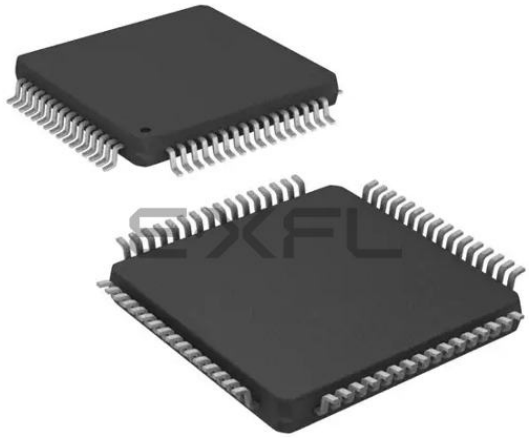


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

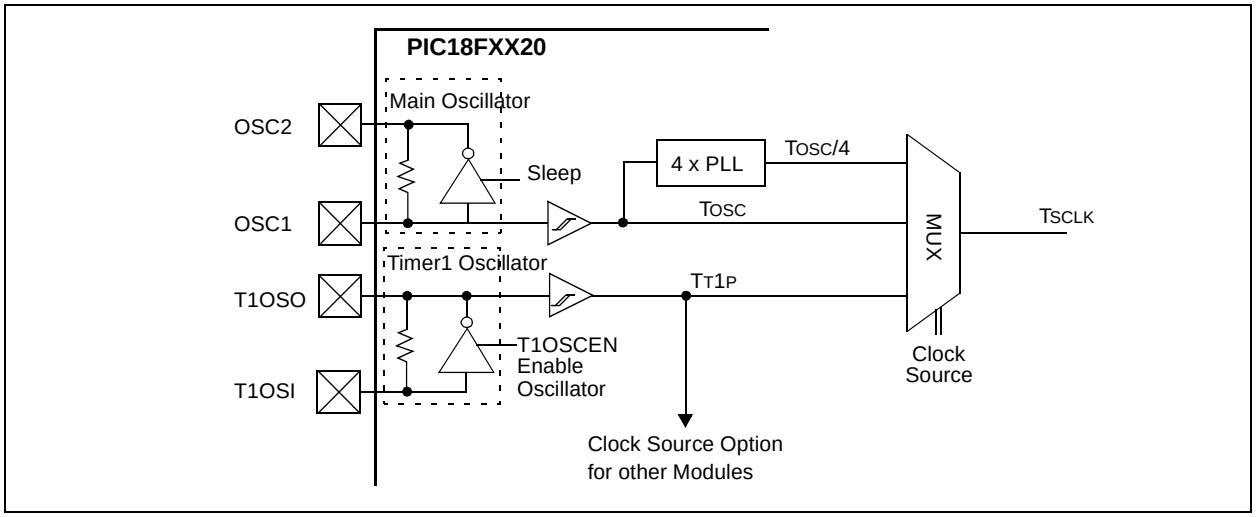
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	25MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	52
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.75K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f6720-i-pt

2.6 Oscillator Switching Feature

The PIC18FXX20 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low-frequency clock source. For the PIC18FXX20 devices, this alternate clock source is the Timer1 oscillator. If a low-frequency crystal (32 kHz, for example) has been attached to the Timer1 oscillator pins and the Timer1 oscillator has been enabled, the device can switch to a low-power execution mode. Figure 2-7 shows a block diagram of the system clock sources. The clock switching feature is enabled by programming the Oscillator Switching Enable (OSCSEN) bit in Configuration Register 1H to a '0'. Clock switching is disabled in an erased device. See **Section 12.0 "Timer1 Module"** for further details of the Timer1 oscillator. See **Section 23.0 "Special Features of the CPU"** for Configuration register details.

FIGURE 2-7: DEVICE CLOCK SOURCES



4.3 Fast Register Stack

A “fast interrupt return” option is available for interrupts. A Fast Register Stack is provided for the Status, WREG and BSR registers and is only one in depth. The stack is not readable or writable and is loaded with the current value of the corresponding register when the processor vectors for an interrupt. The values in the registers are then loaded back into the working registers, if the FAST RETURN instruction is used to return from the interrupt.

A low or high priority interrupt source will push values into the stack registers. If both low and high priority interrupts are enabled, the stack registers cannot be used reliably for low priority interrupts. If a high priority interrupt occurs while servicing a low priority interrupt, the stack register values stored by the low priority interrupt will be overwritten.

If high priority interrupts are not disabled during low priority interrupts, users must save the key registers in software during a low priority interrupt.

If no interrupts are used, the fast register stack can be used to restore the Status, WREG and BSR registers at the end of a subroutine call. To use the fast register stack for a subroutine call, a FAST CALL instruction must be executed.

Example 4-1 shows a source code example that uses the fast register stack.

EXAMPLE 4-1: FAST REGISTER STACK CODE EXAMPLE

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
        •
        •
SUB1    •
        •
        •
   RETURN FAST     ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

4.4 PCL, PCLATH and PCLATU

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide. The low byte is called the PCL register; this register is readable and writable. The high byte is called the PCH register. This register contains the PC<15:8> bits and is not directly readable or writable; updates to the PCH register may be performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits and is not directly readable or writable; updates to the PCU register may be performed through the PCLATU register.

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the LSB of the PCL is fixed to a value of ‘0’. The PC increments by 2 to address sequential instructions in the program memory.

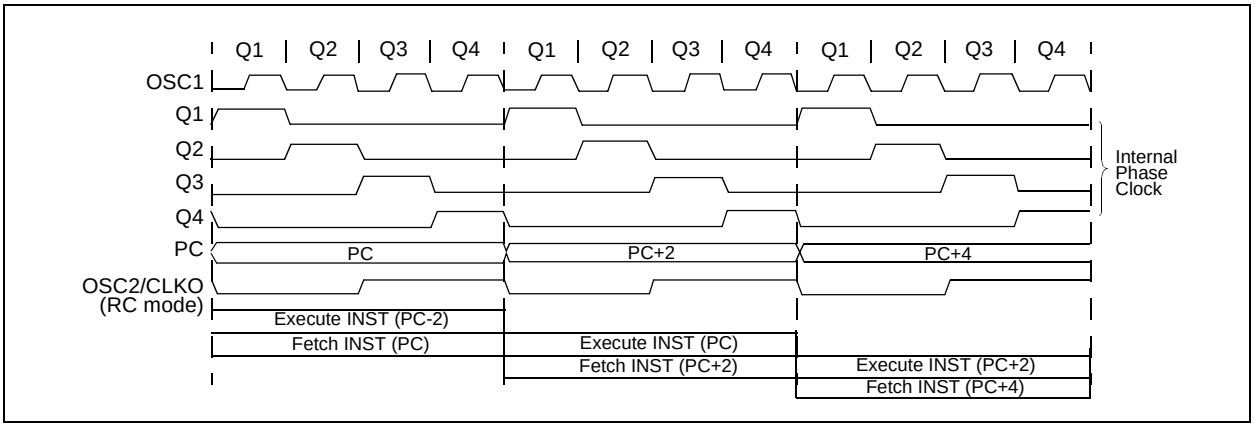
The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

The contents of PCLATH and PCLATU will be transferred to the program counter by an operation that writes PCL. Similarly, the upper two bytes of the program counter will be transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 4.8.1 “Computed GOTO”**).

4.5 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 4-4.

FIGURE 4-4: CLOCK/INSTRUCTION CYCLE



4.7.1 TWO-WORD INSTRUCTIONS

The PIC18FXX20 devices have four two-word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the 4 MSBs set to '1's and is a special kind of NOP instruction. The lower 12 bits of the second word contain data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two-word instruction is preceded by a conditional instruction that changes the PC. A program example that demonstrates this concept is shown in Example 4-3. Refer to **Section 24.0 "Instruction Set Summary"** for further details of the instruction set.

EXAMPLE 4-3: TWO-WORD INSTRUCTIONS

CASE 1:			
Object Code	**Source Code**		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; No, execute 2-word instruction
1111 0100 0101 0110			; 2nd operand holds address of REG2
0010 0100 0000 0000	ADDWF	REG3	; continue code
CASE 2:			
Object Code	**Source Code**		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; Yes
1111 0100 0101 0110			; 2nd operand becomes NOP
0010 0100 0000 0000	ADDWF	REG3	; continue code

4.8 Look-up Tables

Look-up tables are implemented two ways. These are:

- Computed GOTO
- Table Reads

4.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW 0xnn instructions. WREG is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW 0xnn instructions, that returns the value 0xnn to the calling function.

The offset value (value in WREG) specifies the number of bytes that the program counter should advance.

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

4.8.2 TABLE READS/TABLE WRITES

A better method of storing data in program memory allows 2 bytes of data to be stored in each instruction location.

Look-up table data may be stored 2 bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) specifies the byte address and the Table Latch (TABLAT) contains the data that is read from, or written to program memory. Data is transferred to/from program memory, one byte at a time.

A description of the table read/table write operation is shown in **Section 5.0 "Flash Program Memory"**.

5.2.2 TABLAT – TABLE LATCH REGISTER

The Table Latch (TABLAT) is an 8-bit register mapped into the SFR space. The Table Latch is used to hold 8-bit data during data transfers between program memory and data RAM.

5.2.3 TBLPTR – TABLE POINTER REGISTER

The Table Pointer (TBLPTR) addresses a byte within the program memory. The TBLPTR is comprised of three SFR registers: Table Pointer Upper Byte, Table Pointer High Byte and Table Pointer Low Byte (TBLPTRU:TBLPTRH:TBLPTRL). These three registers join to form a 22-bit wide pointer. The low-order 21 bits allow the device to address up to 2 Mbytes of program memory space. The 22nd bit allows access to the Device ID, the User ID and the configuration bits.

The Table Pointer, TBLPTR, is used by the `TBLRD` and `TBLWT` instructions. These instructions can update the TBLPTR in one of four ways, based on the table operation. These operations are shown in Table 5-1. These operations on the TBLPTR only affect the low-order 21 bits.

5.2.4 TABLE POINTER BOUNDARIES

TBLPTR is used in reads, writes and erases of the Flash program memory.

When a `TBLRD` is executed, all 22 bits of the Table Pointer determine which byte is read from program memory into TABLAT.

When a `TBLWT` is executed, the three LSbs of the Table Pointer (TBLPTR<2:0>) determine which of the eight program memory holding registers is written to. When the timed write to program memory (long write) begins, the 19 MSbs of the Table Pointer, TBLPTR (TBLPTR<21:3>), will determine which program memory block of 8 bytes is written to. For more detail, see **Section 5.5 “Writing to Flash Program Memory”**.

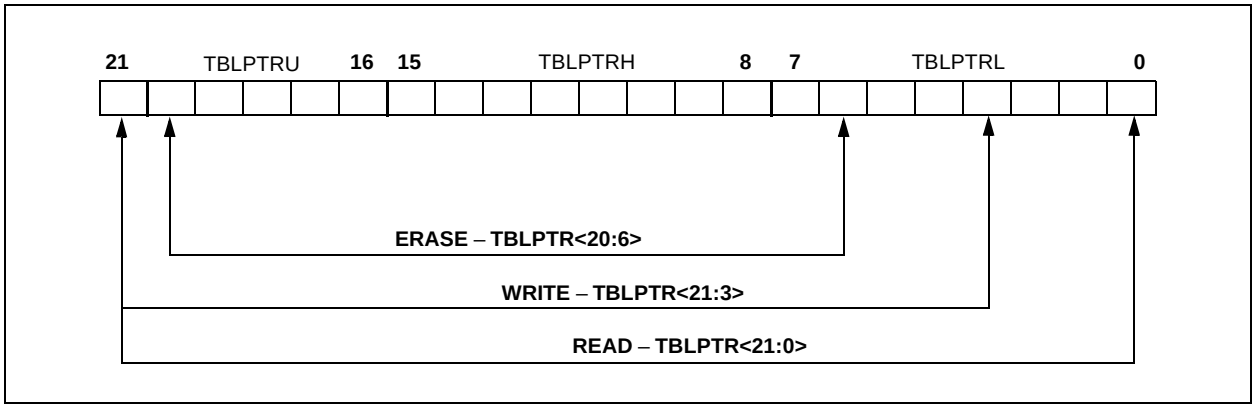
When an erase of program memory is executed, the 16 MSbs of the Table Pointer (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

Figure 5-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

TABLE 5-1: TABLE POINTER OPERATIONS WITH `TBLRD` AND `TBLWT` INSTRUCTIONS

Example	Operation on Table Pointer
`TBLRD*` `TBLWT*`	TBLPTR is not modified
`TBLRD*+` `TBLWT*+`	TBLPTR is incremented after the read/write
`TBLRD*-` `TBLWT*-`	TBLPTR is decremented after the read/write
`TBLRD+*` `TBLWT+*`	TBLPTR is incremented before the read/write

FIGURE 5-3: TABLE POINTER BOUNDARIES BASED ON OPERATION



6.0 EXTERNAL MEMORY INTERFACE

Note: The External Memory Interface is not implemented on PIC18F6X20 (64-pin) devices.

The External Memory Interface is a feature of the PIC18F8X20 devices that allows the controller to access external memory devices (such as Flash, EPROM, SRAM, etc.) as program or data memory.

The physical implementation of the interface uses 27 pins. These pins are reserved for external address/data bus functions; they are multiplexed with I/O port pins on four ports. Three I/O ports are multiplexed with the address/data bus, while the fourth port is multiplexed with the bus control signals. The I/O port functions are enabled when the EBDIS bit in the MEMCON register is set (see Register 6-1). A list of the multiplexed pins and their functions is provided in Table 6-1.

As implemented in the PIC18F8X20 devices, the interface operates in a similar manner to the external memory interface introduced on PIC18C601/801 microcontrollers. The most notable difference is that the interface on PIC18F8X20 devices only operates in 16-bit modes. The 8-bit mode is not supported.

For a more complete discussion of the operating modes that use the external memory interface, refer to **Section 4.1.1 "PIC18F8X20 Program Memory Modes"**.

6.1 Program Memory Modes and the External Memory Interface

As previously noted, PIC18F8X20 controllers are capable of operating in any one of four program memory modes, using combinations of on-chip and external program memory. The functions of the multiplexed port pins depend on the program memory mode selected, as well as the setting of the EBDIS bit.

In **Microprocessor Mode**, the external bus is always active and the port pins have only the external bus function.

In **Microcontroller Mode,** the bus is not active and the pins have their port functions only. Writes to the MEMCOM register are not permitted.

In **Microprocessor with Boot Block** or **Extended Microcontroller Mode,** the external program memory bus shares I/O port functions on the pins. When the device is fetching or doing table read/table write operations on the external program memory space, the pins will have the external bus function. If the device is fetching and accessing internal program memory locations only, the EBDIS control bit will change the pins from external memory to I/O port functions. When EBDIS = 0, the pins function as the external bus. When EBDIS = 1, the pins function as I/O ports.

Note: Maximum FOSC for the PIC18FX520 is limited to 25 MHz when using the external memory interface.

REGISTER 6-1: MEMCON REGISTER

R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0
bit7							bit0

bit 7 **EBDIS:** External Bus Disable bit
1 = External system bus disabled, all external bus drivers are mapped as I/O ports
0 = External system bus enabled and I/O ports are disabled

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

bit 5-4 **WAIT<1:0>:** Table Reads and Writes Bus Cycle Wait Count bits
11 = Table reads and writes will wait 0 TCY
10 = Table reads and writes will wait 1 TCY
01 = Table reads and writes will wait 2 TCY
00 = Table reads and writes will wait 3 TCY

bit 3-2 **Unimplemented:** Read as '0'

bit 1-0 **WM<1:0>:** TBLWRT Operation with 16-bit Bus bits
1x = Word Write mode: TABLAT<0> and TABLAT<1> word output, $\overline{WRH}$ active when TABLAT<1> written
01 = Byte Select mode: TABLAT data copied on both MSB and LSB, $\overline{WRH}$ and ($\overline{UB}$ or $\overline{LB}$) will activate
00 = Byte Write mode: TABLAT data copied on both MSB and LSB, $\overline{WRH}$ or $\overline{WRL}$ will activate

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

REGISTER 7-1: EECON1 REGISTER (ADDRESS FA6h)

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7 **EEPGD:** Flash Program/Data EEPROM Memory Select bit
1 = Access Flash program memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
1 = Access configuration or calibration registers
0 = Access Flash program or data EEPROM memory

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

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation)
0 = Perform write only

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit
1 = A write operation is prematurely terminated (any MCLR or any WDT Reset during self-timed programming in normal operation)
0 = The write operation completed

Note: When a WRERR occurs, the EEPGD or FREE bits are not cleared. This allows tracing of the error condition.

bit 2 **WREN:** Flash Program/Data EEPROM Write Enable bit
1 = Allows write cycles to Flash program/data EEPROM
0 = Inhibits write cycles to Flash program/data EEPROM

bit 1 **WR:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle, or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
1 = Initiates an EEPROM read. (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)
0 = Does not initiate an EEPROM read

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

10.6 PORTF, LATF and TRISF Registers

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

Read-modify-write operations on the LATF register, read and write the latched output value for PORTF.

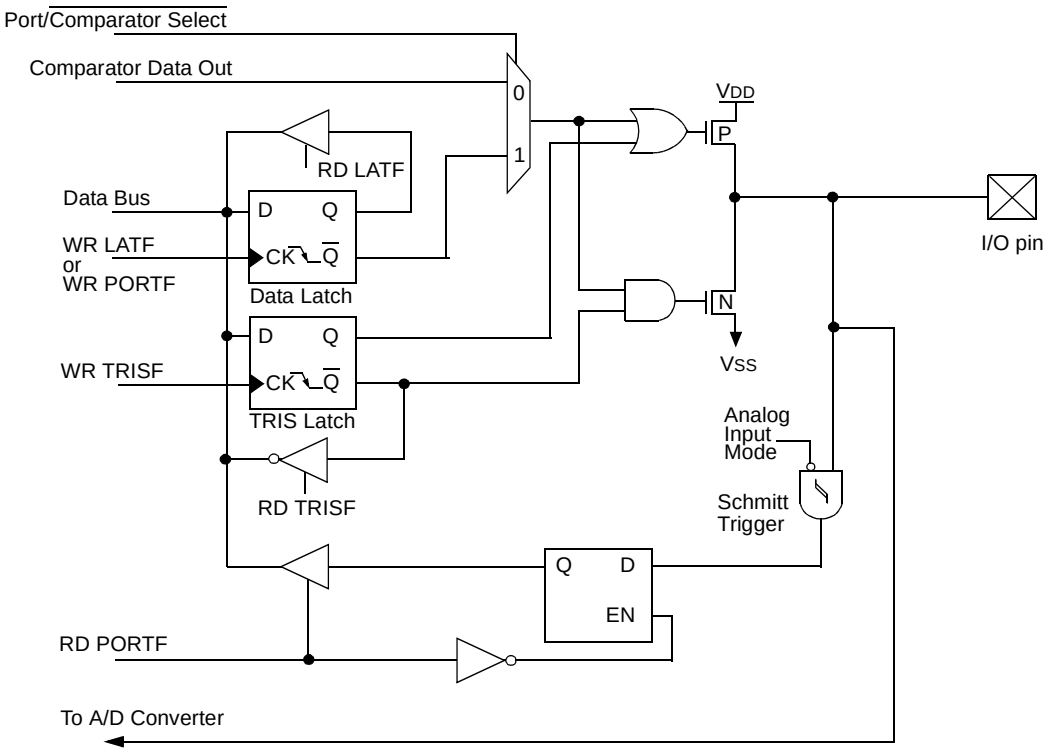
PORTF is multiplexed with several analog peripheral functions, including the A/D converter inputs and comparator inputs, outputs and voltage reference.

Note 1:	On a Power-on Reset, the RF6:RF0 pins are configured as inputs and read as '0'.
2:	To configure PORTF as digital I/O, turn off comparators and set ADCON1 value.

EXAMPLE 10-6: INITIALIZING PORTF

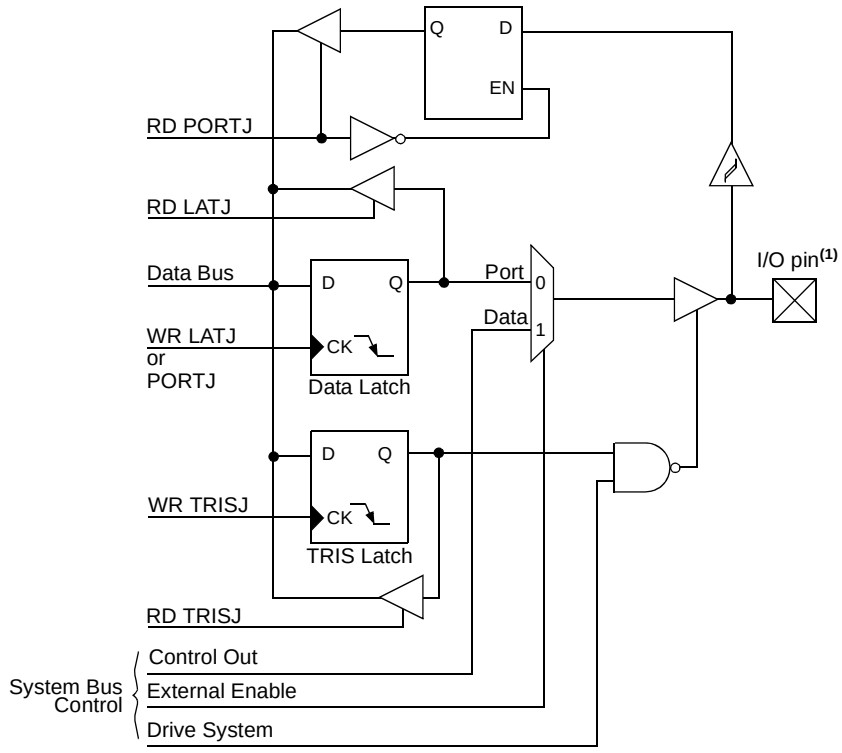
```
CLRF    PORTF   ; Initialize PORTF by
                ; clearing output
                ; data latches
CLRF    LATF    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0x07    ;
MOVWF   CMCON   ; Turn off comparators
MOVLW   0x0F    ;
MOVWF   ADCON1  ; Set PORTF as digital I/O
MOVLW   0xCF    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISF   ; Set RF3:RF0 as inputs
                ; RF5:RF4 as outputs
                ; RF7:RF6 as inputs
```

FIGURE 10-13: PORTF RF1/AN6/C2OUT, RF2/AN7/C1OUT PINS BLOCK DIAGRAM



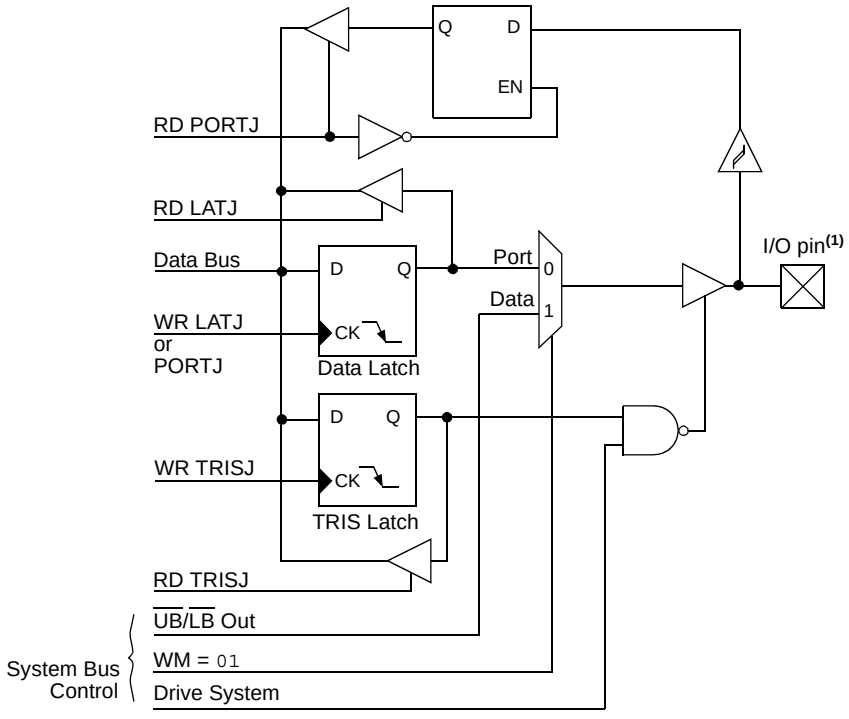
Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-21: RJ4:RJ0 PINS BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-22: RJ7:RJ6 PINS BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

TABLE 10-17: PORTJ FUNCTIONS

Name	Bit#	Buffer Type	Function
RJ0/ALE	bit 0	ST	Input/output port pin or address latch enable control for external memory interface.
RJ1/$\overline{OE}$	bit 1	ST	Input/output port pin or output enable control for external memory interface.
RJ2/$\overline{WRL}$	bit 2	ST	Input/output port pin or write low byte control for external memory interface.
RJ3/$\overline{WRH}$	bit 3	ST	Input/output port pin or write high byte control for external memory interface.
RJ4/BA0	bit 4	ST	Input/output port pin or byte address 0 control for external memory interface.
RJ5/$\overline{CE}$	bit 5	ST	Input/output port pin or chip enable control for external memory interface.
RJ6/$\overline{LB}$	bit 6	ST	Input/output port pin or lower byte select control for external memory interface.
RJ7/$\overline{UB}$	bit 7	ST	Input/output port pin or upper byte select control for external memory interface.

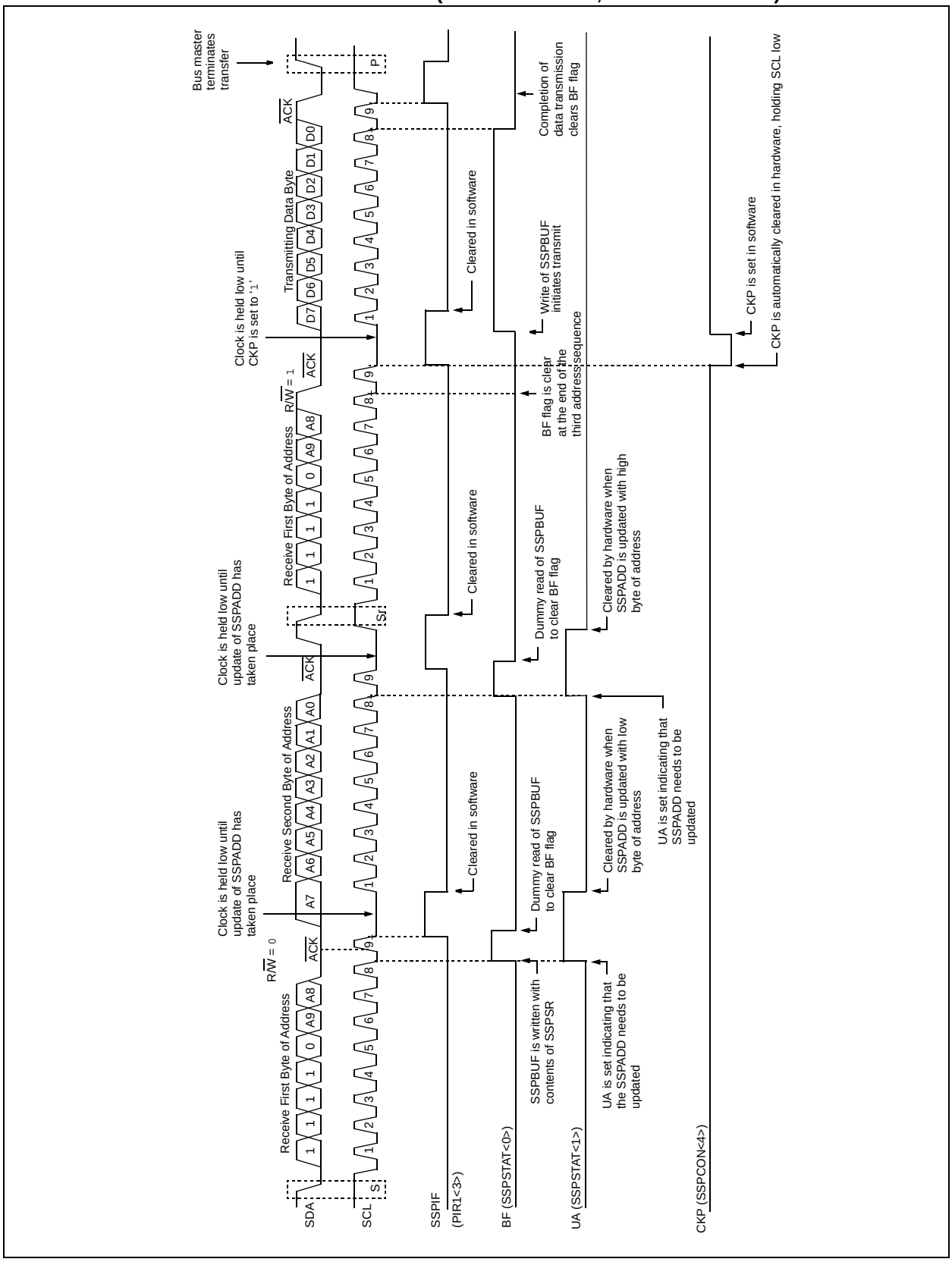
Legend: ST = Schmitt Trigger input

TABLE 10-18: SUMMARY OF REGISTERS ASSOCIATED WITH PORTJ

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
PORTJ	Read PORTJ pin/Write PORTJ Data Latch								xxxx xxxx	uuuu uuuu
LATJ	LATJ Data Output Register								xxxx xxxx	uuuu uuuu
TRISJ	Data Direction Control Register for PORTJ								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

FIGURE 17-11: I²C SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)



18.2.2 USART ASYNCHRONOUS RECEIVER

The USART receiver block diagram is shown in Figure 18-4. The data is received on the pin (RC7/RX1/DT1 or RG2/RX2/DT2) and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 16 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:

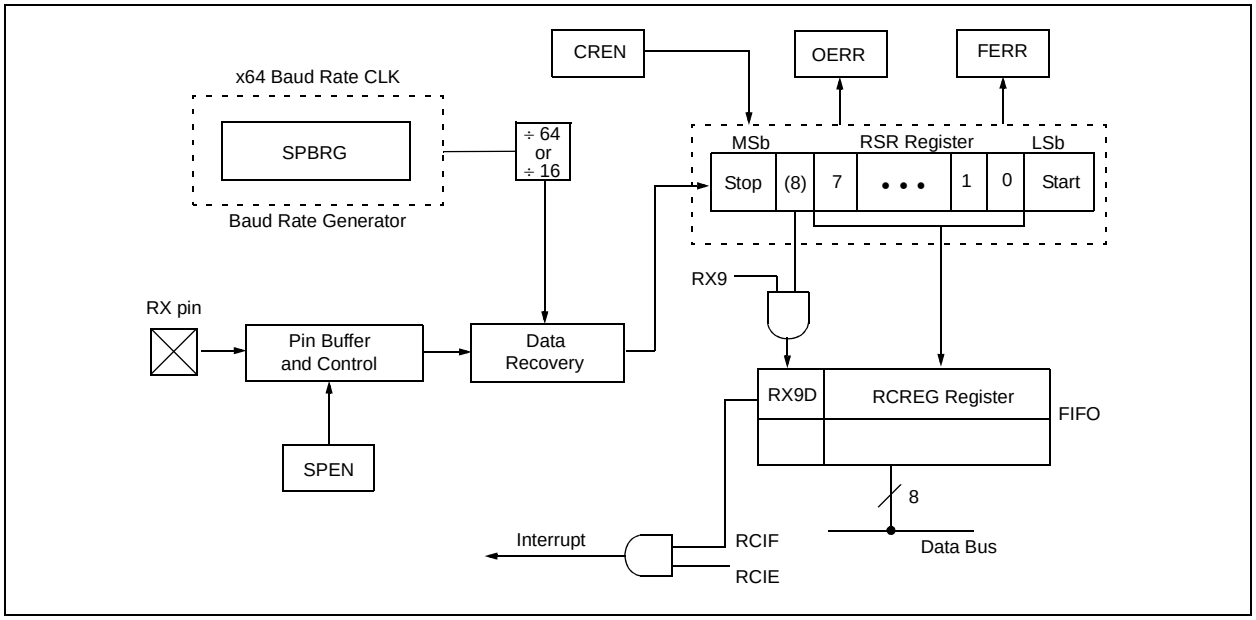
1. Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH (**Section 18.1 "USART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit RCxIE.
4. If 9-bit reception is desired, set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit RCxIF will be set when reception is complete and an interrupt will be generated if enable bit RCxIE was set.
7. Read the RCSTAx register to get the ninth 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.

18.2.3 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 SPBRGx register for the appropriate baud rate. If a high-speed baud rate is required, set the BRGH bit.
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 RCxIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCxIE and GIE bits are set.
8. Read the RCSTAx register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREGx 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 18-4: USART RECEIVE BLOCK DIAGRAM



20.7 Comparator Operation During Sleep

When a comparator is active and the device is placed in Sleep mode, the comparator remains active and the interrupt is functional, if enabled. This interrupt will wake-up the device from Sleep mode, when enabled. While the comparator is powered up, higher Sleep currents than shown in the power-down current specification will occur. Each operational comparator will consume additional current, as shown in the comparator specifications. To minimize power consumption while in Sleep mode, turn off the comparators (CM<2:0> = 111) before entering Sleep. If the device wakes up from Sleep, the contents of the CMCON register are not affected.

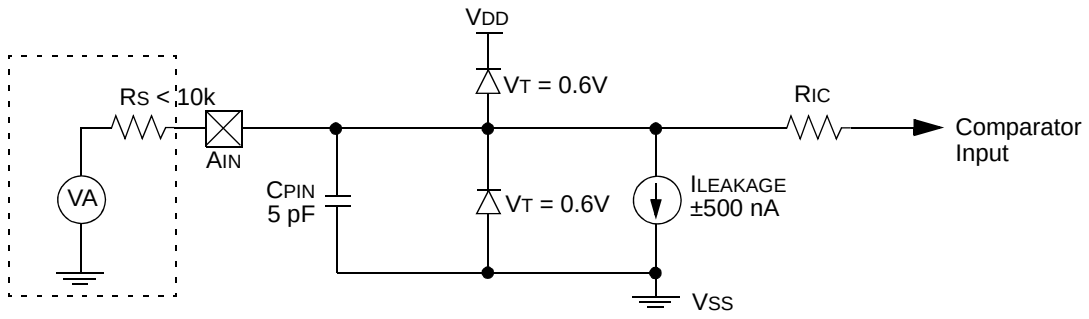
20.8 Effects of a Reset

A device Reset forces the CMCON register to its Reset state, causing the comparator module to be in the Comparator Reset mode, CM<2:0> = 000. This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at Reset time. The comparators will be powered down during the Reset interval.

20.9 Analog Input Connection Considerations

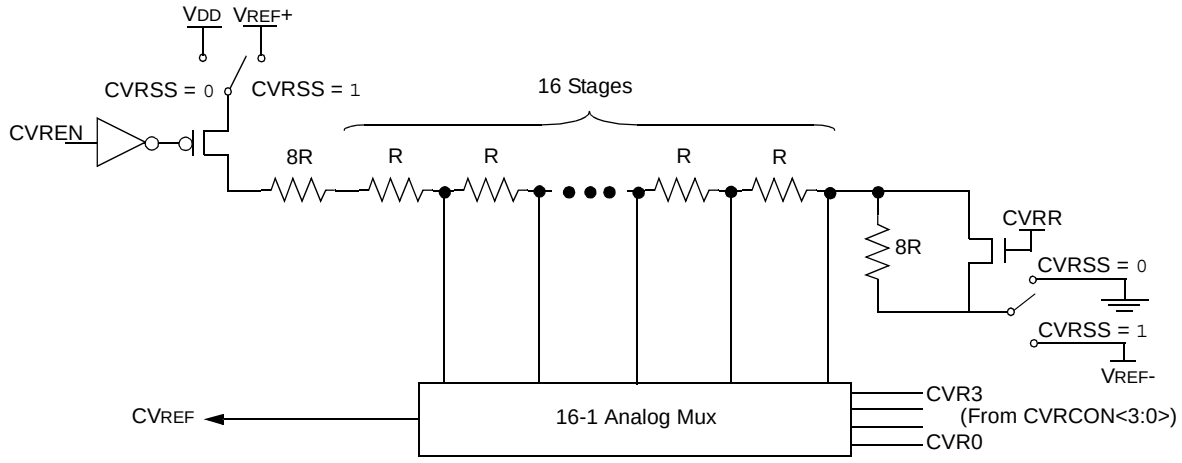
A simplified circuit for an analog input is shown in Figure 20-4. Since the analog pins are connected to a digital output, they have reverse biased diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up condition may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

FIGURE 20-4: COMPARATOR ANALOG INPUT MODEL



Legend:			
	CPIN	=	Input Capacitance
	VT	=	Threshold Voltage
	ILEAKAGE	=	Leakage Current at the pin due to various junctions
	RIC	=	Interconnect Resistance
	RS	=	Source Impedance
	VA	=	Analog Voltage

FIGURE 21-1: VOLTAGE REFERENCE BLOCK DIAGRAM



Note: R is defined in **Section 26.0 "Electrical Characteristics"**.

21.2 Voltage Reference Accuracy/Error

The full range of voltage reference cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 21-1) keep CVREF from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the CVREF output changes with fluctuations in that source. The tested absolute accuracy of the voltage reference can be found in **Section 26.0 "Electrical Characteristics"**.

21.3 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the CVRCON register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

21.4 Effects of a Reset

A device Reset disables the voltage reference by clearing bit CVREN (CVRCON<7>). This Reset also disconnects the reference from the RA2 pin by clearing bit CVROE (CVRCON<6>) and selects the high-voltage range by clearing bit CVRR (CVRCON<5>). The VRSS value select bits, CVRCON<3:0>, are also cleared.

21.5 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RF5 pin if the TRISF<5> bit is set and the CVROE bit is set. Enabling the voltage reference output onto the RF5 pin, configured as a digital input, will increase current consumption. Connecting RF5 as a digital output with VRSS enabled will also increase current consumption.

The RF5 pin can be used as a simple D/A output with limited drive capability. Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to VREF. Figure 21-2 shows an example buffering technique.

22.0 LOW-VOLTAGE DETECT

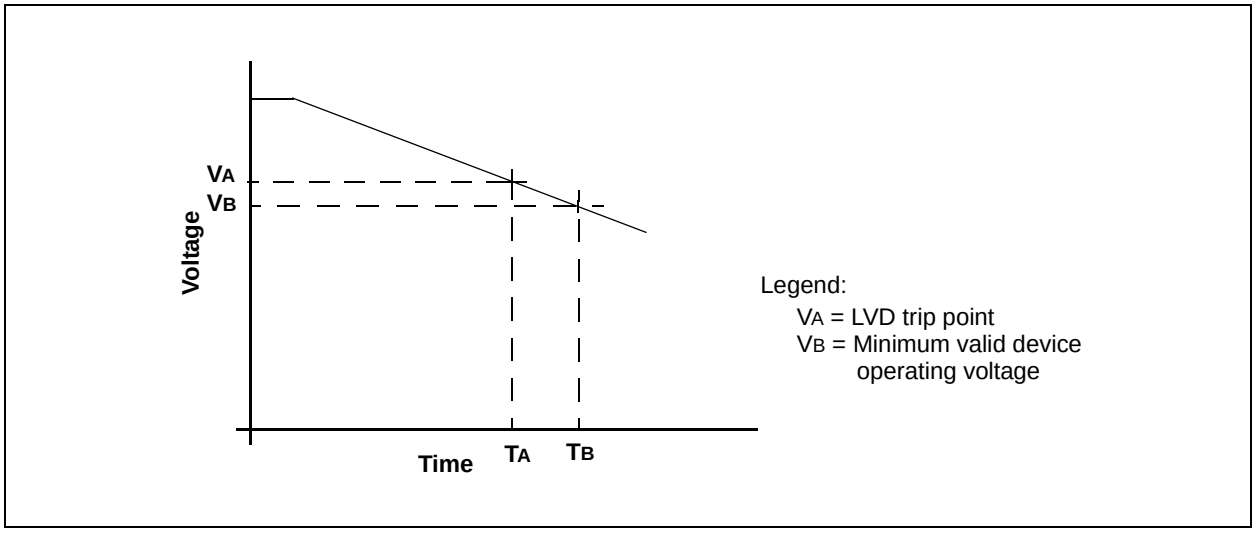
In many applications, the ability to determine if the device voltage (V_{DD}) is below a specified voltage level is a desirable feature. A window of operation for the application can be created, where the application software can do “housekeeping tasks” before the device voltage exits the valid operating range. This can be done using the Low-Voltage Detect module.

This module is a software programmable circuitry, where a device voltage trip point can be specified. When the voltage of the device becomes lower then the specified point, an interrupt flag is set. If the interrupt is enabled, the program execution will branch to the interrupt vector address and the software can then respond to that interrupt source.

The Low-Voltage Detect circuitry is completely under software control. This allows the circuitry to be “turned off” by the software, which minimizes the current consumption for the device.

Figure 22-1 shows a possible application voltage curve (typically for batteries). Over time, the device voltage decreases. When the device voltage equals voltage V_A, the LVD logic generates an interrupt. This occurs at time T_A. The application software then has the time, until the device voltage is no longer in valid operating range, to shut down the system. Voltage point V_B is the minimum valid operating voltage specification. This occurs at time T_B. The difference $T_B - T_A$ is the total time for shutdown.

FIGURE 22-1: TYPICAL LOW-VOLTAGE DETECT APPLICATION



The block diagram for the LVD module is shown in Figure 22-2. A comparator uses an internally generated reference voltage as the set point. When the selected tap output of the device voltage crosses the set point (is lower than), the LVDIF bit is set.

Each node in the resistor divider represents a “trip point” voltage. The “trip point” voltage is the minimum supply voltage level at which the device can operate before the LVD module asserts an interrupt. When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the 1.2V internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal, setting the LVDIF bit. This voltage is software programmable to any one of 16 values (see Figure 22-2). The trip point is selected by programming the LVDL3:LVDL0 bits (LVDCON<3:0>).

CPFSGT **Compare f with W, skip if f > W**

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

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

Operation: (f) – (W),
skip if (f) > (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	010a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If the contents of 'f' are greater than the contents of WREG**,** then the fetched instruction is discarded and a NOP is executed instead, making this a two-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      CPFSGT REG, 0
NGREATER  :
GREATER   :
```

Before Instruction
PC = Address (HERE)
W = ?

After Instruction
If REG > W;
PC = Address (GREATER)
If REG ≤ W;
PC = Address (NGREATER)

CPFSLT **Compare f with W, skip if f < W**

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

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

Operation: (f) – (W),
skip if (f) < (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	000a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank will be selected. If 'a' is '1', the BSR will not be overridden (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    CPFSLT REG, 1
NLESS   :
LESS    :
```

Before Instruction
PC = Address (HERE)
W = ?

After Instruction
If REG < W;
PC = Address (LESS)
If REG ≥ W;
PC = Address (NLESS)

MOVFF **Move f to f**

Syntax: [*label*] MOVFF f_s,f_d

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

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

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

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

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

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

Example: MOVFF REG1, REG2

Before Instruction
REG1 = 0x33
REG2 = 0x11

After Instruction
REG1 = 0x33,
REG2 = 0x33

MOVLB **Move literal to low nibble in BSR**

Syntax: [*label*] MOVLB k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow BSR$

Status Affected: None

Encoding:

0000	0001	kkkk	kkkk

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: MOVLB 5

Before Instruction
BSR register = 0x02

After Instruction
BSR register = 0x05

TABLE 26-4: MEMORY PROGRAMMING REQUIREMENTS

DC Characteristics			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Internal Program Memory Programming Specifications (Note 1)					
D110	VPP	Voltage on $\overline{MCLR}$/VPP pin	9.00	—	13.25	V	**(Note 2)**
D112	IPP	Current into $\overline{MCLR}$/VPP pin	—	—	5	μA	
D113	IDDP	Supply Current during Programming	—	—	10	mA	
		Data EEPROM Memory					
D120	ED	Cell Endurance	100K	1M	—	E/W	-40°C to +85°C
D120A	ED	Cell Endurance	10K	100K	—	E/W	+85°C to +125°C
D121	VDRW	VDD for Read/Write	VMIN	—	5.5	V	Using EECON to read/write VMIN = Minimum operating voltage
D122	TDEW	Erase/Write Cycle Time	—	4	—	ms	
D123	TRETD	Characteristic Retention	40	—	—	Year	-40°C to +85°C **(Note 3)**
D123A	TRETD	Characteristic Retention	100	—	—	Year	25°C **(Note 3)**
		Program Flash Memory					
D130	EP	Cell Endurance	10K	100K	—	E/W	-40°C to +85°C
D130A	EP	Cell Endurance	1000	10K	—	E/W	+85°C to +125°C
D131	VPR	VDD for Read	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D132	VIE	VDD for Block Erase	4.5	—	5.5	V	Using ICSP port
D132A	VIW	VDD for Externally Timed Erase or Write	4.5	—	5.5	V	Using ICSP port
D132B	VPEW	VDD for Self-Timed Write	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D133	TIE	ICSP Block Erase Cycle Time	—	5	—	ms	VDD > 4.5V
D133A	TIW	ICSP Erase or Write Cycle Time (externally timed)	1	—	—	ms	VDD > 4.5V
D133A	TIW	Self-Timed Write Cycle Time	—	2.5	—	ms	
D134	TRETD	Characteristic Retention	40	—	—	Year	-40°C to +85°C **(Note 3)**
D134A	TRETD	Characteristic Retention	100	—	—	Year	25°C **(Note 3)**

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: These specifications are for programming the on-chip program memory through the use of table write instructions.

2: The pin may be kept in this range at times other than programming, but it is not recommended.

3: Retention time is valid, provided no other specifications are violated.

FIGURE 26-12: BROWN-OUT RESET TIMING

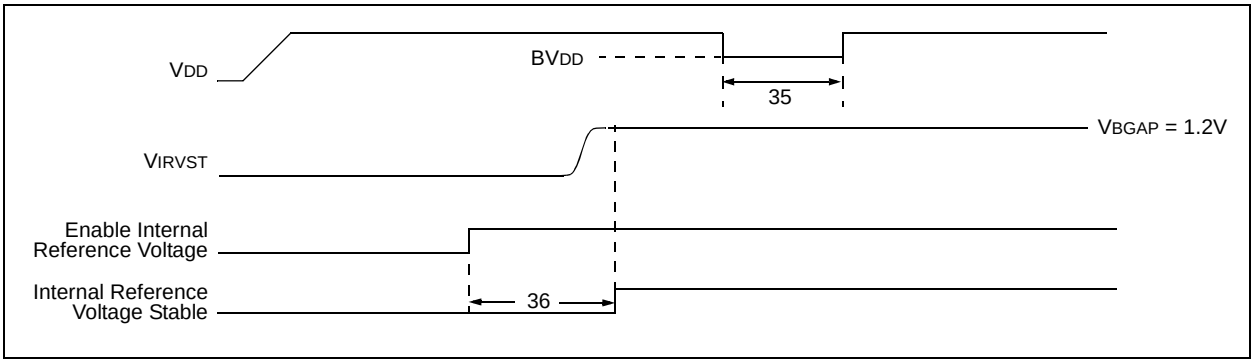
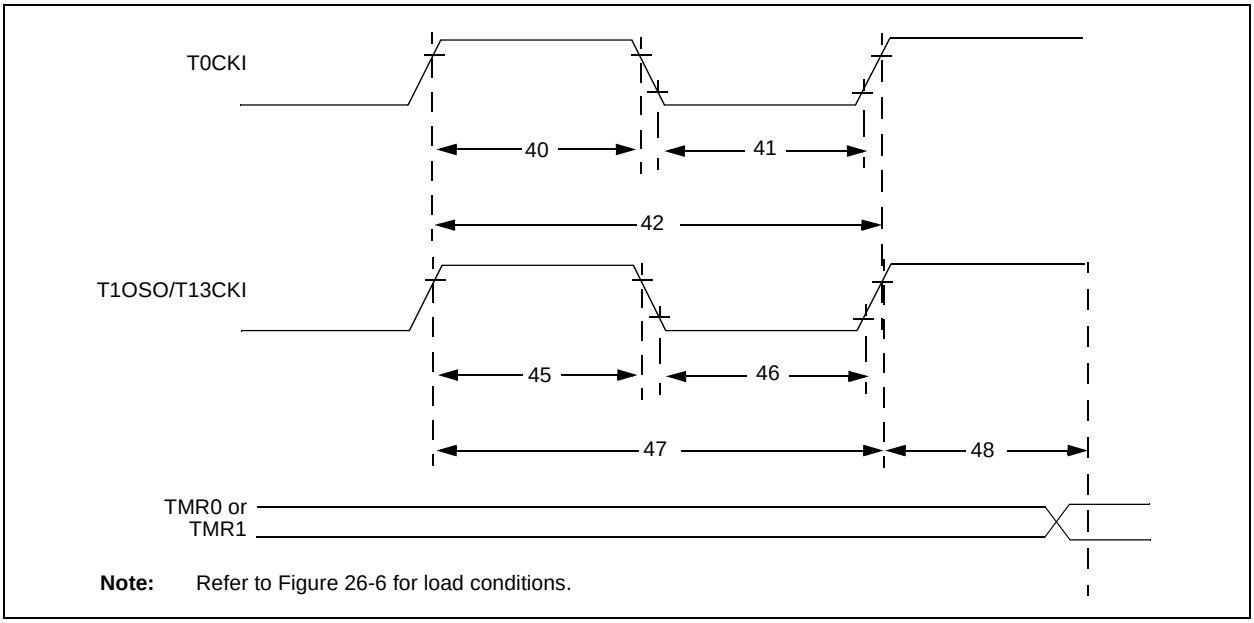


TABLE 26-11: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
30	TMCL	$\overline{MCLR}$ Pulse Width (low)	2	—	—	µs	
31	TWDT	Watchdog Timer Time-out Period (no postscaler)	7	18	33	ms	
32	TOST	Oscillation Start-up Timer Period	1024 TOSC	—	1024 TOSC	—	TOSC = OSC1 period
33	TPWRT	Power-up Timer Period	28	72	132	ms	
34	TIOZ	I/O High-Impedance from $\overline{MCLR}$ Low or Watchdog Timer Reset	—	2	—	µs	
35	TBOR	Brown-out Reset Pulse Width	200	—	—	µs	VDD ≤ BVDD (see D005)
36	TIVRST	Time for Internal Reference Voltage to become stable	—	20	50	µs	
37	TLVD	Low-Voltage Detect Pulse Width	200	—	—	µs	VDD ≤ VLVD

FIGURE 26-13: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS



Note: Refer to Figure 26-6 for load conditions.

FIGURE 27-15: TYPICAL AND MAXIMUM IPD vs. VDD OVER TEMPERATURE (TIMER1 AS MAIN OSCILLATOR, 32.768 kHz, C1 AND C2 = 47 pF)

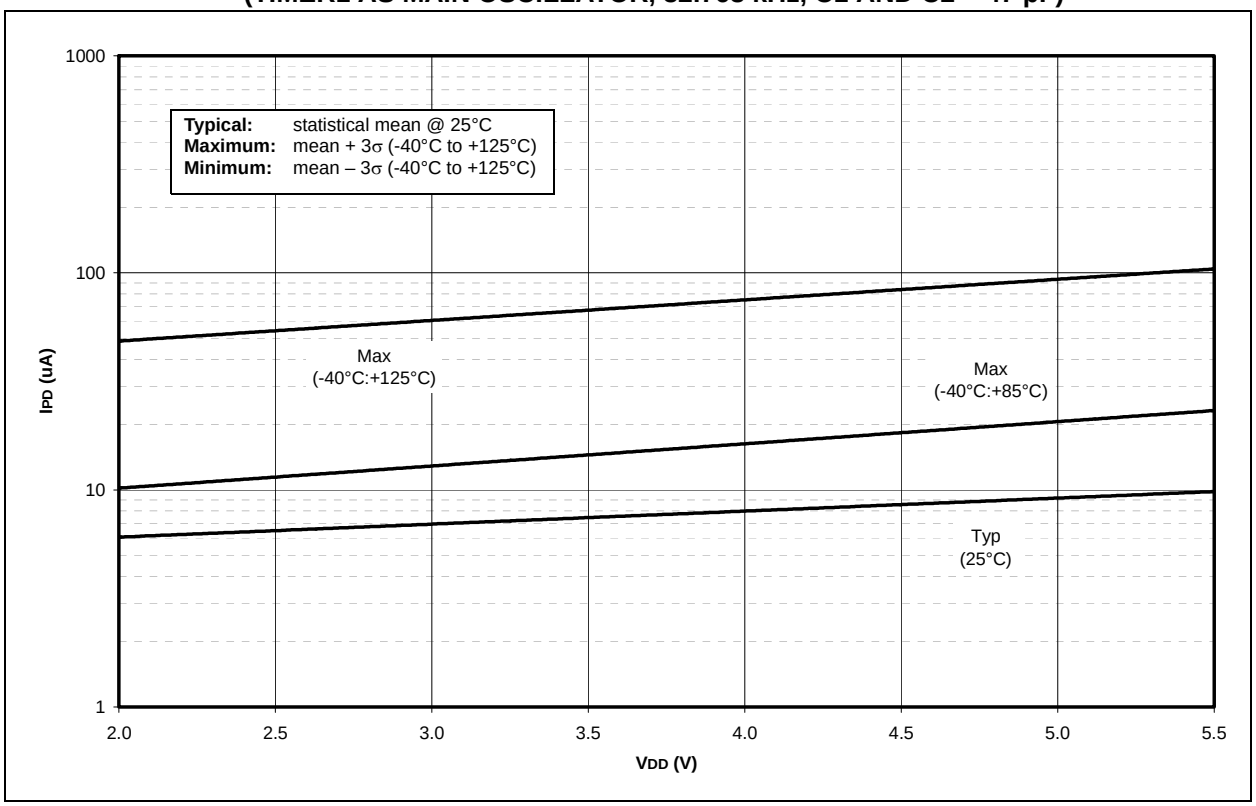
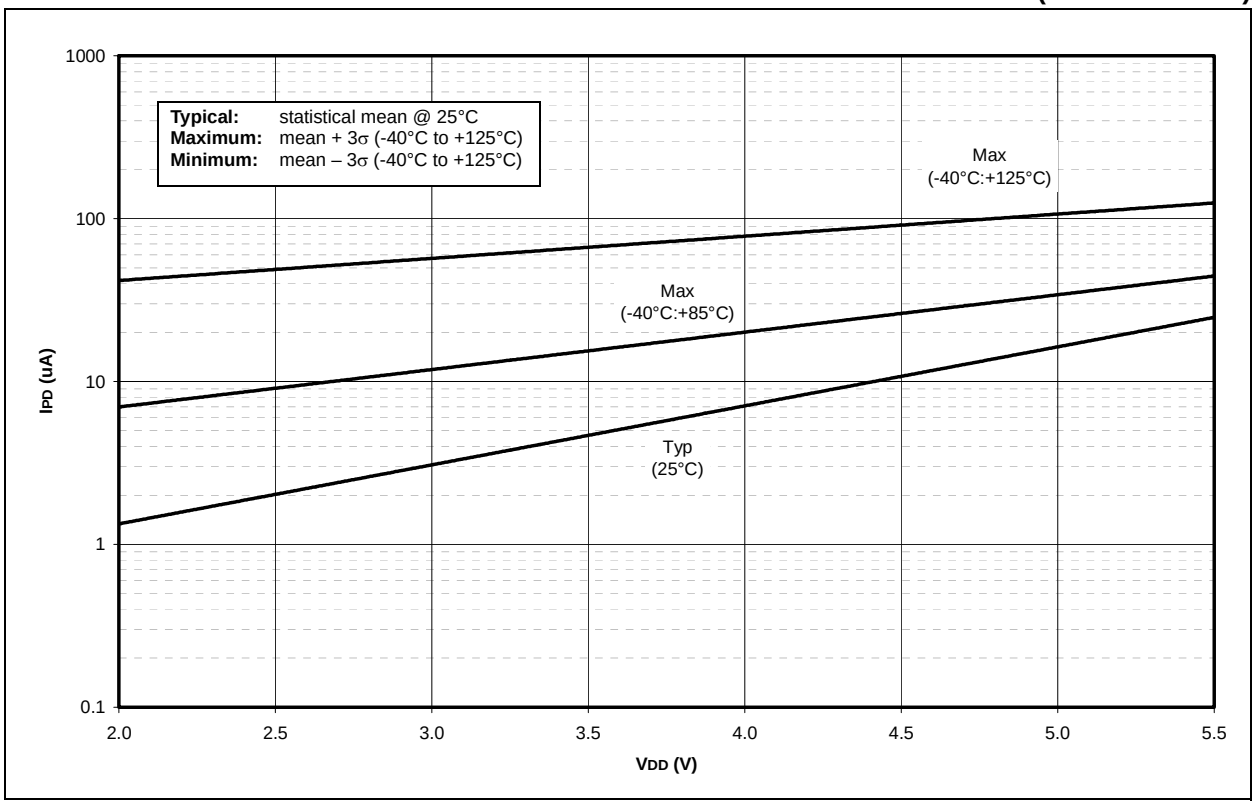


FIGURE 27-16: TYPICAL AND MAXIMUM ΔIWDT vs. VDD OVER TEMPERATURE (WDT ENABLED)



Watchdog Timer 251
BN 268
BNC 269
BNN 269
BNOV 270
BNZ 270
BOR. *See* Brown-out Reset.
BOV 273
BRA 271
BRG. *See* Baud Rate Generator.
Brown-out Reset (BOR) 30
BSF 271
BTFSC 272
BTFSS 272
BTG 273
BZ 274

C

C Compilers
MPLAB C18 302
CALL 274
Capture (CCP Module) 151
Associated Registers 153
CCP Pin Configuration 151
CCPR1H:CCPR1L Registers 151
Software Interrupt 151
Timer1/Timer3 Mode Selection 151
Capture/Compare/PWM (CCP) 149
Capture Mode. *See* Capture.
CCP Mode and Timer Resources 150
CCPRxH Register 150
CCPRxL Register 150
Compare Mode. *See* Compare.
Interconnect Configurations 150
Module Configuration 150
PWM Mode. *See* PWM.
Capture/Compare/PWM Requirements (All CCP Modules) ... 327
CLKO and I/O Timing Requirements 322, 323
Clocking Scheme/Instruction Cycle 44
CLRF 275
CLRWDT 275
Code Examples
16 x 16 Signed Multiply Routine 86
16 x 16 Unsigned Multiply Routine 86
8 x 8 Signed Multiply Routine 85
8 x 8 Unsigned Multiply Routine 85
Changing Between Capture Prescalers 151
Data EEPROM Read 81
Data EEPROM Refresh Routine 82
Data EEPROM Write 81
Erasing a Flash Program Memory Row 66
Fast Register Stack 44
How to Clear RAM (Bank 1) Using Indirect Addressing . 57
Implementing a Real-Time Clock using a Timer1 Interrupt Service 139
Initializing PORTA 103
Initializing PORTB 106
Initializing PORTC 109
Initializing PORTD 111
Initializing PORTE 114
Initializing PORTF 117
Initializing PORTG 120
Initializing PORTH 122
Initializing PORTJ 125
Loading the SSPBUF (SSPSR) Register 160
Reading a Flash Program Memory Word 65
Saving Status, WREG and BSR Registers in RAM . 102
Writing to Flash Program Memory 68–69
Code Protection 239
COMF 276
Comparator 223
Analog Input Connection Considerations 227
Associated Registers 228
Configuration 224
Effects of a Reset 227
Interrupts 226
Operation 225
Operation During Sleep 227
Outputs 225
Reference 225
External Signal 225
Internal Signal 225
Response Time 225
Comparator Specifications 315
Comparator Voltage Reference 229
Accuracy and Error 230
Associated Registers 231
Configuring 229
Connection Considerations 230
Effects of a Reset 230
Operation During Sleep 230
Compare (CCP Module) 152
Associated Registers 153
CCP Pin Configuration 152
CCPR1 Register 152
Software Interrupt 152
Special Event Trigger 138, 145, 152
Timer1/Timer3 Mode Selection 152
Compare (CCP2 Module)
Special Event Trigger 220
Configuration Bits 239
Context Saving During Interrupts 102
Control Registers
EECON1 and EECON2 62
TABLAT (Table Latch) Register 64
TBLPTR (Table Pointer) Register 64
Conversion Considerations 362
CPFSEQ 276
CPFSGT 277
CPFSLT 277
Customer Change Notification Service 375
Customer Notification Service 375
Customer Support 375

D

Data EEPROM Memory
Associated Registers 83
EEADR Register 79
EEADRH Register 79
EECON1 Register 79
EECON2 Register 79
Operation During Code-Protect 82
Protection Against Spurious Write 82
Reading 81
Using 82
Write Verify 82
Writing 81
Data Memory 47
General Purpose Registers 47
Map for PIC18FX520 Devices 48