatable object files for the MPLINK object linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file that contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

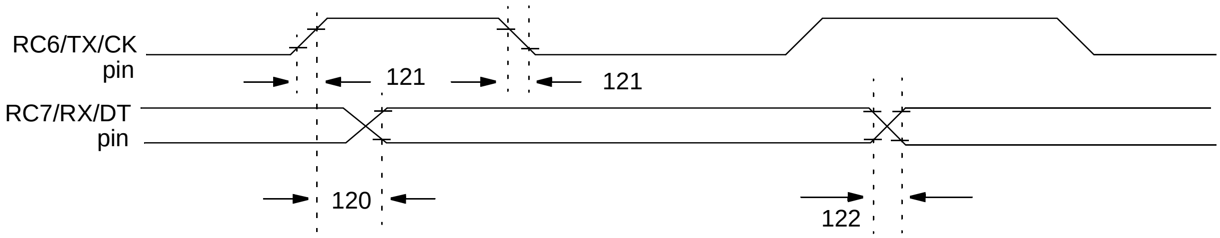
- Integration into MPLAB IDE projects.
- User-defined macros to streamline assembly code.
- Conditional assembly for multi-purpose source files.
- Directives that allow complete control over the assembly process.

### 20.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI ‘C’ compilers for Microchip’s PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

**FIGURE 21-20: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

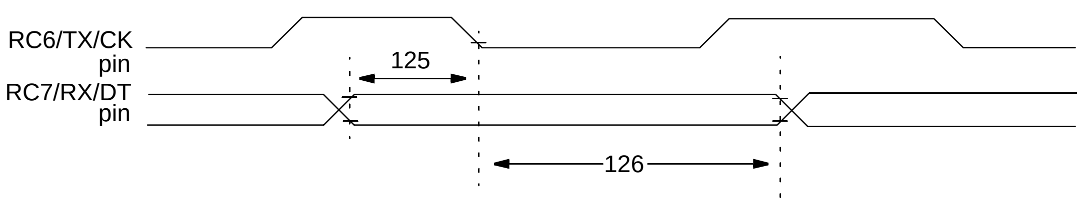



**Note:** Refer to Figure 21-4 for load conditions.

**TABLE 21-19: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 120 | TckH2dtV | SYNC XMIT (MASTER & SLAVE)<br>Clock high to data out valid | PIC18**C**XXX | — | 40 | ns | |
| | | | PIC18**LC**XXX | — | 100 | ns | |
| 121 | Tckrf | Clock out rise time and fall time (Master mode) | PIC18**C**XXX | — | 25 | ns | |
| | | | PIC18**LC**XXX | — | 50 | ns | |
| 122 | Tdtrf | Data out rise time and fall time | PIC18**C**XXX | — | 25 | ns | |
| | | | PIC18**LC**XXX | — | 50 | ns | |

**FIGURE 21-21: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**

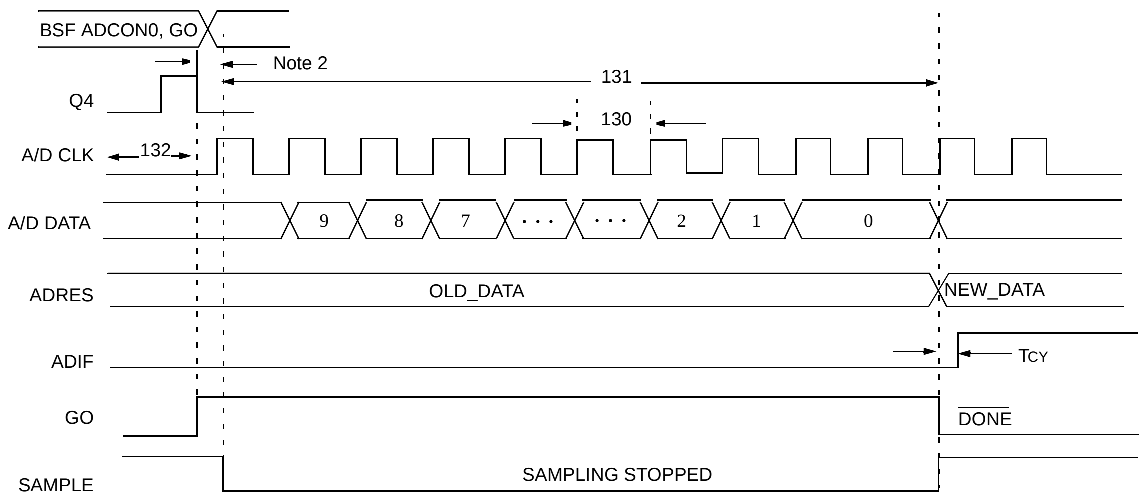



**Note:** Refer to Figure 21-4 for load conditions.

**TABLE 21-20: USART SYNCHRONOUS RECEIVE REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 125 | TdtV2ckl | SYNC RCV (MASTER & SLAVE)<br>Data hold before CK ↓ (DT hold time) | 10 | — | ns | |
| 126 | TckL2dtl | Data hold after CK ↓ (DT hold time) | 15 | — | ns | |

**FIGURE 21-22: A/D CONVERSION TIMING**




**Note 1:** If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

**2:** This is a minimal RC delay (typically 100 ns), which also disconnects the holding capacitor from the analog input.

**TABLE 21-22: A/D CONVERSION REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 130 | TAD | A/D clock period | PIC18**C**XXX | 1.6 | 20(5) | μs | TOSC based, VREF ≥ 3.0V |
| | | | PIC18**LC**XXX | 3.0 | 20(5) | μs | TOSC based, VREF full range |
| | | | PIC18**C**XXX | 2.0 | 6.0 | μs | A/D RC mode |
| | | | PIC18**LC**XXX | 3.0 | 9.0 | μs | A/D RC mode |
| 131 | TCNV | Conversion time (not including acquisition time) **(Note 1)** | | 11 | 12 | TAD | |
| 132 | TACQ | Acquisition time **(Note 3)** | | 15<br>10 | —<br>— | μs<br>μs | -40°C ≤ Temp ≤ 125°C<br>0°C ≤ Temp ≤ 125°C |
| 135 | TSWC | Switching Time from convert → sample | | — | **(Note 4)** | | |
| 136 | TAMP | Amplifier settling time **(Note 2)** | | 1 | — | μs | This may be used if the “new” input voltage has not changed by more than 1 LSb (i.e., 5 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD). |

**Note 1:** ADRES register may be read on the following TCY cycle.

**2:** See Section 16.0 for minimum conditions, when input voltage has changed more than 1 LSb.

**3:** The time for the holding capacitor to acquire the “New” input voltage, when the voltage changes full scale after the conversion (AVDD to AVSS, or AVSS to AVDD). The source impedance (*RS*) on the input channels is 50 Ω.

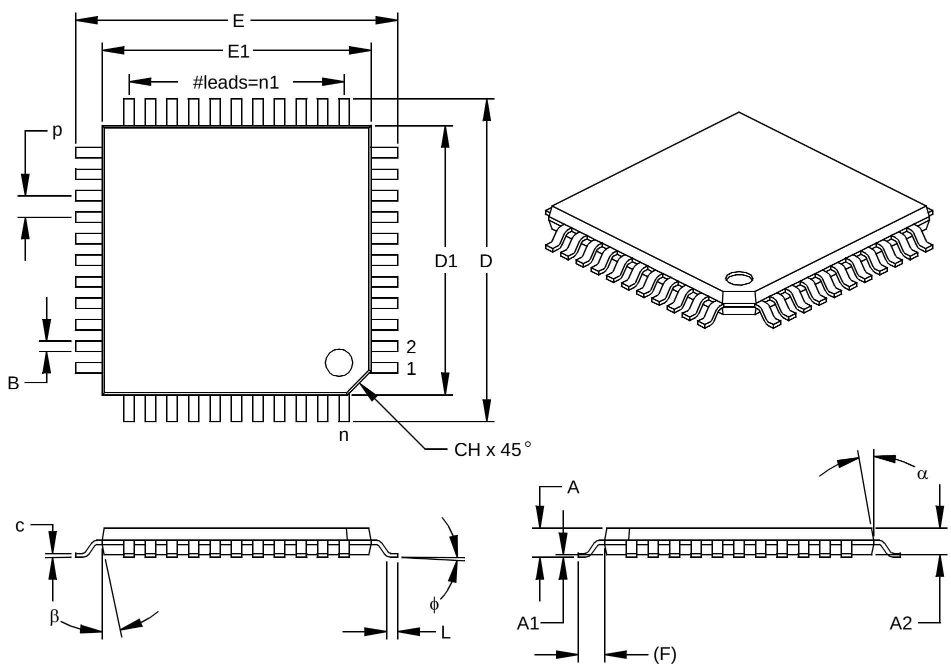
**4:** On the next Q4 cycle of the device clock.

**5:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

### 44-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



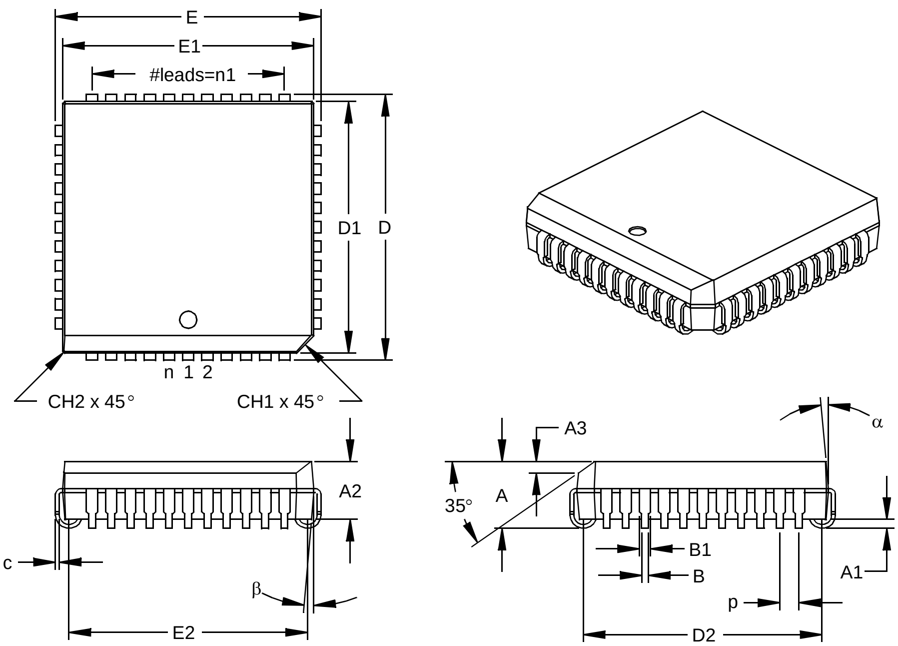

| Units | | INCHES | | | MILLIMETERS* | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 44 | | | 44 | |
| Pitch | p | | .031 | | | 0.80 | |
| Pins per Side | n1 | | 11 | | | 11 | |
| Overall Height | A | .039 | .043 | .047 | 1.00 | 1.10 | 1.20 |
| Molded Package Thickness | A2 | .037 | .039 | .041 | 0.95 | 1.00 | 1.05 |
| Standoff § | A1 | .002 | .004 | .006 | 0.05 | 0.10 | 0.15 |
| Foot Length | L | .018 | .024 | .030 | 0.45 | 0.60 | 0.75 |
| Footprint (Reference) | (F) | | .039 | | 1.00 | | |
| Foot Angle | ϕ | 0 | 3.5 | 7 | 0 | 3.5 | 7 |
| Overall Width | E | .463 | .472 | .482 | 11.75 | 12.00 | 12.25 |
| Overall Length | D | .463 | .472 | .482 | 11.75 | 12.00 | 12.25 |
| Molded Package Width | E1 | .390 | .394 | .398 | 9.90 | 10.00 | 10.10 |
| Molded Package Length | D1 | .390 | .394 | .398 | 9.90 | 10.00 | 10.10 |
| Lead Thickness | c | .004 | .006 | .008 | 0.09 | 0.15 | 0.20 |
| Lead Width | B | .012 | .015 | .017 | 0.30 | 0.38 | 0.44 |
| Pin 1 Corner Chamfer | CH | .025 | .035 | .045 | 0.64 | 0.89 | 1.14 |
| Mold Draft Angle Top | α | 5 | 10 | 15 | 5 | 10 | 15 |
| Mold Draft Angle Bottom | β | 5 | 10 | 15 | 5 | 10 | 15 |

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-076

## 44-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | INCHES* | | | MILLIMETERS | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 44 | | | 44 | |
| Pitch | p | | .050 | | | 1.27 | |
| Pins per Side | n1 | | 11 | | | 11 | |
| Overall Height | A | .165 | .173 | .180 | 4.19 | 4.39 | 4.57 |
| Molded Package Thickness | A2 | .145 | .153 | .160 | 3.68 | 3.87 | 4.06 |
| Standoff § | A1 | .020 | .028 | .035 | 0.51 | 0.71 | 0.89 |
| Side 1 Chamfer Height | A3 | .024 | .029 | .034 | 0.61 | 0.74 | 0.86 |
| Corner Chamfer 1 | CH1 | .040 | .045 | .050 | 1.02 | 1.14 | 1.27 |
| Corner Chamfer (others) | CH2 | .000 | .005 | .010 | 0.00 | 0.13 | 0.25 |
| Overall Width | E | .685 | .690 | .695 | 17.40 | 17.53 | 17.65 |
| Overall Length | D | .685 | .690 | .695 | 17.40 | 17.53 | 17.65 |
| Molded Package Width | E1 | .650 | .653 | .656 | 16.51 | 16.59 | 16.66 |
| Molded Package Length | D1 | .650 | .653 | .656 | 16.51 | 16.59 | 16.66 |
| Footprint Width | E2 | .590 | .620 | .630 | 14.99 | 15.75 | 16.00 |
| Footprint Length | D2 | .590 | .620 | .630 | 14.99 | 15.75 | 16.00 |
| Lead Thickness | c | .008 | .011 | .013 | 0.20 | 0.27 | 0.33 |
| Upper Lead Width | B1 | .026 | .029 | .032 | 0.66 | 0.74 | 0.81 |
| Lower Lead Width | B | .013 | .020 | .021 | 0.33 | 0.51 | 0.53 |
| Mold Draft Angle Top | α | 0 | 5 | 10 | 0 | 5 | 10 |
| Mold Draft Angle Bottom | β | 0 | 5 | 10 | 0 | 5 | 10 |

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-048