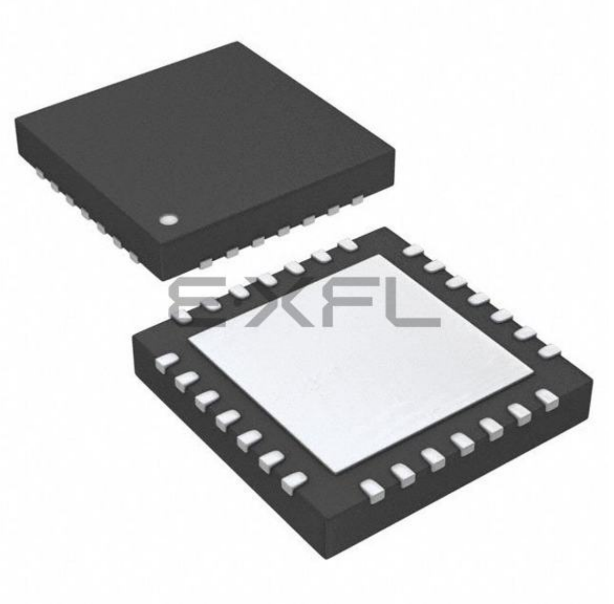


What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	16
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f1320t-i-ml

1.3 Details on Individual Family Members

Devices in the PIC18F1220/1320 family are available in 18-pin, 20-pin and 28-pin packages. A block diagram for this device family is shown in Figure 1-1.

The devices are differentiated from each other only in the amount of on-chip Flash program memory (4 Kbytes for the PIC18F1220 device, 8 Kbytes for the PIC18F1320 device). These and other features are summarized in Table 1-1.

A block diagram of the PIC18F1220/1320 device architecture is provided in Figure 1-1. The pinouts for this device family are listed in Table 1-2.

TABLE 1-1: DEVICE FEATURES

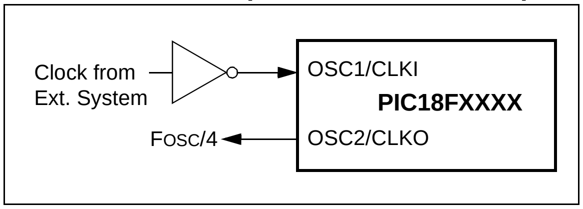
Features	PIC18F1220	PIC18F1320
Operating Frequency	DC – 40 MHz	DC – 40 MHz
Program Memory (Bytes)	4096	8192
Program Memory (Instructions)	2048	4096
Data Memory (Bytes)	256	256
Data EEPROM Memory (Bytes)	256	256
Interrupt Sources	15	15
I/O Ports	Ports A, B	Ports A, B
Timers	4	4
Enhanced Capture/Compare/PWM Modules	1	1
Serial Communications	Enhanced USART	Enhanced USART
10-bit Analog-to-Digital Module	7 input channels	7 input channels
Resets (and Delays)	POR, BOR, `RESET` Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{\text{MCLR}}$ (optional), WDT	POR, BOR, `RESET` Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{\text{MCLR}}$ (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes
Programmable Brown-out Reset	Yes	Yes
Instruction Set	75 Instructions	75 Instructions
Packages	18-pin SDIP 18-pin SOIC 20-pin SSOP 28-pin QFN	18-pin SDIP 18-pin SOIC 20-pin SSOP 28-pin QFN

2.4 External Clock Input

The EC and ECIO Oscillator modes require an external clock source to be connected to the OSC1 pin. There is no oscillator start-up time required after a Power-on Reset, or after an exit from Sleep mode.

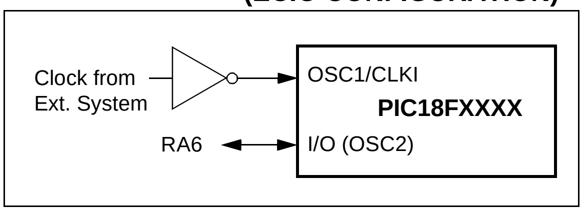
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes, or to synchronize other logic. Figure 2-4 shows the pin connections for the EC Oscillator mode.

FIGURE 2-4: EXTERNAL CLOCK INPUT OPERATION (EC CONFIGURATION)



The ECIO Oscillator mode functions like the EC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6). Figure 2-5 shows the pin connections for the ECIO Oscillator mode.

FIGURE 2-5: EXTERNAL CLOCK INPUT OPERATION (ECIO CONFIGURATION)

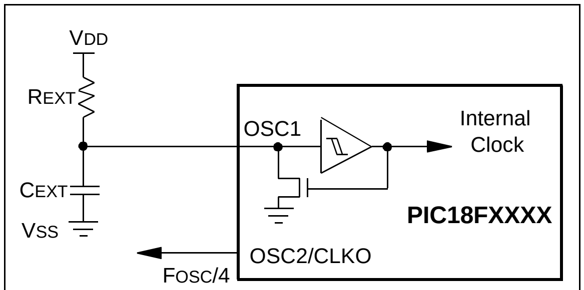


2.5 RC Oscillator

For timing insensitive applications, the "RC" and "RCIO" device options offer additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal manufacturing variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low CEXT values. The user also needs to take into account variation, due to tolerance of external R and C components used. Figure 2-6 shows how the R/C combination is connected.

In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes, or to synchronize other logic.

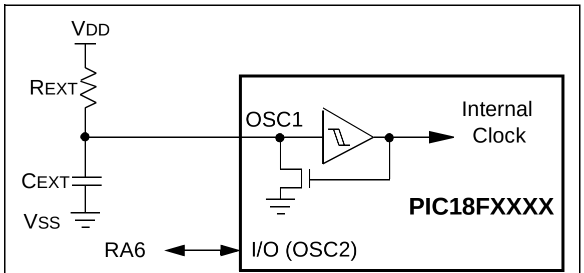
FIGURE 2-6: RC OSCILLATOR MODE



Recommended values: 3 kΩ ≤ REXT ≤ 100 kΩ
CEXT > 20 pF

The RCIO Oscillator mode (Figure 2-7) functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

FIGURE 2-7: RCIO OSCILLATOR MODE



Recommended values: 3 kΩ ≤ REXT ≤ 100 kΩ
CEXT > 20 pF

TABLE 4-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
TOSU	1220	1320	---0 0000	---0 0000	---0 uuuu(3)
TOSH	1220	1320	0000 0000	0000 0000	uuuu uuuu(3)
TOSL	1220	1320	0000 0000	0000 0000	uuuu uuuu(3)
STKPTR	1220	1320	00-0 0000	00-0 0000	uu-u uuuu(3)
PCLATU	1220	1320	---0 0000	---0 0000	---u uuuu
PCLATH	1220	1320	0000 0000	0000 0000	uuuu uuuu
PCL	1220	1320	0000 0000	0000 0000	PC + 2(2)
TBLPTRU	1220	1320	--00 0000	--00 0000	--uu uuuu
TBLPTRH	1220	1320	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	1220	1320	0000 0000	0000 0000	uuuu uuuu
TABLAT	1220	1320	0000 0000	0000 0000	uuuu uuuu
PRODH	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	1220	1320	0000 000x	0000 000u	uuuu uuuu(1)
INTCON2	1220	1320	1111 -1-1	1111 -1-1	uuuu -u-u(1)
INTCON3	1220	1320	11-0 0-00	11-0 0-00	uu-u u-uu(1)
INDF0	1220	1320	N/A	N/A	N/A
POSTINC0	1220	1320	N/A	N/A	N/A
POSTDEC0	1220	1320	N/A	N/A	N/A
PREINC0	1220	1320	N/A	N/A	N/A
PLUSW0	1220	1320	N/A	N/A	N/A
FSR0H	1220	1320	---- 0000	---- 0000	---- uuuu
FSR0L	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	1220	1320	N/A	N/A	N/A
POSTINC1	1220	1320	N/A	N/A	N/A
POSTDEC1	1220	1320	N/A	N/A	N/A
PREINC1	1220	1320	N/A	N/A	N/A
PLUSW1	1220	1320	N/A	N/A	N/A
FSR1H	1220	1320	---- 0000	---- 0000	---- uuuu
FSR1L	1220	1320	xxxx xxxx	uuuu uuuu	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: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

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

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

4: See Table 4-2 for Reset value for specific condition.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the Oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

6: Bit 5 of PORTA is enabled if MCLR is disabled.

5.0 MEMORY ORGANIZATION

There are three memory types in Enhanced MCU devices. These memory types are:

- Program Memory
- Data RAM
- Data EEPROM

Data and program memory use separate busses, which allows for concurrent access of these types.

Additional detailed information for Flash program memory and data EEPROM is provided in **Section 6.0 "Flash Program Memory"** and **Section 7.0 "Data EEPROM Memory"**, respectively.

5.1 Program Memory Organization

A 21-bit program counter is capable of addressing the 2-Mbyte program memory space. Accessing a location between the physically implemented memory and the 2-Mbyte address will cause a read of all '0's (a NOP instruction).

The PIC18F1220 has 4 Kbytes of Flash memory and can store up to 2,048 single-word instructions.

The PIC18F1320 has 8 Kbytes of Flash memory and can store up to 4,096 single-word instructions.

The Reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

The program memory maps for the PIC18F1220 and PIC18F1320 devices are shown in Figure 5-1 and Figure 5-2, respectively.

FIGURE 5-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F1220

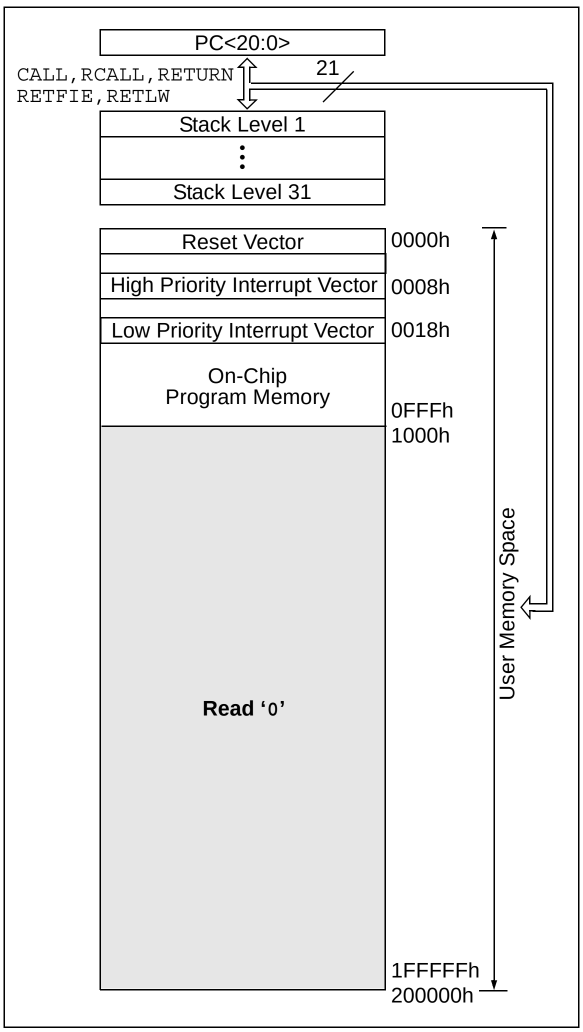
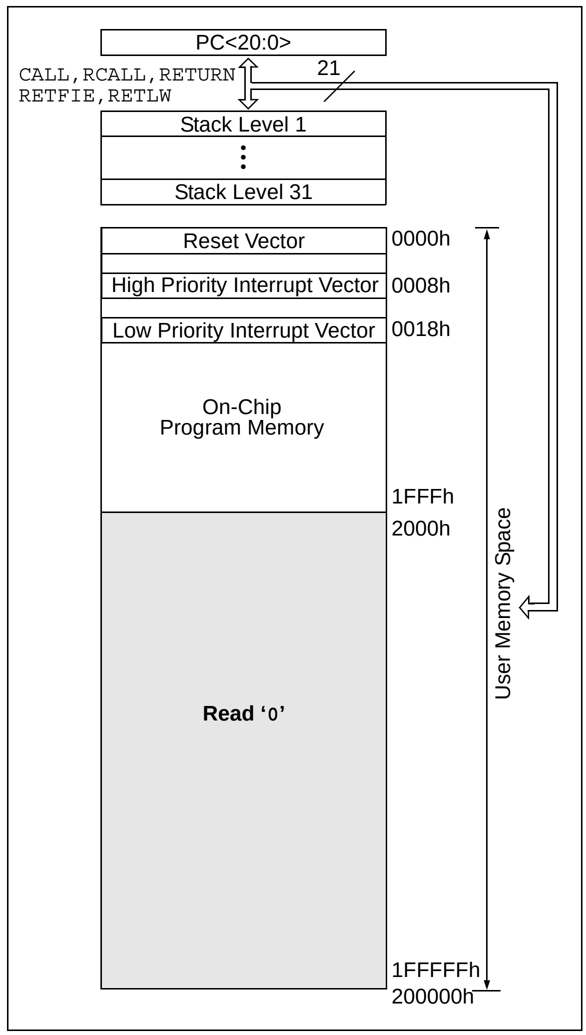


FIGURE 5-2: PROGRAM MEMORY MAP AND STACK FOR PIC18F1320



9.1 INTCON Registers

The INTCON registers are readable and writable registers, which contain various enable, priority and flag bits.

Note:	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. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R-0/0
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** Timer0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit(1)
1 = At least one of the RB<7:4> pins changed state (must be cleared in software)
0 = None of the RB<7:4> pins have changed state

Note 1: A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

9.2 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, 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 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1

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

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

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

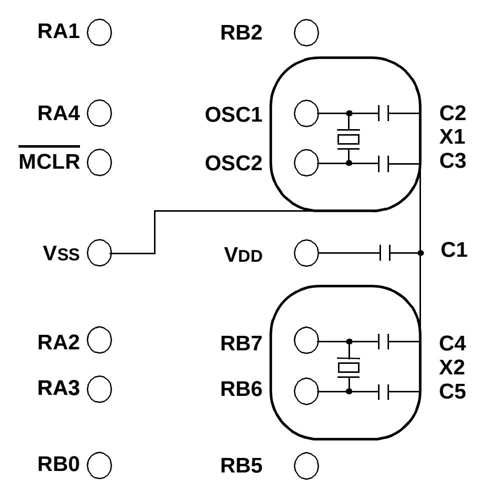
12.3 Timer1 Oscillator Layout Considerations

The Timer1 oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity.

The oscillator circuit, shown in Figure 12-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.

If a high-speed circuit must be located near the oscillator (such as the CCP1 pin in output compare or PWM mode, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as shown in Figure 12-4, may be helpful when used on a single sided PCB, or in addition to a ground plane.

FIGURE 12-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING



Note: Not drawn to scale.

12.4 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 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 Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

12.5 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 (see **Section 15.4.4 “Special Event Trigger”** for more information).

> **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.

12.6 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.

REGISTER 16-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R-0/0	R-0/0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don’t care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don’t care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, generates RCIF interrupt and loads RCREG when RX9D is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
Don’t care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.

16.2 EUSART Baud Rate Generator (BRG)

The BRG is a dedicated 8-bit or 16-bit generator, that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode; setting the BRG16 bit (BAUDCTL<3>) selects 16-bit mode.

The SPBRGH:SPBRG register pair controls the period of a free running timer. In Asynchronous mode, bits BRGH (TXSTA<2>) and BRG16 also control the baud rate. In Synchronous mode, bit BRGH is ignored. Table 16-1 shows the formula for computation of the baud rate for different EUSART modes which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRGH:SPBRG registers can be calculated using the formulas in Table 16-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 16-1. Typical baud rates and error values for the various asynchronous modes are shown in Table 16-2. It may be advantageous to use the high baud rate (BRGH = 1), or the 16-bit BRG to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGH:SPBRG registers causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

16.2.1 POWER MANAGED MODE OPERATION

The system clock is used to generate the desired baud rate; however, when a power managed mode is entered, the clock source may be operating at a different frequency than in PRI_RUN mode. In Sleep mode, no clocks are present and in PRI_IDLE mode, the primary clock source continues to provide clocks to the Baud Rate Generator; however, in other power managed modes, the clock frequency will probably change. This may require the value in SPBRG to be adjusted.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RCIDL bit and make sure that the receive operation is Idle before changing the system clock.

16.2.2 SAMPLING

The data on the RB4/AN6/RX/DT/KBI0 pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 16-1: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	FOSC/[64 (n + 1)]
0	0	1	8-bit/Asynchronous	FOSC/[16 (n + 1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	FOSC/[4 (n + 1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

EXAMPLE 16-1: 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%

The value in the ADRESH/ADRESL registers is not modified for a Power-on Reset. The ADRESH/ADRESL registers will contain unknown data after a Power-on Reset.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see **Section 17.1 "A/D Acquisition Requirements"**. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/$\overline{\text{DONE}}$ bit and the actual start of the conversion.

To do an A/D Conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON2)
   - Select A/D conversion clock (ADCON2)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

   OR

   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit, ADIF, if required.
7. For the next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as T_{AD}. A minimum wait of 2 T_{AD} is required before the next acquisition starts.

FIGURE 17-2: ANALOG INPUT MODEL

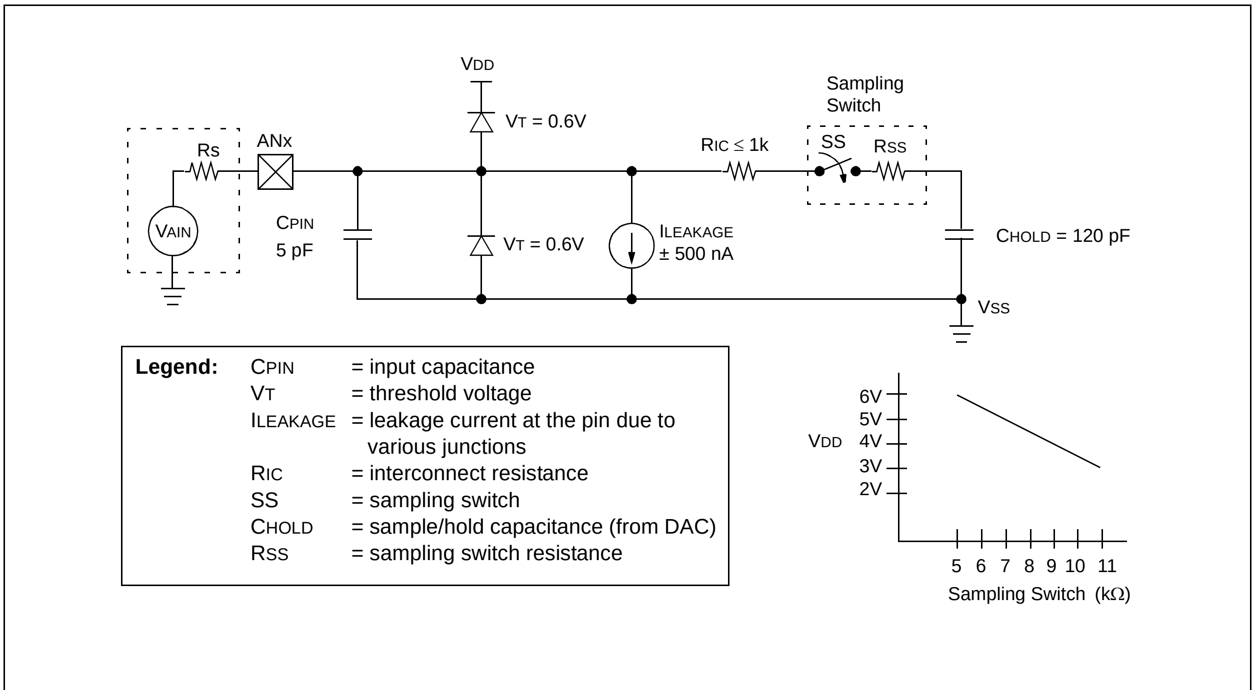


TABLE 19-2: SUMMARY OF WATCHDOG TIMER REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS2	WDTPS0	WDTEN
RCON	IPEN	—	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
WDTCON	—	—	—	—	—	—	—	SWDTEN

Legend: Shaded cells are not used by the Watchdog Timer.

19.3 Two-Speed Start-up

The Two-Speed Start-up feature helps to minimize the latency period from oscillator start-up to code execution by allowing the microcontroller to use the INTRC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO bit in Configuration Register 1H (CONFIG1H<7>).

Two-Speed Start-up is available only if the primary oscillator mode is LP, XT, HS or HSPLL (crystal-based modes). Other sources do not require an OST start-up delay; for these, Two-Speed Start-up is disabled.

When enabled, Resets and wake-ups from Sleep mode cause the device to configure itself to run from the internal oscillator block as the clock source, following the time-out of the Power-up Timer after a Power-on Reset is enabled. This allows almost immediate code execution while the primary oscillator starts and the OST is running. Once the OST times out, the device automatically switches to PRI_RUN mode.

Because the OSCCON register is cleared on Reset events, the INTOSC (or postscaler) clock source is not initially available after a Reset event; the INTRC clock is used directly at its base frequency. To use a higher clock speed on wake-up, the INTOSC or postscaler clock sources can be selected to provide a higher clock speed by setting bits, IFRC2:IFRC0, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting IFRC2:IFRC0 prior to entering Sleep mode.

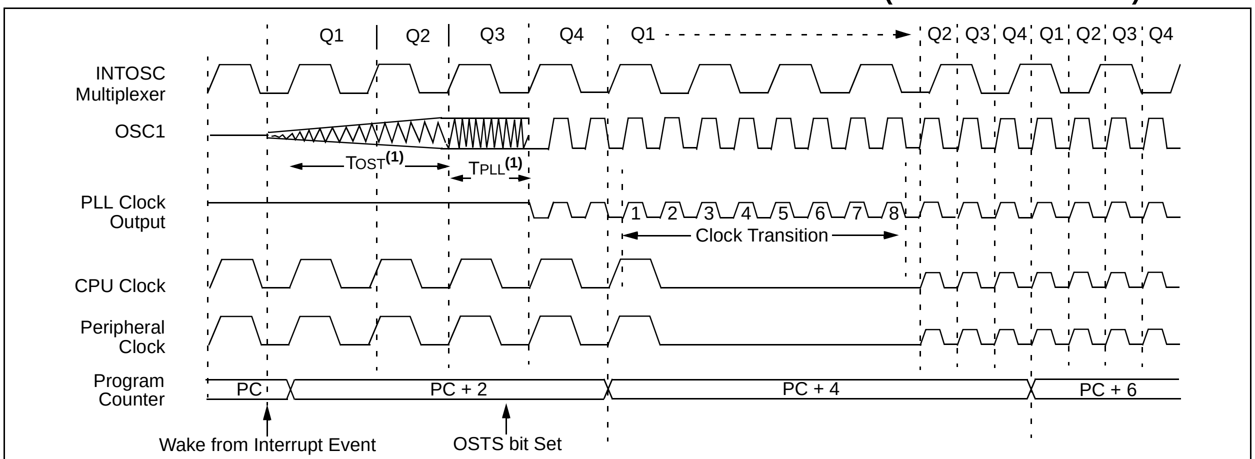
In all other power managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

19.3.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTRC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power managed modes, including serial SLEEP instructions (refer to **Section 3.1.3 "Multiple Sleep Commands"**). In practice, this means that user code can change the SCS1:SCS0 bit settings and issue SLEEP commands before the OST times out. This would allow an application to briefly wake-up, perform routine "housekeeping" tasks and return to Sleep before the device starts to operate from the primary oscillator.

User code can also check if the primary clock source is currently providing the system clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the system clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

FIGURE 19-2: TIMING TRANSITION FOR TWO-SPEED START-UP (INTOSC TO HSPLL)



Note 1: T_{OST} = 1024 T_{OSC}; T_{PLL} = 2 ms (approx). These intervals are not shown to scale.

19.5 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 Flash devices differs significantly from other PIC devices.

The user program memory is divided into three blocks. One of these is a boot block of 512 bytes. The remainder of the memory is divided into two blocks on binary boundaries.

Each of the three blocks has three protection bits associated with them. They are:

- Code-Protect bit (CPn)
- Write-Protect bit (WRTn)
- External Block Table Read bit (EBTRn)

Figure 19-5 shows the program memory organization for 4 and 8-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 19-3.

FIGURE 19-5: CODE-PROTECTED PROGRAM MEMORY FOR PIC18F1220/1320

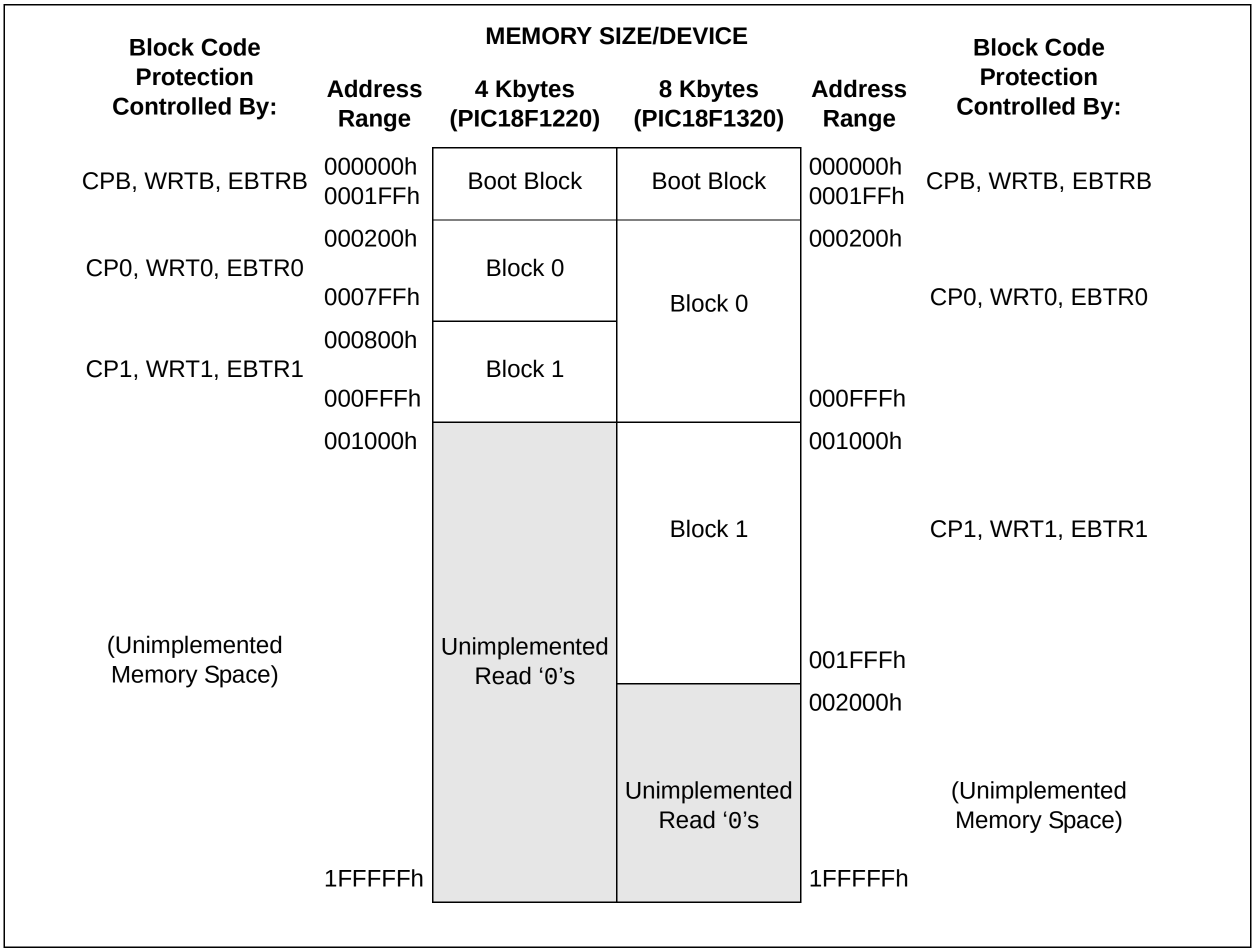


TABLE 19-3: SUMMARY OF CODE PROTECTION REGISTERS

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h	CONFIG5L	—	—	—	—	—	—	CP1	CP0
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—
30000Ah	CONFIG6L	—	—	—	—	—	—	WRT1	WRT0
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—
30000Ch	CONFIG7L	—	—	—	—	—	—	EBTR1	EBTR0
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—

Legend: Shaded cells are unimplemented.

BTFSC **Bit Test File, Skip if Clear**

Syntax:	[*label*] BTFSC f,b[,a]
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]
Operation:	skip if (f<b>) = 0
Status Affected:	None
Encoding:	1011 bbba ffff ffff
Description:	If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    BTFSC   FLAG, 1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

BTFSS **Bit Test File, Skip if Set**

Syntax:	[*label*] BTFSS f,b[,a]
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]
Operation:	skip if (f<b>) = 1
Status Affected:	None
Encoding:	1010 bbba ffff ffff
Description:	If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    BTFSS   FLAG, 1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)

INCFSZ	**Increment f, skip if 0**
Syntax:	[*label*] INCFSZ f [,d [,a]]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) + 1 → dest, skip if result = 0
Status Affected:	None
Encoding:	0011 | 11da | ffff | ffff
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE     INCFSZ   CNT
NZERO    :
ZERO     :
```

Before Instruction
PC = Address (HERE)

After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

INFSNZ	**Increment f, skip if not 0**
Syntax:	[*label*] INFSNZ f [,d [,a]]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) + 1 → dest, skip if result ≠ 0
Status Affected:	None
Encoding:	0100 | 10da | ffff | ffff
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE     INFSNZ REG
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)

After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

MOVLW **Move literal to W**

Syntax:	[*label*] MOVLW k
Operands:	$0 \le k \le 255$
Operation:	$k \rightarrow W$
Status Affected:	None
Encoding:	0000 1110 kkkk kkkk
Description:	The 8-bit literal 'k' is loaded into W.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `MOVLW 0x5A`

After Instruction

W = 0x5A

MOVWF **Move W to f**

Syntax:	[*label*] MOVWF f [,a]
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	$(W) \rightarrow f$
Status Affected:	None
Encoding:	0110 111a ffff ffff
Description:	Move data from W to register 'f'. 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 register 'f'

Example: `MOVWF REG`

Before Instruction

W = 0x4F
REG = 0xFF

After Instruction

W = 0x4F
REG = 0x4F

NEGF **Negate f**

Syntax:	[*label*] NEGF f [,a]
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	$(\bar{f}) + 1 \rightarrow f$
Status Affected:	N, OV, C, DC, Z
Encoding:	0110 110a ffff ffff
Description:	Location ‘f’ is negated using two’s complement. The result is placed in the data memory location ‘f’. 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.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write register ‘f’

Example: `NEGF REG, 1`

Before Instruction
```
REG = 0011 1010 [0x3A]
```
After Instruction
```
REG = 1100 0110 [0xC6]
```

NOP **No Operation**

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	0000 0000 0000 0000 1111 xxxx xxxx xxxx
Description:	No operation.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example:

None.

SUBWFB **Subtract W from f with Borrow**

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

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

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

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

Encoding:

0101	10da	ffff	ffff

Description: Subtract W and the Carry flag (borrow) from register 'f' (2's complement method). 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' is '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 1: `SUBWFB REG, 1, 0`

Before Instruction
REG = 0x19 (0001 1001)
W = 0x0D (0000 1101)
C = 0x01

After Instruction
REG = 0x0C (0000 1011)
W = 0x0D (0000 1101)
C = 0x01
Z = 0x00
N = 0x00 ; result is positive

Example 2: `SUBWFB REG, 0, 0`

Before Instruction
REG = 0x1B (0001 1011)
W = 0x1A (0001 1010)
C = 0x00

After Instruction
REG = 0x1B (0001 1011)
W = 0x00
C = 0x01
Z = 0x01 ; result is zero
N = 0x00

Example 3: `SUBWFB REG, 1, 0`

Before Instruction
REG = 0x03 (0000 0011)
W = 0x0E (0000 1101)
C = 0x01

After Instruction
REG = 0xF5 (1111 0100) ; [2's comp]
W = 0x0E (0000 1101)
C = 0x00
Z = 0x00
N = 0x01 ; result is negative

SWAPF **Swap f**

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

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

Operation: (f<3:0>) $\rightarrow$ dest<7:4>,
(f<7:4>) $\rightarrow$ dest<3:0>

Status Affected: None

Encoding:

0011	10da	ffff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default). 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 (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `SWAPF REG`

Before Instruction
REG = 0x53

After Instruction
REG = 0x35

TABLE 22-2: LOW-VOLTAGE DETECT CHARACTERISTICS (CONTINUED)

PIC18LF1220/1320 (Industrial)				**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
PIC18F1220/1320 (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.	**Symbol**	**Characteristic**		**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
D420F		LVD Voltage on VDD Transition High-to-Low		Industrial Low Voltage **(-40°C to -10°C)**				
		PIC18LF1220/1320	LVDL<3:0> = 0000	N/A	N/A	N/A	V	Reserved
			LVDL<3:0> = 0001	N/A	N/A	N/A	V	Reserved
			LVDL<3:0> = 0010	1.99	2.26	2.53	V	
			LVDL<3:0> = 0011	2.16	2.45	2.75	V	
			LVDL<3:0> = 0100	2.25	2.55	2.86	V	
			LVDL<3:0> = 0101	2.43	2.77	3.10	V	
			LVDL<3:0> = 0110	2.53	2.87	3.21	V	
			LVDL<3:0> = 0111	2.70	3.07	3.43	V	
			LVDL<3:0> = 1000	2.96	3.36	3.77	V	
			LVDL<3:0> = 1001	3.14	3.57	4.00	V	
			LVDL<3:0> = 1010	3.23	3.67	4.11	V	
			LVDL<3:0> = 1011	3.41	3.87	4.34	V	
			LVDL<3:0> = 1100	3.58	4.07	4.56	V	
			LVDL<3:0> = 1101	3.76	4.28	4.79	V	
			LVDL<3:0> = 1110	4.04	4.60	5.15	V	
		LVD Voltage on VDD Transition High-to-Low		Industrial **(-10°C to +85°C)**				
D420G		PIC18F1220/1320	LVDL<3:0> = 1101	3.93	4.28	4.62	V	
			LVDL<3:0> = 1110	4.23	4.60	4.96	V	
		LVD Voltage on VDD Transition High-to-Low		Industrial **(-40°C to -10°C)**				
D420H		PIC18F1220/1320	LVDL<3:0> = 1101	3.76	4.28	4.79	V	
			LVDL<3:0> = 1110	4.04	4.60	5.15	V	
		LVD Voltage on VDD Transition High-to-Low		Extended **(-10°C to +85°C)**				
D420J		PIC18F1220/1320	LVDL<3:0> = 1101	3.94	4.28	4.62	V	
			LVDL<3:0> = 1110	4.23	4.60	4.96	V	
		LVD Voltage on VDD Transition High-to-Low		Extended **(-40°C to -10°C, +85°C to +125°C)**				
D420K		PIC18F1220/1320	LVDL<3:0> = 1101	3.77	4.28	4.79	V	
			LVDL<3:0> = 1110	4.05	4.60	5.15	V	

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

† Production tested at TAMB = 25°C. Specifications over temperature limits ensured by characterization.

FIGURE 23-25: VOH vs. IOH OVER TEMPERATURE (-40°C TO +125°C), VDD = 5.0V

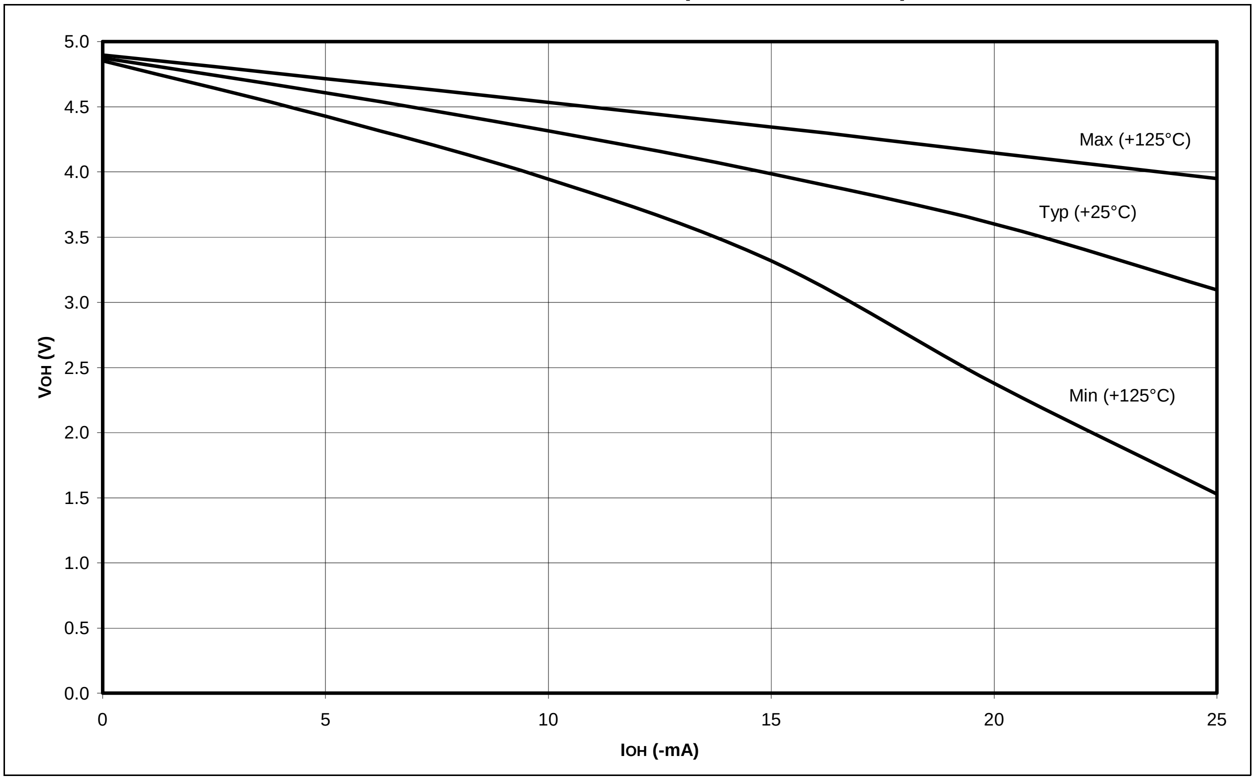
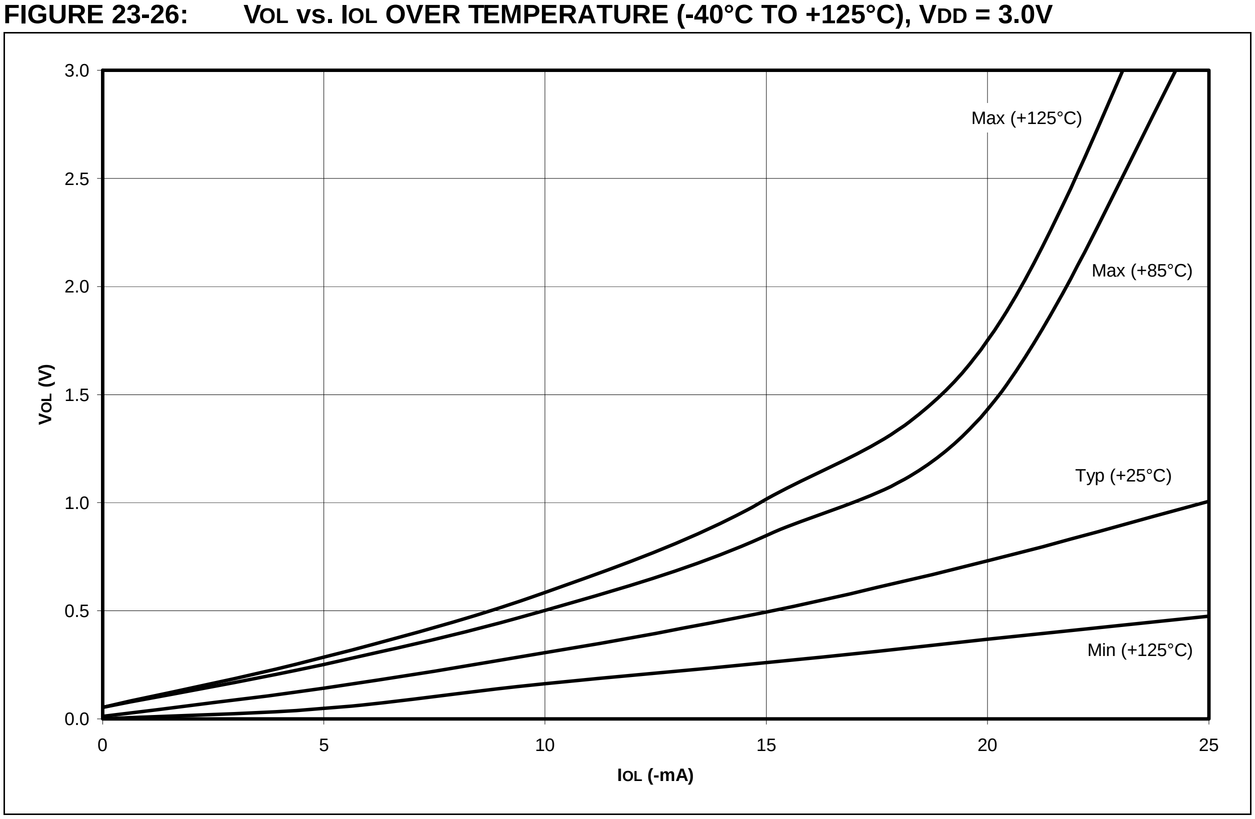


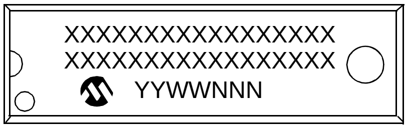
FIGURE 23-26: VOL vs. IOL OVER TEMPERATURE (-40°C TO +125°C), VDD = 3.0V



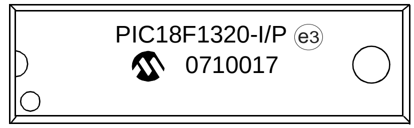
24.0 PACKAGING INFORMATION

24.1 Package Marking Information

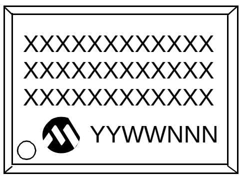
18-Lead PDIP



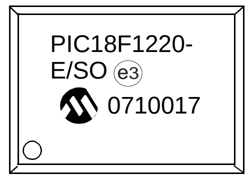
Example



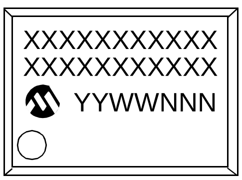
18-Lead SOIC



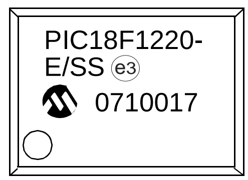
Example



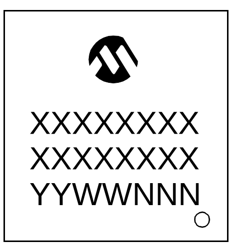
20-Lead SSOP



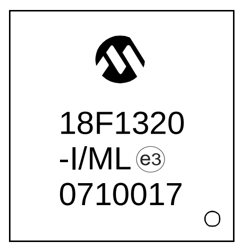
Example



28-Lead QFN



Example



Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	e3	Pb-free JEDEC designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

