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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	4MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	33
Program Memory Size	14KB (8K x 14)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	368 x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf877t-04-pt

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h[3]	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
01h	TMR0	Timer0 Module Register								xxxx xxxx	47
02h[3]	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
03h[3]	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
04h[3]	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	29
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	33
08h[4]	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	35
09h[4]	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	36
0Ah[1,3]	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
0Bh[3]	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
0Ch	PIR1	PSPIF[3]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	22
0Dh	PIR2	—	(5)	—	EEIF	BCLIF	—	—	CCP2IF	-r-0 0--0	24
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	51
11h	TMR2	Timer2 Module Register								0000 0000	55
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	55
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	70, 73
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	67
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	57
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	57
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	58
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	96
19h	TXREG	USART Transmit Data Register								0000 0000	99
1Ah	RCREG	USART Receive Data Register								0000 0000	101
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	57
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	57
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	58
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	116
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	111

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

2.2.2.2 OPTION_REG Register

The OPTION_REG Register is a readable and writable register, which contains various control bits to configure the TMR0 prescaler/WDT postscaler (single assignable register known also as the prescaler), the External INT Interrupt, TMR0 and the weak pull-ups on PORTB.

> **Note:** To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

REGISTER 2-2:   OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

bit 7   bit 0

bit 7   **RBPU:** PORTB Pull-up Enable bit

1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6   **INTEDG**: Interrupt Edge Select bit

1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5   **T0CS**: TMR0 Clock Source Select bit

1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4   **T0SE**: TMR0 Source Edge Select bit

1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3   **PSA**: Prescaler Assignment bit

1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0   **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

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

> **Note:** When using low voltage ICSP programming (LVP) and the pull-ups on PORTB are enabled, bit 3 in the TRISB register must be cleared to disable the pull-up on RB3 and ensure the proper operation of the device

2.2.2.8   PCON Register

The Power Control (PCON) Register contains flag bits to allow differentiation between a Power-on Reset (POR), a Brown-out Reset (BOR), a Watchdog Reset (WDT), and an external $\overline{\text{MCLR}}$ Reset.

Note:	$\overline{\text{BOR}}$ is unknown on POR. It must be set by the user and checked on subsequent RESETS to see if BOR is clear, indicating a brown-out has occurred. The BOR status bit is a "don't care" and is not predictable if the brown-out circuit is disabled (by clearing the BODEN bit in the configuration word).

REGISTER 2-8:   PCON REGISTER (ADDRESS 8Eh)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-1
—	—	—	—	—	—	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
bit 7							bit 0

bit 7-2   **Unimplemented:** Read as '0'

bit 1   **POR**: Power-on Reset Status bit

1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0   **BOR**: Brown-out Reset Status bit

1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

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

PIC16F87X

TABLE 3-1: **PORTA FUNCTIONS**

Name	Bit#	Buffer	Function
RA0/AN0	bit0	TTL	Input/output or analog input.
RA1/AN1	bit1	TTL	Input/output or analog input.
RA2/AN2	bit2	TTL	Input/output or analog input.
RA3/AN3/V_{REF}	bit3	TTL	Input/output or analog input or V$_{REF}$.
RA4/T0CKI	bit4	ST	Input/output or external clock input for Timer0. Output is open drain type.
RA5/$\overline{SS}$/AN4	bit5	TTL	Input/output or slave select input for synchronous serial port or analog input.

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

TABLE 3-2: **SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

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
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	`--0x 0000`	`--0u 0000`
85h	TRISA	—	—	PORTA Data Direction Register						`--11 1111`	`--11 1111`
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	`--0- 0000`	`--0- 0000`

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

Note: When using the SSP module in SPI Slave mode and $\overline{SS}$ enabled, the A/D converter must be set to one of the following modes, where PCFG3:PCFG0 = `0100`,`0101`, `011x`, `1101`, `1110`, `1111`.

PIC16F87X

TABLE 3-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output.
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I^{2}C modes.
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^{2}C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit6	ST	Input/output port pin or USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	bit7	ST	Input/output port pin or USART Asynchronous Receive or Synchronous Data.

Legend: ST = Schmitt Trigger input

TABLE 3-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

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
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

REGISTER 3-1: TRISE REGISTER (ADDRESS 89h)

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	Bit2	Bit1	Bit0

bit 7   bit 0

Parallel Slave Port Status/Control Bits:

bit 7   **IBF:** Input Buffer Full Status bit

   1 = A word has been received and is waiting to be read by the CPU
   0 = No word has been received

bit 6   **OBF**: Output Buffer Full Status bit

   1 = The output buffer still holds a previously written word
   0 = The output buffer has been read

bit 5   **IBOV**: Input Buffer Overflow Detect bit (in Microprocessor mode)

   1 = A write occurred when a previously input word has not been read (must be cleared in software)
   0 = No overflow occurred

bit 4   **PSPMODE**: Parallel Slave Port Mode Select bit

   1 = PORTD functions in Parallel Slave Port mode
   0 = PORTD functions in general purpose I/O mode

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

   PORTE Data Direction Bits:

bit 2   **Bit2**: Direction Control bit for pin RE2/$\overline{\text{CS}}$/AN7

   1 = Input
   0 = Output

bit 1   **Bit1**: Direction Control bit for pin RE1/$\overline{\text{WR}}$/AN6

   1 = Input
   0 = Output

bit 0   **Bit0**: Direction Control bit for pin RE0/$\overline{\text{RD}}$/AN5

   1 = Input
   0 = Output

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

NOTES:

4.2 Reading the EEPROM Data Memory

Reading EEPROM data memory only requires that the desired address to access be written to the EEADR register and clear the EEPGD bit. After the RD bit is set, data will be available in the EEDATA register on the very next instruction cycle. EEDATA will hold this value until another read operation is initiated or until it is written by firmware.

The steps to reading the EEPROM data memory are:

1. Write the address to EEDATA. Make sure that the address is not larger than the memory size of the PIC16F87X device.
2. Clear the EEPGD bit to point to EEPROM data memory.
3. Set the RD bit to start the read operation.
4. Read the data from the EEDATA register.

EXAMPLE 4-1:   EEPROM DATA READ

```
BSF    STATUS, RP1    ;
BCF    STATUS, RP0    ;Bank 2
MOVF   ADDR, W        ;Write address
MOVWF  EEADR          ;to read from
BSF    STATUS, RP0    ;Bank 3
BCF    EECON1, EEPGD  ;Point to Data memory
BSF    EECON1, RD     ;Start read operation
BCF    STATUS, RP0    ;Bank 2

MOVF   EEDATA, W      ;W = EEDATA
```

4.3 Writing to the EEPROM Data Memory

There are many steps in writing to the EEPROM data memory. Both address and data values must be written to the SFRs. The EEPGD bit must be cleared, and the WREN bit must be set, to enable writes. The WREN bit should be kept clear at all times, except when writing to the EEPROM data. The WR bit can only be set if the WREN bit was set in a previous operation, i.e., they both cannot be set in the same operation. The WREN bit should then be cleared by firmware after the write. Clearing the WREN bit before the write actually completes will not terminate the write in progress.

Writes to EEPROM data memory must also be prefaced with a special sequence of instructions, that prevent inadvertent write operations. This is a sequence of five instructions that must be executed without interruptions. The firmware should verify that a write is not in progress, before starting another cycle.

The steps to write to EEPROM data memory are:

1. If step 10 is not implemented, check the WR bit to see if a write is in progress.
2. Write the address to EEADR. Make sure that the address is not larger than the memory size of the PIC16F87X device.
3. Write the 8-bit data value to be programmed in the EEDATA register.
4. Clear the EEPGD bit to point to EEPROM data memory.
5. Set the WREN bit to enable program operations.
6. Disable interrupts (if enabled).
7. Execute the special five instruction sequence:
   • Write 55h to EECON2 in two steps (first to W, then to EECON2)
   • Write AAh to EECON2 in two steps (first to W, then to EECON2)
   • Set the WR bit
8. Enable interrupts (if using interrupts).
9. Clear the WREN bit to disable program operations.
10. At the completion of the write cycle, the WR bit is cleared and the EEIF interrupt flag bit is set. (EEIF must be cleared by firmware.) If step 1 is not implemented, then firmware should check for EEIF to be set, or WR to clear, to indicate the end of the program cycle.

EXAMPLE 4-2:   EEPROM DATA WRITE

```
BSF    STATUS, RP1    ;
BSF    STATUS, RP0    ;Bank 3
BTFSC  EECON1, WR     ;Wait for
GOTO   $-1            ;write to finish
BCF    STATUS, RP0    ;Bank 2
MOVF   ADDR, W        ;Address to
MOVWF  EEADR          ;write to
MOVF   VALUE, W       ;Data to
MOVWF  EEDATA         ;write
BSF    STATUS, RP0    ;Bank 3
BCF    EECON1, EEPGD  ;Point to Data memory
BSF    EECON1, WREN   ;Enable writes
                      ;Only disable interrupts
BCF    INTCON, GIE    ;if already enabled,
                      ;otherwise discard
MOVLW  0x55           ;Write 55h to
MOVWF  EECON2         ;EECON2
MOVLW  0xAA           ;Write AAh to
MOVWF  EECON2         ;EECON2
BSF    EECON1, WR     ;Start write operation
                      ;Only enable interrupts
BSF    INTCON, GIE    ;if using interrupts,
                      ;otherwise discard
BCF    EECON1, WREN   ;Disable writes
```

7.0 TIMER2 MODULE

Timer2 is an 8-bit timer with a prescaler and a postscaler. It can be used as the PWM time-base for the PWM mode of the CCP module(s). The TMR2 register is readable and writable, and is cleared on any device RESET.

The input clock (Fosc/4) has a prescale option of 1:1, 1:4, or 1:16, selected by control bits T2CKPS1:T2CKPS0 (T2CON<1:0>).

The Timer2 module has an 8-bit period register, PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon RESET.

The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit TMR2IF, (PIR1<1>)).

Timer2 can be shut-off by clearing control bit TMR2ON (T2CON<2>), to minimize power consumption.

Register 7-1 shows the Timer2 control register.

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

FIGURE 7-1:   TIMER2 BLOCK DIAGRAM

Note 1: TMR2 register output can be software selected by the SSP module as a baud clock.
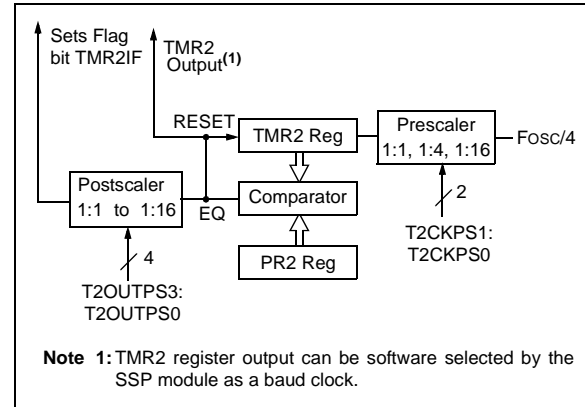
REGISTER 7-1:   T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7   **Unimplemented:** Read as '0'

bit 6-3   **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits

0000 = 1:1 Postscale
0001 = 1:2 Postscale
0010 = 1:3 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2   **TMR2ON**: Timer2 On bit

1 = Timer2 is on
0 = Timer2 is off

bit 1-0   **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits

00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

| 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

8.3 PWM Mode (PWM)

In Pulse Width Modulation mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

> **Note:** Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 8-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see Section 8.3.3.

FIGURE 8-3: SIMPLIFIED PWM BLOCK DIAGRAM
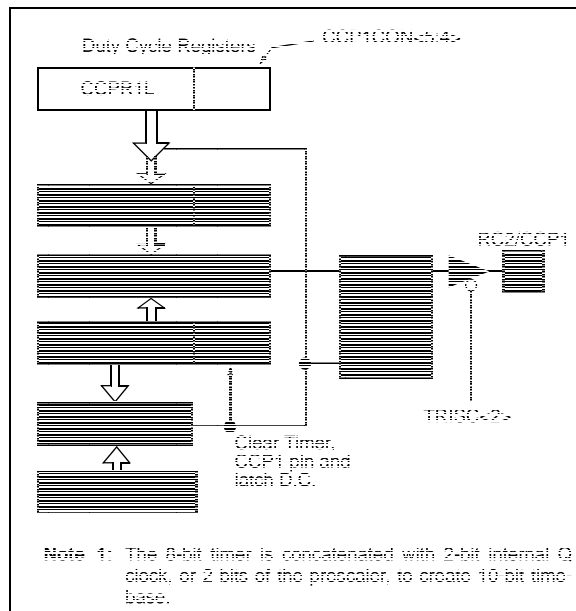
A PWM output (Figure 8-4) has a time-base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 8-4: PWM OUTPUT
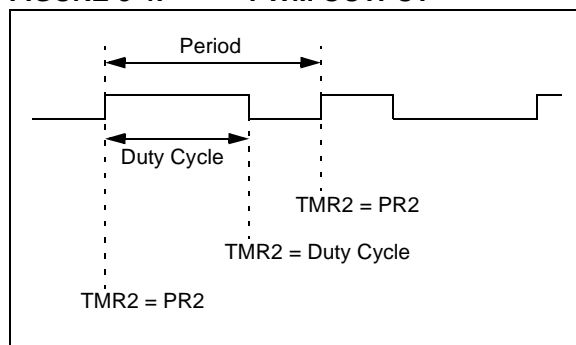
8.3.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

PWM frequency is defined as 1 / [PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

• TMR2 is cleared
• The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
• The PWM duty cycle is latched from CCPR1L into CCPR1H

> **Note:** The Timer2 postscaler (see Section 7.1) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

8.3.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM duty cycle} = (CCPR1L:CCP1CON<5:4>) \cdot T_{OSC} \cdot (\text{TMR2 prescale value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitch-free PWM operation.

When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock, or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by the formula:

$$\text{Resolution} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

> **Note:** If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

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)

TABLE 9-2: **DATA TRANSFER RECEIVED BYTE ACTIONS**

Status Bits as Data Transfer is Received		SSPSR → SSPBUF	Generate $\overline{ACK}$ Pulse	Set bit SSPIF (SSP Interrupt occurs if enabled)
BF	SSPOV			
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	Yes	No	Yes

Note: Shaded cells show the conditions where the user software did not properly clear the overflow condition.

9.2.1.3 Slave Transmission

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{ACK}$ pulse will be sent on the ninth bit, and the SCL pin is held low. The transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then, the SCL pin should be enabled by setting bit CKP (SSPCON<4>). The master must monitor the SCL pin prior to asserting another clock pulse. The slave devices may be holding off the master by stretching the clock. The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time (Figure 9-7).

An SSP interrupt is generated for each data transfer byte. The SSPIF flag bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte transfer. The SSPIF flag bit is set on the falling edge of the ninth clock pulse.

As a slave-transmitter, the $\overline{ACK}$ pulse from the master receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not $\overline{ACK}$), then the data transfer is complete. When the not $\overline{ACK}$ is latched by the slave, the slave logic is reset and the slave then monitors for another occurrence of the START bit. If the SDA line was low ($\overline{ACK}$), the transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then the SCL pin should be enabled by setting the CKP bit.

FIGURE 9-6: **I^{2}C WAVEFORMS FOR RECEPTION (7-BIT ADDRESS)**
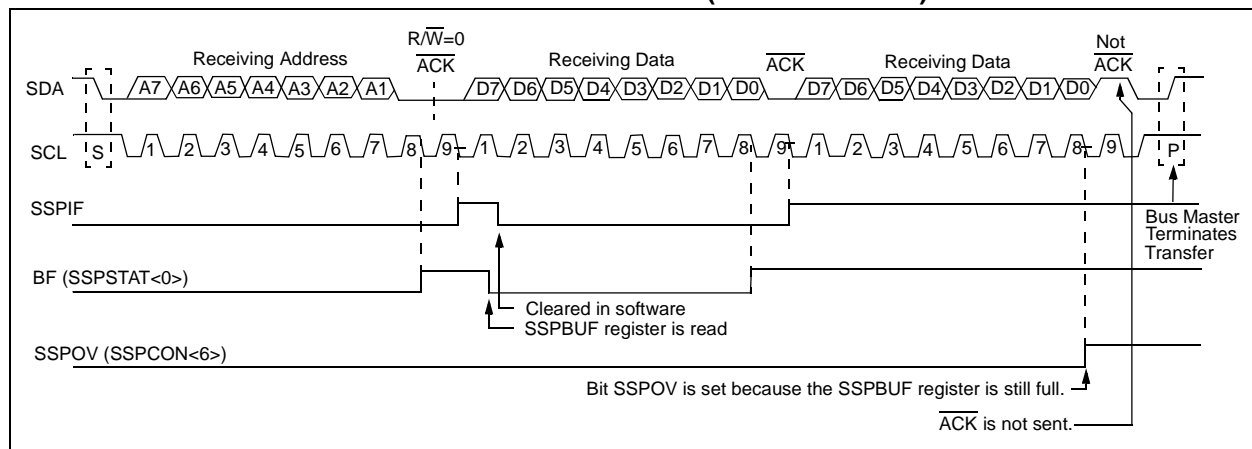
PIC16F87X

9.2.18.1 Bus Collision During a START Condition

During a START condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the START condition (Figure 9-20).

b) SCL is sampled low before SDA is asserted low (Figure 9-21).

During a START condition, both the SDA and the SCL pins are monitored. If either the SDA pin or the SCL pin is already low, then these events all occur:

*   the START condition is aborted,
*   and the BCLIF flag is set,
*   and the SSP module is reset to its IDLE state (Figure 9-20).

The START condition begins with the SDA and SCL pins de-asserted. When the SDA pin is sampled high, the baud rate generator is loaded from SSPADD<6:0> and counts down to 0. If the SCL pin is sampled low while SDA is high, a bus collision occurs, because it is assumed that another master is attempting to drive a data '1' during the START condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 9-22). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The baud rate generator is then reloaded and counts down to 0. During this time, if the SCL pins are sampled as '0', a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

Note:	The reason that bus collision is not a factor during a START condition is that no two bus masters can assert a START condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision, because the two masters must be allowed to arbitrate the first address following the START condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated START, or STOP conditions.

FIGURE 9-20: **BUS COLLISION DURING START CONDITION (SDA ONLY)**
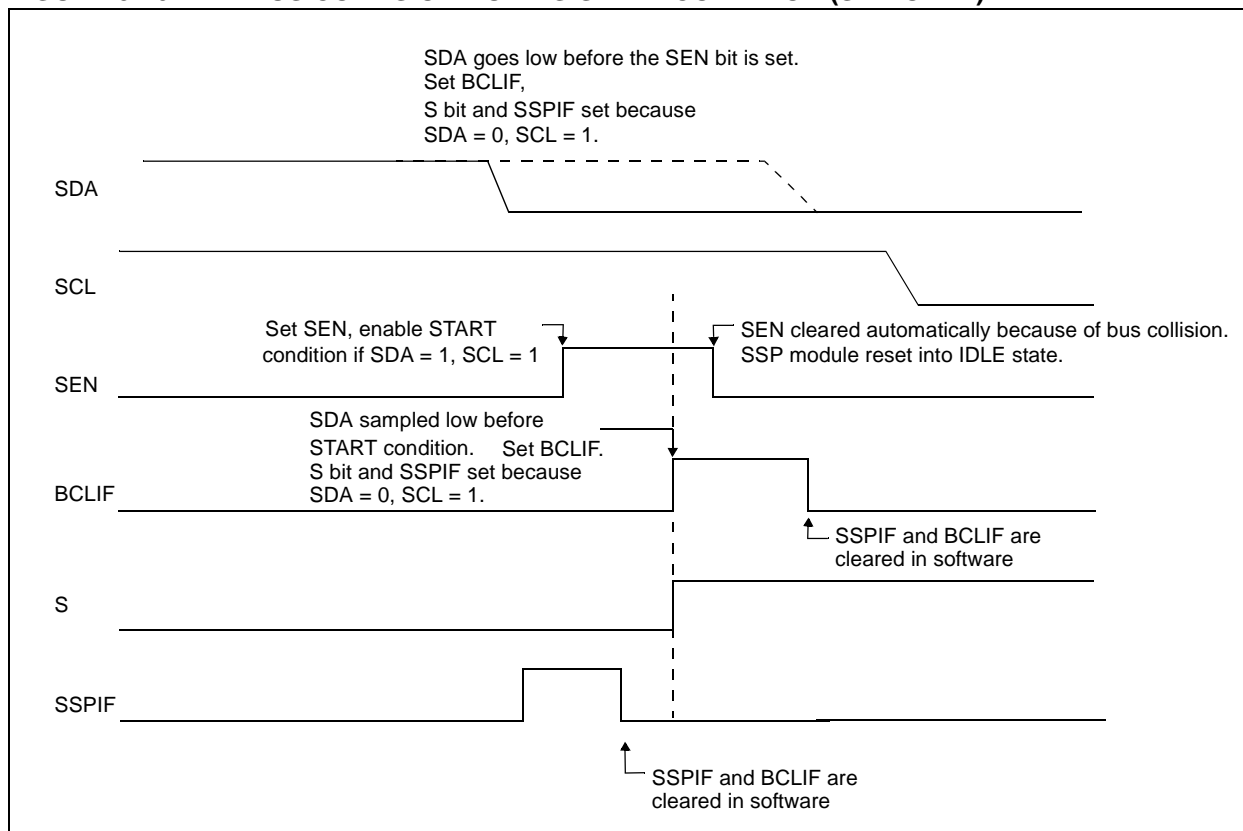
13.1 Instruction Descriptions

ADDLW	Add Literal and W
Syntax:	[*label*] ADDLW k
Operands:	$0 \leq k \leq 255$
Operation:	$(W) + k \rightarrow (W)$
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(W) + (f) \rightarrow (destination)$
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

ANDLW	AND Literal with W
Syntax:	[*label*] ANDLW k
Operands:	$0 \leq k \leq 255$
Operation:	$(W) .AND. (k) \rightarrow (W)$
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.

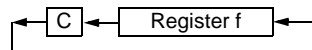| ANDWF	AND W with f
Syntax:	[*label*] ANDWF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(W) .AND. (f) \rightarrow (destination)$
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b \leq 7$
Operation:	$0 \rightarrow (f<b>)$
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b \leq 7$
Operation:	$1 \rightarrow (f<b>)$
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b < 7$
Operation:	skip if $(f<b>) = 1$
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	$0 \leq f \leq 127$ $0 \leq b \leq 7$
Operation:	skip if $(f<b>) = 0$
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

PIC16F87X

CALL	Call Subroutine
Syntax:	[*label*] CALL k
Operands:	$0 \leq k \leq 2047$
Operation:	(PC)+ 1$\rightarrow$ TOS, k $\rightarrow$ PC<10:0>, (PCLATH<4:3>) $\rightarrow$ PC<12:11>
Status Affected:	None
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

CLRWDT	Clear Watchdog Timer
Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h $\rightarrow$ WDT 0 $\rightarrow$ WDT prescaler, 1 $\rightarrow$ $\overline{TO}$ 1 $\rightarrow$ $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	$0 \leq f \leq 127$
Operation:	00h $\rightarrow$ (f) 1 $\rightarrow$ Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

COMF	Complement f
Syntax:	[*label*] COMF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(\bar{f}) \rightarrow$ (destination)
Status Affected:	Z
Description:	The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.

CLRW	Clear W
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h $\rightarrow$ (W) 1 $\rightarrow$ Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

DECF	Decrement f
Syntax:	[*label*] DECF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(f) - 1 $\rightarrow$ (destination)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

RLF — Rotate Left f through Carry

Syntax:	[*label*] RLF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'.

```
  ┌──[ C ]◄──[ Register f ]◄──┐
  └───────────────────────────┘
```

RETURN — Return from Subroutine

Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS $\rightarrow$ PC
Status Affected:	None
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

RRF — Rotate Right f through Carry

Syntax:	[*label*] RRF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

```
  ┌──►[ C ]──►[ Register f ]──►┐
  └───────────────────────────┘
```

SLEEP

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h $\rightarrow$ WDT, 0 $\rightarrow$ WDT prescaler, 1 $\rightarrow$ $\overline{\text{TO}}$, 0 $\rightarrow$ $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Description:	The power-down status bit, $\overline{\text{PD}}$ is cleared. Time-out status bit, $\overline{\text{TO}}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.

SUBLW — Subtract W from Literal

Syntax:	[*label*] SUBLW k
Operands:	$0 \le k \le 255$
Operation:	k - (W) $\rightarrow$ (W)
Status Affected:	C, DC, Z
Description:	The W register is subtracted (2's complement method) from the eight-bit literal 'k'. The result is placed in the W register.

SUBWF — Subtract W from f

Syntax:	[*label*] SUBWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - (W) $\rightarrow$ (destination)
Status Affected:	C, DC, Z
Description:	Subtract (2's complement method) W register from register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

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°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial
PIC16F873/874/876/877-04 PIC16F873/874/876/877-20 (Commercial, Industrial)							Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial
Param No.	Symbol	Characteristic/ Device	Min	Typ†	Max	Units	Conditions
	IDD	**Supply Current**[2,5]					
D010		16LF87X	—	0.6	2.0	mA	XT, RC osc configuration F_{OSC} = 4 MHz, V_{DD} = 3.0V
D010		16F87X	—	1.6	4	mA	RC osc configurations F_{OSC} = 4 MHz, V_{DD} = 5.5V
D010A		16LF87X	—	20	35	μA	LP osc configuration F_{OSC} = 32 kHz, V_{DD} = 3.0V, WDT disabled
D013		16F87X	—	7	15	mA	HS osc configuration, F_{OSC} = 20 MHz, V_{DD} = 5.5V
D015	ΔIBOR	**Brown-out Reset Current**[6]	—	85	200	μA	BOR enabled, V_{DD} = 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 V_{DD} 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 I_{DD} measurements in active operation mode are:
$\overline{OSC1}$ = external square wave, from rail to rail; all I/O pins tri-stated, pulled to V_{DD};
$\overline{MCLR}$ = V_{DD}; 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 V_{DD} and V_{SS}.

4: For RC osc configuration, current through R_{EXT} is not included. The current through the resistor can be estimated by the formula Ir = $V_{DD}/2R_{EXT}$ (mA) with R_{EXT} 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 I_{DD} or I_{PD} measurement.

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

FIGURE 16-5: TYPICAL I_{DD} vs. F_{OSC} OVER V_{DD} (LP MODE)
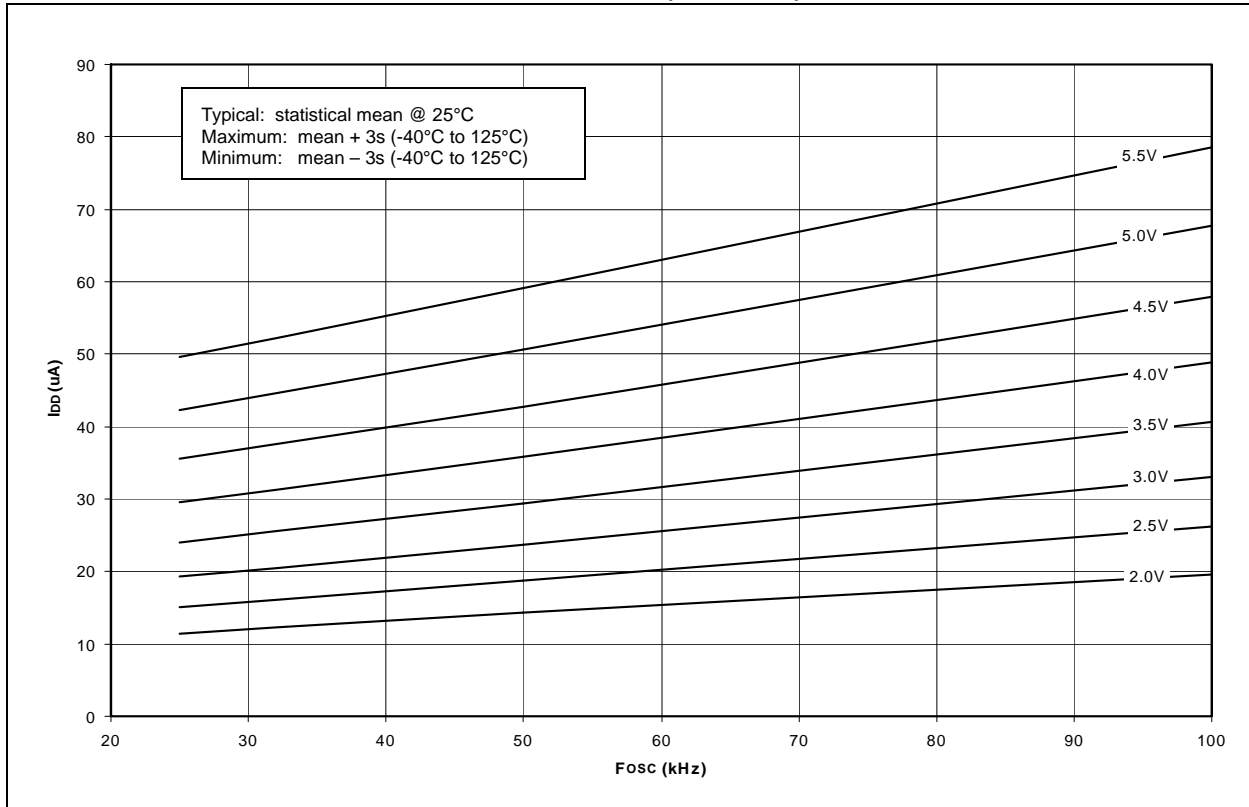
FIGURE 16-6: MAXIMUM I_{DD} vs. F_{OSC} OVER V_{DD} (XT MODE)
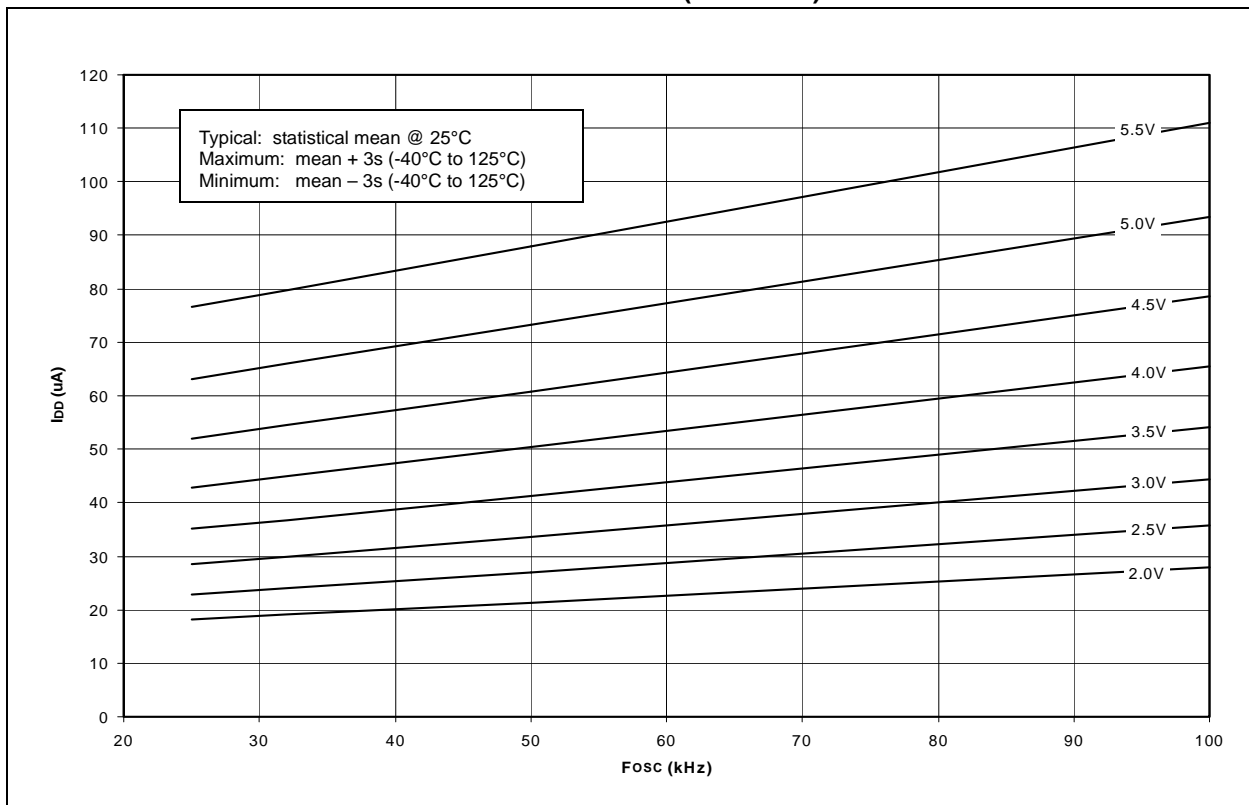
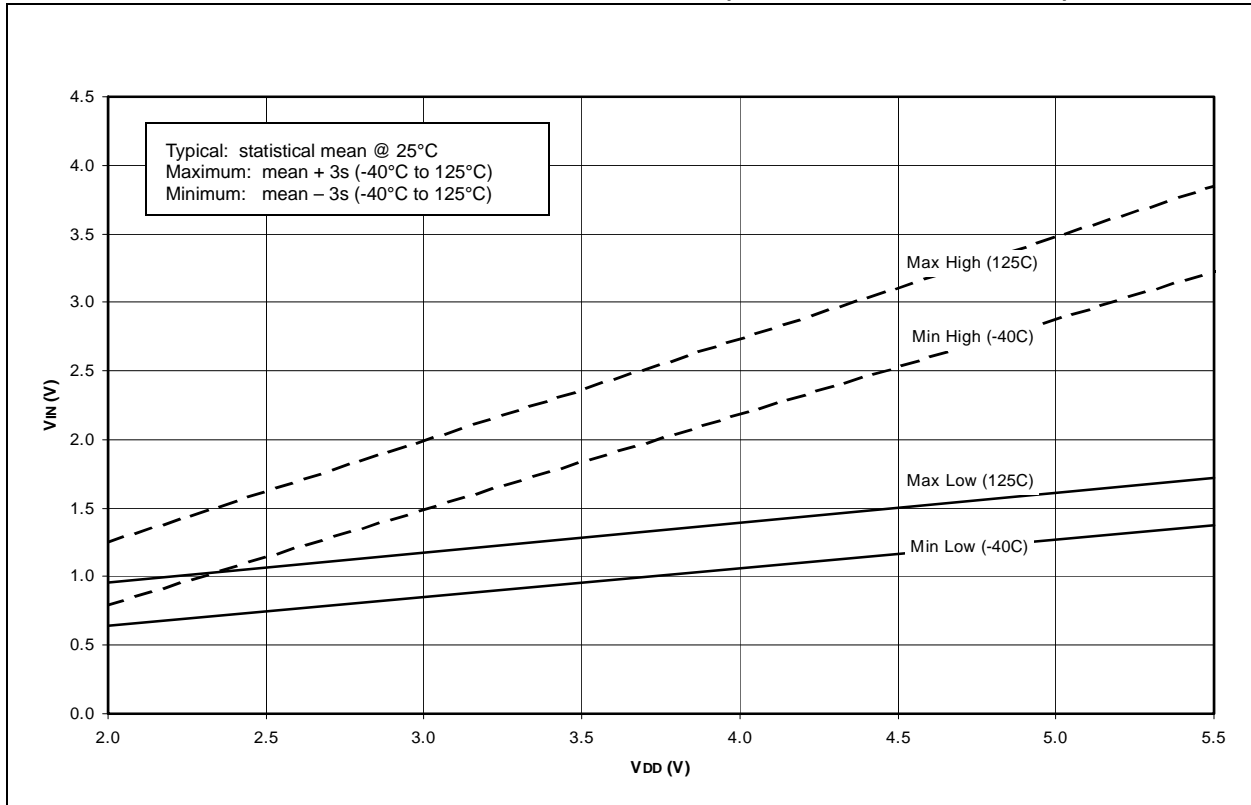**FIGURE 16-21:** MINIMUM AND MAXIMUM Vɪɴ vs. Vᴅᴅ (ST INPUT, -40°C TO 125°C)
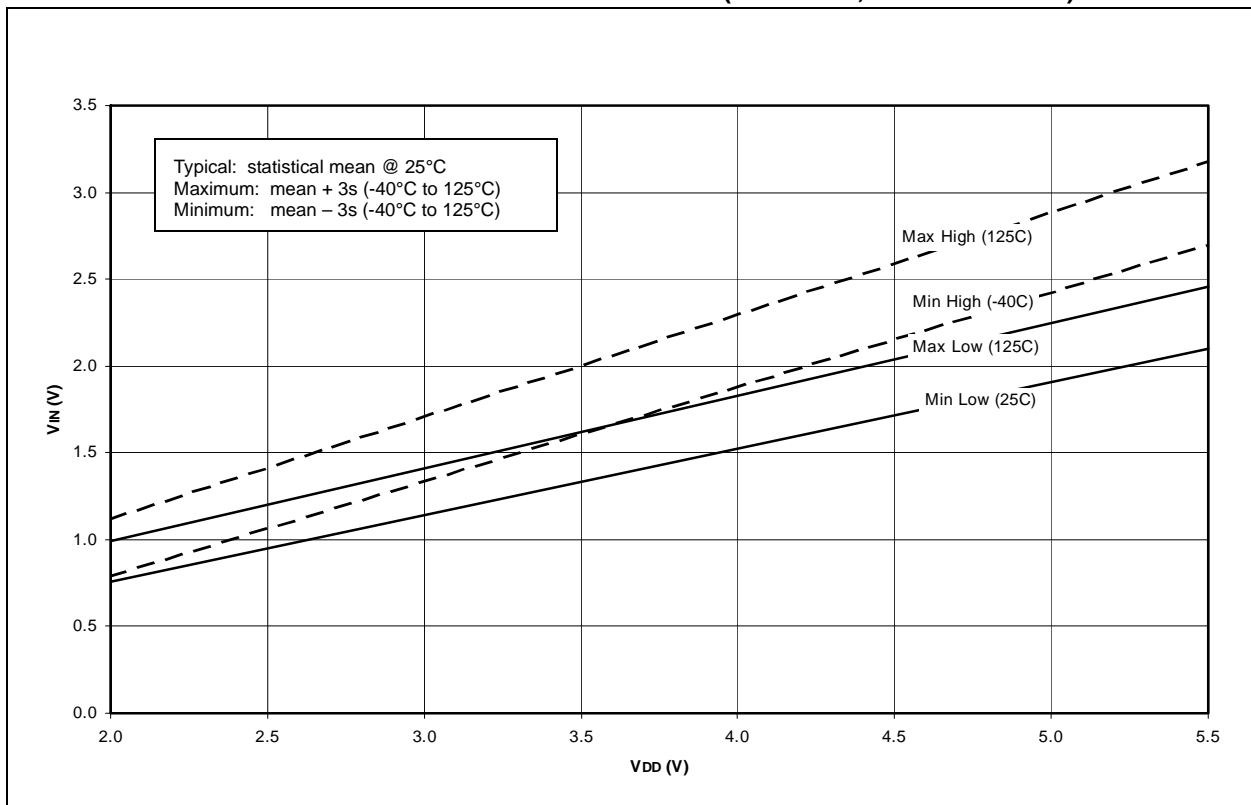

FIGURE 16-22: MINIMUM AND MAXIMUM Vɪɴ vs. Vᴅᴅ (I²C INPUT, -40°C TO 125°C)