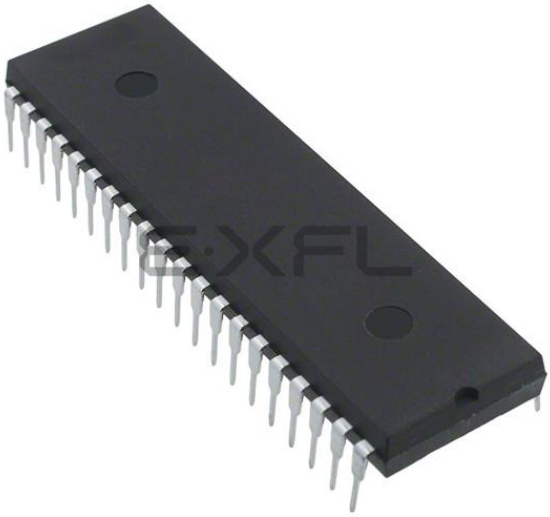


Welcome to E-XFL.COM

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	48MHz
Connectivity	UART/USART, USB
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	34
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 13x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f4450-i-p

TABLE 4-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
UEP9	2450	4450	---0 0000	---0 0000	---u uuuu
UEP8	2450	4450	---0 0000	---0 0000	---u uuuu
UEP7	2450	4450	---0 0000	---0 0000	---u uuuu
UEP6	2450	4450	---0 0000	---0 0000	---u uuuu
UEP5	2450	4450	---0 0000	---0 0000	---u uuuu
UEP4	2450	4450	---0 0000	---0 0000	---u uuuu
UEP3	2450	4450	---0 0000	---0 0000	---u uuuu
UEP2	2450	4450	---0 0000	---0 0000	---u uuuu
UEP1	2450	4450	---0 0000	---0 0000	---u uuuu
UEP0	2450	4450	---0 0000	---0 0000	---u uuuu
UCFG	2450	4450	00-0 0000	00-0 0000	uu-u uuuu
UADDR	2450	4450	-000 0000	-000 0000	-uuu uuuu
UCON	2450	4450	-0x0 000-	-0x0 000-	-uuu uuu-
USTAT	2450	4450	-xxx xxx-	-xxx xxx-	-uuu uuu-
UEIE	2450	4450	0--0 0000	0--0 0000	u--u uuuu
UEIR	2450	4450	0--0 0000	0--0 0000	u--u uuuu
UIE	2450	4450	-000 0000	-000 0000	-uuu uuuu
UIR	2450	4450	-000 0000	-000 0000	-uuu uuuu
UFRMH	2450	4450	---- -xxx	---- -xxx	---- -uuu
UFRML	2450	4450	xxxx xxxx	xxxx xxxx	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: 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.

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

3: 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).

4: See Table 4-3 for Reset value for specific condition.

5: PORTA<6>, LATA<6> and TRISA<6> are enabled depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

5.1.2.4 Stack Full and Underflow Resets

Device Resets on stack overflow and stack underflow conditions are enabled by setting the STVREN bit in Configuration Register 4L. When STVREN is set, a full or underflow condition will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. When STVREN is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit but not cause a device Reset. The STKFUL or STKUNF bits are cleared by user software or a Power-on Reset.

5.1.3 FAST REGISTER STACK

A Fast Register Stack is provided for the STATUS, WREG and BSR registers to provide a “fast return” option for interrupts. Each stack is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers if the `RETFIE, FAST` instruction is used to return from the interrupt.

If both low and high-priority interrupts are enabled, the stack registers cannot be used reliably to return from 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. In these cases, users must save the key registers in software during a low-priority interrupt.

If interrupt priority is not used, all interrupts may use the Fast Register Stack for returns from 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 `CALL label, FAST` instruction must be executed to save the STATUS, WREG and BSR registers to the Fast Register Stack. A `RETURN, FAST` instruction is then executed to restore these registers from the Fast Register Stack.

Example 5-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

EXAMPLE 5-1: 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
```

5.1.4 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed `GOTO`
- Table Reads

5.1.4.1 Computed `GOTO`

A computed `GOTO` is accomplished by adding an offset to the program counter. An example is shown in Example 5-2.

A look-up table can be formed with an `ADDWF PCL` instruction and a group of `RETLW nn` instructions. The W register 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 nn` instructions that returns the value ‘`nn`’ to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of 2 (LSb = `0`).

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

EXAMPLE 5-2: COMPUTED `GOTO` USING AN OFFSET VALUE

```
        MOVF   OFFSET, W
        CALL   TABLE
ORG     nn00h
TABLE   ADDWF  PCL
        RETLW  nnh
        RETLW  nnh
        RETLW  nnh
        .
        .
        .
```

5.1.4.2 Table Reads and Table Writes

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

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory. Data is transferred to or from program memory one byte at a time.

Table read and table write operations are discussed further in **Section 6.1 “Table Reads and Table Writes”**.

TABLE 5-2: REGISTER FILE SUMMARY (PIC18F2450/4450) (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on Page:
OSCCON	IDLEN	—	—	—	OSTS	—	SCS1	SCS0	0--- q-00	50, 31
HLVDCON	VDIRMAG	—	IRVST	HLVDEN	HLVDL3	HLVDL2	HLVDL1	HLVDL0	0-00 0101	50, 185
WDTCON	—	—	—	—	—	—	—	SWDTEN	--- ---0	50, 204
RCON	IPEN	SBOREN(2)	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	0q-1 11q0	50, 42
TMR1H	Timer1 Register High Byte								xxxx xxxx	50, 120
TMR1L	Timer1 Register Low Byte								xxxx xxxx	50, 120
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	0000 0000	50, 115
TMR2	Timer2 Register								0000 0000	50, 122
PR2	Timer2 Period Register								1111 1111	50, 122
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	50, 121
ADRESH	A/D Result Register High Byte								xxxx xxxx	50, 184
ADRESL	A/D Result Register Low Byte								xxxx xxxx	50, 184
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON	--00 0000	50, 175
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 qqqq	50, 176
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	50, 177
CCPR1H	Capture/Compare/PWM Register 1 High Byte								xxxx xxxx	50, 124
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx	50, 124
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	50, 123,
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	01-0 0-00	51, 156,
SPBRGH	EUSART Baud Rate Generator Register High Byte								0000 0000	50, 157
SPBRG	EUSART Baud Rate Generator Register Low Byte								0000 0000	50, 157
RCREG	EUSART Receive Register								0000 0000	50, 165
TXREG	EUSART Transmit Register								0000 0000	51, 163
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	51, 154
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	51, 155
EECON2	Data Memory Control Register 2 (not a physical register)								0000 0000	51, 74
EECON1	—	CFGS	—	FREE	WRERR	WREN	WR	—	-x-0 x00-	51, 75
IPR2	OSCFIP	—	USBIP	—	—	HLVDIP	—	—	1-1- -1--	51, 95
PIR2	OSCFIF	—	USBIF	—	—	HLVDIF	—	—	0-0- -0--	51, 91
PIE2	OSCFIE	—	USBIE	—	—	HLVDIE	—	—	0-0- -0--	51, 93
IPR1	—	ADIP	RCIP	TXIP	—	CCP1IP	TMR2IP	TMR1IP	-111 -111	51, 94
PIR1	—	ADIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	-000 -000	51, 90
PIE1	—	ADIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE	-000 -000	51, 92
TRISE(3)	—	—	—	—	—	TRISE2	TRISE1	TRISE0	---- -111	51, 110
TRISD(3)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	1111 1111	51, 108
TRISC	TRISC7	TRISC6	—	—	—	TRISC2	TRISC1	TRISC0	11-- -111	51, 106
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	51, 103
TRISA	—	TRISA6(4)	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	-111 1111	51, 100
LATE(3)	—	—	—	—	—	LATE2	LATE1	LATE0	---- -xxx	51, 110
LATD(3)	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	xxxx xxxx	51, 108
LATC	LATC7	LATC6	—	—	—	LATC2	LATC1	LATC0	xx-- -xxx	51, 106
LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	xxxx xxxx	51, 103
LATA	—	LATA6(4)	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	-xxx xxxx	51, 100
PORTE	—	—	—	—	RE3(5)	RE2(3)	RE1(3)	RE0(3)	---- x000	51, 109
PORTD(3)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	51, 108

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

Note 1: Bit 21 of the TBLPTRU allows access to the device Configuration bits.

2: The SBOREN bit is only available when BOREN<1:0> = 01; otherwise, the bit reads as '0'.

3: These registers and/or bits are not implemented on 28-pin devices and are read as '0'. Reset values are shown for 40/44-pin devices; individual unimplemented bits should be interpreted as '-'.

4: RA6 is configured as a port pin based on various primary oscillator modes. When the port pin is disabled, all of the associated bits read '0'.

5: RE3 is only available as a port pin when the MCLRE Configuration bit is clear; otherwise, the bit reads as '0'.

6: RC5 and RC4 are only available as port pins when the USB module is disabled (UCON<3> = 0).

8.3 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Request (Flag) registers (PIR1 and PIR2).

Note 1: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE (INTCON<7>).

2: User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

REGISTER 8-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

U-0	R/W-0	R-0	R-0	U-0	R/W-0	R/W-0	R/W-0
—	ADIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

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

bit 7 **Unimplemented:** Read as '0'

bit 6 **ADIF:** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF:** EUSART Receive Interrupt Flag bit
1 = The EUSART receive buffer, RCREG, is full (cleared when RCREG is read)
0 = The EUSART receive buffer is empty

bit 4 **TXIF:** EUSART Transmit Interrupt Flag bit
1 = The EUSART transmit buffer, TXREG, is empty (cleared when TXREG is written)
0 = The EUSART transmit buffer is full

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

bit 2 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

10.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt on overflow

The T0CON register (Register 10-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 10-1. Figure 10-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

REGISTER 10-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

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

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-Bit/16-Bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is not assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

11.7 Considerations in Asynchronous Counter Mode

Following a Timer1 interrupt and an update to the TMR1 registers, the Timer1 module uses a falling edge on its clock source to trigger the next register update on the rising edge. If the update is completed after the clock input has fallen, the next rising edge will not be counted.

If the application can reliably update TMR1 before the timer input goes low, no additional action is needed. Otherwise, an adjusted update can be performed following a later Timer1 increment. This can be done by monitoring TMR1L within the interrupt routine until it increments, and then updating the TMR1H:TMR1L register pair while the clock is low, or one-half of the period of the clock source. Assuming that Timer1 is being used as a Real-Time Clock, the clock source is a 32.768 kHz crystal oscillator. In this case, one-half period of the clock is 15.25 μs.

The Real-Time Clock application code in Example 11-1 shows a typical ISR for Timer1, as well as the optional code required if the update cannot be done reliably within the required interval.

EXAMPLE 11-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE

```
RTCinit
            MOVLW     80h             ; Preload TMR1 register pair
            MOVWF     TMR1H           ; for 1 second overflow
            CLRF      TMR1L
            MOVLW     b'00001111'     ; Configure for external clock,
            MOVWF     T1CON           ; Asynchronous operation, external oscillator
            CLRF      secs            ; Initialize timekeeping registers
            CLRF      mins            ;
            MOVLW     .12
            MOVWF     hours
            BSF       PIE1, TMR1IE    ; Enable Timer1 interrupt
            RETURN
RTCisr
                                      ; Insert the next 4 lines of code when TMR1
                                      ; cannot be reliably updated before clock pulse goes low
            BTFSC     TMR1L,0         ; wait for TMR1L to become clear
            BRA       $-2             ; (may already be clear)
            BTFSS     TMR1L,0         ; wait for TMR1L to become set
            BRA       $-2             ; TMR1 has just incremented
                                      ; If TMR1 update can be completed before clock pulse goes low
                                      ; Start ISR here
            BSF       TMR1H, 7        ; Preload for 1 sec overflow
            BCF       PIR1, TMR1IF    ; Clear interrupt flag
            INCF      secs, F         ; Increment seconds
            MOVLW     .59             ; 60 seconds elapsed?
            CPFSGT    secs
            RETURN                    ; No, done
            CLRF      secs            ; Clear seconds
            INCF      mins, F         ; Increment minutes
            MOVLW     .59             ; 60 minutes elapsed?
            CPFSGT    mins
            RETURN                    ; No, done
            CLRF      mins            ; clear minutes
            INCF      hours, F        ; Increment hours
            MOVLW     .23             ; 24 hours elapsed?
            CPFSGT    hours
            RETURN                    ; No, done
            CLRF      hours           ; Reset hours
            RETURN                    ; Done
```

13.1 CCP Module Configuration

The Capture/Compare/PWM module is associated with a control register (generically, CCP1CON) and a data register (CCPR1). The data register, in turn, is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). All registers are both readable and writable.

13.1.1 CCP MODULE AND TIMER RESOURCES

The CCP module utilizes Timer1 or Timer2, depending on the mode selected. Timer1 is available to the module in Capture or Compare modes, while Timer2 is available for modules in PWM mode.

TABLE 13-1: CCP MODE – TIMER RESOURCE

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

In Timer1 in Asynchronous Counter mode, the capture operation will not work.

13.2 Capture Mode

In Capture mode, the CCPR1H:CCPR1L register pair captures the 16-bit value of the TMR1 register when an event occurs on the corresponding CCP1 pin. An event is defined as one of the following:

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

The event is selected by the mode select bits, CCP1M3:CCP1M0 (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit, CCP1IF, is set; it must be cleared in software. If another capture occurs before the value in register CCPR1 is read, the old captured value is overwritten by the new captured value.

13.2.1 CCP1 PIN CONFIGURATION

In Capture mode, the CCP1 pin should be configured as an input by setting the corresponding TRIS direction bit.

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

13.2.2 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCP1IE interrupt enable bit clear to avoid false interrupts. The interrupt flag bit, CCP1IF, should also be cleared following any such change in operating mode.

13.2.3 CCP PRESCALER

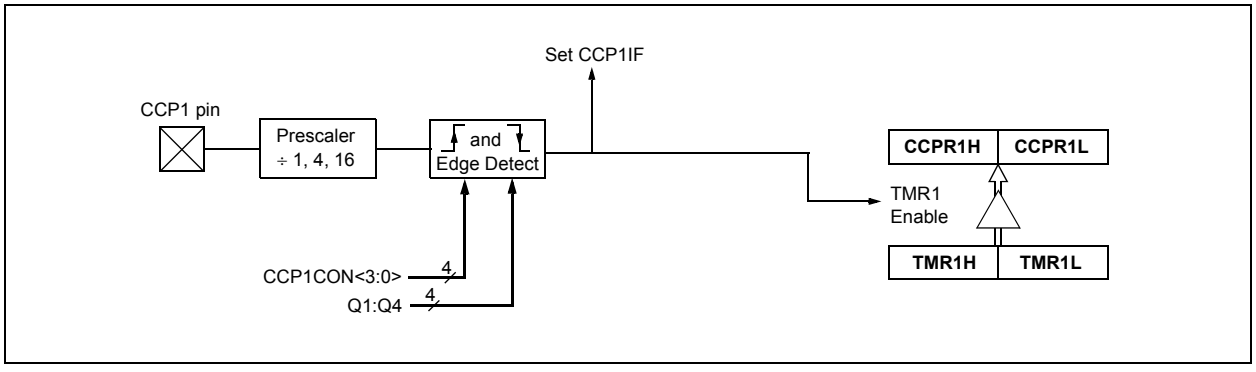
There are four prescaler settings in Capture mode. They are specified as part of the operating mode selected by the mode select bits (CCP1M3:CCP1M0). Whenever the CCP module is turned off or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler 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 (CCP1 SHOWN)

```
CLRF   CCP1CON          ; 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



15.1.3 AUTO-BAUD RATE DETECT

The Enhanced USART module supports the automatic detection and calibration of baud rate. This feature is active only in Asynchronous mode and while the WUE bit is clear.

The automatic baud rate measurement sequence (Figure 15-1) begins whenever a Start bit is received and the ABDEN bit is set. The calculation is self-averaging.

In the Auto-Baud Rate Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. In ABD mode, the internal Baud Rate Generator is used as a counter to time the bit period of the incoming serial byte stream.

Once the ABDEN bit is set, the state machine will clear the BRG and look for a Start bit. The Auto-Baud Rate Detection must receive a byte with the value 55h (ASCII “U”, which is also the LIN bus Sync character) in order to calculate the proper bit rate. The measurement is taken over both a low and a high bit time in order to minimize any effects caused by asymmetry of the incoming signal. After a Start bit, the SPBRG begins counting up, using the preselected clock source on the first rising edge of RX. After eight bits on the RX pin, or the fifth rising edge, an accumulated value totalling the proper BRG period is left in the SPBRGH:SPBRG register pair. Once the 5th edge is seen (this should correspond to the Stop bit), the ABDEN bit is automatically cleared.

If a rollover of the BRG occurs (an overflow from FFFFh to 0000h), the event is trapped by the ABDOVF status bit (BAUDCON<7>). It is set in hardware by BRG rollovers and can be set or cleared by the user in software. ABD mode remains active after rollover events and the ABDEN bit remains set (Figure 15-2).

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. Note that the BRG clock will be configured by the BRG16 and BRGH bits. Independent of the BRG16 bit setting, both the SPBRG and SPBRGH will be used as a 16-bit counter. This allows the user to verify that no carry occurred for 8-bit modes by checking for 00h in the SPBRGH register. Refer to Table 15-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the EUSART state machine is held in Idle. The RCIF interrupt is set once the fifth rising edge on RX is detected. The value in the RCREG needs to be read to clear the RCIF interrupt. The contents of RCREG should be discarded.

Note 1: If the WUE bit is set with the ABDEN bit, Auto-Baud Rate Detection will occur on the byte *following* the Break character.

2: It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible due to bit error rates. Overall system timing and communication baud rates must be taken into consideration when using the Auto-Baud Rate Detection feature.

TABLE 15-4: BRG COUNTER CLOCK RATES

BRG16	BRGH	BRG Counter Clock
0	0	FOSC/512
0	1	FOSC/128
1	0	FOSC/128
1	1	FOSC/32

Note: During the ABD sequence, SPBRG and SPBRGH are both used as a 16-bit counter, independent of the BRG16 setting.

15.1.3.1 ABD and EUSART Transmission

Since the BRG clock is reversed during ABD acquisition, the EUSART transmitter cannot be used during ABD. This means that whenever the ABDEN bit is set, TXREG cannot be written to. Users should also ensure that ABDEN does not become set during a transmit sequence. Failing to do this may result in unpredictable EUSART operation.

15.2 EUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTA<4>). In this mode, the EUSART uses the standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine data bits and one Stop bit). The most common data format is eight bits. An on-chip dedicated 8-bit/16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent but use the same data format and baud rate. The Baud Rate Generator produces a clock, either x16 or x64 of the bit shift rate depending on the BRGH and BRG16 bits (TXSTA<2> and BAUDCON<3>). Parity is not supported by the hardware but can be implemented in software and stored as the ninth data bit.

When operating in Asynchronous mode, the EUSART module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver
- Auto-Wake-up on Sync Break Character
- 12-Bit Break Character Transmit
- Auto-Baud Rate Detection

15.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 15-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREG register (if available).

Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and the TXIF flag bit (PIR1<4>) is set. This interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXIE (PIE1<4>). TXIF will be set regardless of the state of TXIE; it cannot be cleared in software. TXIF is also not cleared immediately upon loading TXREG but becomes valid in the second instruction cycle following the load instruction. Polling TXIF immediately following a load of TXREG will return invalid results.

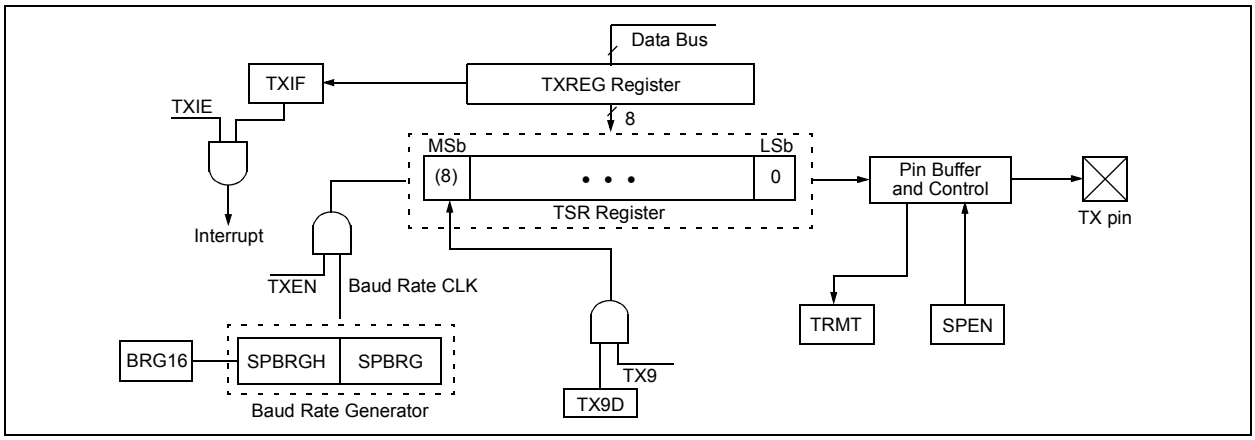
While TXIF indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty.

Note 1:	The TSR register is not mapped in data memory so it is not available to the user.
2:	Flag bit, TXIF, is set when enable bit, TXEN, is set.

To set up an Asynchronous Transmission:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, TXIE.
4. If 9-bit transmission is desired, set transmit bit, TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit, TXEN, which will also set bit, TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Load data to the TXREG register (starts transmission).
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 15-3: EUSART TRANSMIT BLOCK DIAGRAM



17.6 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the HLVDIF bit will be set and the device will wake-up from Sleep. Device execution will continue from the interrupt vector address if interrupts have been globally enabled.

17.7 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

TABLE 17-1: REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
HLVDCON	VDIRMAG	—	IRVST	HLVDEN	HLVDL3	HLVDL2	HLVDL1	HLVDL0	50
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR2	OSCFIF	—	USBIF	—	—	HLVDIF	—	—	51
PIE2	OSCFIE	—	USBIE	—	—	HLVDIE	—	—	51
IPR2	OSCFIP	—	USBIP	—	—	HLVDIP	—	—	51

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the HLVD module.

REGISTER 18-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

R/P-0	R/P-0	U-0	U-0	R/P-0	R/P-1	R/P-1	R/P-1
IESO	FCMEN	—	—	FOSC3(1)	FOSC2(1)	FOSC1(1)	FOSC0(1)
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed		u = Unchanged from programmed state

bit 7 **IESO:** Internal/External Oscillator Switchover bit
1 = Oscillator Switchover mode enabled
0 = Oscillator Switchover mode disabled

bit 6 **FCMEN:** Fail-Safe Clock Monitor Enable bit
1 = Fail-Safe Clock Monitor enabled
0 = Fail-Safe Clock Monitor disabled

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **FOSC3:FOSC0:** Oscillator Selection bits(1)
111x = HS oscillator, PLL enabled (HSPLL)
110x = HS oscillator (HS)
1011 = Internal oscillator, HS oscillator used by USB (INTHS)
1010 = Internal oscillator, XT used by USB (INTXT)
1001 = Internal oscillator, CLKO function on RA6, EC used by USB (INTCKO)
1000 = Internal oscillator, port function on RA6, EC used by USB (INTIO)
0111 = EC oscillator, PLL enabled, CLKO function on RA6 (ECPLL)
0110 = EC oscillator, PLL enabled, port function on RA6 (ECPIO)
0101 = EC oscillator, CLKO function on RA6 (EC)
0100 = EC oscillator, port function on RA6 (ECIO)
001x = XT oscillator, PLL enabled (XTPLL)
000x = XT oscillator (XT)

Note 1: The microcontroller and USB module both use the selected oscillator as their clock source in XT, HS and EC modes. The USB module uses the indicated XT, HS or EC oscillator as its clock source whenever the microcontroller uses the internal oscillator.

TABLE 19-1: OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7).
BSR	Bank Select Register. Used to select the current RAM bank.
C, DC, Z, OV, N	ALU Status bits: **C**arry, **D**igit **C**arry, **Z**ero, **Ov**erflow, **N**egative.
d	Destination select bit d = 0: store result in WREG d = 1: store result in file register f
dest	Destination: either the WREG register or the specified register file location.
f	8-bit register file address (00h to FFh) or 2-bit FSR designator (0h to 3h).
f_s	12-bit register file address (000h to FFFh). This is the source address.
f_d	12-bit register file address (000h to FFFh). This is the destination address.
GIE	Global Interrupt Enable bit.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
label	Label name.
mm * *+ *- +*	The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions: No change to register (such as TBLPTR with table reads and writes) Post-Increment register (such as TBLPTR with table reads and writes) Post-Decrement register (such as TBLPTR with table reads and writes) Pre-Increment register (such as TBLPTR with table reads and writes)
n	The relative address (2's complement number) for relative branch instructions or the direct address for Call/Branch and Return instructions.
PC	Program Counter.
PCL	Program Counter Low Byte.
PCH	Program Counter High Byte.
PCLATH	Program Counter High Byte Latch.
PCLATU	Program Counter Upper Byte Latch.
$\overline{PD}$	Power-Down bit.
PRODH	Product of Multiply High Byte.
PRODL	Product of Multiply Low Byte.
s	Fast Call/Return mode select bit s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
TBLPTR	21-bit Table Pointer (points to a program memory location).
TABLAT	8-bit Table Latch.
$\overline{TO}$	Time-out bit.
TOS	Top-of-Stack.
u	Unused or unchanged.
WDT	Watchdog Timer.
WREG	Working register (accumulator).
x	Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
z_s	7-bit offset value for indirect addressing of register files (source).
z_d	7-bit offset value for indirect addressing of register files (destination).
{ }	Optional argument.
[text]	Indicates an indexed address.
(text)	The contents of text.
[expr]<n>	Specifies bit n of the register indicated by the pointer, expr.
→	Assigned to.
< >	Register bit field.
∈	In the set of.
italics	User-defined term (font is Courier New).

ADDWFC **ADD W and Carry bit to f**

Syntax: ADDWFC f {,d {,a}}

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

Operation: (W) + (f) + (C) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

0010	00da	ffff	ffff

Description: Add W, the Carry flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 19.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDWFC REG, 0, 1`

Before Instruction
Carry bit = 1
REG = 02h
W = 4Dh
After Instruction
Carry bit = 0
REG = 02h
W = 50h

ANDLW **AND Literal with W**

Syntax: ANDLW k

Operands: $0 \le k \le 255$

Operation: (W) .AND. k → W

Status Affected: N, Z

Encoding:

0000	1011	kkkk	kkkk

Description: The contents of W are ANDed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `ANDLW 05Fh`

Before Instruction
W = A3h
After Instruction
W = 03h

BCF **Bit Clear f**

Syntax: BCF f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $0 \rightarrow f<b>$

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 19.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

BN **Branch if Negative**

Syntax: BN n

Operands: $-128 \le n \le 127$

Operation: if Negative bit is '1',
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0110	nnnn	nnnn

Description: If the Negative bit is '1', then the program will branch.
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 BN Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
If Negative = 1;
PC = address `(Jump)`
If Negative = 0;
PC = address `(HERE + 2)`

19.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18F2450/4450 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for Indexed Addressing. Two of the instructions, `ADDFSR` and `SUBFSR`, each have an additional special instantiation for using FSR2. These versions (`ADDULNK` and `SUBULNK`) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- Dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- Function Pointer invocation
- Software Stack Pointer manipulation
- Manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 19-3. Detailed descriptions are provided in **Section 19.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 19-1 (page 214) apply to both the standard and extended PIC18 instruction sets.

Note:	The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

19.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of Indexed Addressing, it is enclosed in square brackets ("[ ]"). This is done to indicate that the argument is used as an index or offset. The MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 19.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**.

Note:	In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{ }").

TABLE 19-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected
				MSb			LSb	
ADDFSR	f, k	Add Literal to FSR	1	1110	1000	ffkk	kkkk	None
ADDULNK	k	Add Literal to FSR2 and Return	2	1110	1000	11kk	kkkk	None
CALLW		Call Subroutine using WREG	2	0000	0000	0001	0100	None
MOVSF	z_s, f_d	Move z_s (source) to 1st word f_d (destination) 2nd word	2	1110 1111	1011 ffff	0zzz ffff	zzzz ffff	None
MOVSS	z_s, z_d	Move z_s (source) to 1st word z_d (destination) 2nd word	2	1110 1111	1011 xxxx	1zzz xzzz	zzzz zzzz	None
PUSHL	k	Store Literal at FSR2, Decrement FSR2	1	1110	1010	kkkk	kkkk	None
SUBFSR	f, k	Subtract Literal from FSR	1	1110	1001	ffkk	kkkk	None
SUBULNK	k	Subtract Literal from FSR2 and Return	2	1110	1001	11kk	kkkk	None

21.2 DC Characteristics: Power-Down and Supply Current PIC18F2450/4450 (Industrial) PIC18LF2450/4450 (Industrial) (Continued)

PIC18LF2450/4450 (Industrial)		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18F2450/4450 (Industrial)		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
D022 (ΔIWDT)	**Module Differential Currents (ΔIWDT, ΔIBOR, ΔILVD, ΔIOSCB, ΔIAD)**						
	Watchdog Timer	1.3	3.8	μA	-40°C	VDD = 2.0V	
		1.5	3.8	μA	+25°C		
		2.3	3.8	μA	+85°C		
		1.8	4.6	μA	-40°C	VDD = 3.0V	
		2.0	4.6	μA	+25°C		
		3.0	4.6	μA	+85°C		
		3.3	10	μA	-40°C	VDD = 5.0V	
		3.6	10	μA	+25°C		
		3.9	10	μA	+85°C		
D022A (ΔIBOR)	**Brown-out Reset(4)**	40	52	μA	-40°C to +85°C	VDD = 3.0V	
		45	63	μA	-40°C to +85°C	VDD = 5.0V	
		0	2	μA	-40°C to +85°C		Sleep mode, BOREN1:BOREN0 = 10
D022B (ΔILVD)	**High/Low-Voltage Detect(4)**	22	47	μA	-40°C to +85°C	VDD = 2.0V	
		25	58	μA	-40°C to +85°C	VDD = 3.0V	
		29	69	μA	-40°C to +85°C	VDD = 5.0V	
D025 (ΔIOSCB)	**Timer1 Oscillator**	1.5	4.5	μA	-40°C	VDD = 2.0V	32 kHz on Timer1(3)
		1.2	4.5	μA	+25°C		
		1.6	4.5	μA	+85°C		
		1.7	6.0	μA	-40°C	VDD = 3.0V	32 kHz on Timer1(3)
		1.8	6.0	μA	+25°C		
		2.0	6.0	μA	+85°C		
		1.4	8.0	μA	-40°C	VDD = 5.0V	32 kHz on Timer1(3)
		1.5	8.0	μA	+25°C		
		1.9	8.0	μA	+85°C		
D026 (ΔIAD)	**A/D Converter**	0.2	2.0	μA	-40°C to +85°C	VDD = 2.0V	A/D on, not converting
		0.2	2.0	μA	-40°C to +85°C	VDD = 3.0V	
		0.2	2.0	μA	-40°C to +85°C	VDD = 5.0V	

Legend: Shading of rows is to assist in readability of the table.

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD or VSS;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: Standard low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

4: BOR and HLVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

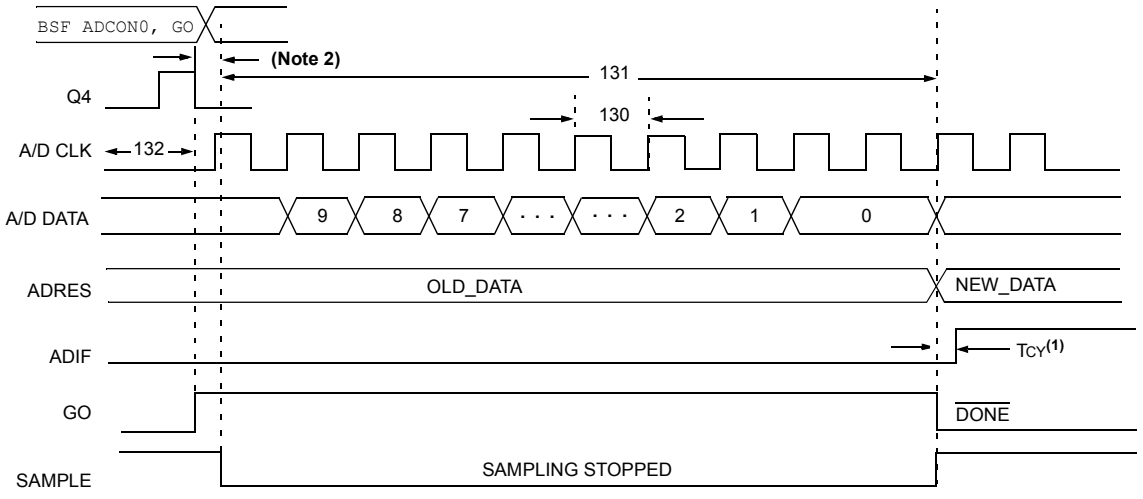
TABLE 21-17: A/D CONVERTER CHARACTERISTICS: PIC18F2450/4450 (INDUSTRIAL) PIC18LF2450/4450 (INDUSTRIAL)

Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
A01	NR	Resolution	—	—	10	bit	ΔVREF ≥ 3.0V
A03	EIL	Integral Linearity Error	—	—	<±1	LSb	ΔVREF ≥ 3.0V
A04	EDL	Differential Linearity Error	—	—	<±1	LSb	ΔVREF ≥ 3.0V
A06	EOFF	Offset Error	—	—	<±2	LSb	ΔVREF ≥ 3.0V
A07	EGN	Gain Error	—	—	<±1	LSb	ΔVREF ≥ 3.0V
A10	—	Monotonicity	Guaranteed(1)			—	VSS ≤ VAIN ≤ VREF
A20	ΔVREF	Reference Voltage Range (VREFH – VREFL)	1.8 3	— —	— —	V V	VDD < 3.0V VDD ≥ 3.0V
A21	VREFH	Reference Voltage High	VSS	—	VREFH	V	
A22	VREFL	Reference Voltage Low	VSS – 0.3V	—	VDD – 3.0V	V	
A25	VAIN	Analog Input Voltage	VREFL	—	VREFH	V	
A30	ZAIN	Recommended Impedance of Analog Voltage Source	—	—	2.5	kΩ	
A50	IREF	VREF Input Current(2)	— —	— —	5 150	μA μA	During VAIN acquisition. During A/D conversion cycle.

Note 1: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

2: VREFH current is from RA3/AN3/VREF+ pin or VDD, whichever is selected as the VREFH source. VREFL current is from RA2/AN2/VREF- pin or VSS, whichever is selected as the VREFL source.

FIGURE 21-14: A/D CONVERSION TIMING

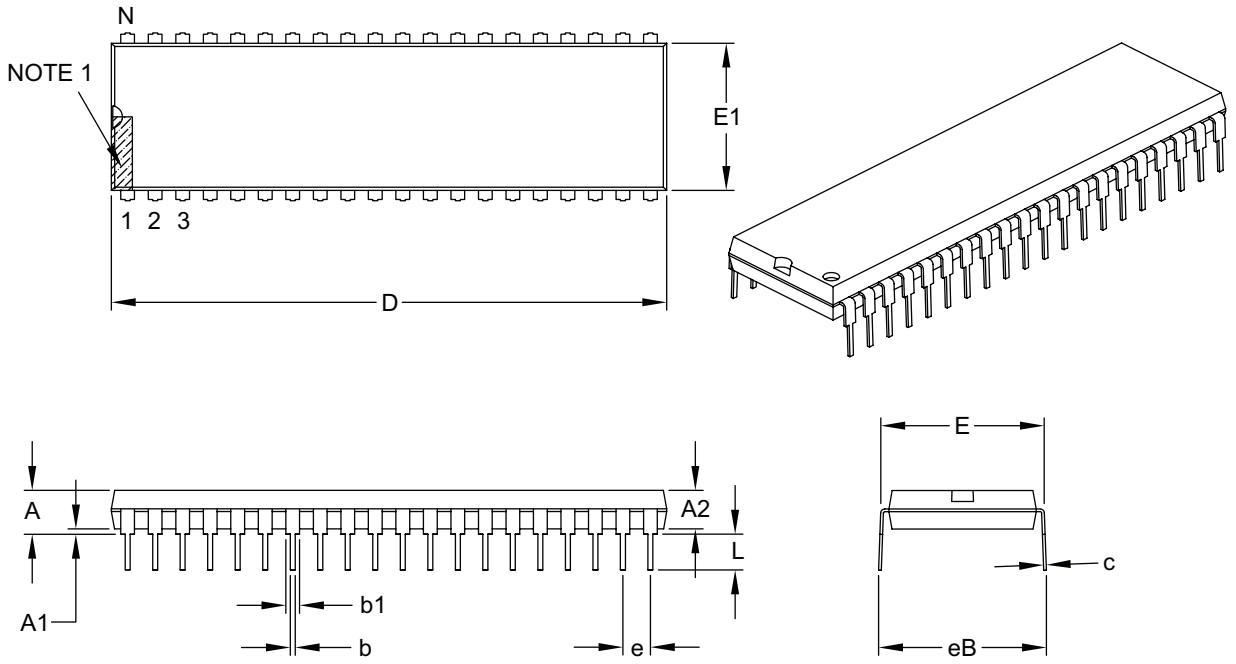


Note 1: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

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

40-Lead Plastic Dual In-Line (P) – 600 mil Body [PDIP]

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



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		40	
Pitch	e		.100 BSC	
Top to Seating Plane	A	–	–	.250
Molded Package Thickness	A2	.125	–	.195
Base to Seating Plane	A1	.015	–	–
Shoulder to Shoulder Width	E	.590	–	.625
Molded Package Width	E1	.485	–	.580
Overall Length	D	1.980	–	2.095
Tip to Seating Plane	L	.115	–	.200
Lead Thickness	c	.008	–	.015
Upper Lead Width	b1	.030	–	.070
Lower Lead Width	b	.014	–	.023
Overall Row Spacing §	eB	–	–	.700

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-016B

Q

R

S

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

To: Technical Publications Manager Total Pages Sent _____

RE: Reader Response

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

Telephone: (_____) _____ - _____ FAX: (_____) _____ - _____

Application (optional):

Would you like a reply? ___Y ___N

Device: PIC18F2450/4450 Literature Number: DS39760D

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this document easy to follow? If not, why?

4. What additions to the document do you think would enhance the structure and subject?

5. What deletions from the document could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?