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	48MHz
Connectivity	EBI/EMI, I²C, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	65
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 12x10b
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/pic18f86j50-i-pt

PIC18F87J50 FAMILY

64/80-Pin High-Performance, 1-Mbit Flash USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant SIE
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 3.9-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver

Flexible Oscillator Structure:

- High-Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal 31 kHz Oscillator, Tunable Internal Oscillator, 31 kHz to 8 MHz
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source 25 mA/25mA (PORTB and PORTC)
- Four Programmable External Interrupts
- Four Input Change Interrupts
- Two Capture/Compare/PWM (CCP) modules
- Three Enhanced Capture/Compare/PWM (ECCP) modules:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Two Master Synchronous Serial Port (MSSP) modules supporting 3-Wire SPI (all 4 modes) and I^2C™ Master and Slave modes
- 8-Bit Parallel Master Port/Enhanced Parallel Slave Port with 16 Address Lines
- Dual Analog Comparators with Input Multiplexing

Peripheral Highlights (continued):

- 10-Bit, up to 12-Channel Analog-to-Digital (A/D) Converter module:
  - Auto-acquisition capability
  - Conversion available during Sleep
- Two Enhanced USART modules:
  - Supports RS-485, RS-232 and LIN 1.2
  - Auto-wake-up on Start bit
  - Auto-Baud Detect

External Memory Bus (80-pin devices only):

- Address Capability of up to 2 Mbytes
- 8-Bit or 16-Bit Interface
- 12-Bit, 16-Bit and 20-Bit Addressing modes

Special Microcontroller Features:

- 5.5V Tolerant Inputs (digital-only pins)
- Low-Power, High-Speed CMOS Flash Technology
- C Compiler Optimized Architecture for Re-Entrant Code
- Power Management Features:
  - Run: CPU on, peripherals on
  - Idle: CPU off, peripherals on
  - Sleep: CPU off, peripherals off
- Priority Levels for Interrupts
- Self-Programmable under Software Control
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-Supply In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) with 3 Breakpoints via Two Pins
- Operating Voltage Range of 2.0V to 3.6V
- On-Chip 2.5V Regulator
- Flash Program Memory of 10000 Erase/Write Cycles and 20-Year Data Retention

TABLE 1-1: DEVICE FEATURES FOR THE PIC18F6XJ5X (64-PIN DEVICES)

Features	PIC18F65J50	PIC18F66J50	PIC18F66J55	PIC18F67J50
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	32K	64K	96K	128K
Program Memory (Instructions)	16384	32768	49152	65536
Data Memory (Bytes)	3904	3904	3904	3904
Interrupt Sources	30			
I/O Ports	Ports A, B, C, D, E, F, G			
Timers	5			
Capture/Compare/PWM Modules	2			
Enhanced Capture/ Compare/PWM Modules	3			
Serial Communications	MSSP (2), Enhanced USART (2), USB			
Parallel Communications (PMP)	Yes			
10-Bit Analog-to-Digital Module	8 Input Channels			
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow, $\overline{\text{MCLR}}$, WDT (PWRT, OST)			
Instruction Set	75 Instructions, 83 with Extended Instruction Set Enabled			
Packages	64-Pin TQFP			

TABLE 1-2: DEVICE FEATURES FOR THE PIC18F8XJ5X (80-PIN DEVICES)

Features	PIC18F85J50	PIC18F86J50	PIC18F86J55	PIC18F87J50
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	32K	64K	96K	128K
Program Memory (Instructions)	16384	32768	49152	65536
Data Memory (Bytes)	3904	3904	3904	3904
Interrupt Sources	30			
I/O Ports	Ports A, B, C, D, E, F, G, H, J			
Timers	5			
Capture/Compare/PWM Modules	2			
Enhanced Capture/ Compare/PWM Modules	3			
Serial Communications	MSSP (2), Enhanced USART (2), USB			
Parallel Communications (PMP)	Yes			
10-Bit Analog-to-Digital Module	12 Input Channels			
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow, $\overline{\text{MCLR}}$, WDT (PWRT, OST)			
Instruction Set	75 Instructions, 83 with Extended Instruction Set Enabled			
Packages	80-Pin TQFP			

TABLE 1-3: PIC18F6XJ5X PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	64-TQFP			
				PORTD is a bidirectional I/O port.
RD0/PMD0	58			
RD0		I/O	ST	Digital I/O.
PMD0		I/O	TTL	Parallel Master Port data.
RD1/PMD1	55			
RD1		I/O	ST	Digital I/O.
PMD1		I/O	TTL	Parallel Master Port data.
RD2/PMD2	54			
RD2		I/O	ST	Digital I/O.
PMD2		I/O	TTL	Parallel Master Port data.
RD3/PMD3	53			
RD3		I/O	ST	Digital I/O.
PMD3		I/O	TTL	Parallel Master Port data.
RD4/PMD4/SDO2	52			
RD4		I/O	ST	Digital I/O.
PMD4		I/O	TTL	Parallel Master Port data.
SDO2		O	—	SPI data out.
RD5/PMD5/SDI2/SDA2	51			
RD5		I/O	ST	Digital I/O.
PMD5		I/O	TTL	Parallel Master Port data.
SDI2		I	ST	SPI data in.
SDA2		I/O	ST	I^2C™ data I/O.
RD6/PMD6/SCK2/SCL2	50			
RD6		I/O	ST	Digital I/O.
PMD6		I/O	TTL	Parallel Master Port data.
SCK2		I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL2		I/O	ST	Synchronous serial clock input/output for I^2C mode.
RD7/PMD7/$\overline{SS2}$	49			
RD7		I/O	ST	Digital I/O.
PMD7		I/O	TTL	Parallel Master Port data.
$\overline{SS2}$		I	TTL	SPI slave select input.

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

Note 1: Default assignment for ECCP2/P2A when CCP2MX Configuration bit is set.

2: Alternate assignment for ECCP2/P2A when CCP2MX Configuration bit is cleared.

3: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

Example 8-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 8-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES3:RES0).

EQUATION 8-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

EXAMPLE 8-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    MOVF    ARG1H, W        ;
    MULWF   ARG2L           ; ARG1H * ARG2L->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
```

Example 8-4 shows the sequence to do a 16 x 16 signed multiply. Equation 8-2 shows the algorithm used. The 32-bit result is stored in four registers (RES3:RES0). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H\langle 7\rangle \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H\langle 7\rangle \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    MOVF    ARG1H, W        ;
    MULWF   ARG2L           ; ARG1H * ARG2L ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    BTFSS   ARG2H, 7        ; ARG2H:ARG2L neg?
    BRA     SIGN_ARG1       ; no, check ARG1
    MOVF    ARG1L, W        ;
    SUBWF   RES2            ;
    MOVF    ARG1H, W        ;
    SUBWFB  RES3
;
SIGN_ARG1
    BTFSS   ARG1H, 7        ; ARG1H:ARG1L neg?
    BRA     CONT_CODE       ; no, done
    MOVF    ARG2L, W        ;
    SUBWF   RES2            ;
    MOVF    ARG2H, W        ;
    SUBWFB  RES3
;
CONT_CODE
   :
```

9.0 INTERRUPTS

Members of the PIC18F87J10 family of devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high-priority level or a low-priority level. The high-priority interrupt vector is at 0008h and the low-priority interrupt vector is at 0018h. High-priority interrupt events will interrupt any low-priority interrupts that may be in progress.

There are thirteen registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

In general, interrupt sources have three bits to control their operation. They are:

- **Flag bit** to indicate that an interrupt event occurred
- **Enable bit** that allows program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** to select high priority or low priority

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 0008h or 0018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit which enables/disables all interrupt sources. All interrupts branch to address 0008h in Compatibility mode.

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High-priority interrupt sources can interrupt a low-priority interrupt. Low-priority interrupts are not processed while high-priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The “return from interrupt” instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used) which re-enables interrupts.

For external interrupt events, such as the INTx pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set regardless of the status of their corresponding enable bit or the GIE bit.

Note:	Do not use the `MOVFF` instruction to modify any of the interrupt control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior.

REGISTER 9-3: 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.

10.10 PORTJ, TRISJ and LATJ Registers

Note: PORTJ is available only on 80-pin devices.

PORTJ is an 8-bit wide, bidirectional port. All pins on PORTJ are digital only and tolerate voltages up to 5.5V.

All pins on PORTJ are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Note: These pins are configured as digital inputs on any device Reset.

When the external memory interface is enabled, all of the PORTJ pins function as control outputs for the interface. This occurs automatically when the interface is enabled by clearing the EBDIS control bit (MEMCON<7>). The TRISJ bits are also overridden.

Each of the PORTJ pins has a weak internal pull-up. The pull-ups are provided to keep the inputs at a known state for the external memory interface while powering up. A single control bit can turn off all the pull-ups. This is performed by clearing bit RJPU (PORTG<5>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on any device Reset.

EXAMPLE 10-9: INITIALIZING PORTJ

```
CLRF    PORTJ   ; Initialize PORTG by
                ; clearing output
                ; data latches
CLRF    LATJ    ; Alternate method to clear
                ; output data latches
MOVLW   0CFh    ; Value used to initialize
                ; data direction
MOVWF   TRISJ   ; Set RJ3:RJ0 as inputs
                ; RJ5:RJ4 as output
                ; RJ7:RJ6 as inputs
```

13.2 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 13-2). When the RD16 control bit, T1CON<7>, is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 High Byte Buffer register. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, has become invalid due to a rollover between reads.

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

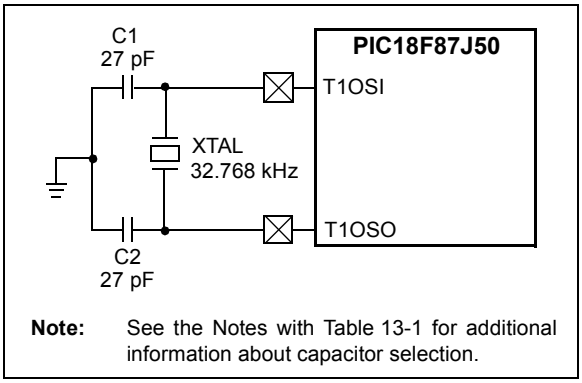
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

13.3 Timer1 Oscillator

An on-chip crystal oscillator circuit is incorporated between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting the Timer1 Oscillator Enable bit, T1OSCEN (T1CON<3>). The oscillator is a low-power circuit rated for 32 kHz crystals. It will continue to run during all power-managed modes. The circuit for a typical LP oscillator is shown in Figure 13-3. Table 13-1 shows the capacitor selection for the Timer1 oscillator.

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

FIGURE 13-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR



Note: See the Notes with Table 13-1 for additional information about capacitor selection.

TABLE 13-1: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR(2,3,4)

Oscillator Type	Freq.	C1	C2
LP	32 kHz	27 pF(1)	27 pF(1)

Note 1: Microchip suggests these values as a starting point in validating the oscillator circuit.

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

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

4: Capacitor values are for design guidance only.

13.3.1 USING TIMER1 AS A CLOCK SOURCE

The Timer1 oscillator is also available as a clock source in power-managed modes. By setting the clock select bits, SCS1:SCS0 (OSCCON<1:0>), to '01', the device switches to SEC_RUN mode; both the CPU and peripherals are clocked from the Timer1 oscillator. If the IDLEN bit (OSCCON<7>) is cleared and a SLEEP instruction is executed, the device enters SEC_IDLE mode. Additional details are available in **Section 3.0 "Power-Managed Modes"**.

Whenever the Timer1 oscillator is providing the clock source, the Timer1 system clock status flag, T1RUN (T1CON<6>), is set. This can be used to determine the controller's current clocking mode. It can also indicate the clock source being currently used by the Fail-Safe Clock Monitor. If the Clock Monitor is enabled and the Timer1 oscillator fails while providing the clock, polling the T1RUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

13.3.2 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

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

The oscillator circuit, shown in Figure 13-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.

REGISTER 19-2: SSPxCON1: MSSPx CONTROL REGISTER 1 (SPI MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV(1)	SSPEN(2)	CKP	SSPM3(3)	SSPM2(3)	SSPM1(3)	SSPM0(3)
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 **WCOL:** Write Collision Detect bit
1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit(1)
SPI Slave mode:
1 = A new byte is received while the SSPxBUF register is still holding the previous data. In case of overflow, the data in SSPxSR is lost. Overflow can only occur in Slave mode. The user must read the SSPxBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
0 = No overflow

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(2)
1 = Enables serial port and configures SCKx, SDOx, SDIx and $\overline{SSx}$ as serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Master Synchronous Serial Port Mode Select bits(3)
0101 = SPI Slave mode, clock = SCKx pin, $\overline{SSx}$ pin control disabled, $\overline{SSx}$ can be used as I/O pin
0100 = SPI Slave mode, clock = SCKx pin, $\overline{SSx}$ pin control enabled
0011 = SPI Master mode, clock = TMR2 output/2
0010 = SPI Master mode, clock = FOSC/64
0001 = SPI Master mode, clock = FOSC/16
0000 = SPI Master mode, clock = FOSC/4

Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register.
2: When enabled, this pin must be properly configured as input or output.
3: Bit combinations not specifically listed here are either reserved or implemented in I^2C™ mode only.

REGISTER 19-4: SSPxCON1: MSSPx CONTROL REGISTER 1 (I²C™ MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN(1)	CKP	SSPM3(2)	SSPM2(2)	SSPM1(2)	SSPM0(2)
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 **WCOL:** Write Collision Detect bit

In Master Transmit mode:
1 = A write to the SSPxBUF register was attempted while the I²C conditions were not valid for a transmission to be started (must be cleared in software)
0 = No collision

In Slave Transmit mode:
1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

In Receive mode (Master or Slave modes):
This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:
1 = A byte is received while the SSPxBUF register is still holding the previous byte (must be cleared in software)
0 = No overflow

In Transmit mode:
This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(1)

1 = Enables the serial port and configures the SDAx and SCLx pins as the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCKx Release Control bit

In Slave mode:
1 = Releases clock
0 = Holds clock low (clock stretch), used to ensure data setup time

In Master mode:
Unused in this mode.

bit 3-0 **SSPM3:SSPM0:** Master Synchronous Serial Port Mode Select bits(2)

1111 = I²C Slave mode, 10-bit address with Start and Stop bit interrupts enabled
1110 = I²C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1011 = I²C Firmware Controlled Master mode (Slave Idle)
1001 = Load SSPMSK register at SSPADD SFR address(3,4)
1000 = I²C Master mode, clock = FOSC/(4 * (SSPxADD + 1))
0111 = I²C Slave mode, 10-bit address
0110 = I²C Slave mode, 7-bit address

Note 1: When enabled, the SDAx and SCLx pins must be configured as inputs.
2: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.
3: When SSPM3:SSPM0 = 1001, any reads or writes to the SSPxADD SFR address actually accesses the SSPMSK register.
4: This mode is only available when 7-bit Address Masking mode is selected (MSSPMSK Configuration bit is '1').

19.4.8 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Enable bit, SEN (SSPxCON2<0>). If the SDAx and SCLx pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<6:0> and starts its count. If SCLx and SDAx are both sampled high when the Baud Rate Generator times out (TBRG), the SDAx pin is driven low. The action of the SDAx being driven low while SCLx is high is the Start condition and causes the S bit (SSPxSTAT<3>) to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPxADD<6:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit (SSPxCON2<0>) will be automatically cleared by hardware. The Baud Rate Generator is suspended, leaving the SDAx line held low and the Start condition is complete.

Note:	If, at the beginning of the Start condition, the SDAx and SCLx pins are already sampled low or if during the Start condition, the SCLx line is sampled low before the SDAx line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLxIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state.

19.4.8.1 WCOL Status Flag

If the user writes the SSPxBUF when a Start sequence is in progress, the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

Note:	Because queueing of events is not allowed, writing to the lower 5 bits of SSPxCON2 is disabled until the Start condition is complete.

FIGURE 19-21: FIRST START BIT TIMING

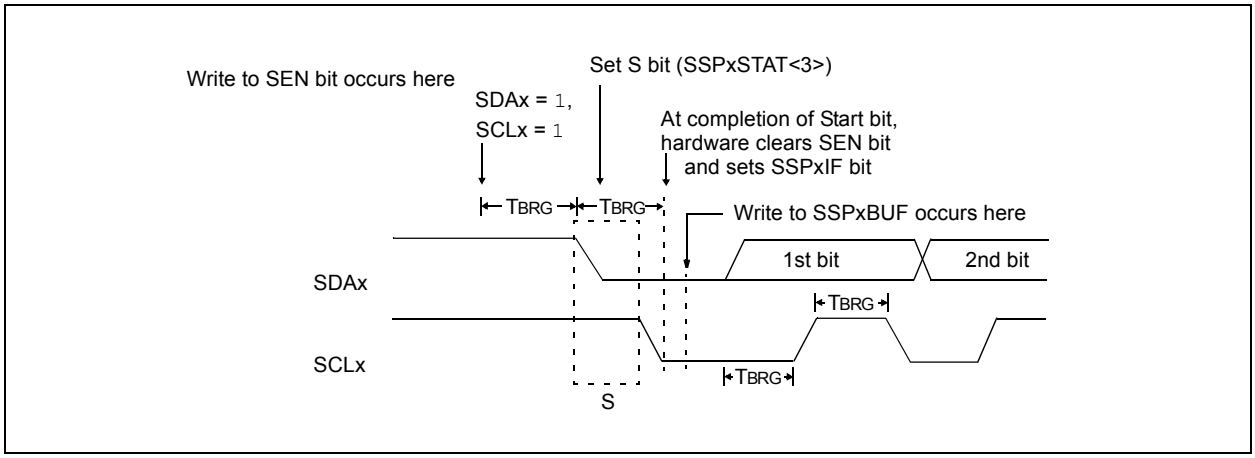
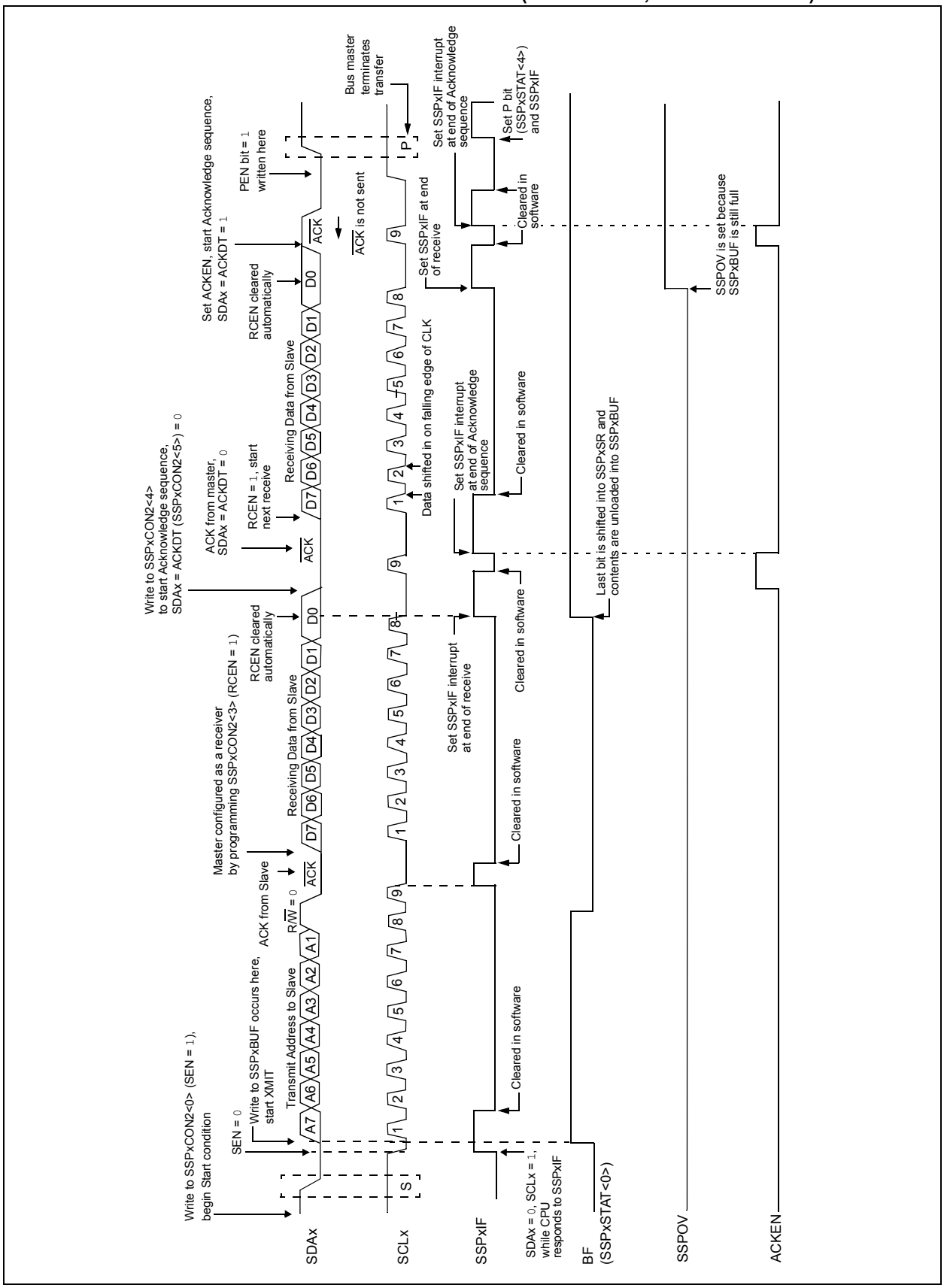


FIGURE 19-24: I²C™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



22.4.1.3 BDnSTAT Register (SIE Mode)

When the BD and its buffer are owned by the SIE, most of the bits in BDnSTAT take on a different meaning. The configuration is shown in Register 22-6. Once UOWN is set, any data or control settings previously written there by the user will be overwritten with data from the SIE.

The BDnSTAT register is updated by the SIE with the token Packet Identifier (PID) which is stored in BDnSTAT<5:3>. The transfer count in the corresponding BDnCNT register is updated. Values that overflow the 8-bit register carry over to the two most significant digits of the count, stored in BDnSTAT<1:0>.

22.4.2 BD BYTE COUNT

The byte count represents the total number of bytes that will be transmitted during an IN transfer. After an IN transfer, the SIE will return the number of bytes sent to the host.

For an OUT transfer, the byte count represents the maximum number of bytes that can be received and stored in USB RAM. After an OUT transfer, the SIE will return the actual number of bytes received. If the number of bytes received exceeds the corresponding byte count, the data packet will be rejected and a NAK handshake will be generated. When this happens, the byte count will not be updated.

The 10-bit byte count is distributed over two registers. The lower 8 bits of the count reside in the BDnCNT register. The upper two bits reside in BDnSTAT<1:0>. This represents a valid byte range of 0 to 1023.

22.4.3 BD ADDRESS VALIDATION

The BD Address register pair contains the starting RAM address location for the corresponding endpoint buffer. No mechanism is available in hardware to validate the BD address.

If the value of the BD address does not point to an address in the USB RAM, or if it points to an address within another endpoint's buffer, data is likely to be lost or overwritten. Similarly, overlapping a receive buffer (OUT endpoint) with a BD location in use can yield unexpected results. When developing USB applications, the user may want to consider the inclusion of software-based address validation in their code.

REGISTER 22-6: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD63STAT), SIE MODE (DATA RETURNED BY THE SIDE TO THE MCU)

R/W-x	U-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN	—	PID3	PID2	PID1	PID0	BC9	BC8
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 **UOWN:** USB Own bit
1 = The SIE owns the BD and its corresponding buffer

bit 6 **Reserved:** Not written by the SIE

bit 5-2 **PID3:PID0:** Packet Identifier bits
The received token PID value of the last transfer (IN, OUT or SETUP transactions only).

bit 1-0 **BC9:BC8:** Byte Count 9 and 8 bits
These bits are updated by the SIE to reflect the actual number of bytes received on an OUT transfer and the actual number of bytes transmitted on an IN transfer.

EQUATION 22-1: ESTIMATING USB TRANSCEIVER CURRENT CONSUMPTION

$$I_{XCVR} = \frac{(60\ mA \bullet V_{USB} \bullet P_{ZERO} \bullet P_{IN} \bullet L_{CABLE})}{(3.3V \bullet 5m)} + I_{PULLUP}$$

Legend: V_{USB} – Voltage applied to the V_{USB} pin in volts. (Should be 3.0V to 3.6V.)

P_{ZERO} – Percentage (in decimal) of the IN traffic bits sent by the PIC® device that are a value of '0'.

P_{IN} – Percentage (in decimal) of total bus bandwidth that is used for IN traffic.

L_{CABLE} – Length (in meters) of the USB cable. The USB 2.0 specification requires that full-speed applications use cables no longer than 5m.

I_{PULLUP} – Current which the nominal, 1.5 kΩ pull-up resistor (when enabled) must supply to the USB cable. On the host or hub end of the USB cable, 15 kΩ nominal resistors (14.25 kΩ to 24.8 kΩ) are present which pull both the D+ and D- lines to ground. During bus Idle conditions (such as between packets or during USB Suspend mode), this results in up to 218 μA of quiescent current drawn at 3.3V.

I_{PULLUP} is also dependant on bus traffic conditions and can be as high as 2.2 mA when the USB bandwidth is fully utilized (either IN or OUT traffic) for data that drives the lines to the "K" state most of the time.

EXAMPLE 22-2: CALCULATING USB TRANSCEIVER CURRENT†

For this example, the following assumptions are made about the application:

- 3.3V will be applied to V_{USB} and V_{DD}, with the core voltage regulator enabled.
- This is a full-speed application that uses one interrupt IN endpoint that can send one packet of 64 bytes every 1 ms, with no restrictions on the values of the bytes being sent. The application may or may not have additional traffic on OUT endpoints.
- A regular USB "B" or "mini-B" connector will be used on the application circuit board.

In this case, P_{ZERO} = 100% = 1, because there should be no restriction on the value of the data moving through the IN endpoint. All 64 kBps of data could potentially be bytes of value, 00h. Since '0' bits cause toggling of the output state of the transceiver, they cause the USB transceiver to consume extra current charging/discharging the cable. In this case, 100% of the data bits sent can be of value '0'. This should be considered the "max" value, as normal data will consist of a fair mix of ones and zeros.

This application uses 64 kBps for IN traffic out of the total bus bandwidth of 1.5 MBps (12 Mbps), therefore:

$$Pin = \frac{64\ kBps}{1.5\ MBps} = 4.3\% = 0.043$$

Since a regular "B" or "mini-B" connector is used in this application, the end user may plug in any type of cable up to the maximum allowed 5 m length. Therefore, we use the worst-case length:

L_{CABLE} = 5 meters

Assume I_{PULLUP} = 2.2 mA. The actual value of I_{PULLUP} will likely be closer to 218 μA, but allow for the worst-case. USB bandwidth is shared between all the devices which are plugged into the root port (via hubs). If the application is plugged into a USB 1.1 hub that has other devices plugged into it, your device may see host to device traffic on the bus, even if it is not addressed to your device. Since any traffic, regardless of source, can increase the I_{PULLUP} current above the base 218 μA, it is safest to allow for the worst case of 2.2 mA.

Therefore:

$$I_{XCVR} = \frac{(60\ mA \bullet 3.3V \bullet 1 \bullet 0.043 \bullet 5m)}{(3.3V \bullet 5m)} + 2.2\ mA = 4.8\ mA$$

† The calculated value should be considered an approximation and additional guardband or application-specific product testing is recommended. The transceiver current is "in addition to" the rest of the current consumed by the PIC18F87J10 family device that is needed to run the core, drive the other I/O lines, power the various modules, etc.

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

Syntax: 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 the 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 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 $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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: 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank (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)

INCFSZ **Increment f, Skip if 0**

Syntax: INCFSZ f {,d {,a}}

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

Operation: (f) + 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

0011	11da	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. (default)

If the result is '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

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 $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

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

If skip and followed by 2-word instruction:

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

Example:
```
HERE    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

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

INFSNZ **Increment f, Skip if not 0**

Syntax: INFSNZ f {,d {,a}}

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

Operation: (f) + 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

0100	10da	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If the result is not '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

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 $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

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

If skip and followed by 2-word instruction:

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

Example:
```
HERE    INFSNZ  REG, 1, 0
ZERO
NZERO
```

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

POP **Pop Top of Return Stack**

Syntax: POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding:

0000	0000	0000	0110

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	POP TOS value	No operation

Example:

```
POP
GOTO    NEW
```

Before Instruction
TOS = 0031A2h
Stack (1 level down) = 014332h

After Instruction
TOS = 014332h
PC = NEW

PUSH **Push Top of Return Stack**

Syntax: PUSH

Operands: None

Operation: (PC + 2) → TOS

Status Affected: None

Encoding:

0000	0000	0000	0101

Description: The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	PUSH PC + 2 onto return stack	No operation	No operation

Example:

```
PUSH
```

Before Instruction
TOS = 345Ah
PC = 0124h

After Instruction
PC = 0126h
TOS = 0126h
Stack (1 level down) = 345Ah

28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J50 Family (Industrial) (Continued)

PIC18F87J50 Family (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Supply Current (IDD) Cont.(2)						
	All devices	0.17	0.35	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 1 MHz (**PRI_RUN** mode, EC oscillator)
		0.18	0.35	mA	+25°C		
		0.20	0.42	mA	+85°C		
	All devices	0.29	0.52	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		0.31	0.52	mA	+25°C		
		0.34	0.61	mA	+85°C		
	All devices	0.59	1.1	mA	-40°C	VDD = 3.3V(5)	
		0.44	0.85	mA	+25°C		
		0.42	0.85	mA	+85°C		
	All devices	0.70	1.25	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 4 MHz (**PRI_RUN** mode, EC oscillator)
		0.75	1.25	mA	+25°C		
		0.79	1.36	mA	+85°C		
	All devices	1.10	1.7	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		1.10	1.7	mA	+25°C		
		1.12	1.82	mA	+85°C		
	All devices	1.55	1.95	mA	-40°C	VDD = 3.3V(5)	
		1.47	1.89	mA	+25°C		
		1.54	1.92	mA	+85°C		
	All devices	9.9	14.8	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	FOSC = 48 MHz (**PRI_RUN** mode, EC oscillator)
		9.5	14.8	mA	+25°C		
		10.1	15.2	mA	+85°C		
	All devices	13.3	23.2	mA	-40°C	VDD = 3.3V(5)	
		12.2	22.7	mA	+25°C		
		12.1	22.7	mA	+85°C		

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. All features that add delta current are disabled (USB module, WDT, etc.).
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 disabled unless otherwise specified.

3: 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.

4: Voltage regulator disabled (ENVREG = 0, tied to VSS).

5: Voltage regulator enabled (ENVREG = 1, tied to VDD), REGSLP = 1.

6: This is the module differential current when the USB module is enabled and clocked at 48 MHz, but with no USB cable attached. When the USB cable is attached or data is being transmitted, the current consumption may be much higher (see **Section 22.6.4 "USB Transceiver Current Consumption"**). During USB Suspend mode (USBEN = 1, SUSPND = 1, bus in Idle state), the USB module current will be dominated by the D+ or D- pull-up resistor. The integrated pull-up resistors use "resistor switching" according to the resistor_ecn supplement to the USB 2.0 specifications, and therefore, may be as low as 900Ω during Idle conditions.

28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J50 Family (Industrial) (Continued)

PIC18F87J50 Family (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
Param No.	Device	Typ	Max	Units	Conditions		
	Supply Current (IDD) Cont.(2)						
	All devices	18	35	µA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 32 kHz(3) (**SEC_RUN** mode, Timer1 as clock)
		19	35	µA	+25°C		
		28	49	µA	+85°C		
	All devices	20	45	µA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		21	45	µA	+25°C		
		32	61	µA	+85°C		
	All devices	0.06	0.11	mA	-40°C	VDD = 3.3V(5)	
		0.07	0.11	mA	+25°C		
		0.09	0.15	mA	+85°C		
	All devices	14	28	µA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 32 kHz(3) (**SEC_IDLE** mode, Timer1 as clock)
		15	28	µA	+25°C		
		24	43	µA	+85°C		
	All devices	15	31	µA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		16	31	µA	+25°C		
		27	50	µA	+85°C		
	All devices	0.05	0.10	mA	-40°C	VDD = 3.3V(5)	
		0.06	0.10	mA	+25°C		
		0.08	0.14	mA	+85°C		

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. All features that add delta current are disabled (USB module, WDT, etc.).
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;
$\overline{\text{MCLR}}$ = VDD; WDT disabled unless otherwise specified.

3: 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.

4: Voltage regulator disabled (ENVREG = 0, tied to VSS).

5: Voltage regulator enabled (ENVREG = 1, tied to VDD), REGSLP = 1.

6: This is the module differential current when the USB module is enabled and clocked at 48 MHz, but with no USB cable attached. When the USB cable is attached or data is being transmitted, the current consumption may be much higher (see **Section 22.6.4 "USB Transceiver Current Consumption"**). During USB Suspend mode (USBEN = 1, SUSPND = 1, bus in Idle state), the USB module current will be dominated by the D+ or D- pull-up resistor. The integrated pull-up resistors use "resistor switching" according to the resistor_ecn supplement to the USB 2.0 specifications, and therefore, may be as low as 900Ω during Idle conditions.

28.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 28-4: EXTERNAL CLOCK TIMING

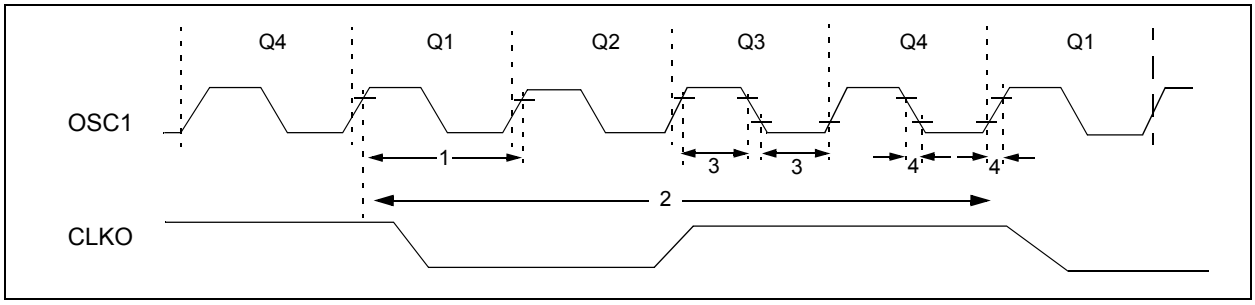


TABLE 28-7: EXTERNAL CLOCK TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	FOSC	External CLKI Frequency(1)	DC	48	MHz	EC Oscillator mode
			DC	48		ECPLL Oscillator mode(2)
		Oscillator Frequency(1)	4	25	MHz	HS Oscillator mode
			4	25		HSPLL Oscillator mode(3)
1	TOSC	External CLKI Period(1)	20.8	—	ns	EC Oscillator mode
			20.8	—		ECPLL Oscillator mode(2)
		Oscillator Period(1)	40.0	250	ns	HS Oscillator mode
			40.0	250		HSPLL Oscillator mode(3)
2	TCY	Instruction Cycle Time(1)	83.3	—	ns	TCY = 4/FOSC, Industrial
3	TOSL, TOSH	External Clock in (OSC1) High or Low Time	10	—	ns	EC Oscillator mode
4	TOSR, TOSF	External Clock in (OSC1) Rise or Fall Time	—	7.5	ns	EC Oscillator mode

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations except PLL. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

2: In order to use the PLL, the external clock frequency must be either 4, 8, 12, 16, 20, 24, 40 or 48 MHz.

3: In order to use the PLL, the crystal/resonator must produce a frequency of either 4, 8, 12, 16, 20 or 24 MHz.