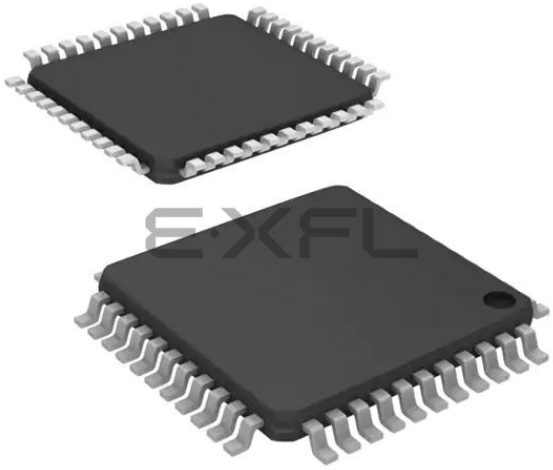


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, Power Control PWM, QEI, POR, PWM, WDT
Number of I/O	36
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 9x10b
Oscillator Type	Internal
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/pic18lf4331t-i-pt

6.7 Data Addressing Modes

The data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

6.7.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all. They either perform an operation that globally affects the device or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include SLEEP, RESET and DAW.

Other instructions work in a similar way but require an additional explicit argument in the opcode. This is known as Literal Addressing mode because they require some literal value as an argument. Examples include ADDLW and MOVLW, which respectively, add or move a literal value to the W register. Other examples include CALL and GOTO, which include a 20-bit program memory address.

6.7.2 DIRECT ADDRESSING

Direct Addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 6.5.4 “Special Function Registers”**) or a location in the Access Bank (**Section 6.5.2 “Access Bank”**) as the data source for the instruction.

The Access RAM bit, ‘a’, determines how the address is interpreted. When ‘a’ is ‘1’, the contents of the BSR (**Section 6.5.1 “Bank Select Register (BSR)”**) are used with the address to determine the complete 12-bit address of the register. When ‘a’ is ‘0’, the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as MOVFF, include the entire 12-bit address (either source or destination) in their op codes. In these cases, the BSR is ignored entirely.

The destination of the operation’s results is determined by the destination bit, ‘d’. When ‘d’ is ‘1’, the results are stored back in the source register, overwriting its original contents. When ‘d’ is ‘0’, the results are stored in the W register. Instructions without the ‘d’ argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

6.7.3 INDIRECT ADDRESSING

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations to be read or written to. Since the FSRs are themselves located in RAM as Special Function Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for Indirect Addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in Example 6-5.

EXAMPLE 6-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR   FSR0, 100h ;
NEXT    CLRF   POSTINC0   ; Clear INDF
                          ; register then
                          ; inc pointer
        BTFSS  FSR0H, 1   ; All done with
                          ; Bank1?
        BRA    NEXT       ; NO, clear next
CONTINUE                  ; YES, continue
```

15.4 Noise Filter

The Timer5 module includes an optional input noise filter, designed to reduce spurious signals in noisy operating environments. The filter ensures that the input is not permitted to change until a stable value has been registered for three consecutive sampling clock cycles.

The noise filter is part of the input filter network associated with the Motion Feedback Module (see **Section 17.0 "Motion Feedback Module"**). All of the filters are controlled using the Digital Filter Control (DFLTCON) register (Register 17-3). The Timer5 filter can be individually enabled or disabled by setting or clearing the FLT4EN bit (DFLTCON<6>). It is disabled on all Brown-out Resets.

For additional information, refer to **Section 17.3 "Noise Filters"** in the Motion Feedback Module.

15.5 Timer5 Interrupt

Timer5 has the ability to generate an interrupt on a period match. When the PR5 register is loaded with a new period value (00FFh), the Timer5 time base increments until its value is equal to the value of PR5. When a match occurs, the Timer5 interrupt is generated on the rising edge of Q4; TMR5IF is set on the next TCY.

The interrupt latency (i.e., the time elapsed from the moment Timer5 rolls over until TMR5IF is set) will not exceed 1 TCY. When the Timer5 clock input is prescaled and a TMR5/PR5 match occurs, the interrupt will be generated on the first Q4 rising edge after TMR5 resets.

15.6 Timer5 Special Event Trigger Output

A Timer5 Special Event Trigger is generated on a TMR5/PR5 match. The Special Event Trigger is generated on the falling edge of Q3.

Timer5 must be configured for either Synchronous mode (Counter or Timer) to take advantage of the Special Event Trigger feature. If Timer5 is running in Asynchronous Counter mode, the Special Event Trigger may not work and should not be used.

15.7 Timer5 Special Event Trigger Reset Input

In addition to the Special Event Trigger output, Timer5 has a Special Event Trigger Reset input that may be used with Input Capture Channel 1 (IC1) of the Motion Feedback Module. To use the Special Event Trigger Reset, the Capture 1 Control register, CAP1CON, must be configured for one of the Special Event Trigger modes (CAP1M<3:0> = 1110 or 1111). The Special Event Trigger Reset can be disabled by setting the $\overline{\text{RESEN}}$ control bit (T5CON<6>).

The Special Event Trigger Reset resets the Timer5 time base. This Reset occurs in either Continuous Count or Single-Shot modes.

15.7.1 WAKE-UP ON IC1 EDGE

The Timer5 Special Event Trigger Reset input can act as a Timer5 wake-up and a start-up pulse. Timer5 must be in Single-Shot mode and disabled (TMR5ON = 0). An active edge on the CAP1 input pin will set TMR5ON. The timer is subsequently incremented on the next following clock according to the prescaler and the Timer5 clock settings. A subsequent hardware time-out (such as TMR5/PR5 match) will clear the TMR5ON bit and stop the timer.

15.7.2 DELAYED ACTION EVENT TRIGGER

An active edge on CAP1 can also be used to initiate some later action delayed by the Timer5 time base. In this case, Timer5 increments as before after being triggered. When the hardware time-out occurs, the Special Event Trigger output is generated and used to trigger another action, such as an A/D conversion. This allows a given hardware action to be referenced from a capture edge on CAP1 and delayed by the timer.

The event timing for the delayed action event trigger is discussed further in **Section 17.1 "Input Capture"**.

15.7.3 SPECIAL EVENT TRIGGER RESET WHILE TIMER5 IS INCREMENTING

In the event that a bus write to Timer5 coincides with a Special Event Trigger Reset, the bus write will always take precedence over the Special Event Trigger Reset.

15.8 Operation in Sleep Mode

When Timer5 is configured for asynchronous operation, it will continue to increment each timer clock (or prescale multiple of clocks). Executing the SLEEP instruction will either stop the timer or let the timer continue, depending on the setting of the Timer5 Sleep Enable bit, T5SEN. If T5SEN is set (= 1), the timer continues to run when the SLEEP instruction is executed and the external clock is selected (TMR5CS = 1). If T5SEN is cleared, the timer stops when a SLEEP instruction is executed, regardless of the state of the TMR5CS bit.

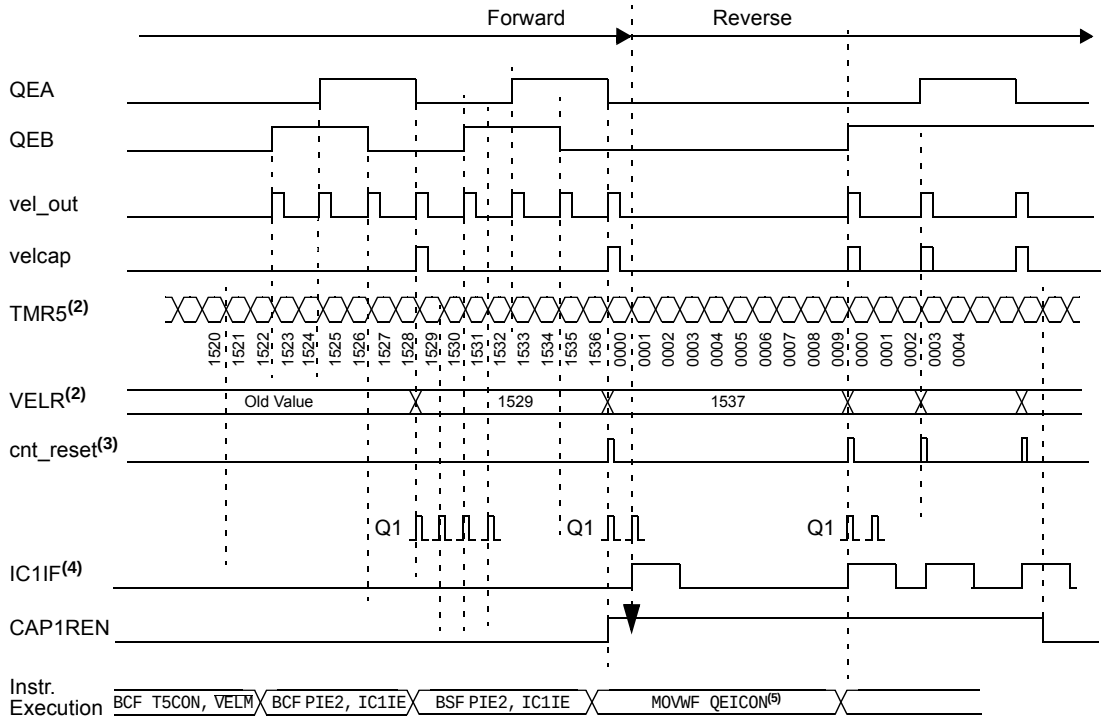
To summarize, Timer5 will continue to increment when a SLEEP instruction is executed only if all of these bits are set:

- TMR5ON
- T5SEN
- TMR5CS
- $\overline{\text{T5SYNC}}$

15.8.1 INTERRUPT DETECT IN SLEEP MODE

When configured as described above, Timer5 will continue to increment on each rising edge on T5CKI while in Sleep mode. When a TMR5/PR5 match occurs, an interrupt is generated which can wake the part.

FIGURE 17-13: VELOCITY MEASUREMENT TIMING(1)



Note 1: Timing shown is for QEIM<2:0> = 101, 110 or 111 (x4 Update mode enabled) and the velocity postscaler divide ratio is set to divide-by-4 (PDEC<1:0> = 01).

2: The VELR register latches the TMR5 count on the "velcap" capture pulse. Timer5 must be set to the Synchronous Timer or Counter mode. In this example, it is set to the Synchronous Timer mode, where the TMR5 prescaler divide ratio = 1 (i.e., Timer5 Clock = TCY).

3: The TMR5 counter is reset on the next Q1 clock cycle following the "velcap" pulse. The TMR5 value is unaffected when the Velocity Measurement mode is first enabled ($\overline{VELM}$ = 0). The velocity postscaler values must be reconfigured to their previous settings when re-entering Velocity Measurement mode. While making speed measurements of very slow rotational speeds (e.g., servo-controller applications), the Velocity Measurement mode may not provide sufficient precision. The Pulse-Width Measurement mode may have to be used to provide the additional precision. In this case, the input pulse is measured on the CAP1 input pin.

4: IC1IF interrupt is enabled by setting IC1IE as follows: BSF PIE2, IC1IE. Assume IC1E bit is placed in the PIE2 (Peripheral Interrupt Enable 2) register in the target device. The actual IC1IF bit is written on the Q2 rising edge.

5: The post decimation value is changed from PDEC = 01 (decimate by 4) to PDEC = 00 (decimate by 1).

17.2.6.2 Velocity Postscaler

The velocity event pulse (velcap, see Figure 17-12) serves as the TMR5 capture trigger to IC1 while in the Velocity mode. The number of velocity events are reduced by the velocity postscaler before they are used as the input capture clock. The velocity event reduction ratio can be set with the PDEC<1:0> control bits (QEICON<1:0>) to 1:4, 1:16, 1:64 or no reduction (1:1).

The velocity postscaler settings are automatically reloaded from their previous values as the Velocity mode is re-enabled.

17.2.6.3 CAP1REN in Velocity Mode

The TMR5 value can be reset (TMR5 register pair = 0000h) on a velocity event capture by setting the CAP1REN bit (CAP1CON<6>). When CAP1REN is cleared, the TMR5 time base will not be reset on any velocity event capture pulse. The VELR register pair, however, will continue to be updated with the current TMR5 value.

18.1 Control Registers

The operation of the PWM module is controlled by a total of 22 registers. Eight of these are used to configure the features of the module:

- PWM Timer Control Register 0 (PTCON0)
- PWM Timer Control Register 1 (PTCON1)
- PWM Control Register 0 (PWMCON0)
- PWM Control Register 1 (PWMCON1)
- Dead-Time Control Register (DTCON)
- Output Override Control Register (OVDCOND)
- Output State Register (OVDCONS)
- Fault Configuration Register (FLTCONFIG)

There are also 14 registers that are configured as seven register pairs of 16 bits. These are used for the configuration values of specific features. They are:

- PWM Time Base Registers (PTMRH and PTMRL)
- PWM Time Base Period Registers (PTPERH and PTPERL)
- PWM Special Event Trigger Compare Registers (SEVTCMPH and SEVTCMPL)
- PWM Duty Cycle #0 Registers (PDC0H and PDC0L)
- PWM Duty Cycle #1 Registers (PDC1H and PDC1L)
- PWM Duty Cycle #2 Registers (PDC2H and PDC2L)
- PWM Duty Cycle #3 Registers (PDC3H and PDC3L)

All of these register pairs are double-buffered.

18.2 Module Functionality

The PWM module supports several modes of operation that are beneficial for specific power and motor control applications. Each mode of operation is described in subsequent sections.

The PWM module is composed of several functional blocks. The operation of each is explained separately in relation to the several modes of operation:

- PWM Time Base
- PWM Time Base Interrupts
- PWM Period
- PWM Duty Cycle
- Dead-Time Generators
- PWM Output Overrides
- PWM Fault Inputs
- PWM Special Event Trigger

18.3 PWM Time Base

The PWM time base is provided by a 12-bit timer with prescaler and postscaler functions. A simplified block diagram of the PWM time base is shown in Figure 18-4. The PWM time base is configured through the PTCON0 and PTCON1 registers. The time base is enabled or disabled by respectively setting or clearing the PTEN bit in the PTCON1 register.

Note:	The PTMR register pair (PTMRL:PTMRH) is not cleared when the PTEN bit is cleared in software.

18.6 PWM Duty Cycle

PWM duty cycle is defined by the PDCx (PDCxL and PDCxH) registers. There are a total of four PWM Duty Cycle registers for four pairs of PWM channels. The Duty Cycle registers have 14-bit resolution by combining six LSbs of PDCxH with the 8 bits of PDCxL. PDCx is a double-buffered register used to set the counting period for the PWM time base.

18.6.1 PWM DUTY CYCLE REGISTERS

There are four 14-bit Special Function Registers used to specify duty cycle values for the PWM module:

- PDC0 (PDC0L and PDC0H)
- PDC1 (PDC1L and PDC1H)
- PDC2 (PDC2L and PDC2H)
- PDC3 (PDC3L and PDC3H)

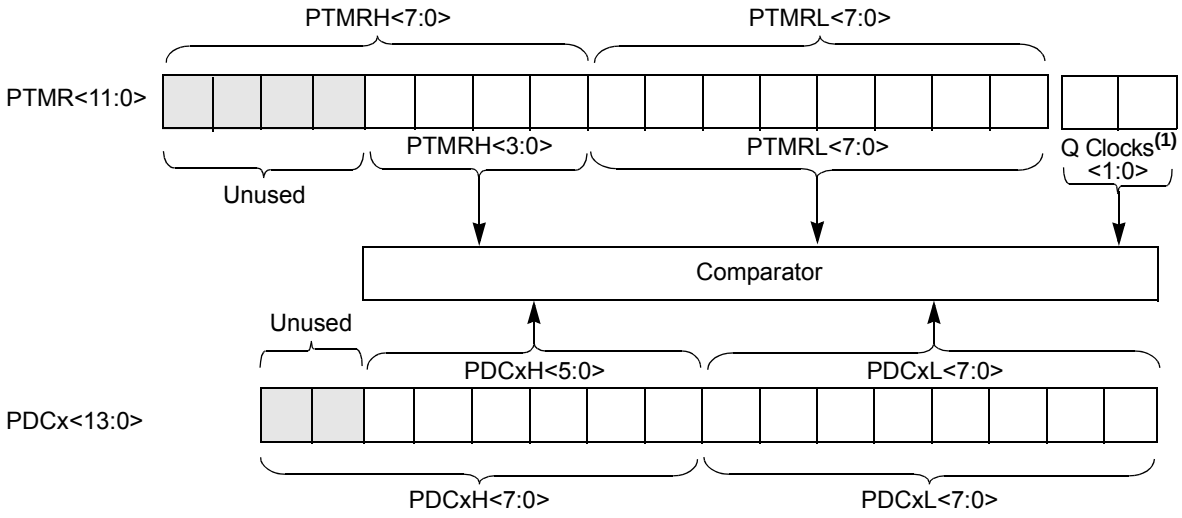
The value in each Duty Cycle register determines the amount of time that the PWM output is in the active state. The upper 12 bits of PDCx holds the actual duty cycle value from PTMRH/L<11:0>, while the lower 2 bits control which internal Q clock the duty cycle match will occur. This 2-bit value is decoded from the Q clocks as shown in Figure 18-11 (when the prescaler is 1:1 or PTCKPS<1:0> = 00).

In Edge-Aligned mode, the PWM period starts at Q1 and ends when the Duty Cycle register matches the PTMR register as follows. The duty cycle match is considered when the upper 12 bits of the PDCx are equal to the PTMR and the lower 2 bits are equal to Q1, Q2, Q3 or Q4, depending on the lower two bits of the PDCx (when the prescaler is 1:1 or PTCKPS<1:0> = 00).

Note:	When the prescaler is not 1:1 (PTCKPS<1:0> ≠ ~00), the duty cycle match occurs at the Q1 clock of the instruction cycle when the PTMR and PDCx match occurs.

Each compare unit has logic that allows override of the PWM signals. This logic also ensures that the PWM signals will complement each other (with dead-time insertion) in Complementary mode (see **Section 18.7 "Dead-Time Generators"**).

FIGURE 18-11: DUTY CYCLE COMPARISON



Note 1: This value is decoded from the Q clocks:
00 = duty cycle match occurs on Q1
01 = duty cycle match occurs on Q2
10 = duty cycle match occurs on Q3
11 = duty cycle match occurs on Q4

NOTES:

EXAMPLE 20-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:
Desired Baud Rate = FOSC/(64 ([SPBRGH:SPBRG] + 1))
Solving for SPBRGH:SPBRG:

X = ((FOSC/Desired Baud Rate)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25

Calculated Baud Rate = 16000000/(64 (25 + 1))
= 9615
Error = (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
= (9615 – 9600)/9600 = 0.16%

TABLE 20-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	56
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	56
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	56
SPBRGH	EUSART Baud Rate Generator Register High Byte								56
SPBRG	EUSART Baud Rate Generator Register Low Byte								56

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the BRG.

TABLE 20-3: BAUD RATES FOR ASYNCHRONOUS MODES

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 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	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.16	207	0.300	-0.16	103	0.300	-0.16	51
1.2	1.202	0.16	51	1.201	-0.16	25	1.201	-0.16	12
2.4	2.404	0.16	25	2.403	-0.16	12	—	—	—
9.6	8.929	-6.99	6	—	—	—	—	—	—
19.2	20.833	8.51	2	—	—	—	—	—	—
57.6	62.500	8.51	0	—	—	—	—	—	—
115.2	62.500	-45.75	0	—	—	—	—	—	—

20.5.2 EUSART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep, or any Idle mode and bit SREN, which is a “don’t care” in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep or any Idle mode, then a word may be received while in this Low-Power mode. Once the word is received, the RSR register will transfer the data to the RCREG register. If the RCIE enable bit is set, the interrupt generated will wake the chip from Low-Power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. If interrupts are desired, set enable bit, RCIE.
3. If 9-bit reception is desired, set bit, RX9.
4. To enable reception, set enable bit, CREN.
5. Flag bit, RCIF, will be set when reception is complete. An interrupt will be generated if enable bit, RCIE, was set.
6. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit, CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 20-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	54
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	57
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	57
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	57
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	56
RCREG	EUSART Receive Register								56
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	56
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	56
SPBRGH	EUSART Baud Rate Generator Register High Byte								56
SPBRG	EUSART Baud Rate Generator Register Low Byte								56

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.

REGISTER 21-6: ANSEL0: ANALOG SELECT REGISTER 0(1)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANS7(2)	ANS6(2)	ANS5(2)	ANS4	ANS3	ANS2	ANS1	ANS0
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 7-0 **ANS<7:0>:** Analog Input Function Select bits
Correspond to pins, AN<7:0>.
1 = Analog input
0 = Digital I/O

Note 1: Setting a pin to an analog input disables the digital input buffer. The corresponding TRIS bit should be set for an input and cleared for an output (analog or digital). The ANSx bits directly correspond to the ANx pins (e.g., ANS0 = AN0, ANS1 = AN1, etc.). Unused ANSx bits are read as '0'.
2: ANS7 through ANS5 are available only on PIC18F4331/4431 devices.

REGISTER 21-7: ANSEL1: ANALOG SELECT REGISTER 1(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-1
—	—	—	—	—	—	—	ANS8(2)
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 7-1 **Unimplemented:** Read as '0'
bit 0 **ANS8:** Analog Input Function Select bit(2)
1 = Analog input
0 = Digital I/O

Note 1: Setting a pin to an analog input disables the digital input buffer. The corresponding TRIS bit should be set for an input and cleared for an output (analog or digital). The ANSx bits directly correspond to the ANx pins (e.g., ANS8 = AN8, ANS9 = AN9, etc.). Unused ANSx bits are read as '0'.
2: ANS8 is available only on PIC18F4331/4431 devices.

MOVFF **Move f to f**

Syntax: [*label*] MOVFF f_s,f_d

Operands: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)	1100	ffff	ffff	$ffff_s$
2nd word (destin.)	1111	ffff	ffff	$ffff_d$

Description: The contents of source register, 'f_s', are moved to destination register, 'f_d'. Location of source, 'f_s', can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination, 'f_d', can also be anywhere from 000h to FFFh.
Either source or destination can be W (a useful special situation).
MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).
The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
The MOVFF instruction should not be used to modify interrupt settings while any interrupt is enabled (see the note on page 97).

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: MOVFF REG1, REG2

Before Instruction
REG1 = 0x33
REG2 = 0x11
After Instruction
REG1 = 0x33
REG2 = 0x33

MOVLB **Move Literal to Low Nibble in BSR**

Syntax: [*label*] MOVLB k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow BSR$

Status Affected: None

Encoding:

0000	0001	0000	kkkk

Description: The 8-bit literal, 'k', is loaded into the Bank Select Register (BSR).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: MOVLB 5

Before Instruction
BSR register = 0x02
After Instruction
BSR register = 0x05

RRNCF **Rotate Right f (No Carry)**

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

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

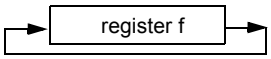
Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

Encoding:

0100	00da	ffff	ffff

Description: The contents of register, 'f', are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register, 'f'. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value.



Words: 1

Cycles: 1

Q Cycle Activity:

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

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

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

SETF **Set f**

Syntax: [*label*] SETF f [,a]

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

Operation: FFh → f

Status Affected: None

Encoding:

0110	100a	ffff	ffff

Description: The contents of the specified register are set to FFh. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
SETF    REG
```
Before Instruction
REG = 0x5A
After Instruction
REG = 0xFF

25.2 MPLAB C Compilers for Various Device Families

The MPLAB C Compiler code development systems are complete ANSI C compilers for Microchip's PIC18, PIC24 and PIC32 families of microcontrollers and the dsPIC30 and dsPIC33 families of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

25.3 HI-TECH C for Various Device Families

The HI-TECH C Compiler code development systems are complete ANSI C compilers for Microchip's PIC family of microcontrollers and the dsPIC family of digital signal controllers. These compilers provide powerful integration capabilities, omniscient code generation and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

The compilers include a macro assembler, linker, preprocessor, and one-step driver, and can run on multiple platforms.

25.4 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

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, absolute LST files that contain source lines and generated machine code and COFF files 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

25.5 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

25.6 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

26.2 DC Characteristics: Power-Down and Supply Current PIC18F2331/2431/4331/4431 (Industrial, Extended) PIC18LF2331/2431/4331/4431 (Industrial) (Continued)

PIC18LF2331/2431/4331/4431 (Industrial)	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial
PIC18F2331/2431/4331/4431 (Industrial, Extended)	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended

Param No.	Device	Typ	Max	Units	Conditions		
	Supply Current (IDD)(2,3)						
	PIC18LF2X31/4X31	8	40	µA	-40°C	VDD = 2.0V	FOSC = 31 kHz (**RC_RUN** mode, Internal oscillator source)
		9	40	µA	+25°C		
		11	40	µA	+85°C		
	PIC18LF2X31/4X31	25	68	µA	-40°C	VDD = 3.0V	
		25	68	µA	+25°C		
		20	68	µA	+85°C		
	All devices	55	180	µA	-40°C	VDD = 5.0V	
		55	180	µA	+25°C		
		50	180	µA	+85°C		
		0.25	1	mA	+125°C		
	PIC18LF2X31/4X31	140	220	µA	-40°C	VDD = 2.0V	FOSC = 1 MHz (**RC_RUN** mode, Internal oscillator source)
		145	220	µA	+25°C		
		155	220	µA	+85°C		
	PIC18LF2X31/4X31	215	330	µA	-40°C	VDD = 3.0V	
		225	330	µA	+25°C		
		235	330	µA	+85°C		
	All devices	385	550	µA	-40°C	VDD = 5.0V	
		390	550	µA	+25°C		
		405	550	µA	+85°C		
		0.7	2.8	mA	+125°C		
	PIC18LF2X31/4X31	410	600	µA	-40°C	VDD = 2.0V	FOSC = 4 MHz (**RC_RUN** mode, Internal oscillator source)
		425	600	µA	+25°C		
		435	600	µA	+85°C		
	PIC18LF2X31/4X31	650	900	µA	-40°C	VDD = 3.0V	
		670	900	µA	+25°C		
		680	900	µA	+85°C		
	All devices	1.2	1.8	mA	-40°C	VDD = 5.0V	
		1.2	1.8	mA	+25°C		
		1.2	1.8	mA	+85°C		
		2.2	6	mA	+125°C		

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;
MCLR = VDD; WDT enabled/disabled as specified.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula: Ir = VDD/2REXT (mA) with REXT in kΩ.

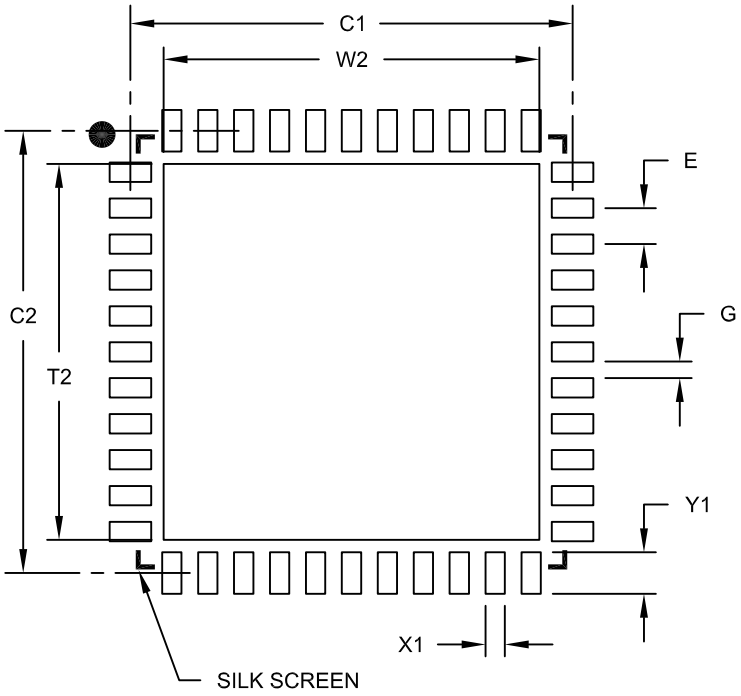
4: Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

TABLE 26-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
30	TMCL	$\overline{\text{MCLR}}$ Pulse Width (low)	2	—	—	μs	
31	TWDT	Watchdog Timer Time-out Period (no postscaler)	—	4.00	—	ms	
32	TOST	Oscillation Start-up Timer Period	1024 TOSC	—	1024 TOSC	—	TOSC = OSC1 period
33	TPWRT	Power-up Timer Period	—	65.5	—	ms	
34	TIOZ	I/O High-impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset	—	2	—	μs	
35	TBOR	Brown-out Reset Pulse Width	200	—	—	μs	VDD ≤ BVDD (see D005)
36	TIRVST	Time for Internal Reference Voltage to become Stable	—	20	50	μs	
37	TLVD	Low-Voltage Detect Pulse Width	200	—	—	μs	VDD ≤ VLVD
38	TCSD	CPU Start-up Time	—	10	—	μs	
39	TIOBST	Time for INTOSC to Stabilize	—	1	—	ms	

44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Optional Center Pad Width	W2			6.80
Optional Center Pad Length	T2			6.80
Contact Pad Spacing	C1		8.00	
Contact Pad Spacing	C2		8.00	
Contact Pad Width (X44)	X1			0.35
Contact Pad Length (X44)	Y1			0.80
Distance Between Pads	G	0.25		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

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

Microchip Technology Drawing No. C04-2103A

NOTES:

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support
- Development Systems Information Line

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://support.microchip.com

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
RE: Reader Response

Total Pages Sent _____

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

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

Application (optional):

Would you like a reply? ___ Y ___ N

Device: PIC18F2331/2431/4331/4431 Literature Number: DS39616D

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?