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	EBI/EMI, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	70
Program Memory Size	48KB (24K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3936 x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf8527-i-pt

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number TQFP	Pin Type	Buffer Type	Description
				PORTE is a bidirectional I/O port.
RE0/AD8/$\overline{RD}$/P2D	4			
RE0		I/O	ST	Digital I/O.
AD8		I/O	TTL	External memory address/data 8.
$\overline{RD}$		I	TTL	Read control for Parallel Slave Port.
P2D		O	—	ECCP2 PWM output D.
RE1/AD9/$\overline{WR}$/P2C	3			
RE1		I/O	ST	Digital I/O.
AD9		I/O	TTL	External memory address/data 9.
$\overline{WR}$		I	TTL	Write control for Parallel Slave Port.
P2C		O	—	ECCP2 PWM output C.
RE2/AD10/$\overline{CS}$/P2B	78			
RE2		I/O	ST	Digital I/O.
AD10		I/O	TTL	External memory address/data 10.
$\overline{CS}$		I	TTL	Chip select control for Parallel Slave Port.
P2B		O	—	ECCP2 PWM output B.
RE3/AD11/P3C	77			
RE3		I/O	ST	Digital I/O.
AD11		I/O	TTL	External memory address/data 11.
P3C**(4)**		O	—	ECCP3 PWM output C.
RE4/AD12/P3B	76			
RE4		I/O	ST	Digital I/O.
AD12		I/O	TTL	External memory address/data 12.
P3B**(4)**		O	—	ECCP3 PWM output B.
RE5/AD13/P1C	75			
RE5		I/O	ST	Digital I/O.
AD13		I/O	TTL	External memory address/data 13.
P1C**(4)**		O	—	ECCP1 PWM output C.
RE6/AD14/P1B	74			
RE6		I/O	ST	Digital I/O.
AD14		I/O	TTL	External memory address/data 14.
P1B**(4)**		O	—	ECCP1 PWM output B.
RE7/AD15/ECCP2/P2A	73			
RE7		I/O	ST	Digital I/O.
AD15		I/O	TTL	External memory address/data 15.
ECCP2**(3)**		I/O	ST	Enhanced Capture 2 input/Compare 2 output/ PWM 2 output.
P2A**(3)**		O	—	ECCP2 PWM output A.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels; Analog = Analog input
I = Input; O = Output
P = Power; I^2C™/SMB = I^2C/SMBus input buffer

Note 1: Alternate assignment for ECCP2 when Configuration bit, CCP2MX, is cleared (all operating modes except Microcontroller mode).
2: Default assignment for ECCP2 in all operating modes (CCP2MX is set).
3: Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).
4: Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).
5: Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

2.6.5.1 Compensating with the EUSART

An adjustment may be required when the EUSART begins to generate framing errors or receives data with errors while in Asynchronous mode. Framing errors indicate that the device clock frequency is too high. To adjust for this, decrement the value in OSCTUNE to reduce the clock frequency. On the other hand, errors in data may suggest that the clock speed is too low. To compensate, increment OSCTUNE to increase the clock frequency.

2.6.5.2 Compensating with the Timers

This technique compares device clock speed to some reference clock. Two timers may be used; one timer is clocked by the peripheral clock, while the other is clocked by a fixed reference source, such as the Timer1 oscillator.

Both timers are cleared, but the timer clocked by the reference generates interrupts. When an interrupt occurs, the internally clocked timer is read and both timers are cleared. If the internally clocked timer value is much greater than expected, then the internal oscillator block is running too fast. To adjust for this, decrement the OSCTUNE register.

2.6.5.3 Compensating with the CCP Module in Capture Mode

A CCP module can use free running Timer1 (or Timer3), clocked by the internal oscillator block and an external event with a known period (i.e., AC power frequency). The time of the first event is captured in the CCPRxH:CCPRxL registers and is recorded for use later. When the second event causes a capture, the time of the first event is subtracted from the time of the second event. Since the period of the external event is known, the time difference between events can be calculated.

If the measured time is much greater than the calculated time, the internal oscillator block is running too fast. To compensate, decrement the OSCTUNE register. If the measured time is much less than the calculated time, the internal oscillator block is running too slow. To compensate, increment the OSCTUNE register.

5.3 Data Memory Organization

Note:	The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See **Section 5.5 "Data Memory and the Extended Instruction Set"** for more information.

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each; the PIC18F8722 family of devices implements all 16 banks. Figure 5-6 shows the data memory organization for the PIC18F8722 family of devices.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '`0`'s.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this subsection.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to SFRs and the lower portion of GPR Bank 0 without using the BSR. **Section 5.3.2 "Access Bank"** provides a detailed description of the Access RAM.

5.3.1 BANK SELECT REGISTER (BSR)

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 16 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 12-bit address, or an 8-bit low-order address and a 4-bit Bank Pointer.

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the 4 Most Significant bits of a location's address; the instruction itself includes the 8 Least Significant bits. Only the four lower bits of the BSR are implemented (BSR<3:0>). The upper four bits are unused; they will always read '`0`' and cannot be written to. The BSR can be loaded directly by using the `MOVLB` instruction.

The value of the BSR indicates the bank in data memory; the 8 bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 5-7.

Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 0Fh will end up resetting the program counter.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '`0`'s. Even so, the STATUS register will still be affected as if the operation was successful. The data memory map in Figure 5-6 indicates which banks are implemented.

In the core PIC18 instruction set, only the `MOVFF` instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the Access Bank to locate their target registers.

5.3.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top half of Bank 15 (F60h to FFFh). A list of these registers is given in Table 5-2 and Table 5-3.

The SFRs can be classified into two sets: those associated with the “core” device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The Reset and interrupt registers are described in their respective chapters, while the ALU’s STATUS register is described later in this section. Registers related to the operation of a peripheral feature are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as ‘0’s.

TABLE 5-2: SPECIAL FUNCTION REGISTER MAP FOR THE PIC18F8722 FAMILY OF DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(1)	FBFh	CCPR1H	F9Fh	IPR1	F7Fh	SPBRGH1
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	CCPR1L	F9Eh	PIR1	F7Eh	BAUDCON1
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	CCP1CON	F9Dh	PIE1	F7Dh	SPBRGH2
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR2H	F9Ch	MEMCON	F7Ch	BAUDCON2
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCPR2L	F9Bh	OSCTUNE	F7Bh	—(2)
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	TRISJ(3)	F7Ah	—(2)
FF9h	PCL	FD9h	FSR2L	FB9h	CCPR3H	F99h	TRISH(3)	F79h	ECCP1DEL
FF8h	TBLPTRU	FD8h	STATUS	FB8h	CCPR3L	F98h	TRISG	F78h	TMR4
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	CCP3CON	F97h	TRISF	F77h	PR4
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS	F96h	TRISE	F76h	T4CON
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD	F75h	CCPR4H
FF4h	PRODH	FD4h	—(2)	FB4h	CMCON	F94h	TRISC	F74h	CCPR4L
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB	F73h	CCP4CON
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA	F72h	CCPR5H
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ(3)	F71h	CCPR5L
FF0h	INTCON3	FD0h	RCON	FB0h	PSPCON	F90h	LATH(3)	F70h	CCP5CON
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG1	F8Fh	LATG	F6Fh	SPBRG2
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	RCREG1	F8Eh	LATF	F6Eh	RCREG2
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG1	F8Dh	LATE	F6Dh	TXREG2
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA1	F8Ch	LATD	F6Ch	TXSTA2
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA1	F8Bh	LATC	F6Bh	RCSTA2
FEAh	FSR0H	FCAh	T2CON	FAAh	EEADRH	F8Ah	LATB	F6Ah	ECCP3AS
FE9h	FSR0L	FC9h	SSP1BUF	FA9h	EEADR	F89h	LATA	F69h	ECCP3DEL
FE8h	WREG	FC8h	SSP1ADD	FA8h	EEDATA	F88h	PORTJ(3)	F68h	ECCP2AS
FE7h	INDF1(1)	FC7h	SSP1STAT	FA7h	EECON2(1)	F87h	PORTH(3)	F67h	ECCP2DEL
FE6h	POSTINC1(1)	FC6h	SSP1CON1	FA6h	EECON1	F86h	PORTG	F66h	SSP2BUF
FE5h	POSTDEC1(1)	FC5h	SSP1CON2	FA5h	IPR3	F85h	PORTF	F65h	SSP2ADD
FE4h	PREINC1(1)	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE	F64h	SSP2STAT
FE3h	PLUSW1(1)	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD	F63h	SSP2CON1
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC	F62h	SSP2CON2
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB	F61h	—(2)
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA	F60h	—(2)

Note 1: This is not a physical register.
2: Unimplemented registers are read as ‘0’.
3: This register is not available on 64-pin devices.

REGISTER 10-3: INTCON3: INTERRUPT CONTROL REGISTER 3

R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
bit 7							bit 0

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

bit 7 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **INT3IE:** INT3 External Interrupt Enable bit
1 = Enables the INT3 external interrupt
0 = Disables the INT3 external interrupt

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

bit 2 **INT3IF:** INT3 External Interrupt Flag bit
1 = The INT3 external interrupt occurred (must be cleared in software)
0 = The INT3 external interrupt did not occur

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global interrupt enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

11.0 I/O PORTS

Depending on the device selected and features enabled, there are up to nine ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

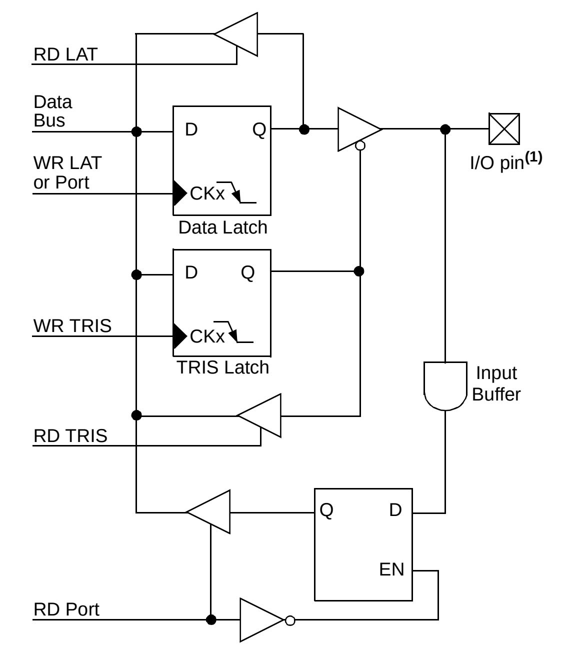
Each port has three registers for its operation. These registers are:

- TRIS register (Data Direction register)
- Port register (reads the levels on the pins of the device)
- LAT register (output latch)

The Data Latch (LAT register) is useful for read-modify-write operations on the value that the I/O pins are driving.

A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 11-1.

FIGURE 11-1: GENERIC I/O PORT OPERATION



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

11.1 PORTA, TRISA and LATA Registers

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

Reading the PORTA register reads the status of the pins, whereas writing to it, will write to the port latch.

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

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. Pins RA6 and RA7 are multiplexed with the main oscillator pins; they are enabled as oscillator or I/O pins by the selection of the main oscillator in the Configuration register (see **Section 25.1 "Configuration Bits"** for details). When they are not used as port pins, RA6 and RA7 and their associated TRIS and LAT bits are read as '`0`'.

The other PORTA pins are multiplexed with the analog VREF+ and VREF- inputs. The operation of pins RA5:RA0 as A/D converter inputs is selected by clearing or setting the PCFG<3:0> control bits in the ADCON1 register.

Note:	On a Power-on Reset, RA5 and RA<3:0> are configured as analog inputs and read as '`0`'. RA4 is configured as a digital input.

The RA4/T0CKI pin is a Schmitt Trigger input and an open-drain output. All other PORTA pins have TTL input levels and full CMOS output drivers.

The TRISA register controls the direction of the PORTA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 11-1: INITIALIZING PORTA

```
CLRF    PORTA   ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF    LATA    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0Fh     ; Configure A/D
MOVWF   ADCON1  ; for digital inputs
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISA   ; Set RA<3:0> as inputs
                ; RA<5:4> as outputs
```

11.2 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 11-2: INITIALIZING PORTB

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

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 $\overline{\text{RBPU}}$ (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.

Four of the PORTB pins (RB<7:4>) 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 ORed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

This interrupt can wake the device from power-managed modes. 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 `MOVSF, MOVSS, MOVFF (ANY), PORTB` 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.

For 80-pin devices, RB3 can be configured as the alternate peripheral pin for the ECCP2 module by clearing the CCP2MX Configuration bit. This applies only when the device is in one of the operating modes other than the default Microcontroller mode. If the device is in Microcontroller mode, the alternate assignment for ECCP2 is RE7. As with other ECCP2 configurations, the user must ensure that the TRISB<3> bit is set appropriately for the intended operation.

19.4.4 CLOCK STRETCHING

Both 7-Bit and 10-Bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPxCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCLx pin to be held low at the end of each data receive sequence.

19.4.4.1 Clock Stretching for 7-Bit Slave Receive Mode (SEN = 1)

In 7-Bit Slave Receive mode, on the falling edge of the ninth clock at the end of the $\overline{\text{ACK}}$ sequence, if the BF bit is set, the CKP bit in the SSPxCON1 register is automatically cleared, forcing the SCLx output to be held low. The CKP being cleared to '0' will assert the SCLx line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and read the contents of the SSPxBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 19-13).

> **Note 1:** If the user reads the contents of the SSPxBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.
>
> **2:** The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

19.4.4.2 Clock Stretching for 10-Bit Slave Receive Mode (SEN = 1)

In 10-Bit Slave Receive mode during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{\text{W}}$ bit cleared to '0'. The release of the clock line occurs upon updating SSPxADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

> **Note:** If the user polls the UA bit and clears it by updating the SSPxADD register before the falling edge of the ninth clock occurs and if the user hasn't cleared the BF bit by reading the SSPxBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

19.4.4.3 Clock Stretching for 7-Bit Slave Transmit Mode

The 7-Bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and load the contents of the SSPxBUF before the master device can initiate another transmit sequence (see Figure 19-9).

> **Note 1:** If the user loads the contents of SSPxBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.
>
> **2:** The CKP bit can be set in software regardless of the state of the BF bit.

19.4.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence which contains the high-order bits of the 10-bit address and the R/$\overline{\text{W}}$ bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-Bit Slave Transmit mode (see Figure 19-11).

24.2 HLVD Setup

The following steps are needed to set up the HLVD module:

1. Write the value to the HLVDL<3:0> bits that selects the desired HLVD trip point.
2. Set the VDIRMAG bit to detect high voltage (VDIRMAG = 1) or low voltage (VDIRMAG = 0).
3. Enable the HLVD module by setting the HLVDEN bit.
4. Clear the HLVD interrupt flag (PIR2<2>), which may have been set from a previous interrupt.
5. Enable the HLVD interrupt if interrupts are desired by setting the HLVDIE and GIE bits (PIE2<2> and INTCON<7>). An interrupt will not be generated until the IRVST bit is set.

24.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled and will consume static current. The total current consumption, when enabled, is specified in electrical specification parameter D022B (**Section 28.2 "DC Characteristics"**).

Depending on the application, the HLVD module does not need to be operating constantly. To decrease the current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After doing the check, the HLVD module may be disabled.

24.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification parameter D420 (**Section 28.2 "DC Characteristics"**), may be used by other internal circuitry, such as the Programmable Brown-out Reset. If the HLVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time, TIRVST, is an interval that is independent of device clock speed. It is specified in electrical specification parameter 36 (Table 28-12).

The HLVD interrupt flag is not enabled until TIRVST has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval. Refer to Figure 24-2 or Figure 24-3.

FIGURE 24-2: LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 0)

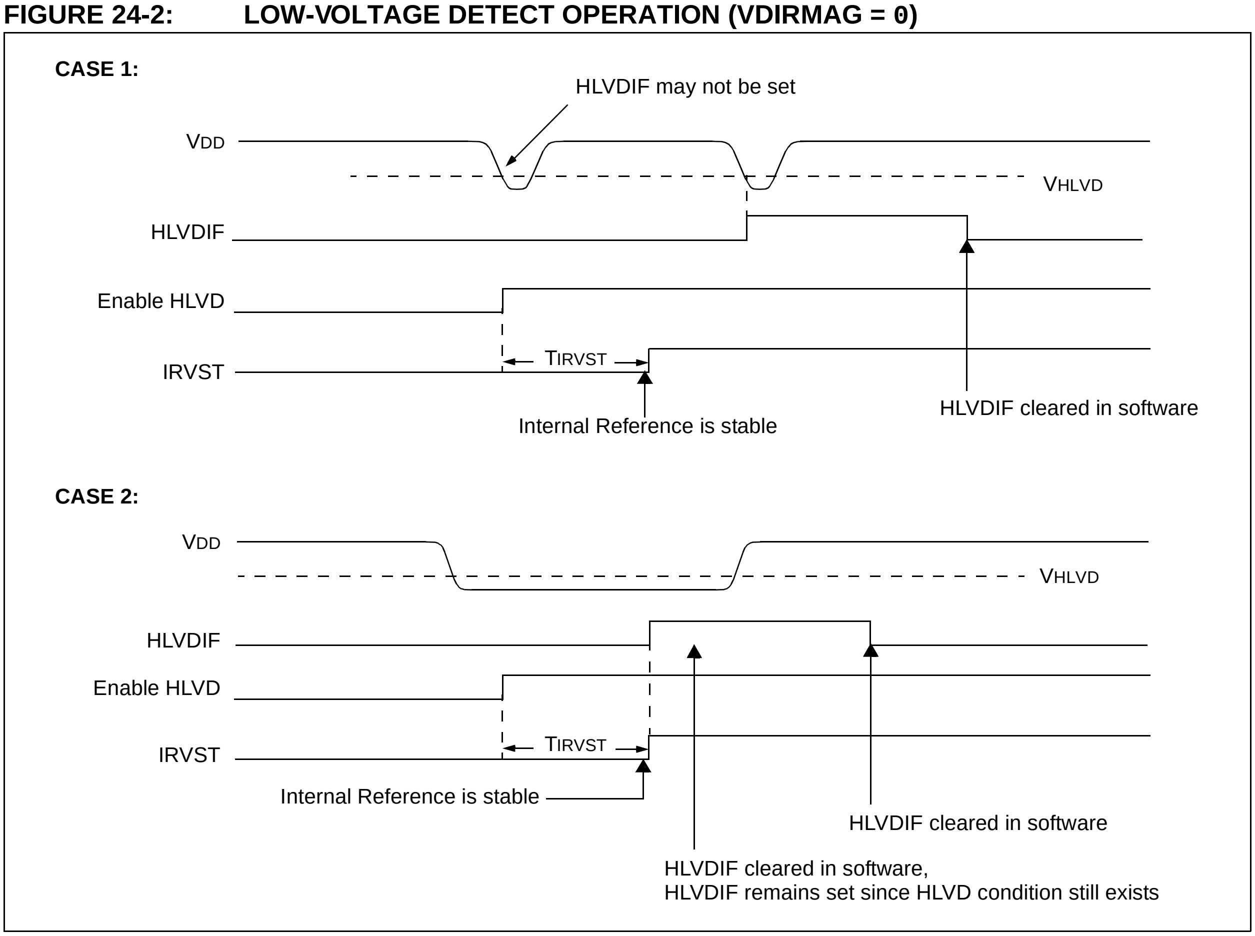


TABLE 26-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	n, s	Call Subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	n	Go to Address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop Top of Return Stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push Top of Return Stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software Device Reset	1	0000	0000	1111	1111	All	
RETFIE	s	Return from Interrupt Enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

RETURN — Return from Subroutine

Syntax:	RETURN {s}
Operands:	s ∈ [0,1]
Operation:	(TOS) → PC, if s = 1 (WS) → W, (STATUSS) → STATUS, (BSRS) → BSR, PCLATU, PCLATH are unchanged
Status Affected:	None
Encoding:	`0000` `0000` `0001` `001s`
Description:	Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If ‘s’= `1`, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers W, STATUS and BSR. If ‘s’ = `0`, no update of these registers occurs (default).
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	POP PC from stack
No operation	No operation	No operation	No operation

Example: `RETURN`

After Instruction:
PC = TOS

RLCF — Rotate Left f through Carry

Syntax:	RLCF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f<n>) → dest<n + 1>, (f<7>) → C, (C) → dest<0>
Status Affected:	C, N, Z
Encoding:	`0011` `01da` `ffff` `ffff`
Description:	The contents of register ‘f’ are rotated one bit to the left through the Carry flag. If ‘d’ is ‘`0`’, the result is placed in W. If ‘d’ is ‘`1`’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default). If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `RLCF REG, 0, 0`

Before Instruction
```
REG  =  1110 0110
C    =  0
```
After Instruction
```
REG  =  1110 0110
W    =  1100 1100
C    =  1
```

TBLRD	**Table Read**
Syntax:	TBLRD (*; *+; *-; +*)
Operands:	None
Operation:	if TBLRD *, (Prog Mem (TBLPTR)) → TABLAT; TBLPTR – No Change if TBLRD *+, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) + 1 → TBLPTR if TBLRD *-, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) – 1 → TBLPTR if TBLRD +*, (TBLPTR) + 1 → TBLPTR; (Prog Mem (TBLPTR)) → TABLAT
Status Affected:	None

Encoding:

`0000`	`0000`	`0000`	`10nn`
			`nn=0 *`
			`=1 *+`
			`=2 *-`
			`=3 +*`

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR<0> = `0`:Least Significant Byte of Program Memory Word

TBLPTR<0> = `1`:Most Significant Byte of Program Memory Word

The `TBLRD` instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)

TBLRD **Table Read (Continued)**

Example 1: `TBLRD *+ ;`

Before Instruction

TABLAT	=	55h
TBLPTR	=	00A356h
MEMORY(00A356h)	=	34h

After Instruction

TABLAT	=	34h
TBLPTR	=	00A357h

Example 2: `TBLRD +* ;`

Before Instruction

TABLAT	=	AAh
TBLPTR	=	01A357h
MEMORY(01A357h)	=	12h
MEMORY(01A358h)	=	34h

After Instruction

TABLAT	=	34h
TBLPTR	=	01A358h

TBLWT **Table Write**

Syntax: TBLWT (*; *+; *-; +*)

Operands: None

Operation:
if TBLWT*,
(TABLAT) → Holding Register;
TBLPTR – No Change
if TBLWT*+,
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR
if TBLWT*-,
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register

Status Affected: None

Encoding:

`0000`	`0000`	`0000`	`11nn`
			`nn=0 *`
			`=1 *+`
			`=2 *-`
			`=3 +*`

Description: This instruction uses the 3 LSBs of TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 5.0 “Memory Organization”** for additional details on programming Flash memory.)

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR<0> = 0:Least Significant Byte of Program Memory Word

TBLPTR<0> = 1:Most Significant Byte of Program Memory Word

The `TBLWT` instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

TBLWT **Table Write (Continued)**

Example 1: `TBLWT *+;`

Before Instruction

TABLAT	=	55h
TBLPTR	=	00A356h
HOLDING REGISTER (00A356h)	=	FFh

After Instructions (table write completion)

TABLAT	=	55h
TBLPTR	=	00A357h
HOLDING REGISTER (00A356h)	=	55h

Example 2: `TBLWT +*;`

Before Instruction

TABLAT	=	34h
TBLPTR	=	01389Ah
HOLDING REGISTER (01389Ah)	=	FFh
HOLDING REGISTER (01389Bh)	=	FFh

After Instruction (table write completion)

TABLAT	=	34h
TBLPTR	=	01389Bh
HOLDING REGISTER (01389Ah)	=	FFh
HOLDING REGISTER (01389Bh)	=	34h

SUBFSR	Subtract Literal from FSR
Syntax:	SUBFSR f, k
Operands:	$0 \leq k \leq 63$ $f \in [\ 0, 1, 2\ ]$
Operation:	FSRf – k → FSRf
Status Affected:	None
Encoding:	`1110` `1001` `ffkk` `kkkk`
Description:	The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.
Words:	1
Cycles:	1

Q Cycle Activity:

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

<u>Example:</u> `SUBFSR 2, 23h`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 03DCh

SUBULNK	Subtract Literal from FSR2 and Return
Syntax:	SUBULNK k
Operands:	$0 \leq k \leq 63$
Operation:	FSR2 – k → FSR2 (TOS) → PC
Status Affected:	None
Encoding:	`1110` `1001` `11kk` `kkkk`
Description:	The 6-bit literal 'k' is subtracted from the contents of the FSR2. A `RETURN` is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a `NOP` is performed during the second cycle. This may be thought of as a special case of the `SUBFSR` instruction, where f = 3 (binary '`11`'); it operates only on FSR2.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination
No Operation	No Operation	No Operation	No Operation

<u>Example:</u> `SUBULNK 23h`

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 03DCh
PC = (TOS)

26.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set for the PIC18F8722 family. This includes the MPLAB C18 C Compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option or dialog box within the environment that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

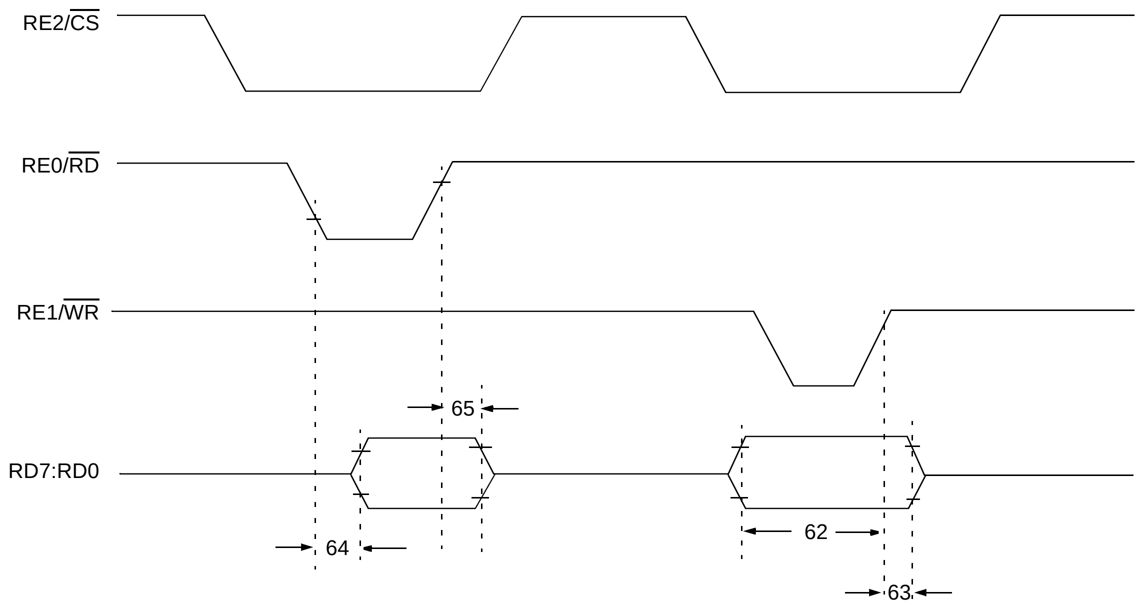
28.4 AC (Timing) Characteristics

28.4.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS		3. TCC:ST	(I^2C™ specifications only)
2. TppS		4. Ts	(I^2C specifications only)
T			
F	Frequency	T	Time
Lowercase letters (pp) and their meanings:			
pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{\text{RD}}$
cs	$\overline{\text{CS}}$	rw	$\overline{\text{RD}}$ or $\overline{\text{WR}}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{\text{SS}}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T13CKI
mc	$\overline{\text{MCLR}}$	wr	$\overline{\text{WR}}$
Uppercase letters and their meanings:			
S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (High-Impedance)	V	Valid
L	Low	Z	High-Impedance
I^2C only			
AA	output access	High	High
BUF	Bus free	Low	Low
TCC:ST (I^2C specifications only)			
CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	Stop condition
STA	Start condition		

FIGURE 28-14: PARALLEL SLAVE PORT TIMING (PIC18F8527/8622/8627/8722)



Note: Refer to Figure 28-5 for load conditions.

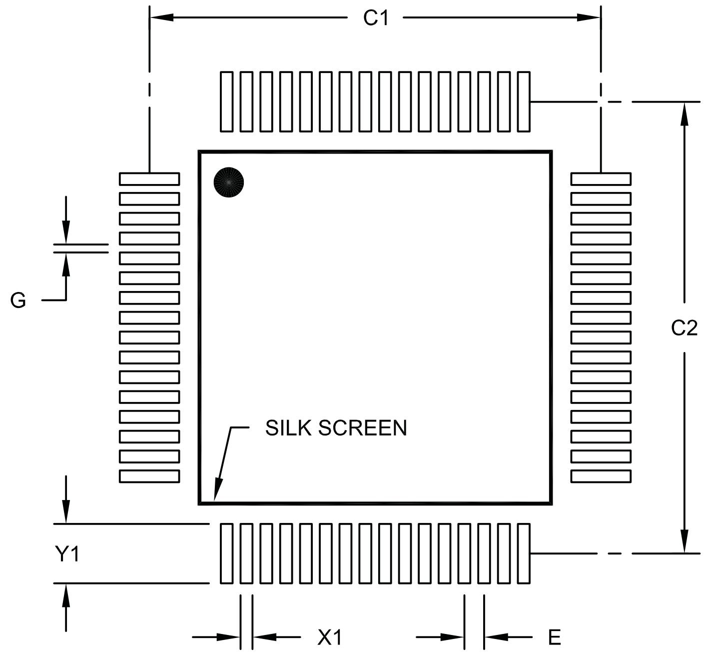
TABLE 28-15: PARALLEL SLAVE PORT REQUIREMENTS (PIC18F8527/8622/8627/8722)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
62	TdtV2wrH	Data In Valid before $\overline{\text{WR}}$ ↑ or $\overline{\text{CS}}$ ↑ (setup time)		20	—	ns	
63	TwrH2dtI	$\overline{\text{WR}}$ ↑ or $\overline{\text{CS}}$ ↑ to Data–In Invalid (hold time)	PIC18**F**XXXX	20	—	ns	
			PIC18**LF**XXXX	35	—	ns	VDD = 2.0V
64	TrdL2dtV	$\overline{\text{RD}}$ ↓ and $\overline{\text{CS}}$ ↓ to Data–Out Valid		—	80	ns	
65	TrdH2dtI	$\overline{\text{RD}}$ ↑ or $\overline{\text{CS}}$ ↓ to Data–Out Invalid		10	30	ns	
66	TibfINH	Inhibit of the IBF Flag bit being Cleared from $\overline{\text{WR}}$ ↑ or $\overline{\text{CS}}$ ↑		—	3 TCY		

NOTES:

64-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Contact Pad Spacing	C1		11.40	
Contact Pad Spacing	C2		11.40	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			1.50
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

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

Microchip Technology Drawing No. C04-2085A