



Welcome to [E-XFL.COM](https://www.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	4MHz
Connectivity	I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	22
Program Memory Size	7KB (4K x 14)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	192 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 5.5V
Data Converters	A/D 5x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16f873-04i-so

PIC16F87X

2.2 Data Memory Organization

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Bits RP1 (STATUS<6>) and RP0 (STATUS<5>) are the bank select bits.

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function Registers. Some frequently used Special Function Registers from one bank may be mirrored in another bank for code reduction and quicker access.

Note: EEPROM Data Memory description can be found in Section 4.0 of this data sheet.

2.2.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly, or indirectly through the File Select Register (FSR).

PIC16F87X

2.2.2.5 PIR1 Register

The PIR1 register contains the individual flag bits for the peripheral interrupts.

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt bits are clear prior to enabling an interrupt.

REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

- bit 7 **PSPIF⁽¹⁾:** Parallel Slave Port Read/Write Interrupt Flag bit
 1 = A read or a write operation has taken place (must be cleared in software)
 0 = No read or write has occurred
- bit 6 **ADIF:** A/D Converter Interrupt Flag bit
 1 = An A/D conversion completed
 0 = The A/D conversion is not complete
- bit 5 **RCIF:** USART Receive Interrupt Flag bit
 1 = The USART receive buffer is full
 0 = The USART receive buffer is empty
- bit 4 **TXIF:** USART Transmit Interrupt Flag bit
 1 = The USART transmit buffer is empty
 0 = The USART transmit buffer is full
- bit 3 **SSPIF:** Synchronous Serial Port (SSP) Interrupt Flag
 1 = The SSP interrupt condition has occurred, and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
- SPI
 - A transmission/reception has taken place.
 - I²C Slave
 - A transmission/reception has taken place.
 - I²C Master
 - A transmission/reception has taken place.
 - The initiated START condition was completed by the SSP module.
 - The initiated STOP condition was completed by the SSP module.
 - The initiated Restart condition was completed by the SSP module.
 - The initiated Acknowledge condition was completed by the SSP module.
 - A START condition occurred while the SSP module was idle (Multi-Master system).
 - A STOP condition occurred while the SSP module was idle (Multi-Master system).
- 0 = No SSP interrupt condition has occurred.
- bit 2 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
 1 = A TMR1 register capture occurred (must be cleared in software)
 0 = No TMR1 register capture occurred
Compare mode:
 1 = A TMR1 register compare match occurred (must be cleared in software)
 0 = No TMR1 register compare match occurred
PWM mode:
 Unused in this mode
- bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
 1 = TMR2 to PR2 match occurred (must be cleared in software)
 0 = No TMR2 to PR2 match occurred
- bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
 1 = TMR1 register overflowed (must be cleared in software)
 0 = TMR1 register did not overflow

Note 1: PSPIF is reserved on PIC16F873/876 devices; always maintain this bit clear.

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

3.3 PORTC and the TRISC Register

PORTC is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

PORTC is multiplexed with several peripheral functions (Table 3-5). PORTC pins have Schmitt Trigger input buffers.

When the I²C module is enabled, the PORTC<4:3> pins can be configured with normal I²C levels, or with SMBus levels by using the CKE bit (SSPSTAT<6>).

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (BSF, BCF, XORWF) with TRISC as destination, should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

FIGURE 3-5: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<2:0>, RC<7:5>

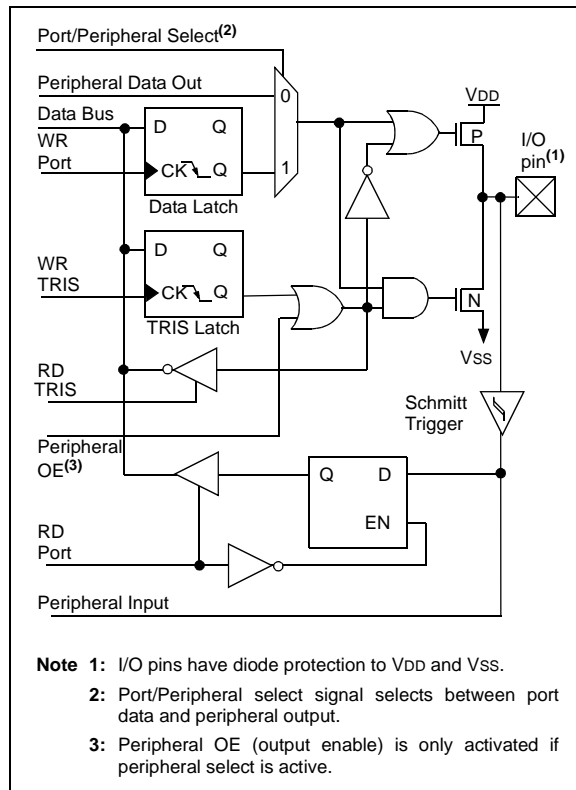
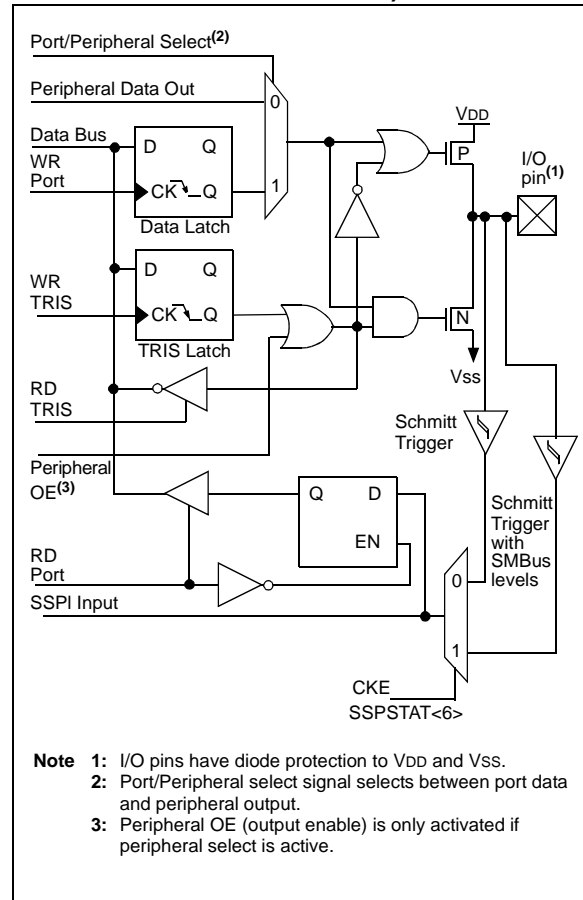


FIGURE 3-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<4:3>



PIC16F87X

3.5 PORTE and TRISE Register

PORTE and TRISE are not implemented on the PIC16F873 or PIC16F876.

PORTE has three pins (RE0/ $\overline{\text{RD}}$ /AN5, RE1/ $\overline{\text{WR}}$ /AN6, and RE2/ $\overline{\text{CS}}$ /AN7) which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become the I/O control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make certain that the TRISE<2:0> bits are set, and that the pins are configured as digital inputs. Also ensure that ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

Register 3-1 shows the TRISE register, which also controls the parallel slave port operation.

PORTE pins are multiplexed with analog inputs. When selected for analog input, these pins will read as '0's.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

Note: On a Power-on Reset, these pins are configured as analog inputs, and read as '0'.

FIGURE 3-8: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)

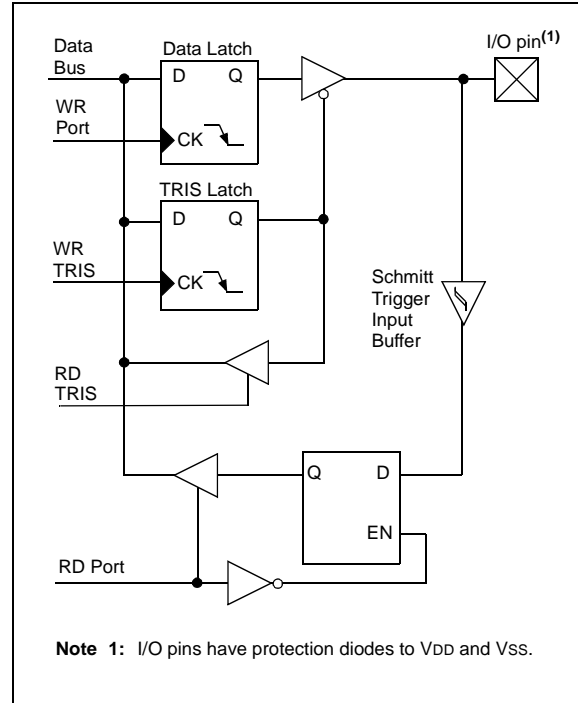


TABLE 3-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/ $\overline{\text{RD}}$ /AN5	bit0	ST/TTL ⁽¹⁾	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{\text{RD}}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected)
RE1/ $\overline{\text{WR}}$ /AN6	bit1	ST/TTL ⁽¹⁾	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{\text{WR}}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected)
RE2/ $\overline{\text{CS}}$ /AN7	bit2	ST/TTL ⁽¹⁾	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{\text{CS}}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 3-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Address	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
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

5.0 TIMER0 MODULE

The Timer0 module timer/counter has the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

Figure 5-1 is a block diagram of the Timer0 module and the prescaler shared with the WDT.

Additional information on the Timer0 module is available in the PIC® MCU Mid-Range Family Reference Manual (DS33023).

Timer mode is selected by clearing bit T0CS (OPTION_REG<5>). In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

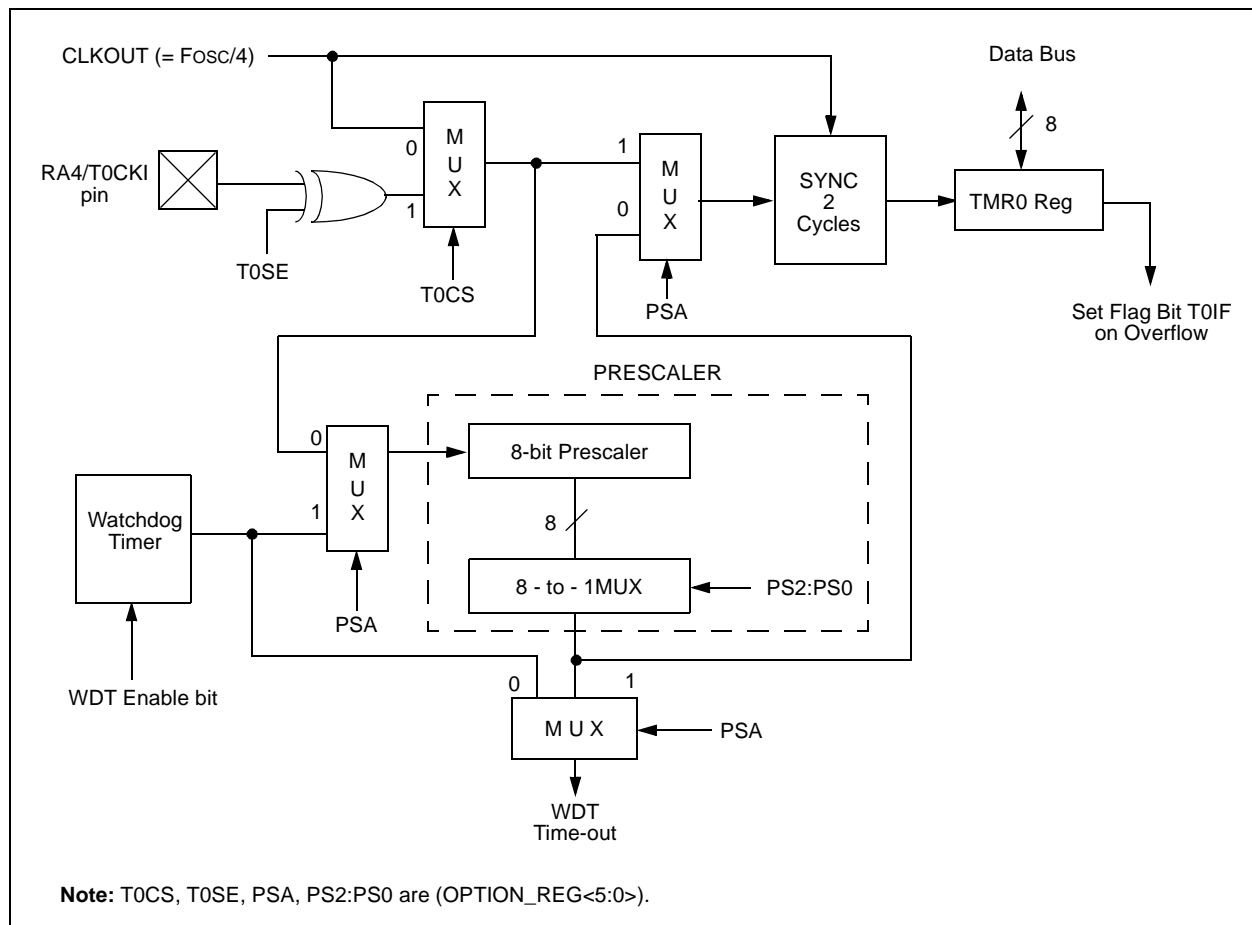
Counter mode is selected by setting bit T0CS (OPTION_REG<5>). In Counter mode, Timer0 will increment either on every rising, or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (OPTION_REG<4>). Clearing bit T0SE selects the rising edge. Restrictions on the external clock input are discussed in detail in Section 5.2.

The prescaler is mutually exclusively shared between the Timer0 module and the Watchdog Timer. The prescaler is not readable or writable. Section 5.3 details the operation of the prescaler.

5.1 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>). The interrupt can be masked by clearing bit T0IE (INTCON<5>). Bit T0IF must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP, since the timer is shut-off during SLEEP.

FIGURE 5-1: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER



9.1 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

- Serial Data Out (SDO)
- Serial Data In (SDI)
- Serial Clock (SCK)

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select (\overline{SS})

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data input sample phase (middle or end of data output time)
- Clock edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

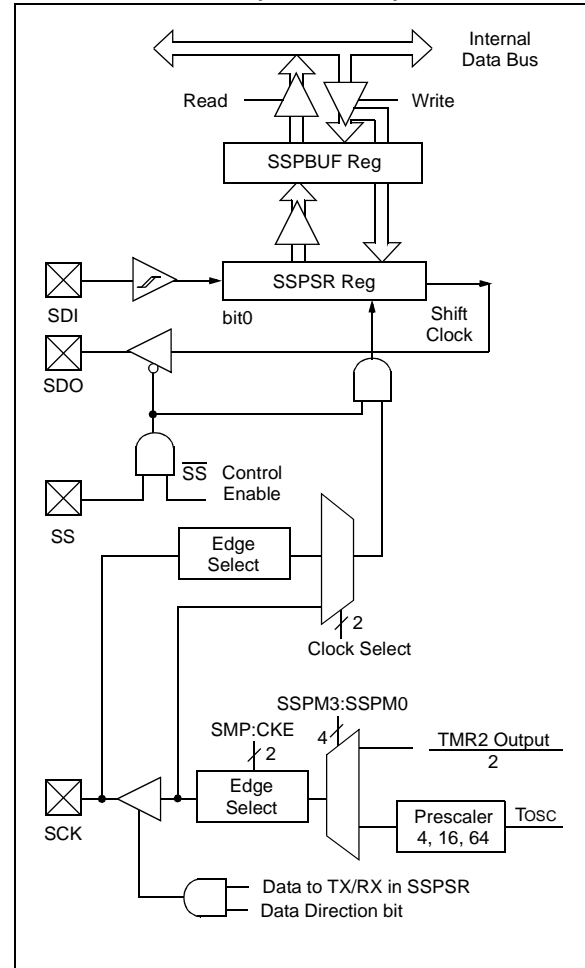
Figure 9-4 shows the block diagram of the MSSP module when in SPI mode.

To enable the serial port, MSSP Enable bit, SSPEN (SSPCON<5>) must be set. To reset or reconfigure SPI mode, clear bit SSPEN, re-initialize the SSPCON registers, and then set bit SSPEN. This configures the SDI, SDO, SCK and \overline{SS} pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed. That is:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> cleared
- SCK (Master mode) must have TRISC<3> cleared
- SCK (Slave mode) must have TRISC<3> set
- \overline{SS} must have TRISA<5> set and register ADCON1 (see Section 11.0: A/D Module) must be set in a way that pin RA5 is configured as a digital I/O

Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

FIGURE 9-1: MSSP BLOCK DIAGRAM (SPI MODE)



9.2.7 I²C MASTER MODE SUPPORT

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON and by setting the SSPEN bit. Once Master mode is enabled, the user has six options:

- Assert a START condition on SDA and SCL.
- Assert a Repeated START condition on SDA and SCL.
- Write to the SSPBUF register initiating transmission of data/address.
- Generate a STOP condition on SDA and SCL.
- Configure the I²C port to receive data.
- Generate an Acknowledge condition at the end of a received byte of data.

Note: The MSSP Module, when configured in I²C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a START condition and immediately write the SSPBUF register to initiate transmission before the START condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

9.2.7.1 I²C Master Mode Operation

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a Repeated START condition. Since the Repeated START condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/W) bit. In this case, the R/W bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/W bit. In this case, the R/W bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an Acknowledge bit is transmitted. START and STOP conditions indicate the beginning and end of transmission.

The baud rate generator used for SPI mode operation is now used to set the SCL clock frequency for either 100 kHz, 400 kHz, or 1 MHz I²C operation. The baud rate generator reload value is contained in the lower 7 bits of the SSPADD register. The baud rate generator will automatically begin counting on a write to the

SSPBUF. Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

A typical transmit sequence would go as follows:

- User generates a START condition by setting the START enable bit (SEN) in SSPCON2.
- SSPIF is set. The module will wait the required start time before any other operation takes place.
- User loads SSPBUF with address to transmit.
- Address is shifted out the SDA pin until all 8 bits are transmitted.
- MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
- MSSP module generates an interrupt at the end of the ninth clock cycle by setting SSPIF.
- User loads SSPBUF with eight bits of data.
- DATA is shifted out the SDA pin until all 8 bits are transmitted.
- MSSP module shifts in the ACK bit from the slave device, and writes its value into the SSPCON2 register (SSPCON2<6>).
- MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
- User generates a STOP condition by setting the STOP enable bit, PEN, in SSPCON2.
- Interrupt is generated once the STOP condition is complete.

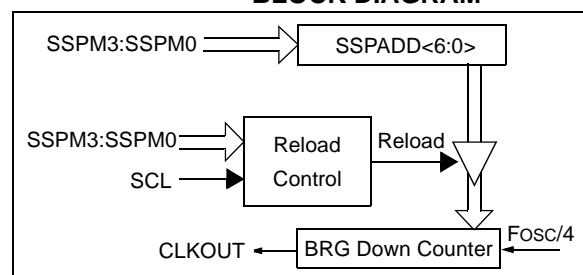
9.2.8 BAUD RATE GENERATOR

In I²C Master mode, the reload value for the BRG is located in the lower 7 bits of the SSPADD register (Figure 9-10). When the BRG is loaded with this value, the BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (Tcy), on the Q2 and Q4 clock.

In I²C Master mode, the BRG is reloaded automatically. If clock arbitration is taking place, the BRG will be reloaded when the SCL pin is sampled high (Figure 9-11).

Note: Baud Rate = $F_{osc} / (4 * (SSPADD + 1))$

FIGURE 9-10: BAUD RATE GENERATOR BLOCK DIAGRAM



9.2.18 MULTI-MASTER COMMUNICATION, BUS COLLISION, AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin = '0', a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF and reset the I²C port to its IDLE state (Figure 9-19).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are de-asserted, and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine, and if the I²C bus is free, the user can resume communication by asserting a START condition.

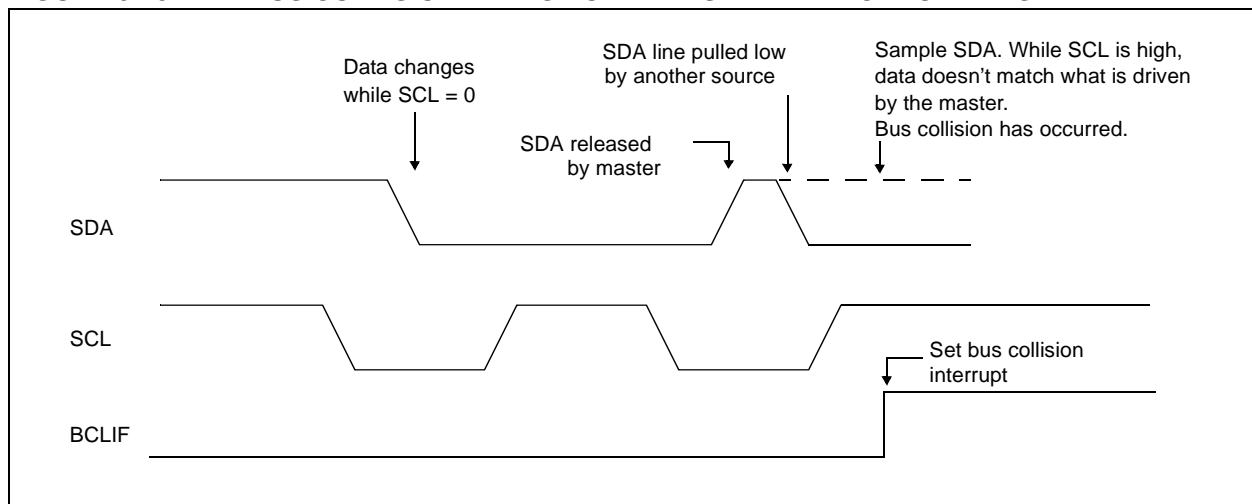
If a START, Repeated START, STOP, or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are de-asserted, and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, and if the I²C bus is free, the user can resume communication by asserting a START condition.

The master will continue to monitor the SDA and SCL pins and if a STOP condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of START and STOP conditions allows the determination of when the bus is free. Control of the I²C bus can be taken when the P bit is set in the SSPSTAT register, or the bus is idle and the S and P bits are cleared.

FIGURE 9-19: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



PIC16F87X

TABLE 12-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

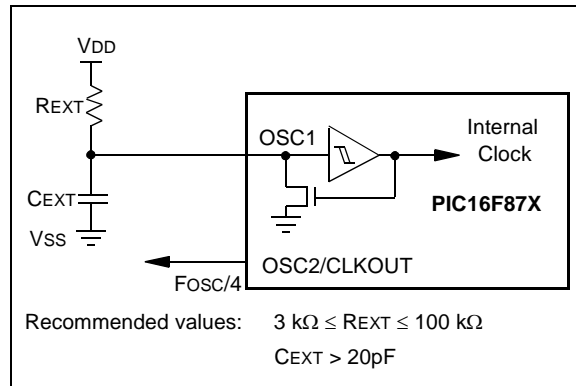
Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF
These values are for design guidance only. See notes following this table.			
Crystals Used			
32 kHz	Epson C-001R32.768K-A	± 20 PPM	
200 kHz	STD XTL 200.000KHz	± 20 PPM	
1 MHz	ECS ECS-10-13-1	± 50 PPM	
4 MHz	ECS ECS-40-20-1	± 50 PPM	
8 MHz	EPSON CA-301 8.000M-C	± 30 PPM	
20 MHz	EPSON CA-301 20.000M-C	± 30 PPM	

- Note 1:** Higher capacitance increases the stability of oscillator, but also increases the start-up time.
- 2:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
- 3:** R_s may be required in HS mode, as well as XT mode, to avoid overdriving crystals with low drive level specification.
- 4:** When migrating from other PIC® MCU devices, oscillator performance should be verified.

12.2.3 RC OSCILLATOR

For timing insensitive applications, the “RC” device option offers additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (R_{EXT}) and capacitor (C_{EXT}) values, and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low C_{EXT} values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 12-3 shows how the R/C combination is connected to the PIC16F87X.

FIGURE 12-3: RC OSCILLATOR MODE



PIC16F87X

12.10.1 INT INTERRUPT

External interrupt on the RB0/INT pin is edge triggered, either rising, if bit INTEDG (OPTION_REG<6>) is set, or falling, if the INTEDG bit is clear. When a valid edge appears on the RB0/INT pin, flag bit INTF (INTCON<1>) is set. This interrupt can be disabled by clearing enable bit INTE (INTCON<4>). Flag bit INTF must be cleared in software in the Interrupt Service Routine before re-enabling this interrupt. The INT interrupt can wake-up the processor from SLEEP, if bit INTE was set prior to going into SLEEP. The status of global interrupt enable bit, GIE, decides whether or not the processor branches to the interrupt vector following wake-up. See Section 12.13 for details on SLEEP mode.

12.10.2 TMR0 INTERRUPT

An overflow (FFh → 00h) in the TMR0 register will set flag bit T0IF (INTCON<2>). The interrupt can be enabled/disabled by setting/clearing enable bit T0IE (INTCON<5>) (Section 5.0).

12.10.3 PORTB INTCON CHANGE

An input change on PORTB<7:4> sets flag bit RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit RBIE (INTCON<4>) (Section 3.2).

12.11 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt, (i.e., W register and STATUS register). This will have to be implemented in software.

For the PIC16F873/874 devices, the register W_TEMP must be defined in both banks 0 and 1 and must be defined at the same offset from the bank base address (i.e., If W_TEMP is defined at 0x20 in bank 0, it must also be defined at 0xA0 in bank 1). The registers, PCLATH_TEMP and STATUS_TEMP, are only defined in bank 0.

Since the upper 16 bytes of each bank are common in the PIC16F876/877 devices, temporary holding registers W_TEMP, STATUS_TEMP, and PCLATH_TEMP should be placed in here. These 16 locations don't require banking and therefore, make it easier for context save and restore. The same code shown in Example 12-1 can be used.

EXAMPLE 12-1: SAVING STATUS, W, AND PCLATH REGISTERS IN RAM

```
MOVWF    W_TEMP          ;Copy W to TEMP register
SWAPF    STATUS,W         ;Swap status to be saved into W
CLRF     STATUS          ;bank 0, regardless of current bank, Clears IRP,RP1,RP0
MOVWF    STATUS_TEMP      ;Save status to bank zero STATUS_TEMP register
MOVF     PCLATH, W        ;Only required if using pages 1, 2 and/or 3
MOVWF    PCLATH_TEMP      ;Save PCLATH into W
CLRF     PCLATH           ;Page zero, regardless of current page
:
: (ISR)                   ; (Insert user code here)
:
MOVF     PCLATH_TEMP, W    ;Restore PCLATH
MOVWF    PCLATH           ;Move W into PCLATH
SWAPF    STATUS_TEMP,W    ;Swap STATUS_TEMP register into W
                        ; (sets bank to original state)
MOVWF    STATUS           ;Move W into STATUS register
SWAPF    W_TEMP,F         ;Swap W_TEMP
SWAPF    W_TEMP,W         ;Swap W_TEMP into W
```

14.0 DEVELOPMENT SUPPORT

The PIC[®] microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
 - MPLAB[®] IDE Software
- Assemblers/Compilers/Linkers
 - MPASM[™] Assembler
 - MPLAB C17 and MPLAB C18 C Compilers
 - MPLINK[™] Object Linker/
MPLIB[™] Object Librarian
- Simulators
 - MPLAB SIM Software Simulator
- Emulators
 - MPLAB ICE 2000 In-Circuit Emulator
 - ICEPIC[™] In-Circuit Emulator
- In-Circuit Debugger
 - MPLAB ICD for PIC16F87X
- Device Programmers
 - PRO MATE[®] II Universal Device Programmer
 - PICSTART[®] Plus Entry-Level Development Programmer
- Low Cost Demonstration Boards
 - PICDEM[™] 1 Demonstration Board
 - PICDEM 2 Demonstration Board
 - PICDEM 3 Demonstration Board
 - PICDEM 17 Demonstration Board
 - KEELQ[®] Demonstration Board

14.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. The MPLAB IDE is a Windows[®]-based application that contains:

- An interface to debugging tools
 - simulator
 - programmer (sold separately)
 - emulator (sold separately)
 - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or 'C')
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
 - source files
 - absolute listing file
 - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

14.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PIC[®] MCUs.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. The MPASM assembler generates relocatable object files for the MPLINK object linker, Intel[®] standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file that contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

- Integration into MPLAB IDE projects.
- User-defined macros to streamline assembly code.
- Conditional assembly for multi-purpose source files.
- Directives that allow complete control over the assembly process.

14.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI 'C' compilers for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

FIGURE 15-3: PIC16F87X-04 VOLTAGE-FREQUENCY GRAPH (ALL TEMPERATURE RANGES)

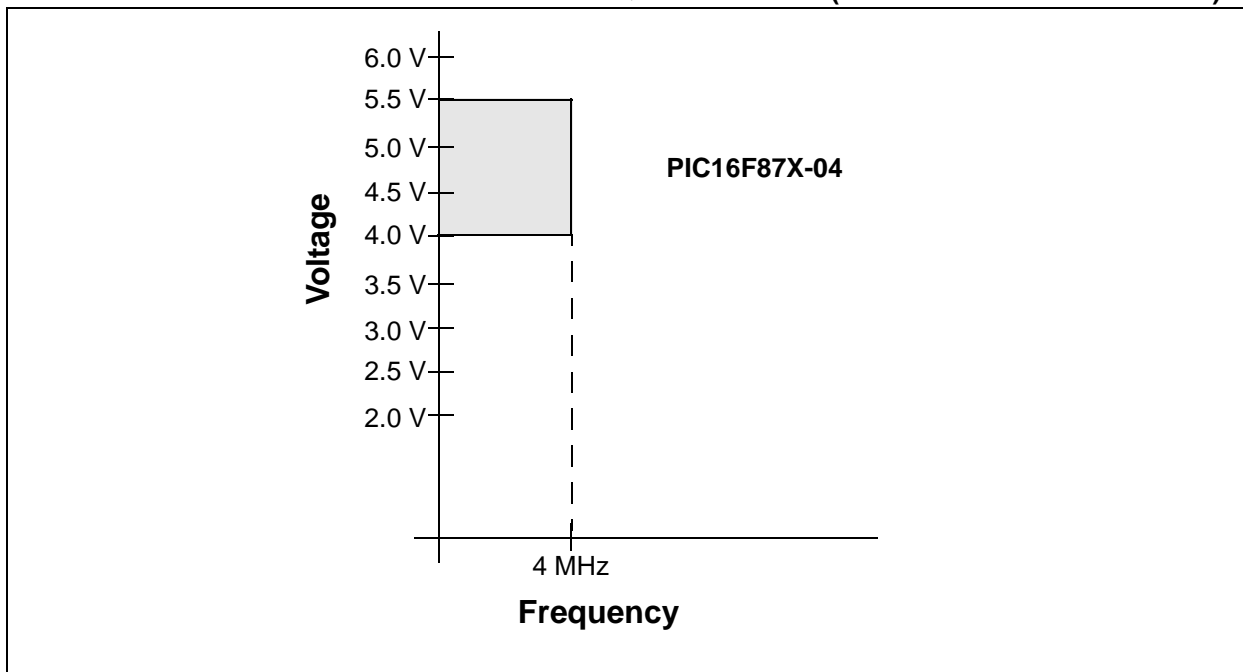
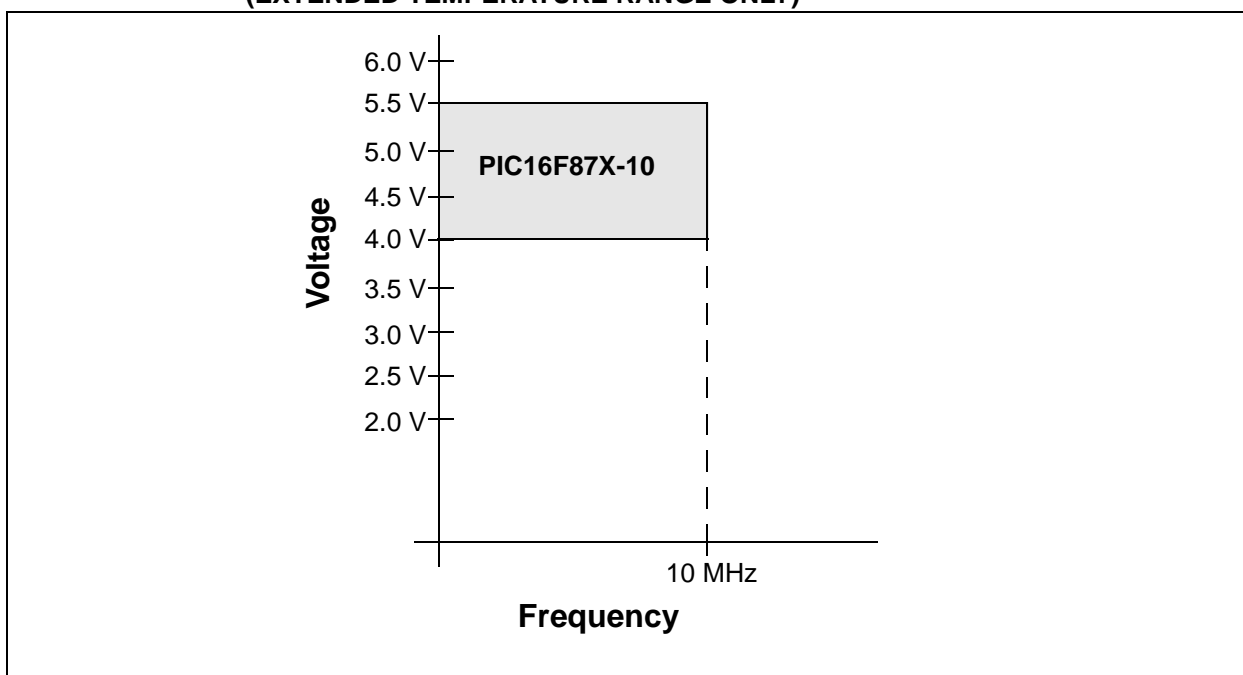


FIGURE 15-4: PIC16F87X-10 VOLTAGE-FREQUENCY GRAPH (EXTENDED TEMPERATURE RANGE ONLY)



15.1 DC Characteristics: PIC16F873/874/876/877-04 (Commercial, Industrial) PIC16F873/874/876/877-20 (Commercial, Industrial) PIC16LF873/874/876/877-04 (Commercial, Industrial) (Continued)

PIC16LF873/874/876/877-04 (Commercial, Industrial)			Standard Operating Conditions (unless otherwise stated)				
			Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ for commercial				
PIC16F873/874/876/877-04 PIC16F873/874/876/877-20 (Commercial, Industrial)			Standard Operating Conditions (unless otherwise stated)				
			Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ for commercial				
Param No.	Symbol	Characteristic/ Device	Min	Typ†	Max	Units	Conditions
D010	IDD	Supply Current^(2,5)					
		16LF87X	—	0.6	2.0	mA	XT, RC osc configuration FOSC = 4 MHz, VDD = 3.0V
D010		16F87X	—	1.6	4	mA	RC osc configurations FOSC = 4 MHz, VDD = 5.5V
D010A		16LF87X	—	20	35	μA	LP osc configuration FOSC = 32 kHz, VDD = 3.0V, WDT disabled
D013		16F87X	—	7	15	mA	HS osc configuration, FOSC = 20 MHz, VDD = 5.5V
D015	ΔIBOR	Brown-out Reset Current⁽⁶⁾	—	85	200	μA	BOR enabled, VDD = 5.0V

Legend: Rows with standard voltage device data only are shaded for improved readability.

† Data in "Typ" column is at 5V, 25°C, unless otherwise stated. These parameters are for design guidance only, and are not tested.

Note 1: This is the limit to which VDD can be lowered without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading, switching rate, oscillator type, internal code execution pattern and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tri-stated, pulled to VDD;

MCLR = VDD; WDT enabled/disabled as specified.

3: 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 hi-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula $I_r = V_{DD}/2R_{EXT}$ (mA) with REXT in kOhm.

5: Timer1 oscillator (when enabled) adds approximately 20 μA to the specification. This value is from characterization and is for design guidance only. This is not tested.

6: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base IDD or IPD measurement.

7: When BOR is enabled, the device will operate correctly until the VBOR voltage trip point is reached.

FIGURE 15-15: SPI SLAVE MODE TIMING (CKE = 0)

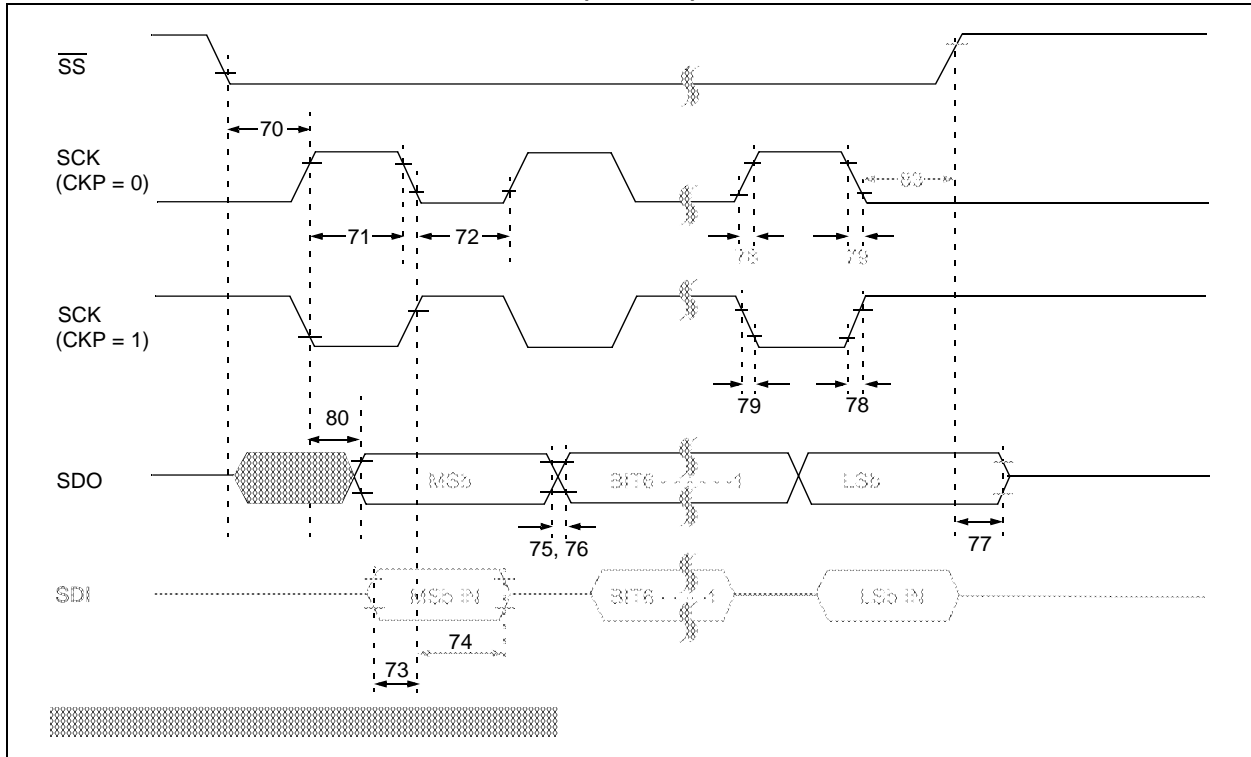
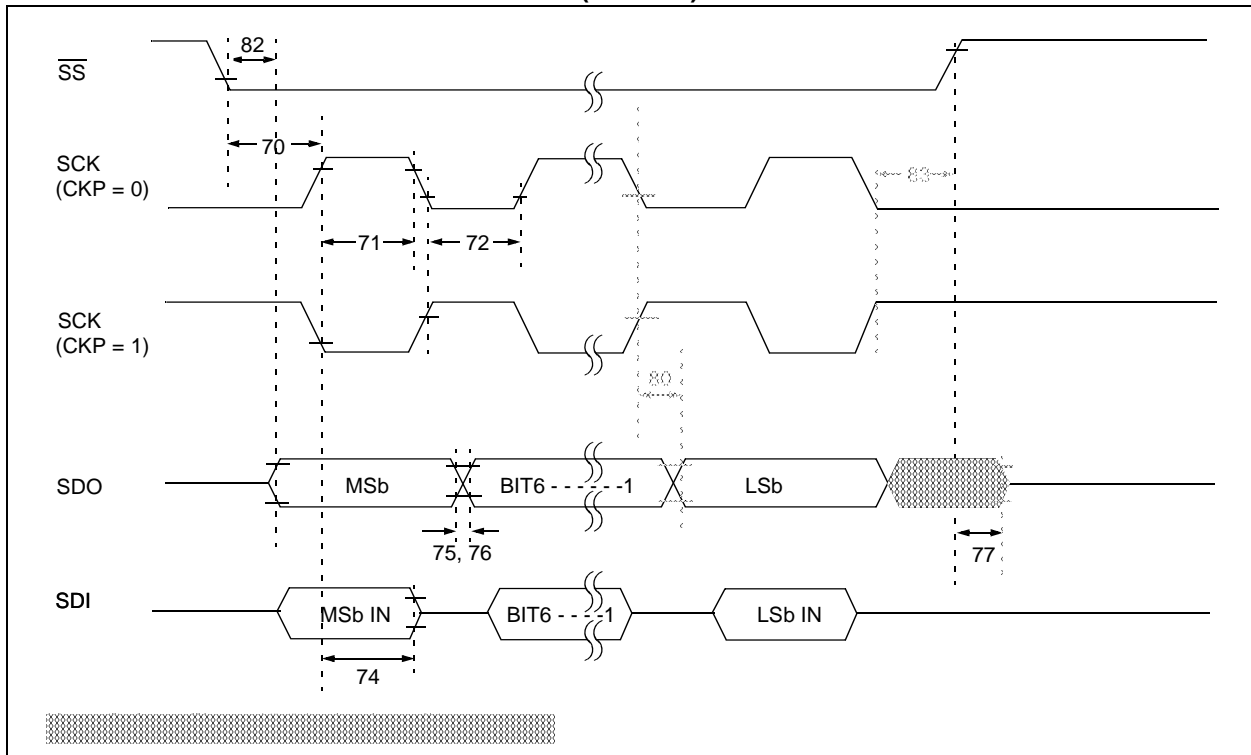


FIGURE 15-16: SPI SLAVE MODE TIMING (CKE = 1)



16.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

The graphs and tables provided in this section are for **design guidance** and are **not tested**.

In some graphs or tables, the data presented is **outside specified operating range** (i.e., outside specified V_{DD} range). This is for **information only** and devices are ensured to operate properly only within the specified range.

The data presented in this section is a **statistical summary** of data collected on units from different lots over a period of time and matrix samples. 'Typical' represents the mean of the distribution at 25°C. 'max' or 'min' represents (mean + 3 σ) or (mean - 3 σ) respectively, where σ is standard deviation, over the whole temperature range.

FIGURE 16-1: TYPICAL I_{DD} vs. F_{osc} OVER V_{DD} (HS MODE)

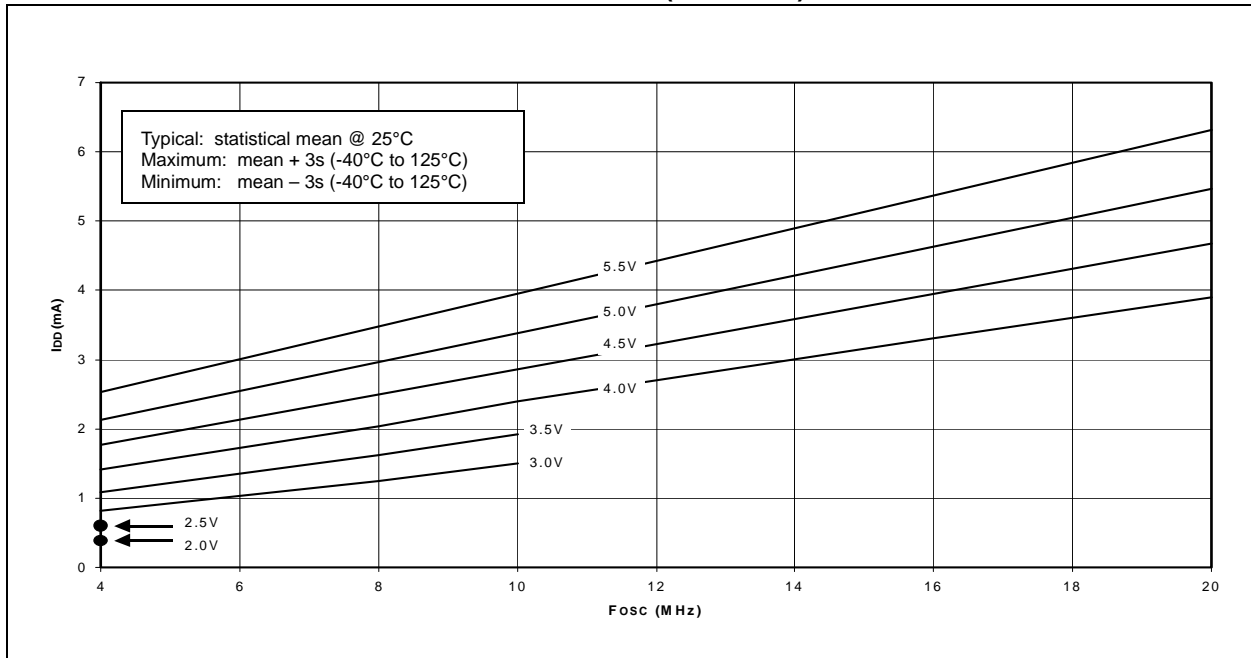
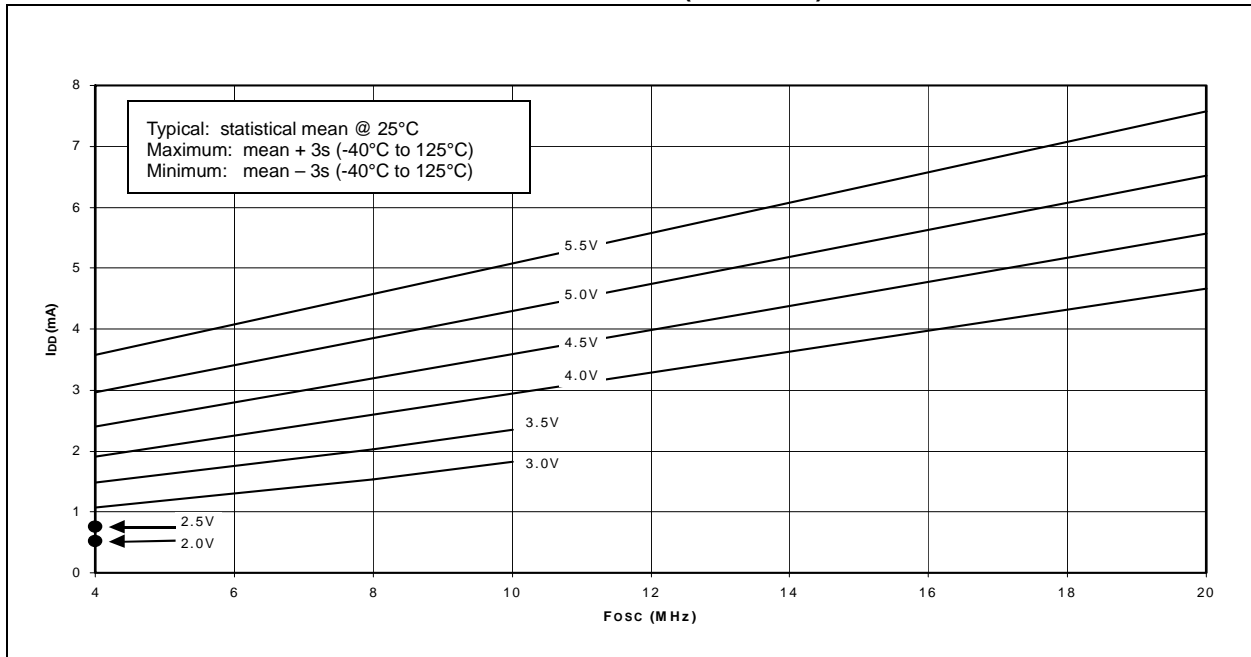
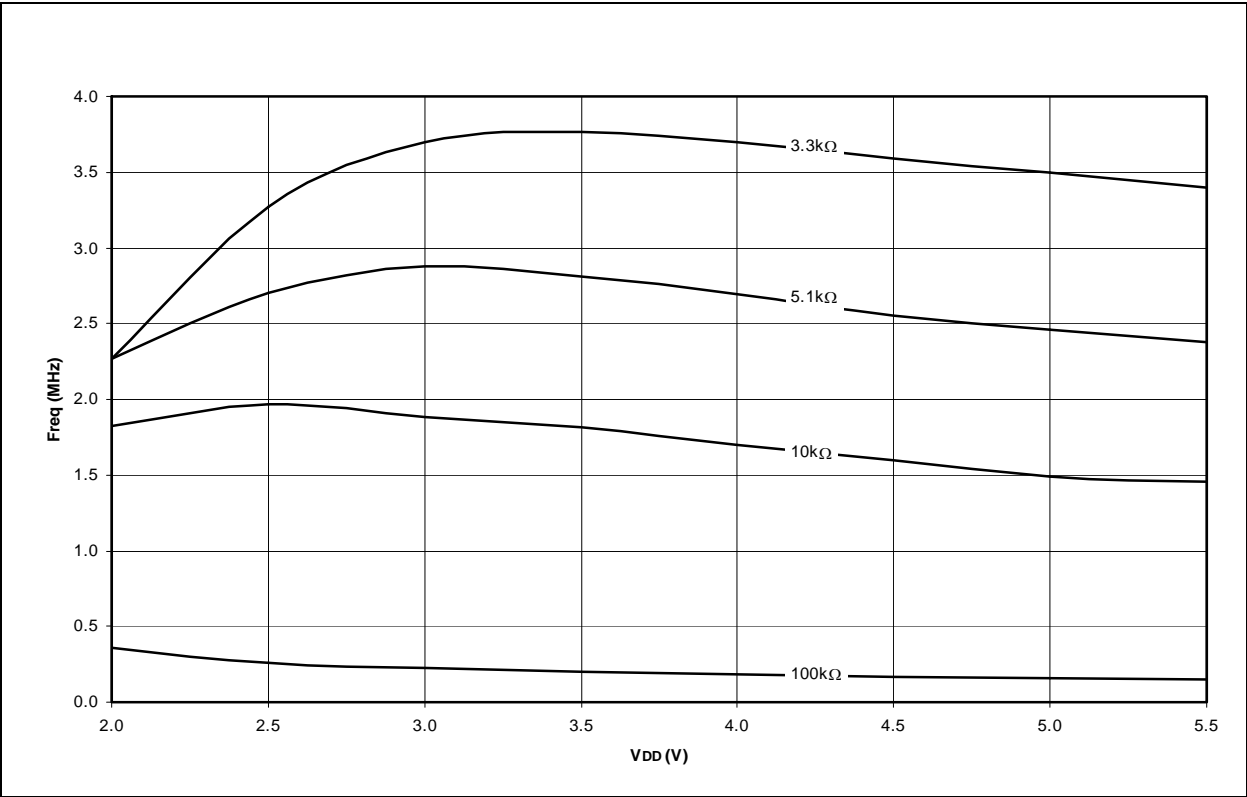


FIGURE 16-2: MAXIMUM I_{DD} vs. F_{osc} OVER V_{DD} (HS MODE)

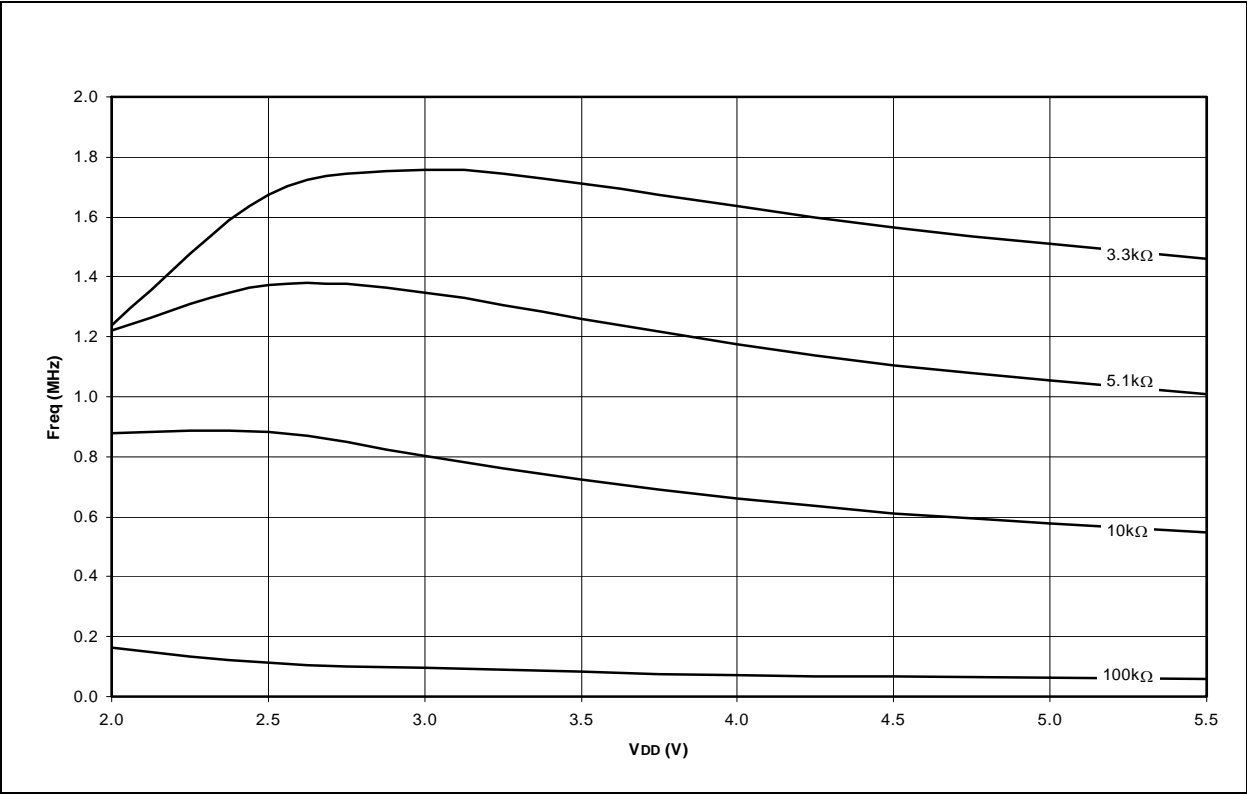


PIC16F87X

**FIGURE 16-7: AVERAGE F_{osc} vs. V_{DD} FOR VARIOUS VALUES OF R
(RC MODE, C = 20 pF, 25°C)**



**FIGURE 16-8: AVERAGE F_{osc} vs. V_{DD} FOR VARIOUS VALUES OF R
(RC MODE, C = 100 pF, 25°C)**



PIC16F87X

FIGURE 16-11: ΔI_{BOR} vs. V_{DD} OVER TEMPERATURE

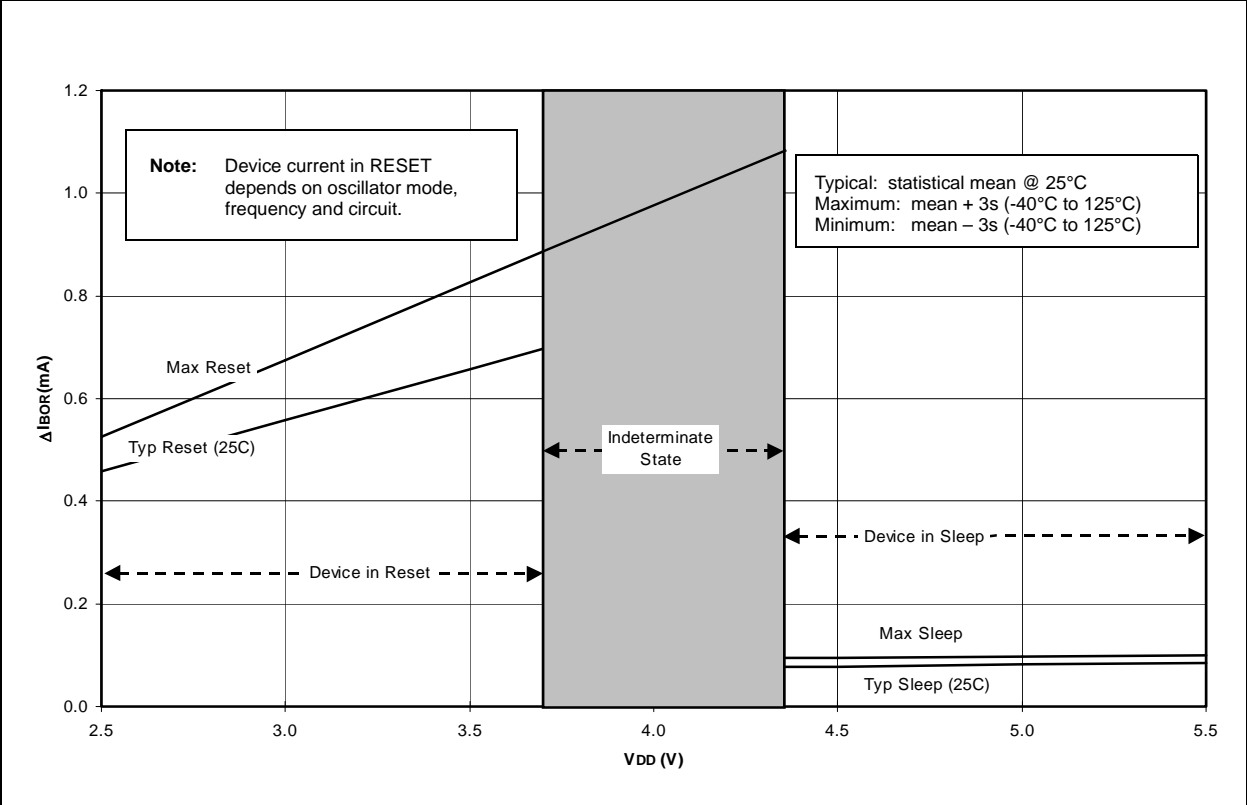
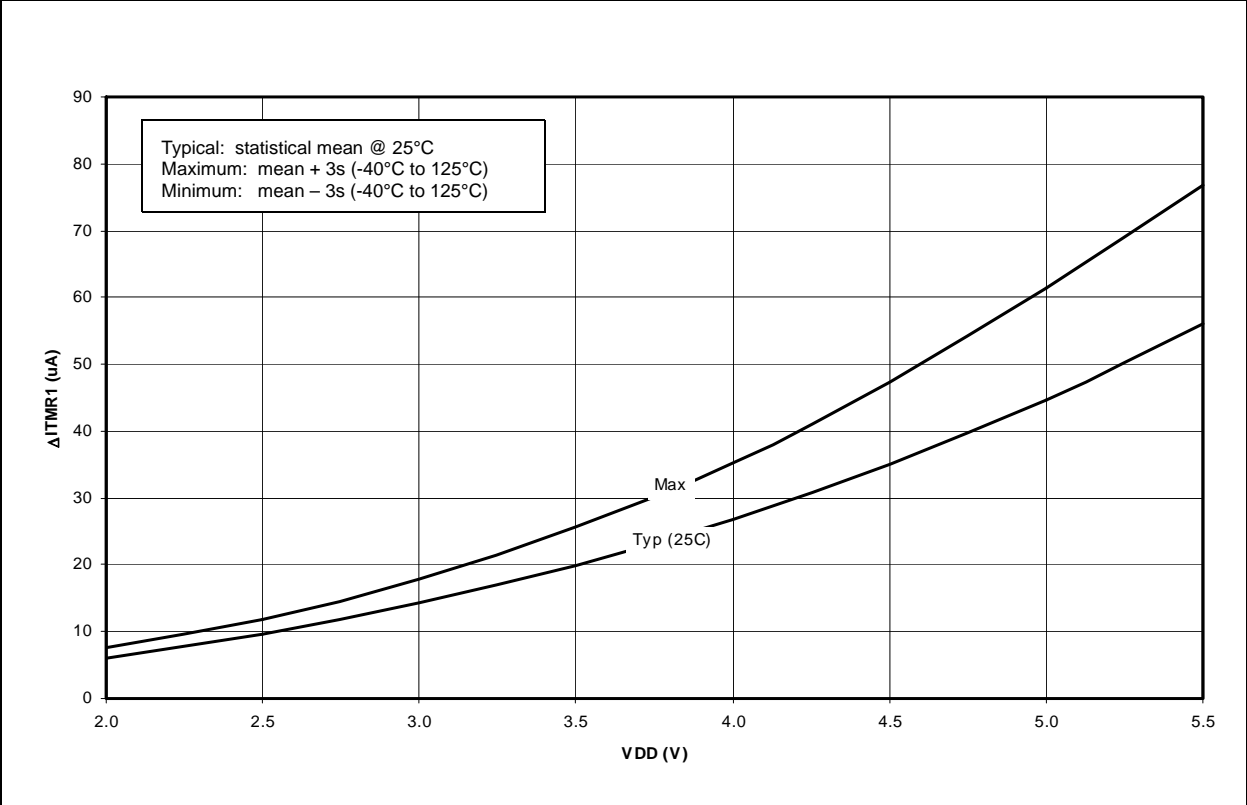


FIGURE 16-12: TYPICAL AND MAXIMUM ΔI_{TMR1} vs. V_{DD} OVER TEMPERATURE (-10°C TO 70°C, TIMER1 WITH OSCILLATOR, XTAL=32 kHz, C1 AND C2=50 pF)



INDEX

A

A/D	111
Acquisition Requirements	114
ADCON0 Register	111
ADCON1 Register	112
ADIF bit	112
Analog Input Model Block Diagram	114
Analog Port Pins	7, 8, 9, 36, 38
Associated Registers and Bits	117
Block Diagram	113
Calculating Acquisition Time	114
Configuring Analog Port Pins	115
Configuring the Interrupt	113
Configuring the Module	113
Conversion Clock	115
Conversions	116
Delays	114
Effects of a RESET	117
GO/DONE bit	112
Internal Sampling Switch (Rss) Impedance	114
Operation During SLEEP	117
Result Registers	116
Sampling Requirements	114
Source Impedance	114
Time Delays	114
Absolute Maximum Ratings	149
ACK	74
Acknowledge Data bit	68
Acknowledge Pulse	74
Acknowledge Sequence Enable bit	68
Acknowledge Status bit	68
ADRES Register	15, 111
Analog Port Pins. See A/D	
Analog-to-Digital Converter. See A/D	
Application Notes	
AN552 (Implementing Wake-up on Key Strokes	
Using PIC16CXXX)	31
AN556 (Implementing a Table Read)	26
AN578 (Use of the SSP Module in the I2C	
Multi-Master Environment)	73
Architecture	
PIC16F873/PIC16F876 Block Diagram	5
PIC16F874/PIC16F877 Block Diagram	6
Assembler	
MPASM Assembler	143

B

Banking, Data Memory	12, 18
Baud Rate Generator	79
BCLIF	24
BF	74, 82, 84
Block Diagrams	
A/D	113
A/D Converter	113
Analog Input Model	114
Baud Rate Generator	79
Capture Mode	59
Compare Mode	60
I ² C Master Mode	78
I ² C Module	73
I ² C Slave Mode	73
Interrupt Logic	129
PIC16F873/PIC16F876	5

PIC16F874/PIC16F877	6
PORTA	
RA3:RA0 and RA5 Pins	29
RA4/T0CKI Pin	29
PORTB	
RB3:RB0 Port Pins	31
RB7:RB4 Port Pins	31
PORTC	
Peripheral Output Override (RC 0:2, 5:7)	33
Peripheral Output Override (RC 3:4)	33
PORTD	35
PORTD and PORTE (Parallel Slave Port)	38
PORTE	36
PWM Mode	61
RESET Circuit	123
SSP (I ² C Mode)	73
SSP (SPI Mode)	69
Timer0/WDT Prescaler	47
Timer1	52
Timer2	55
USART Asynchronous Receive	101
USART Asynchronous Receive (9-bit Mode)	103
USART Transmit	99
Watchdog Timer	131

BOR. See Brown-out Reset

BRG	79
BRGH bit	97
Brown-out Reset (BOR)	119, 123, 125, 126
BOR Status (BOR Bit)	25
Buffer Full bit, BF	74
Bus Arbitration	89
Bus Collision Section	89
Bus Collision During a Repeated START Condition	92
Bus Collision During a START Condition	90
Bus Collision During a STOP Condition	93
Bus Collision Interrupt Flag bit, BCLIF	24

C

Capture/Compare/PWM (CCP)	57
Associated Registers	
Capture, Compare and Timer1	62
PWM and Timer2	63
Capture Mode	59
Block Diagram	59
CCP1CON Register	58
CCP1IF	59
Prescaler	59
CCP Timer Resources	57
CCP1	
RC2/CCP1 Pin	7, 9
CCP2	
RC1/T1OSI/CCP2 Pin	7, 9
Compare	
Special Trigger Output of CCP1	60
Special Trigger Output of CCP2	60
Compare Mode	60
Block Diagram	60
Software Interrupt Mode	60
Special Event Trigger	60
Interaction of Two CCP Modules (table)	57

Bus Collision During a Repeated START Condition (Case 1)	92
Bus Collision During a Repeated START Condition (Case2)	92
Bus Collision During a START Condition (SCL = 0)	91
Bus Collision During a STOP Condition	93
Bus Collision for Transmit and Acknowledge	89
Capture/Compare/PWM	166
CLKOUT and I/O	163
I ² C Bus Data	171
I ² C Bus START/STOP bits	170
I ² C Master Mode First START Bit Timing	80
I ² C Master Mode Reception Timing	85
I ² C Master Mode Transmission Timing	83
Master Mode Transmit Clock Arbitration	88
Power-up Timer	164
Repeat START Condition	81
RESET	164
SPI Master Mode	70
SPI Slave Mode (CKE = 1)	71
SPI Slave Mode Timing (CKE = 0)	71
Start-up Timer	164
STOP Condition Receive or Transmit	87
Time-out Sequence on Power-up	127, 128
Timer0	165
Timer1	165
USART Asynchronous Master Transmission	100
USART Asynchronous Reception	102
USART Synchronous Receive	173
USART Synchronous Reception	108
USART Synchronous Transmission	106, 173
USART, Asynchronous Reception	104
Wake-up from SLEEP via Interrupt	133
Watchdog Timer	164
TMR0	17
TMR0 Register	15
TMR1CS bit	51
TMR1H	17
TMR1H Register	15
TMR1L	17
TMR1L Register	15
TMR1ON bit	51
TMR2	17
TMR2 Register	15
TMR2ON bit	55
TOUTPS0 bit	55
TOUTPS1 bit	55
TOUTPS2 bit	55
TOUTPS3 bit	55
TRISA Register	16
TRISB Register	16
TRISC Register	16
TRISD Register	16
TRISE Register	16, 36, 37
IBF Bit	37
IBOV Bit	37
OBF Bit	37
PSPMODE Bit	35, 36, 37, 38
TXREG	17

TXSTA Register	95
BRGH Bit	95
CSRC Bit	95
SYNC Bit	95
TRMT Bit	95
TX9 Bit	95
TX9D Bit	95
TXEN Bit	95

U

UA	66
Universal Synchronous Asynchronous Receiver Transmitter. See USART	
Update Address, UA	66
USART	95
Address Detect Enable (ADDEN Bit)	96
Asynchronous Mode	99
Asynchronous Receive	101
Associated Registers	102
Block Diagram	101
Asynchronous Receive (9-bit Mode)	103
Associated Registers	104
Block Diagram	103
Timing Diagram	104
Asynchronous Receive with Address Detect. See Asynchronous Receive (9-bit Mode).	
Asynchronous Reception	102
Asynchronous Transmitter	99
Baud Rate Generator (BRG)	97
Baud Rate Formula	97
Baud Rates, Asynchronous Mode (BRGH=0) ...	98
High Baud Rate Select (BRGH Bit)	95
Sampling	97
Clock Source Select (CSRC Bit)	95
Continuous Receive Enable (CREN Bit)	96
Framing Error (FERR Bit)	96
Mode Select (SYNC Bit)	95
Overrun Error (OERR Bit)	96
RC6/TX/CK Pin	7, 9
RC7/RX/DT Pin	7, 9
RCSTA Register	96
Receive Data, 9th bit (RX9D Bit)	96
Receive Enable, 9-bit (RX9 Bit)	96
Serial Port Enable (SPEN Bit)	95, 96
Single Receive Enable (SREN Bit)	96
Synchronous Master Mode	105
Synchronous Master Reception	107
Associated Registers	107
Synchronous Master Transmission	105
Associated Registers	106
Synchronous Slave Mode	108
Synchronous Slave Reception	109
Associated Registers	109
Synchronous Slave Transmit	108
Associated Registers	108
Transmit Block Diagram	99
Transmit Data, 9th Bit (TX9D)	95
Transmit Enable (TXEN Bit)	95
Transmit Enable, Nine-bit (TX9 Bit)	95
Transmit Shift Register Status (TRMT Bit)	95
TXSTA Register	95

PIC16F87X

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this Data Sheet.

To: Technical Publications Manager

Total Pages Sent

RE: Reader Response

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

Telephone: (____) _____ - _____ FAX: (____) _____ - _____

Application (optional):

Would you like a reply? ____Y ____N

Device: **PIC16F87X**

Literature Number: **DS30292D**

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this data sheet easy to follow? If not, why?

4. What additions to the data sheet do you think would enhance the structure and subject?

5. What deletions from the data sheet could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?

8. How would you improve our software, systems, and silicon products?
