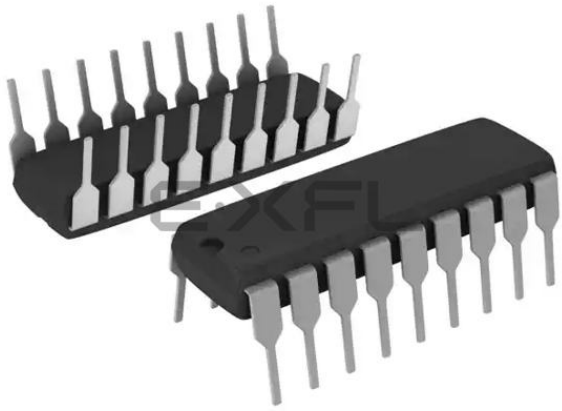


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	16
Program Memory Size	4KB (2K 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	Through Hole
Package / Case	18-DIP (0.300", 7.62mm)
Supplier Device Package	18-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f1220-i-p

3.4 Run Modes

If the IDLEN bit is clear when a SLEEP instruction is executed, the CPU and peripherals are both clocked from the source selected using the SCS1:SCS0 bits. While these operating modes may not afford the power conservation of Idle or Sleep modes, they do allow the device to continue executing instructions by using a lower frequency clock source. RC_RUN mode also offers the possibility of executing code at a frequency greater than the primary clock.

Wake-up from a power managed Run mode can be triggered by an interrupt, or any Reset, to return to full-power operation. As the CPU is executing code in Run modes, several additional exits from Run modes are possible. They include exit to Sleep mode, exit to a corresponding Idle mode and exit by executing a RESET instruction. While the device is in any of the power managed Run modes, a WDT time-out will result in a WDT Reset.

3.4.1 PRI_RUN MODE

The PRI_RUN mode is the normal full-power execution mode. If the SLEEP instruction is never executed, the microcontroller operates in this mode (a SLEEP instruction is executed to enter all other power managed modes). All other power managed modes exit to PRI_RUN mode when an interrupt or WDT time-out occur.

There is no entry to PRI_RUN mode. The OSTS bit is set. The IOFS bit may be set if the internal oscillator block is the primary clock source (see **Section 2.7.1 "Oscillator Control Register"**).

3.4.2 SEC_RUN MODE

The SEC_RUN mode is the compatible mode to the "clock switching" feature offered in other PIC18 devices. In this mode, the CPU and peripherals are clocked from the Timer1 oscillator. This gives users the option of lower power consumption while still using a high accuracy clock source.

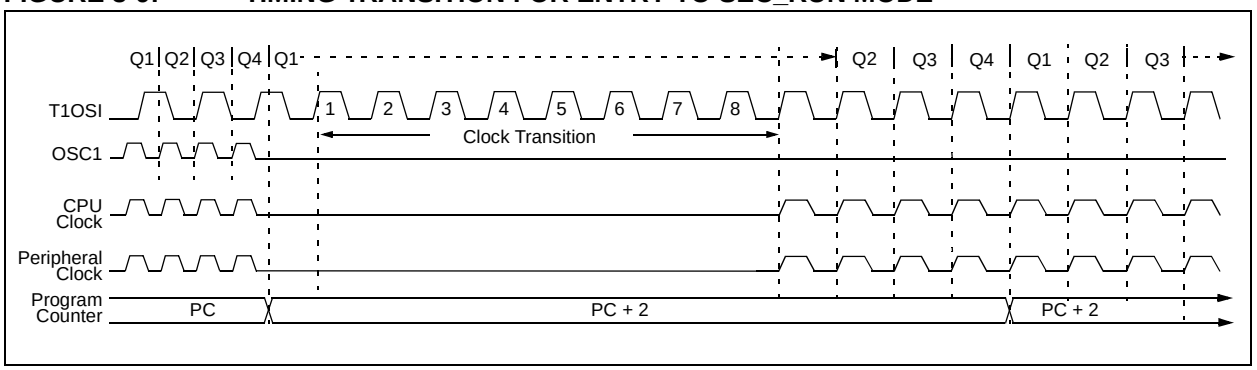
SEC_RUN mode is entered by clearing the IDLEN bit, setting SCS1:SCS0 = 01 and executing a SLEEP instruction. The system clock source is switched to the Timer1 oscillator (see Figure 3-9), the primary oscillator is shut down, the T1RUN bit (T1CON<6>) is set and the OSTS bit is cleared.

Note:	The Timer1 oscillator should already be running prior to entering SEC_RUN mode. If the T1OSCEN bit is not set when the SLEEP instruction is executed, the SLEEP instruction will be ignored and entry to SEC_RUN mode will not occur. If the Timer1 oscillator is enabled, but not yet running, system clocks will be delayed until the oscillator has started; in such situations, initial oscillator operation is far from stable and unpredictable operation may result.

When a wake event occurs, the peripherals and CPU continue to be clocked from the Timer1 oscillator while the primary clock is started. When the primary clock becomes ready, a clock switchback to the primary clock occurs (see Figure 3-6). When the clock switch is complete, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the system clock. The IDLEN and SCS bits are not affected by the wake-up. The Timer1 oscillator continues to run.

Firmware can force an exit from SEC_RUN mode. By clearing the T1OSCEN bit (T1CON<3>), an exit from SEC_RUN back to normal full-power operation is triggered. The Timer1 oscillator will continue to run and provide the system clock, even though the T1OSCEN bit is cleared. The primary clock is started. When the primary clock becomes ready, a clock switchback to the primary clock occurs (see Figure 3-6). When the clock switch is complete, the Timer1 oscillator is disabled, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the system clock. The IDLEN and SCS bits are not affected by the wake-up.

FIGURE 3-9: TIMING TRANSITION FOR ENTRY TO SEC_RUN MODE



5.5 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 5-4.

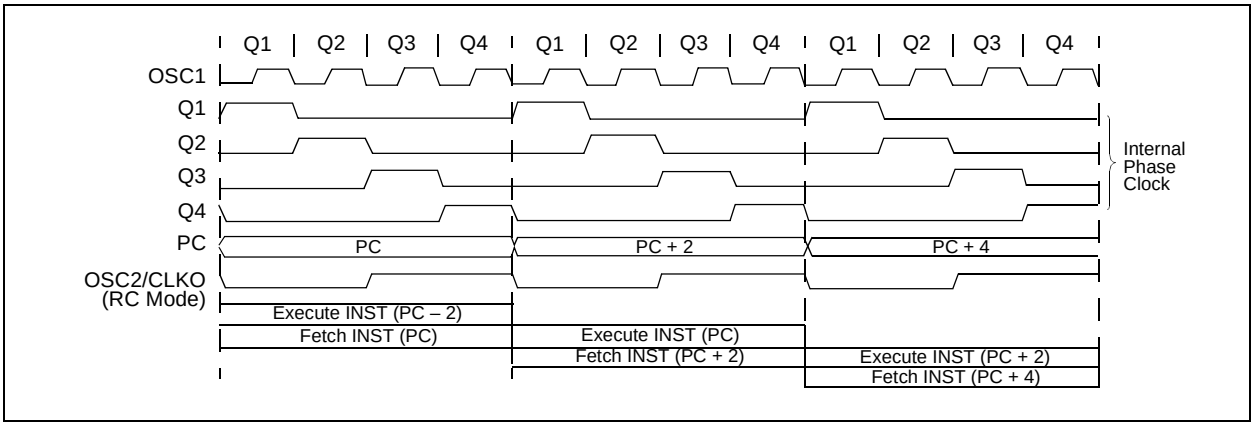
5.6 Instruction Flow/Pipelining

An “Instruction Cycle” consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle, while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), then two cycles are required to complete the instruction (Example 5-2).

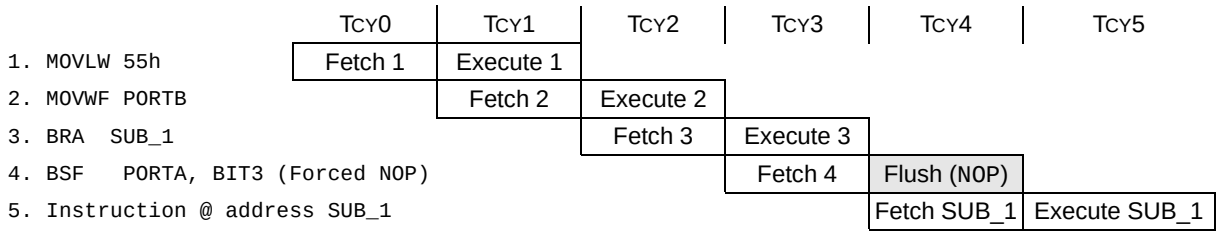
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the “Instruction Register” (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 5-4: CLOCK/INSTRUCTION CYCLE



EXAMPLE 5-2: INSTRUCTION PIPELINE FLOW



All instructions are single cycle, except for any program branches. These take two cycles, since the fetch instruction is “flushed” from the pipeline, while the new instruction is being fetched and then executed.

TABLE 5-2: REGISTER FILE SUMMARY (PIC18F1220/1320)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on Page:
TOSU	—	—	—	Top-of-Stack Upper Byte (TOS<20:16>)					---0 0000	34, 40
TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	34, 40
TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	34, 40
STKPTR	STKFUL	STKUNF	—	Return Stack Pointer					00-0 0000	34, 41
PCLATU	—	—	bit 21(3)	Holding Register for PC<20:16>					---0 0000	34, 42
PCLATH	Holding Register for PC<15:8>								0000 0000	34, 42
PCL	PC Low Byte (PC<7:0>)								0000 0000	34, 42
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--00 0000	34, 58
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	34, 58
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	34, 58
TABLAT	Program Memory Table Latch								0000 0000	34, 58
PRODH	Product Register High Byte								xxxx xxxx	34, 68
PRODL	Product Register Low Byte								xxxx xxxx	34, 68
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	34, 72
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1	34, 73
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	34, 74
INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								N/A	34, 51
POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								N/A	34, 51
POSTDEC0	Uses contents of FSR0 to address data memory– value of FSR0 post-decremented (not a physical register)								N/A	34, 51
PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								N/A	34, 51
PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 offset by W (not a physical register)								N/A	34, 51
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High				---- 0000	34, 51
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	34, 51
WREG	Working Register								xxxx xxxx	34
INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								**N/A**	34, 51
POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								**N/A**	34, 51
POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								**N/A**	34, 51
PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								**N/A**	34, 51
PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 offset by W (not a physical register)								**N/A**	34, 51
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High				---- 0000	34, 51
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	34, 51
BSR	—	—	—	—	Bank Select Register				---- 0000	35, 50
INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								**N/A**	35, 51
POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								**N/A**	35, 51
POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								**N/A**	35, 51
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								**N/A**	35, 51
PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 offset by W (not a physical register)								**N/A**	35, 51
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High				---- 0000	35, 51
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	35, 51
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	35, 53
TMR0H	Timer0 Register High Byte								0000 0000	35, 97
TMR0L	Timer0 Register Low Byte								xxxx xxxx	35, 97
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	35, 95
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0	0000 q000	35, 16
LVDCON	—	—	IVRST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	35, 162
WDTCON	—	—	—	—	—	—	—	SWDTEN	--- ---0	35, 174
RCON	IPEN	—	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	0--1 11q0	33, 54, 81

Legend: x = unknown, u = unchanged, – = unimplemented, q = value depends on condition

Note 1: RA6 and associated bits are configured as port pins in RCIO, ECIO and INTIO2 (with port function on RA6) Oscillator mode only and read '0' in all other oscillator modes.

2: RA7 and associated bits are configured as port pins in INTIO2 Oscillator mode only and read '0' in all other modes.

3: Bit 21 of the PC is only available in Test mode and Serial Programming modes.

4: The RA5 port bit is only available when MCLRE fuse (CONFIG3H<7>) is programmed to '0'. Otherwise, RA5 reads '0'. This bit is read-only.

EXAMPLE 6-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)

```
WRITE_WORD_TO_HREGS
            MOVF    POSTINC0, W             ; get low byte of buffer data and increment FSR0
            MOVWF   TABLAT                  ; present data to table latch
            TBLWT+*                         ; short write
                                            ; to internal TBLWT holding register, increment
                                              TBLPTR
            DECFSZ  COUNTER                 ; loop until buffers are full
            GOTO    WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
            BCF     INTCON, GIE             ; disable interrupts
            MOVLW   55h                     ; required sequence
            MOVWF   EECON2                  ; write 55H
            MOVLW   AAh
            MOVWF   EECON2                  ; write AAH
            BSF     EECON1, WR              ; start program (CPU stall)
            NOP
            BSF     INTCON, GIE             ; re-enable interrupts
            DECFSZ  COUNTER_HI              ; loop until done
            GOTO PROGRAM_LOOP
            BCF     EECON1, WREN            ; disable write to memory
```

6.5.2 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

6.5.3 UNEXPECTED TERMINATION OF WRITE OPERATION

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. The WRERR bit is set when a write operation is interrupted by a MCLR Reset, or a WDT Time-out Reset during normal operation. In these situations, users can check the WRERR bit and rewrite the location.

6.6 Flash Program Operation During Code Protection

See **Section 19.0 “Special Features of the CPU”** for details on code protection of Flash program memory.

TABLE 6-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY

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
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--00 0000	--00 0000
TBPLTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	0000 0000
TBLPTRL	Program Memory Table Pointer High Byte (TBLPTR<7:0>)								0000 0000	0000 0000
TABLAT	Program Memory Table Latch								0000 0000	0000 0000
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
EECON2	EEPROM Control Register 2 (not a physical register)								—	—
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	uu-0 u000
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	TMR3IP	—	1--1 -11-	1--1 -11-
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	TMR3IF	—	0--0 -00-	0--0 -00-
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	TMR3IE	—	0--0 -00-	0--0 -00-

Legend: x = unknown, u = unchanged, – = unimplemented, read as ‘0’.
Shaded cells are not used during Flash/EEPROM access.

10.2 PORTB, TRISB and LATB Registers

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATB) is also memory mapped. Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

EXAMPLE 10-2: INITIALIZING PORTB

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0x70    ; Set RB0, RB1, RB4 as
MOVWF   ADCON1  ; digital I/O pins
MOVLW   0xCF    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

Pins RB0-RB2 are multiplexed with INT0-INT2; pins RB0, RB1 and RB4 are multiplexed with A/D inputs; pins RB1 and RB4 are multiplexed with EUSART; and pins RB2, RB3, RB6 and RB7 are multiplexed with ECCP.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit, RBPU (INTCON2<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Note:	On a Power-on Reset, RB4:RB0 are configured as analog inputs by default and read as '0'; RB7:RB5 are configured as digital inputs.

Four of the PORTB pins (RB7:RB4) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

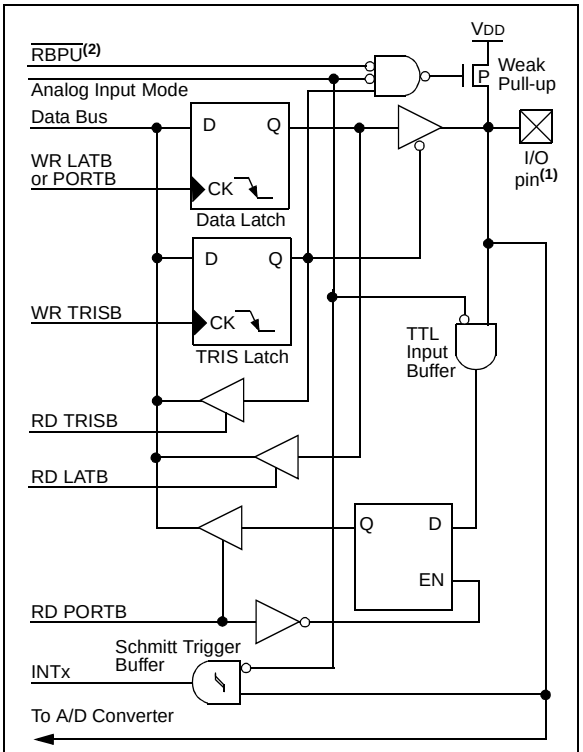
This interrupt can wake the device from Sleep. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB (except with the MOVFF instruction). This will end the mismatch condition.
b) Clear flag bit, RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB will end the mismatch condition and allow flag bit, RBIF, to be cleared.

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

FIGURE 10-7: BLOCK DIAGRAM OF RB0/AN4/INT0 PIN



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2<7>).

12.7 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 12.2 "Timer1 Oscillator"**, above), gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or supercapacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, RTCisr, shown in Example 12-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow, triggers the interrupt and calls the routine, which increments the seconds counter by one; additional counters for minutes and hours are incremented as the previous counter overflow.

Since the register pair is 16 bits wide, counting up to overflow the register directly from a 32.768 kHz clock would take two seconds. To force the overflow at the required one-second intervals, it is necessary to preload it; the simplest method is to set the MSb of TMR1H with a BSF instruction. Note that the TMR1L register is never preloaded or altered; doing so may introduce cumulative error over many cycles.

For this method to be accurate, Timer1 must operate in Asynchronous mode and the Timer1 overflow interrupt must be enabled (PIE1<0> = 1), as shown in the routine, RTCinit. The Timer1 oscillator must also be enabled and running at all times.

EXAMPLE 12-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE

```
RTCinit
            MOVLW   0x80              ; Preload TMR1 register pair
            MOVWF   TMR1H             ; for 1 second overflow
            CLRF    TMR1L
            MOVLW   b'00001111'       ; Configure for external clock,
            MOVWF   T1OSC             ; Asynchronous operation, external oscillator
            CLRF    secs              ; Initialize timekeeping registers
            CLRF    mins              ;
            MOVLW   .12
            MOVWF   hours
            BSF     PIE1, TMR1IE      ; Enable Timer1 interrupt
            RETURN
RTCisr
            BSF     TMR1H, 7          ; Preload for 1 sec overflow
            BCF     PIR1, TMR1IF      ; Clear interrupt flag
            INCF    secs, F           ; Increment seconds
            MOVLW   .59               ; 60 seconds elapsed?
            CPFSGT  secs
            RETURN                    ; No, done
            CLRF    secs              ; Clear seconds
            INCF    mins, F           ; Increment minutes
            MOVLW   .59               ; 60 minutes elapsed?
            CPFSGT  mins
            RETURN                    ; No, done
            CLRF    mins              ; clear minutes
            INCF    hours, F          ; Increment hours
            MOVLW   .23               ; 24 hours elapsed?
            CPFSGT  hours
            RETURN                    ; No, done
            MOVLW   .01               ; Reset hours to 1
            MOVWF   hours
            RETURN                    ; Done
```

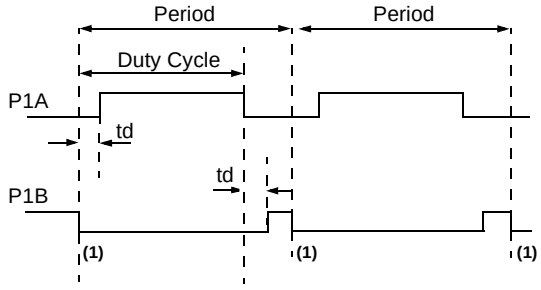
15.5.4 HALF-BRIDGE MODE

In the Half-Bridge Output mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the RB3/CCP1/P1A pin, while the complementary PWM output signal is output on the RB2/P1B/INT2 pin (Figure 15-6). This mode can be used for half-bridge applications, as shown in Figure 15-7, or for full-bridge applications, where four power switches are being modulated with two PWM signals.

In Half-Bridge Output mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of bits, PDC6:PDC0 (PWM1CON<6:0>), sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See **Section 15.5.6 "Programmable Dead-Band Delay"** for more details of the dead-band delay operations.

The TRISB<3> and TRISB<2> bits must be cleared to configure P1A and P1B as outputs.

FIGURE 15-6: HALF-BRIDGE PWM OUTPUT (ACTIVE-HIGH)

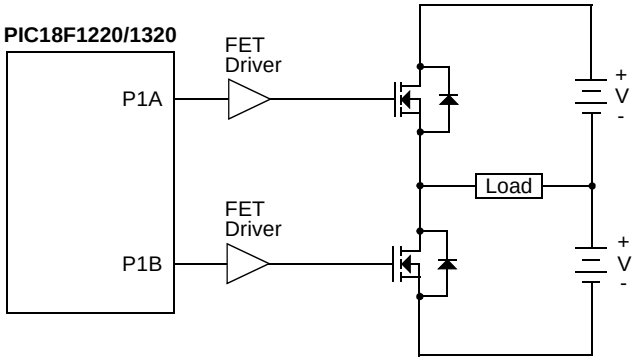


td = Dead-Band Delay

Note 1: At this time, the TMR2 register is equal to the PR2 register.

FIGURE 15-7: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS

Standard Half-Bridge Circuit ("Push-Pull")



Half-Bridge Output Driving a Full-Bridge Circuit

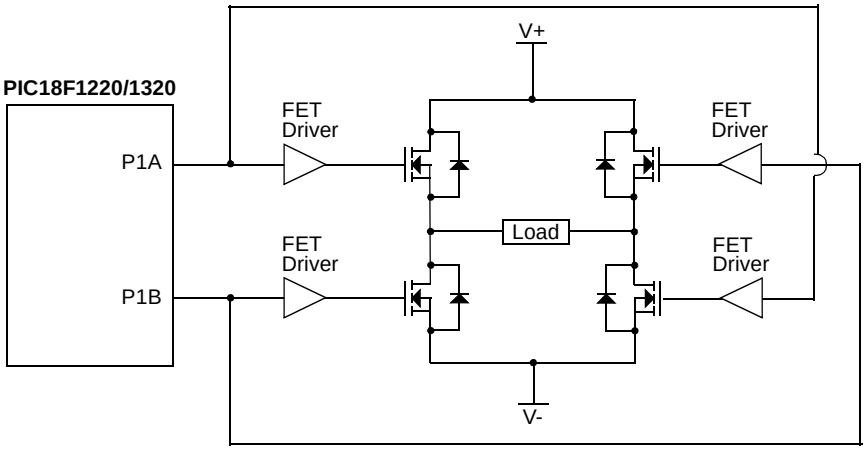


TABLE 16-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE (K)	SYNC = 0, BRGH = 1, 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)
2.4	—	—	—	—	—	—	2.441	1.73	255	2403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, 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	—	—	—	—	—	—	300	-0.16	207
1.2	1.202	0.16	207	1201	-0.16	103	1201	-0.16	51
2.4	2.404	0.16	103	2403	-0.16	51	2403	-0.16	25
9.6	9.615	0.16	25	9615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1											
	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	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082	300	-0.04	1665
1.2	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520	1201	-0.16	415
2.4	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259	2403	-0.16	207
9.6	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64	9615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1								
	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	0.04	832	300	-0.16	415	300	-0.16	207
1.2	1.202	0.16	207	1201	-0.16	103	1201	-0.16	51
2.4	2.404	0.16	103	2403	-0.16	51	2403	-0.16	25
9.6	9.615	0.16	25	9615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

16.3.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 16-5. The data is received on the RB4/AN6/RX/DT/KBI0 pin and drives the data recovery block. The data recovery block is actually a high-speed shifter, operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

To set up an Asynchronous Reception:

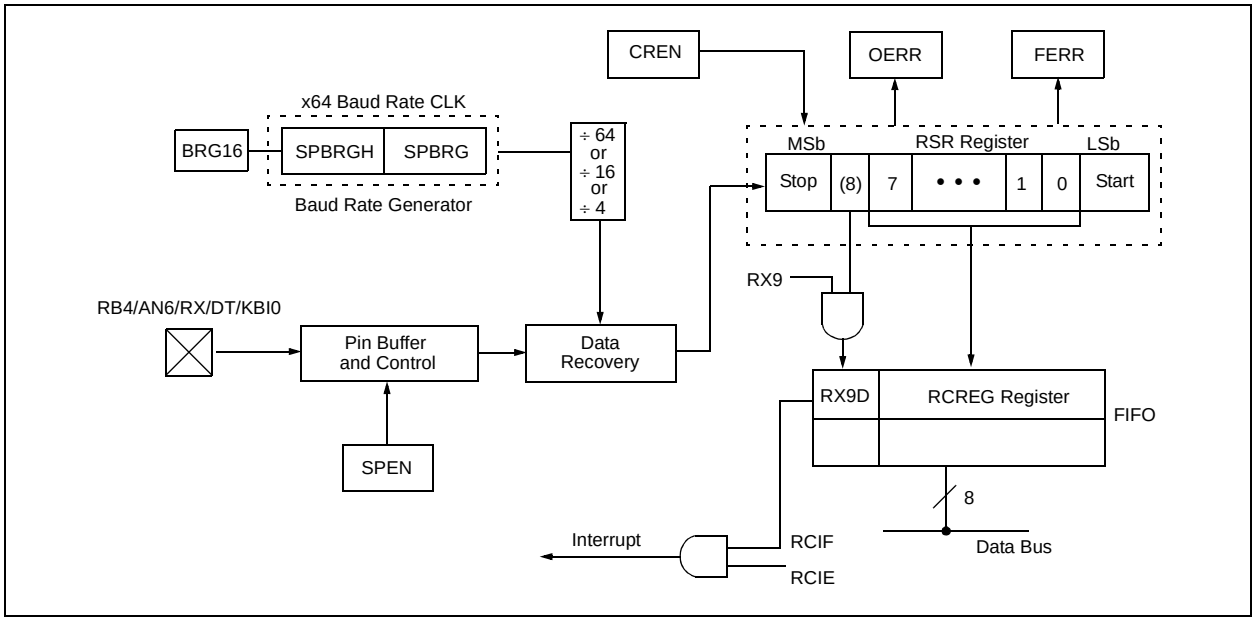
1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit RCIE.
4. If 9-bit reception is desired, set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit, RCIF, will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
7. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

16.3.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCIE and GIE bits are set.
8. Read the RCSTA register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREG to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 16-5: EUSART RECEIVE BLOCK DIAGRAM



REGISTER 17-1: ADCON0: A/D CONTROL REGISTER 0

R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
VCFG1	VCFG0	—	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
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 7-6 **VCFG<1:0>:** Voltage Reference Configuration bits

	A/D VREF+	**A/D VREF-**
00	AVDD	AVSS
01	External VREF+	AVSS
10	AVDD	External VREF-
11	External VREF+	External VREF-

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

bit 4-2 **CHS<2:0>:** Analog Channel Select bits

000 = Channel 0 (AN0)
001 = Channel 1 (AN1)
010 = Channel 2 (AN2)
011 = Channel 3 (AN3)
100 = Channel 4 (AN4)
101 = Channel 5 (AN5)
110 = Channel 6 (AN6)
111 = Unimplemented(1)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit

1 = A/D converter module is enabled
0 = A/D converter module is disabled

Performing a conversion on unimplemented channels returns full-scale results.

Note 1: Performing a conversion on unimplemented channels returns full-scale results.

19.0 SPECIAL FEATURES OF THE CPU

PIC18F1220/1320 devices include several features intended to maximize system reliability, minimize cost through elimination of external components and offer code protection. These are:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Fail-Safe Clock Monitor
- Two-Speed Start-up
- Code Protection
- ID Locations
- In-Circuit Serial Programming

Several oscillator options are available to allow the part to fit the application. The RC oscillator option saves system cost, while the LP crystal option saves power. These are discussed in detail in **Section 2.0 "Oscillator Configurations"**.

A complete discussion of device Resets and interrupts is available in previous sections of this data sheet.

In addition to their Power-up and Oscillator Start-up Timers provided for Resets, PIC18F1220/1320 devices have a Watchdog Timer, which is either permanently enabled via the Configuration bits, or software controlled (if configured as disabled).

The inclusion of an internal RC oscillator also provides the additional benefits of a Fail-Safe Clock Monitor (FSCM) and Two-Speed Start-up. FSCM provides for background monitoring of the peripheral clock and automatic switchover in the event of its failure. Two-Speed Start-up enables code to be executed almost immediately on start-up, while the primary clock source completes its start-up delays.

All of these features are enabled and configured by setting the appropriate Configuration register bits.

19.1 Configuration Bits

The Configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1'), to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh), which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The EECON1 register WR bit starts a self-timed write to the Configuration register. In normal operation mode, a TBLWT instruction, with the TBLPTR pointing to the Configuration register, sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a TBLWT instruction can write a '1' or a '0' into the cell. For additional details on Flash programming, refer to **Section 6.5 "Writing to Flash Program Memory"**.

TABLE 19-1: CONFIGURATION BITS AND DEVICE IDS

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300001h	CONFIG1H	IESO	FSCM	—	—	FOSC3	FOSC2	FOSC1	FOSC0	11-- 1111
300002h	CONFIG2L	—	—	—	—	BORV1	BORV0	BOR	$\overline{\text{PWRTEN}}$	---- 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDT	---1 1111
300005h	CONFIG3H	MCLRE	—	—	—	—	—	—	—	1--- ----
300006h	CONFIG4L	$\overline{\text{DEBUG}}$	—	—	—	—	LVP	—	STVR	1--- -1-1
300008h	CONFIG5L	—	—	—	—	—	—	CP1	CP0	---- --11
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	—	—	WRT1	WRT0	---- --11
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	—	—	EBTR1	EBTR0	---- --11
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1(1)	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	xxxx xxxx(1)
3FFFFFh	DEVID2(1)	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 0111

Legend: x = unknown, u = unchanged, – = unimplemented. Shaded cells are unimplemented, read as '0'.

Note 1: See Register 19-12 for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

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	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
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.

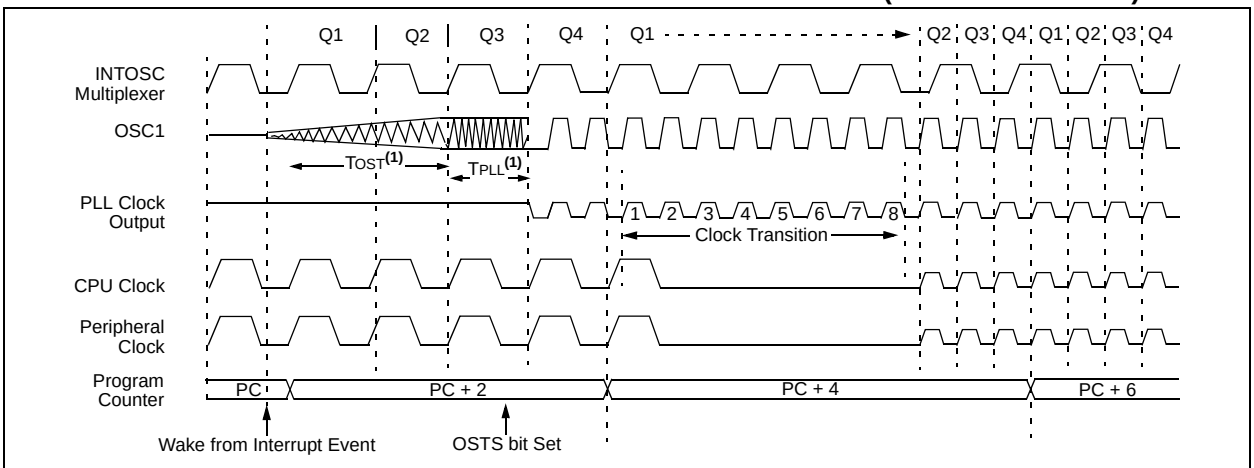
In all other power managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

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: TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

BCF **Bit Clear f**

Syntax: [*label*] BCF f,b[,a]

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

Operation: $0 \rightarrow f<b>$

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared. If 'a' is '0', the Access Bank 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: `BCF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0xC7
After Instruction
FLAG_REG = 0x47

BN **Branch if Negative**

Syntax: [*label*] BN n

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

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

Status Affected: None

Encoding:

1110	0110	nnnn	nnnn

Description: If the Negative bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

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

Example: `HERE BN Jump`

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

21.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICkit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits and Starter Kits
- Third-party development tools

21.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

21.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

21.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

22.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to V_{SS} (except V_{DD}, $\overline{MCLR}$ and RA4) ..... -0.3V to (V_{DD} + 0.3V)
Voltage on V_{DD} with respect to V_{SS} ..... -0.3V to +5.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS} **(Note 2)** ..... 0V to +13.25V
Voltage on RA4 with respect to Vss ..... 0V to +8.5V
Total power dissipation **(Note 1)** 1.0W
Maximum current out of V_{SS} pin ..... 300 mA
Maximum current into V_{DD} pin ..... 250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ±20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD}) ..... ±20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA
Maximum current sunk by all ports ..... 200 mA
Maximum current sourced by all ports ..... 200 mA

Note 1: Power dissipation is calculated as follows:
$Pdis = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

2: Voltage spikes below V_{SS} at the $\overline{MCLR}/V_{PP}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a “low” level to the $\overline{MCLR}/V_{PP}$ pin, rather than pulling this pin directly to V_{SS}.

† NOTICE: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (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.	**Device**	**Typ.**	**Max.**	**Units**	**Conditions**		
	Module Differential Currents (ΔIWDT, ΔIBOR, ΔILVD, ΔIOSCB, ΔIAD)						
D022 (ΔIWDT)	**Watchdog Timer**	1.5	4.0	µA	-40°C	VDD = 2.0V	
		2.2	4.0	µA	+25°C		
		3.1	5.0	µA	+85°C		
		2.5	6.0	µA	-40°C	VDD = 3.0V	
		3.3	6.0	µA	+25°C		
		4.7	7.0	µA	+85°C		
		3.7	10.0	µA	-40°C	VDD = 5.0V	
		4.5	10.0	µA	+25°C		
		6.1	13.0	µA	+85°C		
D022A (ΔIBOR)	**Brown-out Reset**	19	35.0	µA	-40°C to +85°C	VDD = 3.0V	
		24	45.0	µA	-40°C to +85°C	VDD = 5.0V	
D022B (ΔILVD)	**Low-Voltage Detect**	8.5	25.0	µA	-40°C to +85°C	VDD = 2.0V	
		16	35.0	µA	-40°C to +85°C	VDD = 3.0V	
		20	45.0	µA	-40°C to +85°C	VDD = 5.0V	
D025 (ΔIOSCB)	**Timer1 Oscillator**	1.7	3.5	µA	-40°C	VDD = 2.0V	32 kHz on Timer1(4)
		1.8	3.5	µA	+25°C		
		2.1	4.5	µA	+85°C		
		2.2	4.5	µA	-40°C	VDD = 3.0V	32 kHz on Timer1(4)
		2.6	4.5	µA	+25°C		
		2.8	5.5	µA	+85°C		
		3.0	6.0	µA	-40°C	VDD = 5.0V	32 kHz on Timer1(4)
		3.3	6.0	µA	+25°C		
		3.6	7.0	µA	+85°C		
D026 (ΔIAD)	**A/D Converter**	1.0	3.0	µA	-40°C to +85°C	VDD = 2.0V	A/D on, not converting
		1.0	4.0	µA	-40°C to +85°C	VDD = 3.0V	
		2.0	10.0	µA	-40°C to +85°C	VDD = 5.0V	
		1.0	8.0	µA	-40°C to +125°C	VDD = 5.0V	

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

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

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

23.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

Note: The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "Maximum" or "minimum" represents (mean + 3σ) or (mean – 3σ) respectively, where σ is a standard deviation, over the whole temperature range.

FIGURE 23-1: TYPICAL IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, +25°C

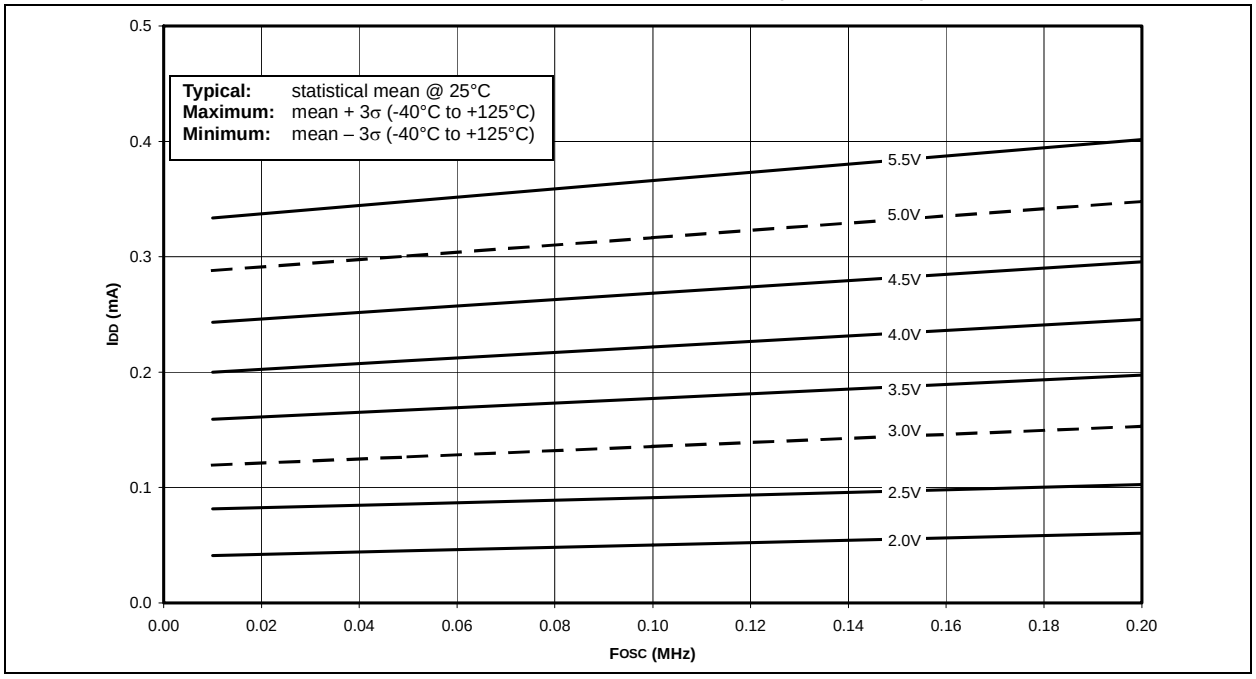


FIGURE 23-2: MAXIMUM IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, -40°C TO +85°C

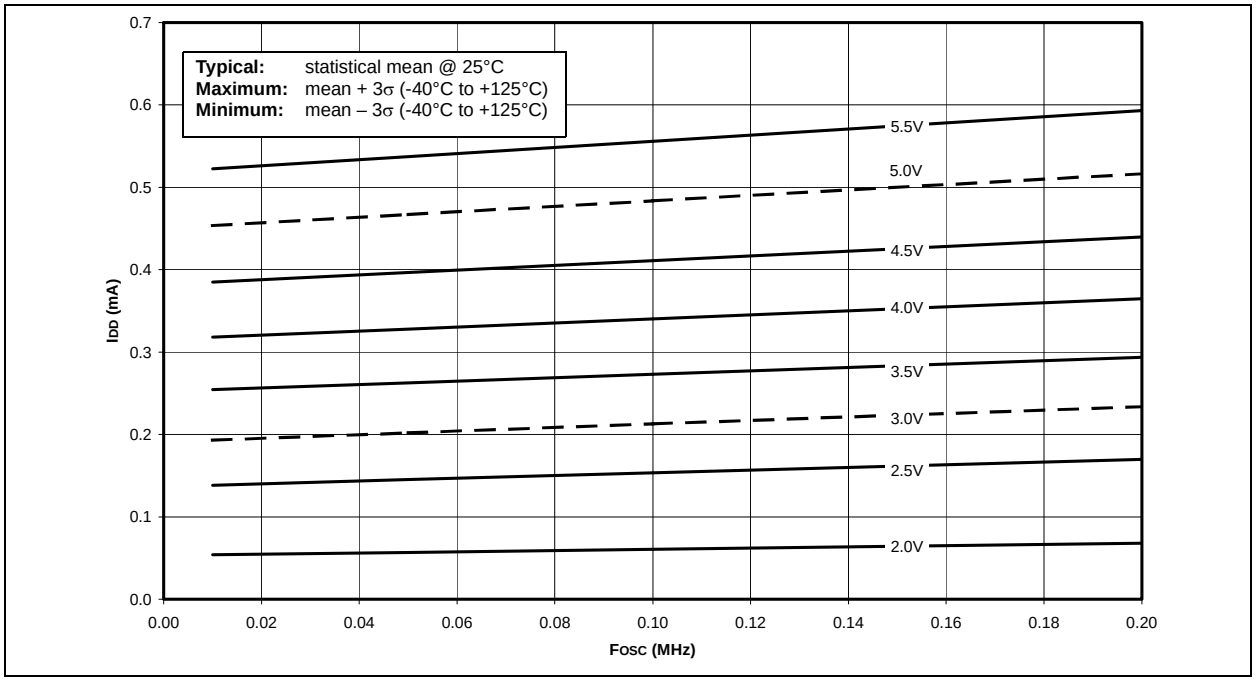


FIGURE 23-9: MAXIMUM IDD vs. FOSC OVER VDD PRI_IDLE, EC MODE, -40°C TO +85°C

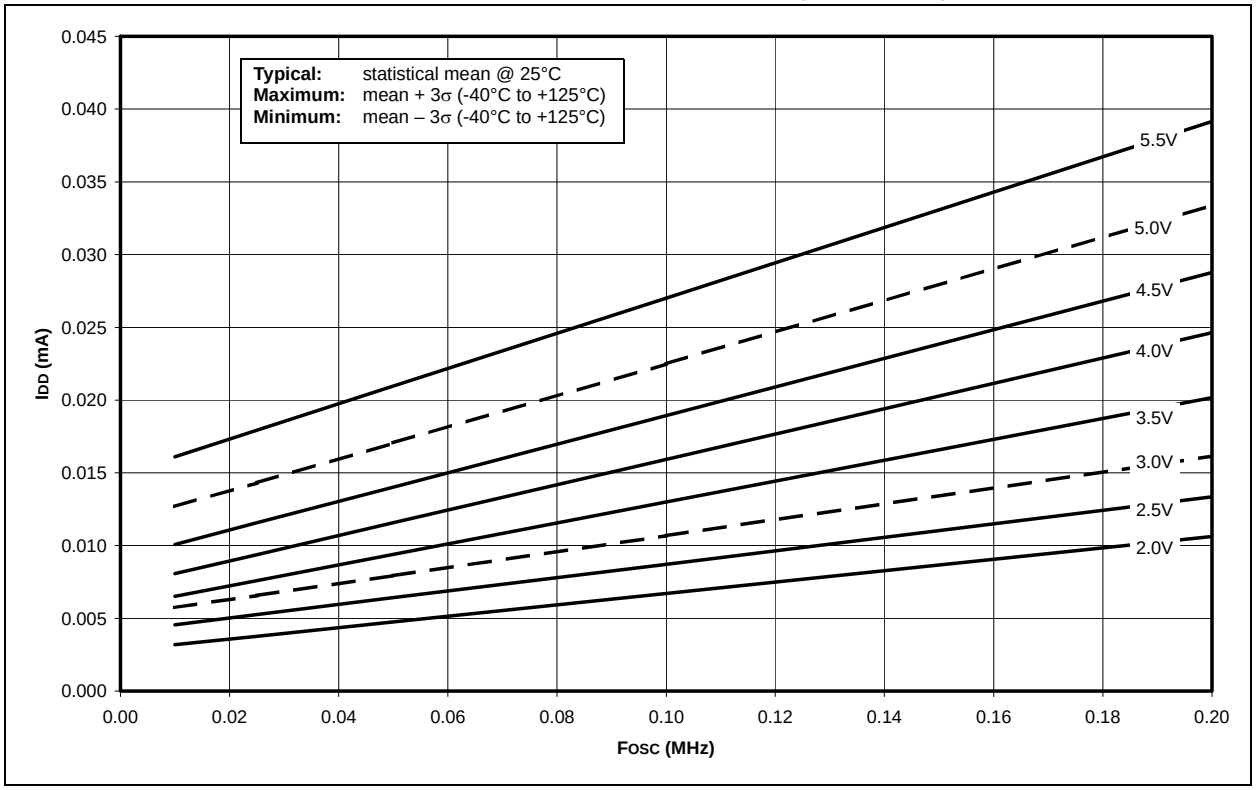


FIGURE 23-10: MAXIMUM IDD vs. FOSC OVER VDD PRI_IDLE, EC MODE, -40°C TO +125°C

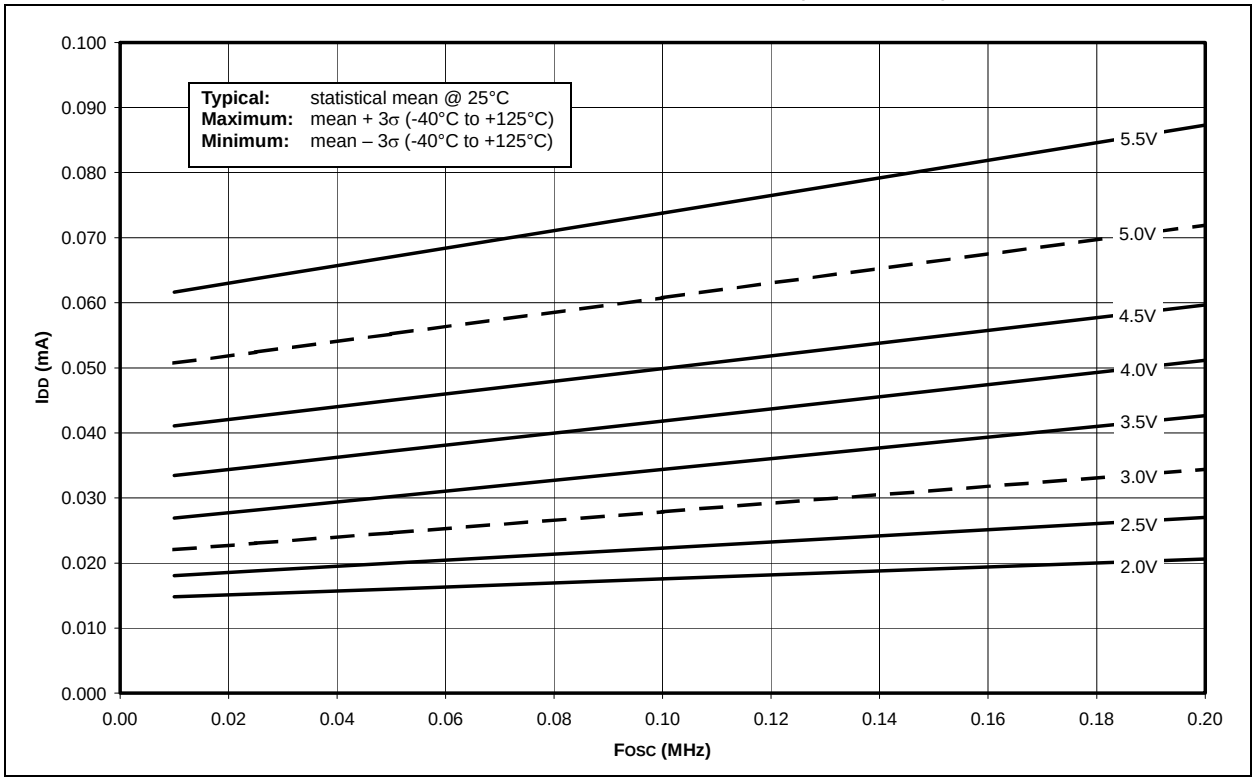


FIGURE 23-25: V_{OH} vs. I_{OH} OVER TEMPERATURE (-40°C TO +125°C), V_{DD} = 5.0V

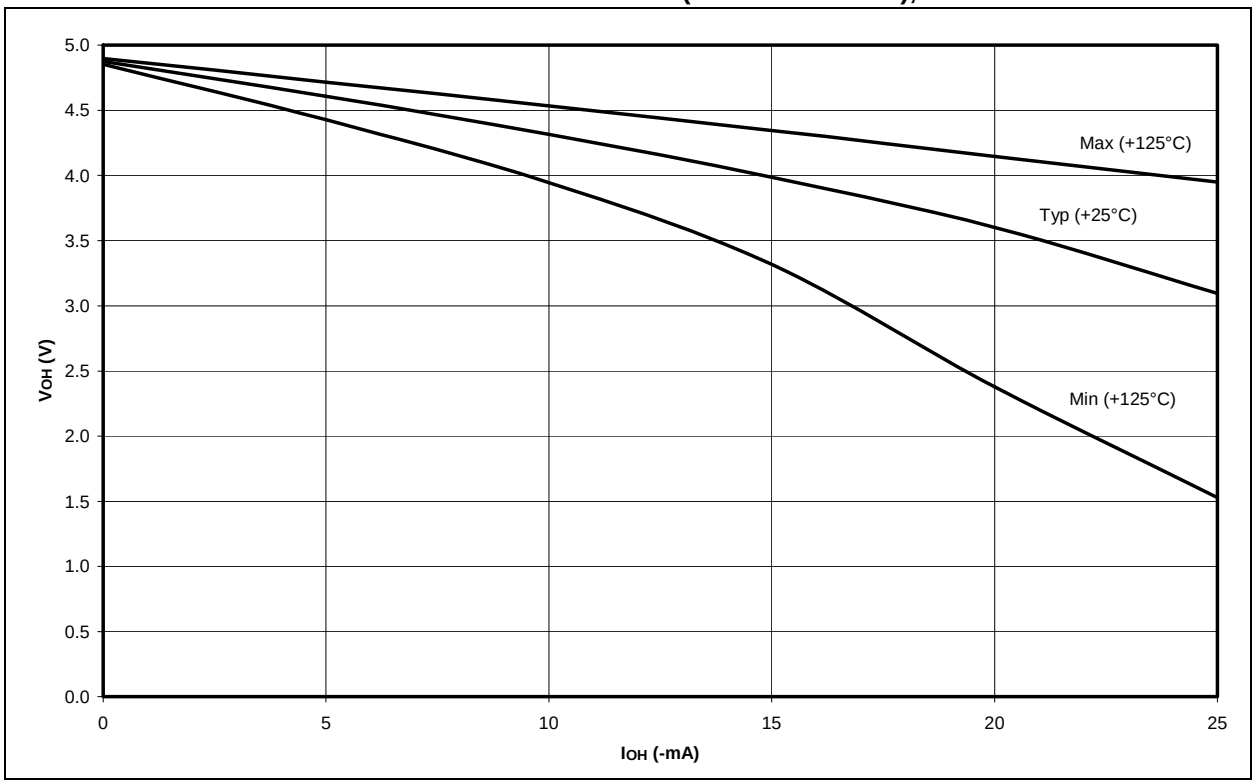


FIGURE 23-26: V_{OL} vs. I_{OL} OVER TEMPERATURE (-40°C TO +125°C), V_{DD} = 3.0V