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	25MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	52
Program Memory Size	48KB (24K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.25K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.23x24.23)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f6585-e-l

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PIC18F6X8X		PIC18F8X8X			
	TQFP	PLCC	TQFP			
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT0	48	60	58			
RB0				I/O	TTL	Digital I/O.
INT0				I	ST	External interrupt 0.
RB1/INT1	47	59	57			
RB1				I/O	TTL	Digital I/O.
INT1				I	ST	External interrupt 1.
RB2/INT2	46	58	56			
RB2				I/O	TTL	Digital I/O.
INT2				I	ST	External interrupt 2.
RB3/INT3/CCP2	45	57	55			
RB3				I/O	TTL	Digital I/O.
INT3				I/O	ST	External interrupt 3.
CCP2(1)				I/O	ST	Capture 2 input/Compare 2 output/ PWM 2 output.
RB4/KBI0	44	56	54			
RB4				I/O	TTL	Digital I/O.
KBI0				I	ST	Interrupt-on-change pin.
RB5/KBI1/PGM	43	55	53			
RB5				I/O	TTL	Digital I/O.
KBI1				I	ST	Interrupt-on-change pin.
PGM				I/O	ST	Low-Voltage ICSP Programming enable pin.
RB6/KBI2/PGC	42	54	52			
RB6				I/O	TTL	Digital I/O.
KBI2				I	ST	Interrupt-on-change pin.
PGC				I/O	ST	In-circuit debugger and ICSP programming clock.
RB7/KBI3/PGD	37	48	47			
RB7				I/O	TTL	Digital I/O.
KBI3				I/O	ST	Interrupt-on-change pin.
PGD						In-circuit debugger and ICSP programming data.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to VDD)

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.

2: Default assignment when CCP2MX is set.

3: External memory interface functions are only available on PIC18F8X8X devices.

4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.

5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.

6: PSP is available in Microcontroller mode only.

7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

4.2 Return Address Stack

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC (Program Counter) is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW, or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit stack pointer, with the stack pointer initialized to 00000b after all Resets. There is no RAM associated with stack pointer 00000b. This is only a Reset value. During a CALL type instruction causing a push onto the stack, the stack pointer is first incremented and the RAM location pointed to by the stack pointer is written with the contents of the PC. During a RETURN type instruction causing a pop from the stack, the contents of the RAM location pointed to by the STKPTR are transferred to the PC and then the stack pointer is decremented.

The stack space is not part of either program or data space. The stack pointer is readable and writable and the address on the top of the stack is readable and writable through SFR registers. Data can also be pushed to or popped from the stack, using the top-of-stack SFRs. Status bits indicate if the stack pointer is at or beyond the 31 levels provided.

4.2.1 TOP-OF-STACK ACCESS

The top of the stack is readable and writable. Three register locations, TOSU, TOSH and TOSL, hold the contents of the stack location pointed to by the STKPTR register. This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU, TOSH and TOSL registers. These values can be placed on a user defined software stack. At return time, the software can replace the TOSU, TOSH and TOSL and do a return.

The user must disable the global interrupt enable bits during this time to prevent inadvertent stack operations.

4.2.2 RETURN STACK POINTER (STKPTR)

The STKPTR register contains the stack pointer value, the STKFUL (Stack Full) status bit, and the STKUNF (Stack Underflow) status bits. Register 4-2 shows the STKPTR register. The value of the stack pointer can be 0 through 31. The stack pointer increments when values are pushed onto the stack and decrements when values are popped off the stack. At Reset, the stack pointer value will be '0'. The user may read and write the stack pointer value. This feature can be used by a Real-Time Operating System for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit can only be cleared in software or by a POR.

The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) configuration bit. Refer to **Section 25.0 "Instruction Set Summary"** for a description of the device configuration bits. If STVREN is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit and reset the device. The STKFUL bit will remain set and the stack pointer will be set to '0'.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the stack pointer will increment to 31. Any additional pushes will not overwrite the 31st push and STKPTR will remain at 31.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and sets the STKUNF bit while the stack pointer remains at '0'. The STKUNF bit will remain set until cleared in software or a POR occurs.

Note:	Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken.

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 is 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 the PCL is fixed to a value of '0'. The PC increments by 2 to address sequential instructions in the program memory.

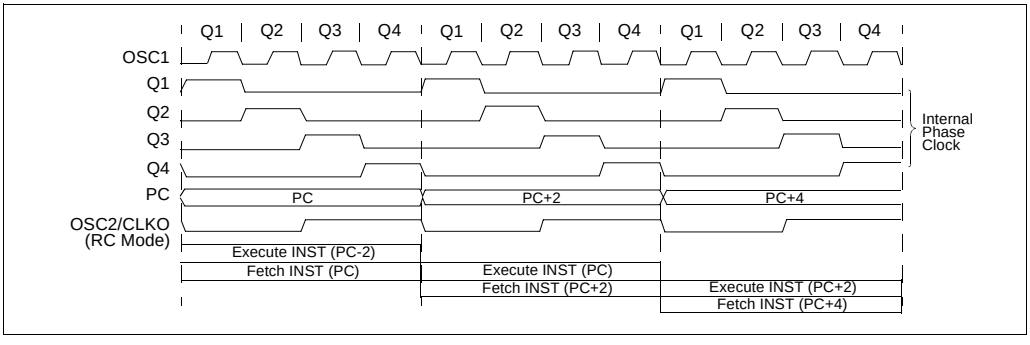
The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

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 "Computed GOTO"**).

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 are shown in Figure 4-5.

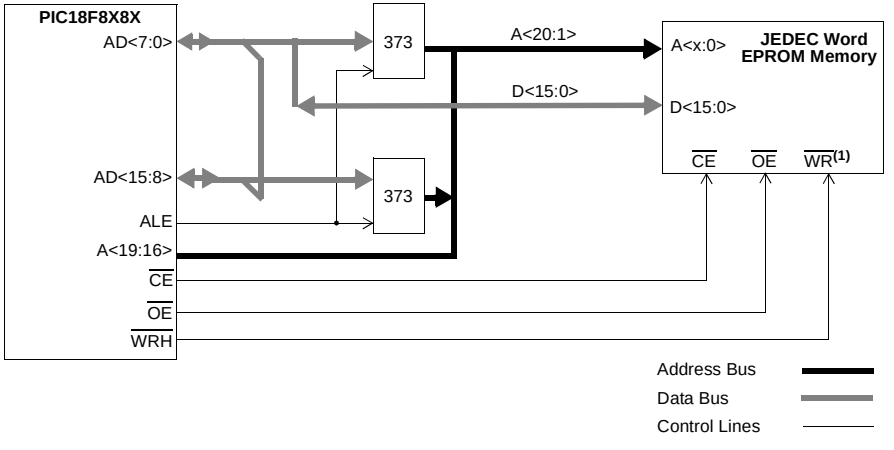
FIGURE 4-5: CLOCK/INSTRUCTION CYCLE



6.2.2 16-BIT WORD WRITE MODE

Figure 6-2 shows an example of 16-bit Word Write mode for PIC18F8X8X devices.

FIGURE 6-2: 16-BIT WORD WRITE MODE EXAMPLE



Note 1: This signal only applies to table writes. See **Section 5.1 "Table Reads and Table Writes"**.

12.2 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit, T1OSCEN (T1CON<3>). The oscillator is a low-power oscillator rated up to 200 kHz. It will continue to run during Sleep. It is primarily intended for a 32 kHz crystal. Table 12-1 shows the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

TABLE 12-1: CAPACITOR SELECTION FOR THE ALTERNATE OSCILLATOR

Osc Type	Freq	C1	C2
LP	32 kHz	TBD(1)	TBD(1)
Crystal to be Tested:			
32.768 kHz	Epson C-001R32.768K-A	± 20 PPM	

Note 1: Microchip suggests 33 pF as a starting point in validating the oscillator circuit.

2: Higher capacitance increases the stability of the oscillator but also increases the start-up time.

3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

4: Capacitor values are for design guidance only.

12.3 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to 0FFFFh and rolls over to 0000h. The TMR1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit, TMR1IE (PIE1<0>).

12.4 Resetting Timer1 Using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

Note: The special event triggers from the CCP1 module will not set interrupt flag bit TMR1IF (PIR1<0>).

Timer1 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1, the write will take precedence.

In this mode of operation, the CCPR1H:CCPR1L register pair effectively becomes the period register for Timer1.

12.5 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 12-2). When the RD16 control bit (T1CON<7>) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 high byte buffer. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, is valid due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H Buffer register. Timer1 high byte is updated with the contents of TMR1H when a write occurs to TMR1L. This allows a user to write all 16 bits to both the high and low bytes of Timer1 at once.

The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

TABLE 12-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0111 1111	0111 1111
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

17.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD<6:0> and counts down to '0'. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 17-31). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 17-32).

FIGURE 17-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)

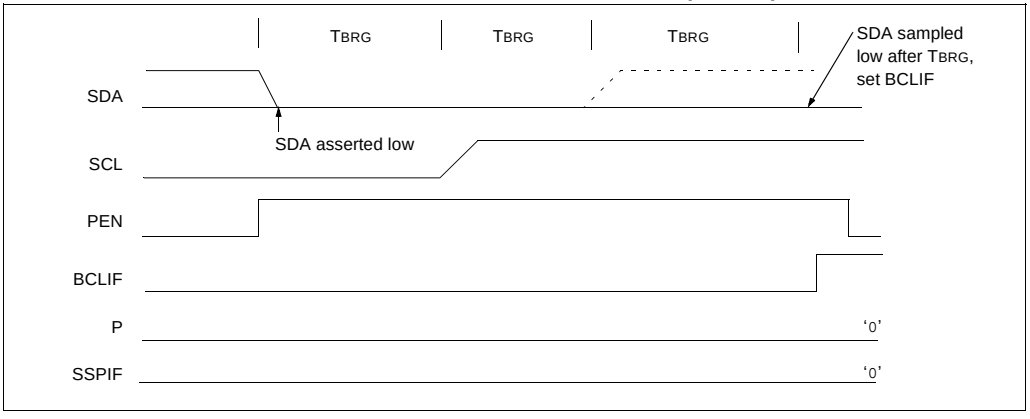
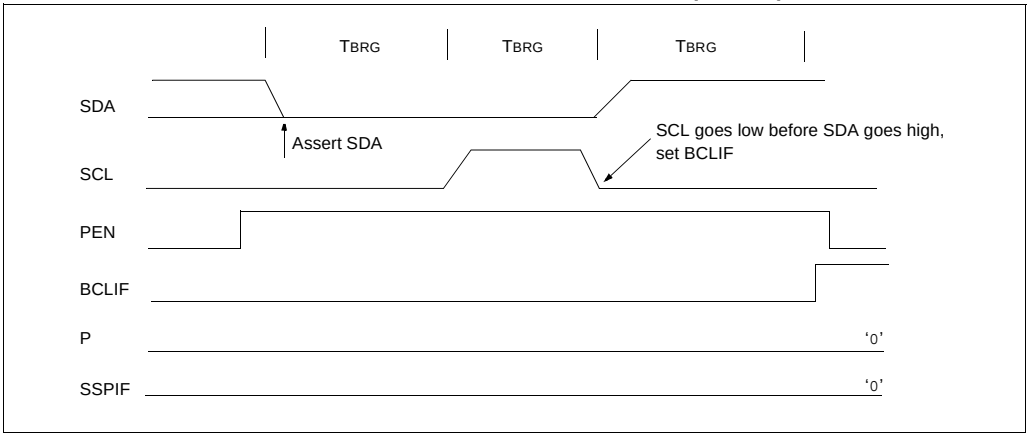


FIGURE 17-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)



18.2.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the USART are suspended. Because of this, the Baud Rate Generator is inactive and a proper byte reception cannot be performed. The auto-wake-up feature allows the controller to wake-up due to activity on the RX/DT line while the USART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCON<1>). Once set, the typical receive sequence on RX/DT is disabled and the USART remains in an Idle state monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a sync break or a wake-up signal character for the LIN protocol.)

Following a wake-up event, the module generates an RCIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 18-7) and asynchronously, if the device is in Sleep mode (Figure 18-8). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RX line following the wake-up event. At this point, the USART module is in Idle mode and returns to normal operation. This signals to the user that the sync break event is over.

18.2.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RX/DT, information with any state changes before the Stop bit may signal a false end-of-character and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all '0's. This can be 00h (8 bytes) for standard RS-232 devices or 000h (12 bits) for LIN bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., XT or HS mode). The sync break (or wake-up signal) character must be of sufficient length and be followed by a sufficient interval to allow enough time for the selected oscillator to start and provide proper initialization of the USART.

18.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the USART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared after this when a rising edge is seen on RX/DT. The interrupt condition is then cleared by reading the RCREG register. Ordinarily, the data in RCREG will be dummy data and should be discarded.

The fact that the WUE bit has been cleared (or is still set) and the RCIF flag is set should not be used as an indicator of the integrity of the data in RCREG. Users should consider implementing a parallel method in firmware to verify received data integrity.

To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

FIGURE 18-7: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION

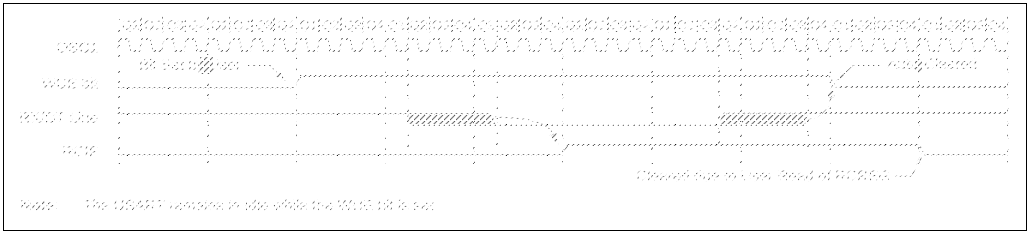
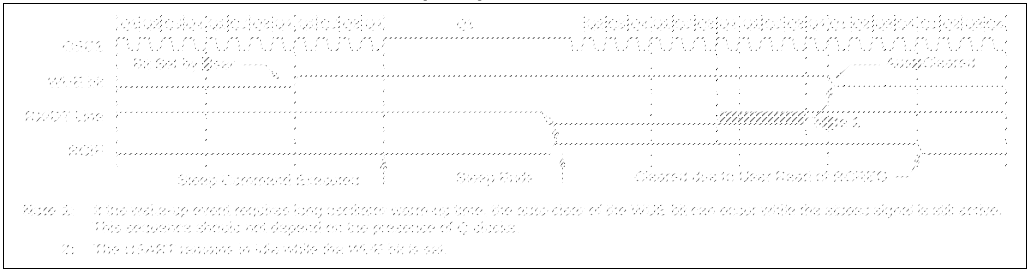


FIGURE 18-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP



18.2.5 BREAK CHARACTER SEQUENCE

The enhanced USART module has the capability of sending the special break character sequences that are required by the LIN bus standard. The break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The frame break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the break character (typically, the sync character in the LIN specification).

Note that the data value written to the TXREG for the break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 18-9 for the timing of the break character sequence.

18.2.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a break, followed by an auto-baud sync byte. This sequence is typical of a LIN bus master.

1. Configure the USART for the desired mode.
2. Set the TXEN and SENDB bits to set up the break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the sync character into the transmit FIFO buffer.
5. After the break has been sent, the SENDB bit is reset by hardware. The sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

18.2.6 RECEIVING A BREAK CHARACTER

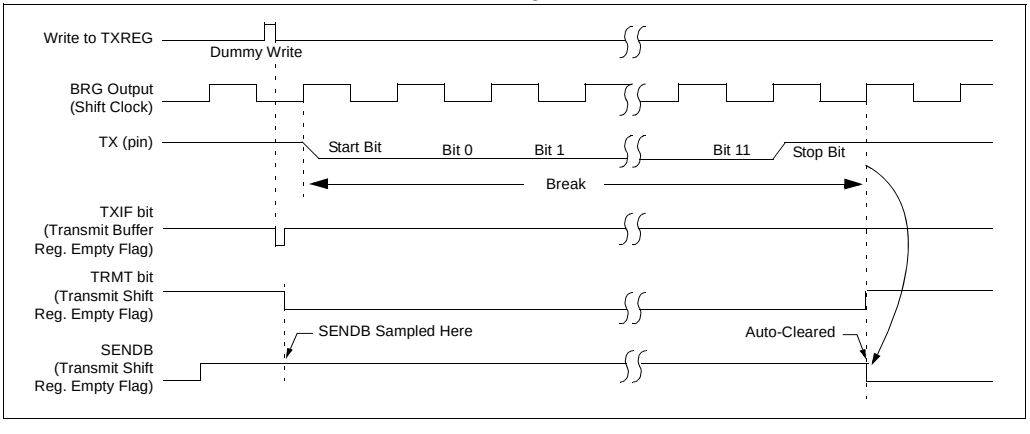
The enhanced USART module can receive a break character in two ways.

The first method forces the configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 18.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the USART will sample the next two transitions on RX/DT, cause an RCIF interrupt, and receive the next data byte followed by another interrupt.

Note that following a break character, the user will typically want to enable the auto-baud rate detect feature. For both methods, the user can set the ABD bit once the TXIF interrupt is observed.

FIGURE 18-9: SEND BREAK CHARACTER SEQUENCE



18.3 USART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTA<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit SYNC (TXSTA<4>). In addition, enable bit, SPEN (RCSTA<7>), is set in order to configure the RC6/TX/CK and RC7/RX/DT I/O pins to CK (clock) and DT (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CK line. Clock polarity is selected with the SCKP bit (BAUDCON<5>); setting SCKP sets the Idle state on CK as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

18.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 18-2. 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 last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available).

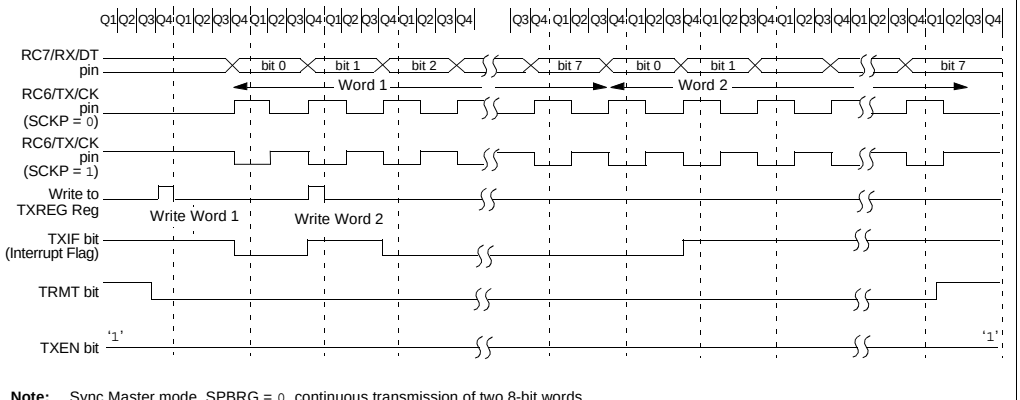
Once the TXREG register transfers the data to the TSR register (occurs in one TCYCLE), the TXREG is empty and interrupt bit TXIF (PIR1<4>) is set. The interrupt can be enabled/disabled by setting/clearing enable bit, TXIE (PIE1<4>). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register.

While flag bit 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 is empty. No interrupt logic is tied to this bit so the user must poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

To set up a synchronous master 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 synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 18-10: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode, SPBRG = 0, continuous transmission of two 8-bit words.

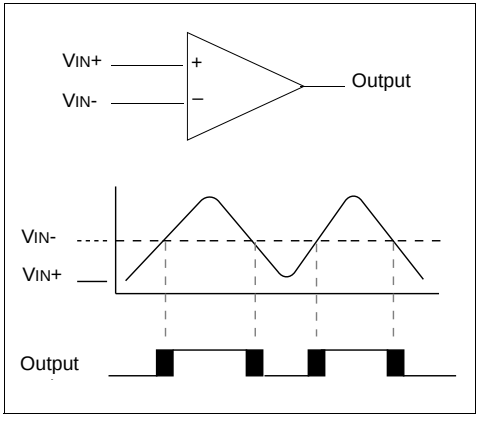
20.2 Comparator Operation

A single comparator is shown in Figure 20-2, along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 20-2 represent the uncertainty due to input offsets and response time.

20.3 Comparator Reference

An external or internal reference signal may be used depending on the Comparator Operating mode. The analog signal present at VIN- is compared to the signal at VIN+ and the digital output of the comparator is adjusted accordingly (Figure 20-2).

FIGURE 20-2: SINGLE COMPARATOR



20.3.1 EXTERNAL REFERENCE SIGNAL

When external voltage references are used, the comparator module can be configured to have the comparators operate from the same or different reference sources. However, threshold detector applications may require the same reference. The reference signal must be between VSS and VDD and can be applied to either pin of the comparator(s).

20.3.2 INTERNAL REFERENCE SIGNAL

The comparator module also allows the selection of an internally generated voltage reference for the comparators. **Section 21.0 "Comparator Voltage Reference Module"** contains a detailed description of the comparator voltage reference module that provides this signal. The internal reference signal is used when comparators are in mode CM<2:0> = 110 (Figure 20-1). In this mode, the internal voltage reference is applied to the VIN+ pin of both comparators.

20.4 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level. If the internal reference is changed, the maximum delay of the internal voltage reference must be considered when using the comparator outputs. Otherwise, the maximum delay of the comparators should be used (**Section 27.0 "Electrical Characteristics"**).

20.5 Comparator Outputs

The comparator outputs are read through the CMCON register. These bits are read-only. The comparator outputs may also be directly output to the RF1 and RF2 I/O pins. When enabled, multiplexors in the output path of the RF1 and RF2 pins will switch and the output of each pin will be the unsynchronized output of the comparator. The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications. Figure 20-3 shows the comparator output block diagram.

The TRISA bits will still function as an output enable/disable for the RF1 and RF2 pins while in this mode.

The polarity of the comparator outputs can be changed using the C2INV and C1INV bits (CMCON<4:5>).

Note 1:	When reading the Port register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert an analog input according to the Schmitt Trigger input specification.
2:	Analog levels on any pin defined as a digital input may cause the input buffer to consume more current than is specified.

TABLE 23-1: CAN CONTROLLER REGISTER MAP (CONTINUED)

Address[1]	Name	Address	Name	Address	Name	Address	Name
E7Fh	CANCON_RO4[2]	E5Fh	CANCON_RO6[2]	E3Fh	CANCON_RO8[2]	E1Fh	—[4]
E7Eh	CANSTAT_RO4[2]	E5Eh	CANSTAT_RO6[2]	E3Eh	CANSTAT_RO8[2]	E1Eh	—[4]
E7Dh	B5D7	E5Dh	B3D7	E3Dh	B1D7	E1Dh	—[4]
E7Ch	B5D6	E5Ch	B3D6	E3Ch	B1D6	E1Ch	—[4]
E7Bh	B5D5	E5Bh	B3D5	E3Bh	B1D5	E1Bh	—[4]
E7Ah	B5D4	E5Ah	B3D4	E3Ah	B1D4	E1Ah	—[4]
E79h	B5D3	E59h	B3D3	E39h	B1D3	E19h	—[4]
E78h	B5D2	E58h	B3D2	E38h	B1D2	E18h	—[4]
E77h	B5D1	E57h	B3D1	E37h	B1D1	E17h	—[4]
E76h	B5D0	E56h	B3D0	E36h	B1D0	E16h	—[4]
E75h	B5DLC	E55h	B3DLC	E35h	B1DLC	E15h	—[4]
E74h	B5EIDL	E54h	B3EIDL	E34h	B1EIDL	E14h	—[4]
E73h	B5EIDH	E53h	B3EIDH	E33h	B1EIDH	E13h	—[4]
E72h	B5SIDL	E52h	B3SIDL	E32h	B1SIDL	E12h	—[4]
E71h	B5SIDH	E51h	B3SIDH	E31h	B1SIDH	E11h	—[4]
E70h	B5CON	E50h	B3CON	E30h	B1CON	E10h	—[4]
E6Fh	CANCON_RO5	E4Fh	CANCON_RO7	E2Fh	CANCON_RO9	E0Fh	—[4]
E6Eh	CANSTAT_RO5	E4Eh	CANSTAT_RO7	E2Eh	CANSTAT_RO9	E0Eh	—[4]
E6Dh	B4D7	E4Dh	B2D7	E2Dh	B0D7	E0Dh	—[4]
E6Ch	B4D6	E4Ch	B2D6	E2Ch	B0D6	E0Ch	—[4]
E6Bh	B4D5	E4Bh	B2D5	E2Bh	B0D5	E0Bh	—[4]
E6Ah	B4D4	E4Ah	B2D4	E2Ah	B0D4	E0Ah	—[4]
E69h	B4D3	E49h	B2D3	E29h	B0D3	E09h	—[4]
E68h	B4D2	E48h	B2D2	E28h	B0D2	E08h	—[4]
E67h	B4D1	E47h	B2D1	E27h	B0D1	E07h	—[4]
E66h	B4D0	E46h	B2D0	E26h	B0D0	E06h	—[4]
E65h	B4DLC	E45h	B2DLC	E25h	B0DLC	E05h	—[4]
E64h	B4EIDL	E44h	B2EIDL	E24h	B0EIDL	E04h	—[4]
E63h	B4EIDH	E43h	B2EIDH	E23h	B0EIDH	E03h	—[4]
E62h	B4SIDL	E42h	B2SIDL	E22h	B0SIDL	E02h	—[4]
E61h	B4SIDH	E41h	B2SIDH	E21h	B0SIDH	E01h	—[4]
E60h	B4CON	E40h	B2CON	E20h	B0CON	E00h	—[4]

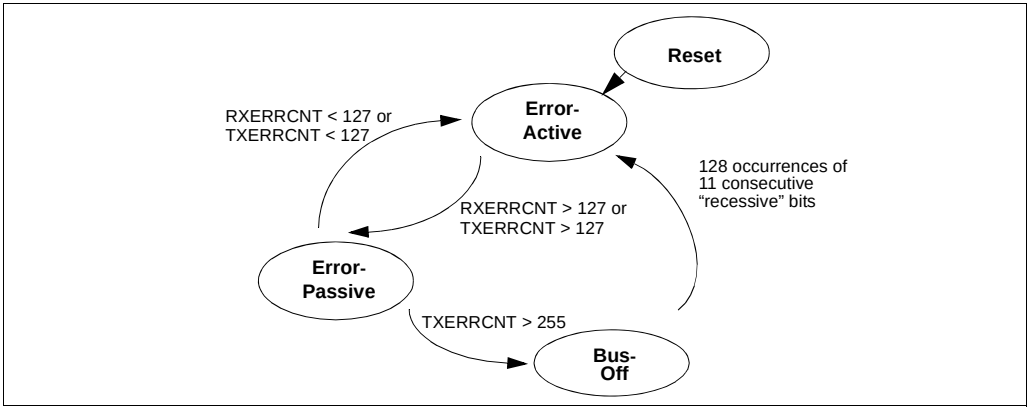
Note 1: Shaded registers are available in Access Bank low area while the rest are available in Bank 15.

2: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

3: These registers are not CAN registers.

4: Unimplemented registers are read as '0'.

FIGURE 23-7: ERROR MODES STATE DIAGRAM



23.15 CAN Interrupts

The module has several sources of interrupts. Each of these interrupts can be individually enabled or disabled. The PIR3 register contains interrupt flags. The PIE3 register contains the enables for the 8 main interrupts. A special set of read-only bits in the CANSTAT register, the ICODE bits, can be used in combination with a jump table for efficient handling of interrupts.

All interrupts have one source with the exception of the error interrupt and buffer interrupts in Mode 1 and 2. Any of the error interrupt sources can set the error interrupt flag. The source of the error interrupt can be determined by reading the Communication Status register, COMSTAT. In Mode 1 and 2, there are two interrupt enable/disable and flag bits – one for all transmit buffers and the other for all receive buffers.

The interrupts can be broken up into two categories: receive and transmit interrupts.

The receive related interrupts are:

- Receive Interrupts
- Wake-up Interrupt
- Receiver Overrun Interrupt
- Receiver Warning Interrupt
- Receiver Error-Passive Interrupt

The transmit related interrupts are:

- Transmit Interrupts
- Transmitter Warning Interrupt
- Transmitter Error-Passive Interrupt
- Bus-Off Interrupt

23.15.1 INTERRUPT CODE BITS

To simplify the interrupt handling process in user firmware, the ECAN module encodes a special set of bits. In Mode 0, these bits are ICODE<2:0> in the CANSTAT register. In Mode 1 and 2, these bits are EICODE<3:0> in the CANSTAT register. Interrupts are internally prioritized such that the higher priority interrupts are assigned lower values. Once the highest priority interrupt condition has been cleared, the code for the next highest priority interrupt that is pending (if any) will be reflected by the ICODE bits. Note that only those interrupt sources that have their associated interrupt enable bit set will be reflected in the ICODE bits.

In Mode 2, when a receive message interrupt occurs, EICODE bits will always consist of ‘10000’. User firmware may use FIFO pointer bits to actually access the next available buffer.

REGISTER 24-5: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

R/P-1	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
MCLRE	—	—	—	—	—	ECCPMX	CCP2MX
bit 7							bit 0

bit 7 **MCLRE:** $\overline{\text{MCLR}}$ Enable bit(1)
1 = $\overline{\text{MCLR}}$ pin enabled, RG5 input pin disabled
0 = RG5 input enabled, $\overline{\text{MCLR}}$ disabled

bit 6-2 **Unimplemented:** Read as '0'

bit 1 **ECCPMX:** CCP1 PWM outputs P1B, P1C mux bit (PIC18F8X8X devices only)(2)
1 = P1B, P1C are multiplexed with RE6, RE5
0 = P1B, P1C are multiplexed with RH7, RH6

bit 0 **CCP2MX:** CCP2 Mux bit
In Microcontroller mode:
1 = CCP2 input/output is multiplexed with RC1
0 = CCP2 input/output is multiplexed with RE7
In Microprocessor, Microprocessor with Boot Block and Extended Microcontroller modes (PIC18F8X8X devices only):
1 = CCP2 input/output is multiplexed with RC1
0 = CCP2 input/output is multiplexed with RB3

Note 1: If $\overline{\text{MCLR}}$ is disabled, either disable low-voltage ICSP or hold RB5/PGM low to ensure proper entry into ICSP mode.
2: Reserved for PIC18F6X8X devices; maintain this bit set.

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

REGISTER 24-6: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/P-1	U-0	U-0	U-0	U-0	R/P-1	U-0	R/P-1
DEBUG	—	—	—	—	LVP	—	STVREN
bit 7							bit 0

bit 7 $\overline{\textbf{DEBUG}}$**:** Background Debugger Enable bit
1 = Background debugger disabled. RB6 and RB7 configured as general purpose I/O pins.
0 = Background debugger enabled. RB6 and RB7 are dedicated to in-circuit debug.

bit 6-3 **Unimplemented:** Read as '0'

bit 2 **LVP:** Low-Voltage ICSP Enable bit
1 = Low-voltage ICSP enabled
0 = Low-voltage ICSP disabled

bit 1 **Unimplemented:** Read as '0'

bit 0 **STVREN:** Stack Full/Underflow Reset Enable bit
1 = Stack full/underflow will cause Reset
0 = Stack full/underflow will not cause Reset

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

25.0 INSTRUCTION SET SUMMARY

The PIC18 instruction set adds many enhancements to the previous PIC MCU instruction sets, while maintaining an easy migration from these PIC MCU instruction sets.

Most instructions are a single program memory word (16 bits) but there are three instructions that require two program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18 instruction set summary in Table 25-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 25-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The destination of the result (specified by 'd')
3. The accessed memory (specified by 'a')

The file register designator 'f' specifies which file register is to be used by the instruction.

The destination designator 'd' specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the WREG register. If 'd' is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The bit in the file register (specified by 'b')
3. The accessed memory (specified by 'a')

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by 'k')
- The desired FSR register to load the literal value into (specified by 'f')
- No operand required (specified by '—')

The **control** instructions may use some of the following operands:

- A program memory address (specified by 'n')
- The mode of the call or return instructions (specified by 's')
- The mode of the table read and table write instructions (specified by 'm')
- No operand required (specified by '—')

All instructions are a single word except for three double-word instructions. These three instructions were made double-word instructions so that all the required information is available in these 32 bits. In the second word, the 4 MSbs are '1's. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single-word instructions are executed in a single instruction cycle unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μs. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 μs. Two-word branch instructions (if true) would take 3 μs.

Figure 25-1 shows the general formats that the instructions can have.

All examples use the format 'nnh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 25-2, lists the instructions recognized by the Microchip Assembler (MPASM™).

Section 25.1 "Instruction Set" provides a description of each instruction.

DAW **Decimal Adjust W Register**

Syntax: [*label*] DAW

Operands: None

Operation: If [W<3:0> >9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;

If [W<7:4> >9] or [C = 1] then
(W<7:4>) + 6 → W<7:4>;
else
(W<7:4>) → W<7:4>;

Status Affected: C

Encoding:

0000	0000	0000	0111

Description: DAW adjusts the eight-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example1: DAW

Before Instruction
W = 0xA5
C = 0
DC = 0

After Instruction
W = 0x05
C = 1
DC = 0

Example 2:

Before Instruction
W = 0xCE
C = 0
DC = 0

After Instruction
W = 0x34
C = 1
DC = 0

DECF **Decrement f**

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

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) – 1 → dest

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

Encoding:

0000	01da	ffff	ffff

Description: Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored 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: DECF CNT, 1, 0

Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

LFSR **Load FSR**

Syntax: [*label*] LFSR f,k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

1110 1111	1110 0000	00ff $k_7$kkk	k_{11}kkk kkkk

Description: The 12-bit literal ‘k’ is loaded into the file select register pointed to by ‘f’.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’ MSB	Process Data	Write literal ‘k’ MSB to FSRfH
Decode	Read literal ‘k’ LSB	Process Data	Write literal ‘k’ to FSRfL

Example: `LFSR 2, 0x3AB`

After Instruction
FSR2H = 0x03
FSR2L = 0xAB

MOVF **Move f**

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

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

Operation: $f \rightarrow dest$

Status Affected: N, Z

Encoding:

0101	00da	ffff	ffff

Description: The contents of register ‘f’ are moved to a destination dependent upon the status of ‘d’. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default). Location ‘f’ can be anywhere in the 256-byte bank. 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 W

Example: `MOVF REG, 0, 0`

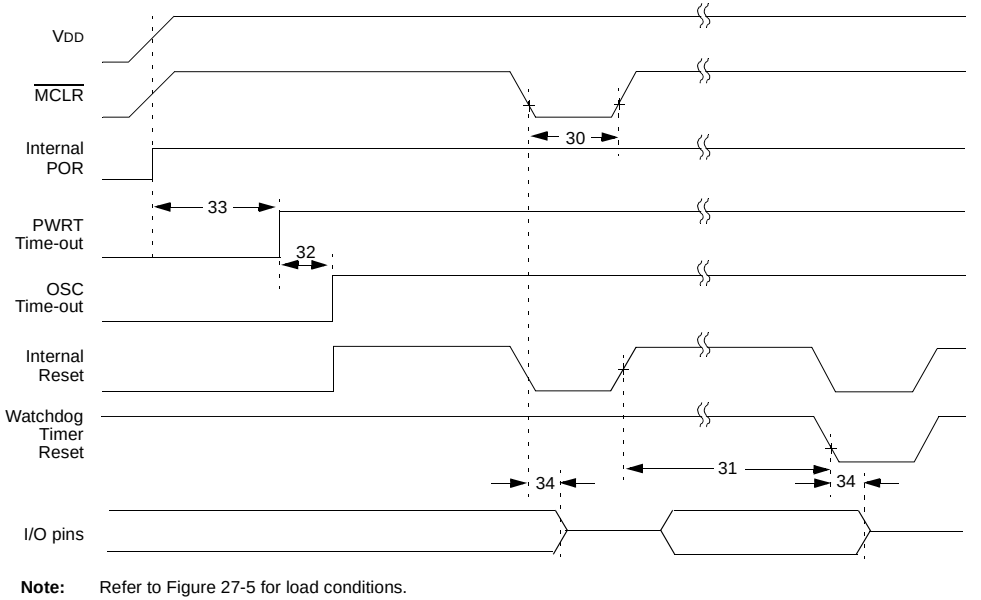
Before Instruction
REG = 0x22
W = 0xFF

After Instruction
REG = 0x22
W = 0x22

TABLE 27-10: PROGRAM MEMORY WRITE TIMING REQUIREMENTS (V_{DD} = 4.2 TO 5.5V)

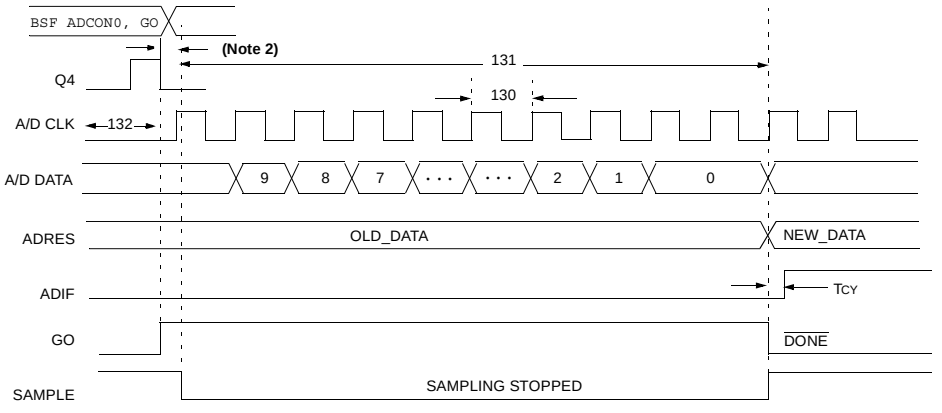
Param. No.	Symbol	Characteristics	Min	Typ	Max	Units
150	TADV2ALL	Address Out Valid to ALE ↓ (address setup time)	0.25 TCY – 10	—	—	ns
151	TALL2ADL	ALE ↓ to Address Out Invalid (address hold time)	5	—	—	ns
153	TWRH2ADL	WRn ↑ to Data Out Invalid (data hold time)	5	—	—	ns
154	TWRL	WRn Pulse Width	0.5 TCY – 5	0.5 TCY	—	ns
156	TADV2WRH	Data Valid before WRn ↑ (data setup time)	0.5 TCY – 10	—	—	ns
157	TBSV2WRL	Byte Select Valid before WRn ↓ (byte select setup time)	0.25 TCY	—	—	ns
157A	TWRH2BSI	WRn ↑ to Byte Select Invalid (byte select hold time)	0.125 TCY – 5	—	—	ns
166	TALH2ALH	ALE ↑ to ALE ↑ (cycle time)	—	0.25 TCY	—	ns
171	TALH2CSL	Chip Enable Active to ALE ↓	—	—	10	ns
171A	TUBL2OEH	AD Valid to Chip Enable Active	0.25 TCY – 20	—	—	ns

FIGURE 27-10: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note: Refer to Figure 27-5 for load conditions.

FIGURE 27-25: 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 27-26: A/D CONVERSION REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
130	TAD	A/D Clock Period	PIC18FXX8X	1.6	20(5)	µs	TOSC based, VREF ≥ 3.0V
			PIC18LFXX8X	3.0	20(5)	µs	TOSC based, VREF full range
			PIC18FXX8X	2.0	6.0	µs	A/D RC mode
			PIC18LFXX8X	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 10	— —	µs µs	-40°C ≤ Temp ≤ +125°C 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 19.0 "10-bit Analog-to-Digital Converter (A/D) Module"** 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 50Ω.

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.

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

Printed on recycled paper.

ISBN: 9781620769638

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.