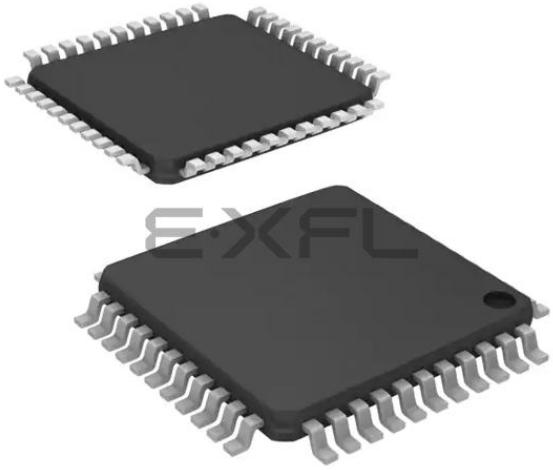


Welcome to E-XFL.COM

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	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)	4V ~ 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/pic16f877-04-pt

2.5 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

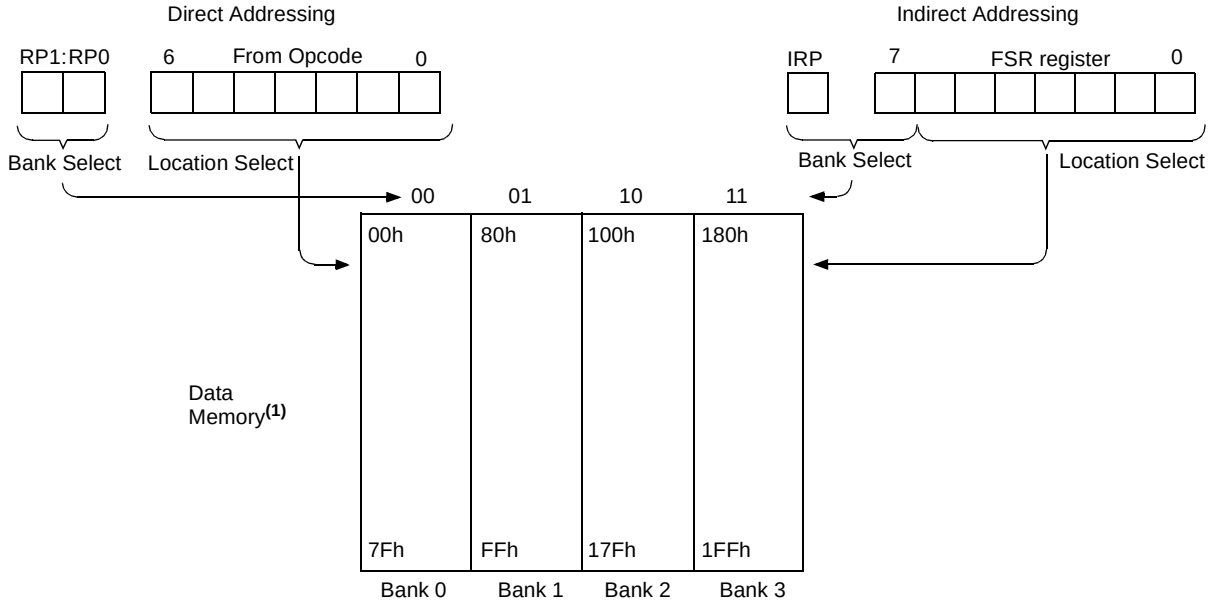
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = '0') will read 00h. Writing to the INDF register indirectly results in a no operation (although status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-6.

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

EXAMPLE 2-2: INDIRECT ADDRESSING

```
          MOVLW  0x20    ;initialize pointer
          MOVWF  FSR     ;to RAM
NEXT      CLRF   INDF    ;clear INDF register
          INCF   FSR,F   ;inc pointer
          BTFSS  FSR,4   ;all done?
          GOTO   NEXT    ;no clear next
CONTINUE
          :              ;yes continue
```

FIGURE 2-6: DIRECT/INDIRECT ADDRESSING



Note 1: For register file map detail, see Figure 2-3.

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^2C modes.
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C 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

Write operations have two control bits, WR and WREN, and two status bits, WRERR and EEIF. The WREN bit is used to enable or disable the write operation. When WREN is clear, the write operation will be disabled. Therefore, the WREN bit must be set before executing a write operation. The WR bit is used to initiate the write operation. It also is automatically cleared at the end of the write operation. The interrupt flag EEIF is used to determine when the memory write completes. This flag must be cleared in software before setting the WR bit. For EEPROM data memory, once the WREN bit and the WR bit have been set, the desired memory address in EEADR will be erased, followed by a write of the data in EEDATA. This operation takes place in parallel with the microcontroller continuing to execute normally. When the write is complete, the EEIF flag bit will be set. For program memory, once the WREN bit and the WR bit have been set, the microcontroller will cease to execute instructions. The desired memory location pointed to by EEADRH:EEADR will be erased. Then, the data value in EEDATH:EEDATA will be programmed. When complete, the EEIF flag bit will be set and the microcontroller will continue to execute code.

The WRERR bit is used to indicate when the PIC16F87X device has been reset during a write operation. WRERR should be cleared after Power-on Reset. Thereafter, it should be checked on any other RESET. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ Reset, or a WDT Time-out Reset, during normal operation. In these situations, following a RESET, the user should check the WRERR bit and rewrite the memory location, if set. The contents of the data registers, address registers and EEPGD bit are not affected by either $\overline{\text{MCLR}}$ Reset, or WDT Time-out Reset, during normal operation.

REGISTER 4-1: EECON1 REGISTER (ADDRESS 18Ch)

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

bit 7 **EEPGD**: Program/Data EEPROM Select bit
1 = Accesses program memory
0 = Accesses data memory
(This bit cannot be changed while a read or write operation is in progress)

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **WRERR**: EEPROM Error Flag bit
1 = A write operation is prematurely terminated
(any $\overline{\text{MCLR}}$ Reset or any WDT Reset during normal operation)
0 = The write operation completed

bit 2 **WREN**: EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM

bit 1 **WR**: Write Control bit
1 = Initiates a write cycle. (The bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

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

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

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.

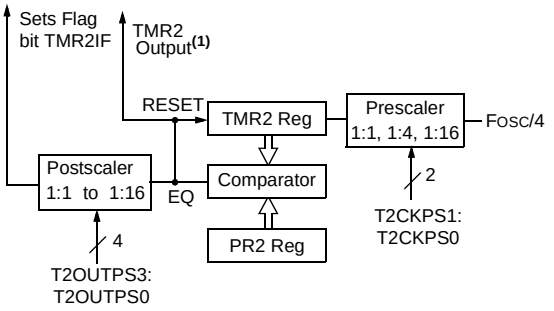
The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit TMR2IF, (PIR1<1>)).

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

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:

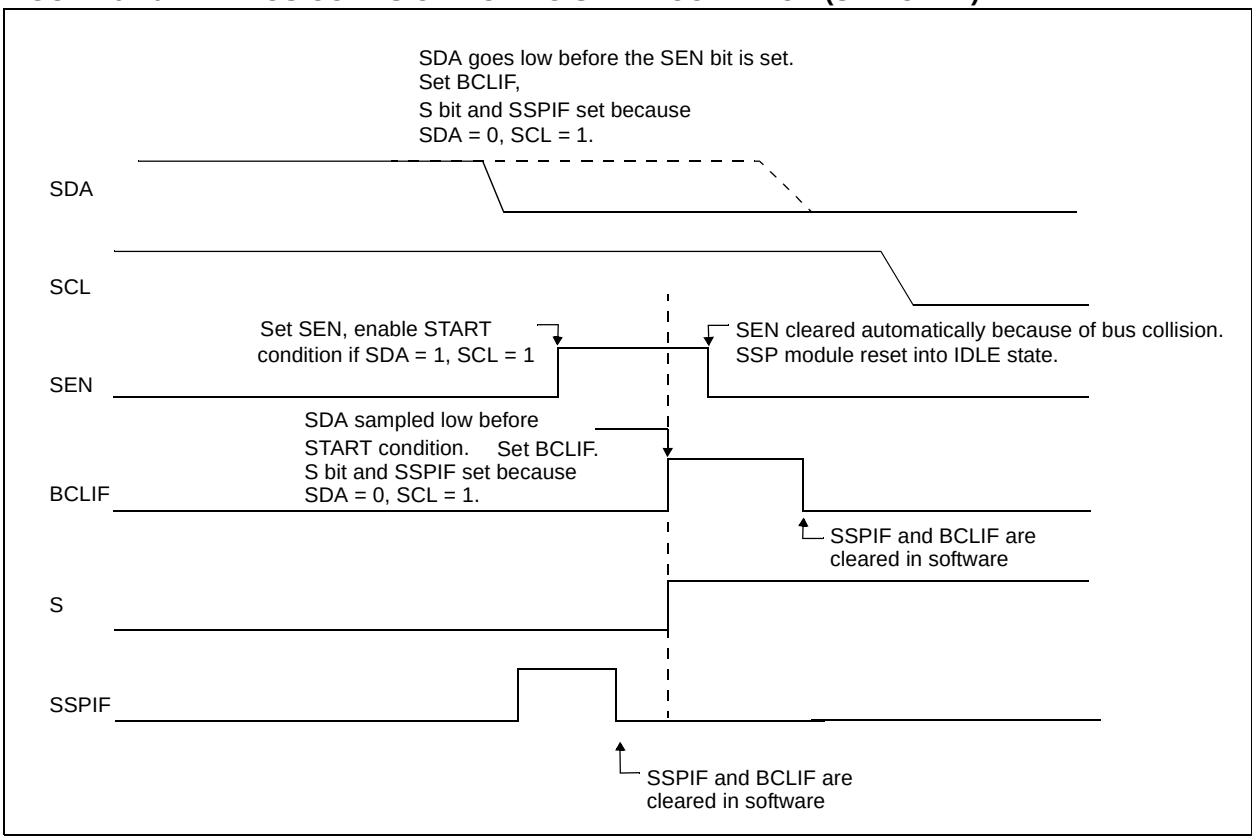
- the START condition is aborted,
- and the BCLIF flag is set,
- and the SSP module is reset to its IDLE state (Figure 9-20).

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)



When setting up an Asynchronous Transmission, follow these steps:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 10.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, then set enable bit TXIE.
4. If 9-bit transmission is desired, then set transmit bit TX9.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).
8. If using interrupts, ensure that GIE and PEIE (bits 7 and 6) of the INTCON register are set.

FIGURE 10-2: ASYNCHRONOUS MASTER TRANSMISSION

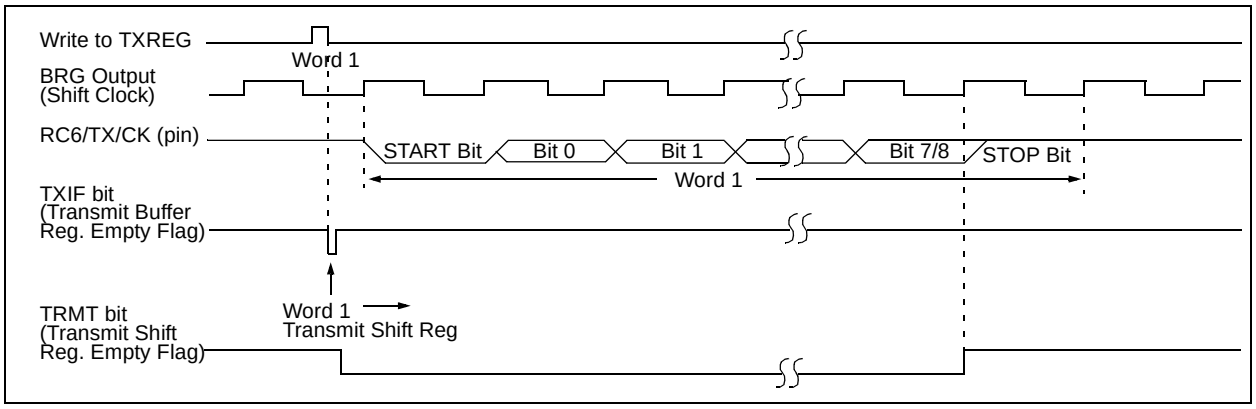
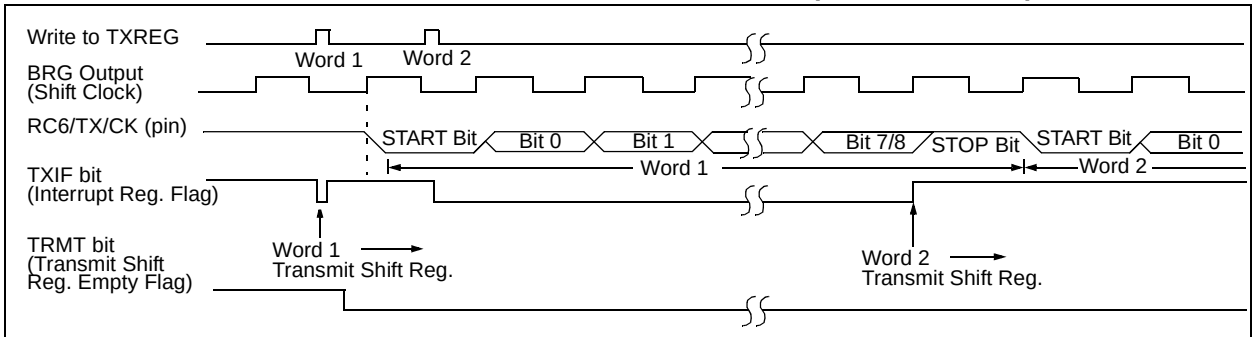


FIGURE 10-3: ASYNCHRONOUS MASTER TRANSMISSION (BACK TO BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 10-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

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
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

11.0 ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has five inputs for the 28-pin devices and eight for the other devices.

The analog input charges a sample and hold capacitor. The output of the sample and hold capacitor is the input into the converter. The converter then generates a digital result of this analog level via successive approximation. The A/D conversion of the analog input signal results in a corresponding 10-bit digital number. The A/D module has high and low voltage reference input that is software selectable to some combination of VDD, VSS, RA2, or RA3.

The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode. To operate in SLEEP, the A/D clock must be derived from the A/D's internal RC oscillator.

The A/D module has four registers. These registers are:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register0 (ADCON0)
- A/D Control Register1 (ADCON1)

The ADCON0 register, shown in Register 11-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 11-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RA3 can also be the voltage reference), or as digital I/O.

Additional information on using the A/D module can be found in the PIC® MCU Mid-Range Family Reference Manual (DS33023).

REGISTER 11-1: ADCON0 REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from the internal A/D module RC oscillator)

bit 5-3 **CHS2:CHS0**: Analog Channel Select bits
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)
010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 4, (RA5/AN4)
101 = channel 5, (RE0/AN5)**(1)**
110 = channel 6, (RE1/AN6)**(1)**
111 = channel 7, (RE2/AN7)**(1)**

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
If ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion)
0 = A/D conversion not in progress (this bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1 **Unimplemented**: Read as '0'

bit 0 **ADON**: A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shut-off and consumes no operating current

Note 1: These channels are not available on PIC16F873/876 devices.

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

11.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 11-2. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), see Figure 11-2. **The maximum recommended impedance for analog sources is 10 kΩ.** As the impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (changed), this acquisition must be done before the conversion can be started.

To calculate the minimum acquisition time, Equation 11-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

To calculate the minimum acquisition time, TACQ, see the PIC® MCU Mid-Range Reference Manual (DS33023).

EQUATION 11-1: ACQUISITION TIME

TACQ = Amplifier Settling Time + Hold Capacitor Charging Time + Temperature Coefficient

= TAMP + TC + TCOFF
= 2µs + TC + [(Temperature -25°C)(0.05µs/°C)]
TC = CHOLD (RIC + RSS + RS) In(1/2047)
= - 120pF (1kΩ + 7kΩ + 10kΩ) In(0.0004885)
= 16.47µs
TACQ = 2µs + 16.47µs + [(50°C -25°C)(0.05µs/°C)
= 19.72µs

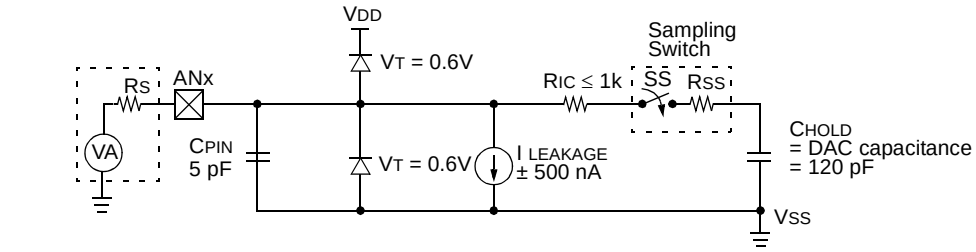
Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

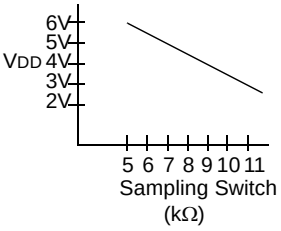
3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

4: After a conversion has completed, a 2.0TAD delay must complete before acquisition can begin again. During this time, the holding capacitor is not connected to the selected A/D input channel.

FIGURE 11-2: ANALOG INPUT MODEL



Legend
CPIN = input capacitance
VT = threshold voltage
I LEAKAGE = leakage current at the pin due to various junctions
RIC = interconnect resistance
SS = sampling switch
CHOLD = sample/hold capacitance (from DAC)



12.17 In-Circuit Serial Programming

PIC16F87X microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data and three other lines for power, ground, and the programming voltage. This allows customers to manufacture boards with unprogrammed devices, and then program the microcontroller just before shipping the product. This also allows the most recent firmware, or a custom firmware to be programmed.

When using ICSP, the part must be supplied at 4.5V to 5.5V, if a bulk erase will be executed. This includes reprogramming of the code protect, both from an on-state to off-state. For all other cases of ICSP, the part may be programmed at the normal operating voltages. This means calibration values, unique user IDs, or user code can be reprogrammed or added.

For complete details of serial programming, please refer to the EEPROM Memory Programming Specification for the PIC16F87X (DS39025).

12.18 Low Voltage ICSP Programming

The LVP bit of the configuration word enables low voltage ICSP programming. This mode allows the microcontroller to be programmed via ICSP using a VDD source in the operating voltage range. This only means that VPP does not have to be brought to VIHH, but can instead be left at the normal operating voltage. In this mode, the RB3/PGM pin is dedicated to the programming function and ceases to be a general purpose I/O pin. During programming, VDD is applied to the MCLR pin. To enter Programming mode, VDD must be applied to the RB3/PGM, provided the LVP bit is set. The LVP bit defaults to on (‘1’) from the factory.

Note 1: The High Voltage Programming mode is always available, regardless of the state of the LVP bit, by applying VIHH to the MCLR pin.

2: While in Low Voltage ICSP mode, the RB3 pin can no longer be used as a general purpose I/O pin.

3: 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.

4: RB3 should not be allowed to float if LVP is enabled. An external pull-down device should be used to default the device to normal operating mode. If RB3 floats high, the PIC16F87X device will enter Programming mode.

5: LVP mode is enabled by default on all devices shipped from Microchip. It can be disabled by clearing the LVP bit in the CONFIG register.

6: Disabling LVP will provide maximum compatibility to other PIC16CXXX devices.

If Low Voltage Programming mode is not used, the LVP bit can be programmed to a '0' and RB3/PGM becomes a digital I/O pin. However, the LVP bit may only be programmed when programming is entered with VIHH on MCLR. The LVP bit can only be charged when using high voltage on MCLR.

It should be noted, that once the LVP bit is programmed to 0, only the High Voltage Programming mode is available and only High Voltage Programming mode can be used to program the device.

When using low voltage ICSP, the part must be supplied at 4.5V to 5.5V, if a bulk erase will be executed. This includes reprogramming of the code protect bits from an on-state to off-state. For all other cases of low voltage ICSP, the part may be programmed at the normal operating voltage. This means calibration values, unique user IDs, or user code can be reprogrammed or added.

MOVF	Move f
Syntax:	[*label*] MOVF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) → (destination)
Status Affected:	Z
Description:	The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register, since status flag Z is affected.

MOVLW	Move Literal to W
Syntax:	[*label*] MOVLW k
Operands:	$0 \le k \le 255$
Operation:	k → (W)
Status Affected:	None
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	$0 \le f \le 127$
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.

NOP	No Operation
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.

RETFIE	Return from Interrupt
Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None

RETLW	Return with Literal in W
Syntax:	[*label*] RETLW k
Operands:	$0 \le k \le 255$
Operation:	k → (W); TOS → PC
Status Affected:	None
Description:	The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

14.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can also link relocatable objects from pre-compiled libraries, using directives from a linker script.

The MPLIB object librarian is a librarian for pre-compiled code to be used with the MPLINK object linker. When a routine from a library is called from another source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications. The MPLIB object librarian manages the creation and modification of library files.

The MPLINK object linker features include:

- Integration with MPASM assembler and MPLAB C17 and MPLAB C18 C compilers.
- Allows all memory areas to be defined as sections to provide link-time flexibility.

The MPLIB object librarian features include:

- Easier linking because single libraries can be included instead of many smaller files.
- Helps keep code maintainable by grouping related modules together.
- Allows libraries to be created and modules to be added, listed, replaced, deleted or extracted.

14.5 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC-hosted environment by simulating the PIC MCU series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user-defined key press, to any of the pins. The execution can be performed in single step, execute until break, or trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and the MPLAB C18 C compilers and the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent multi-project software development tool.

14.6 MPLAB ICE High Performance Universal In-Circuit Emulator with MPLAB IDE

The MPLAB ICE universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC MCU microcontrollers (MCUs). Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment (IDE), which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PIC microcontrollers.

The MPLAB ICE in-circuit emulator system has been designed as a real-time emulation system, with advanced features that are generally found on more expensive development tools. The PC platform and Microsoft® Windows environment were chosen to best make these features available to you, the end user.

14.7 ICEPIC In-Circuit Emulator

The ICEPIC low cost, in-circuit emulator is a solution for the Microchip Technology PIC16C5X, PIC16C6X, PIC16C7X and PIC16CXXX families of 8-bit One-Time-Programmable (OTP) microcontrollers. The modular system can support different subsets of PIC16C5X or PIC16CXXX products through the use of interchangeable personality modules, or daughter boards. The emulator is capable of emulating without target application circuitry being present.

15.5 Timing Parameter Symbology

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

1. TppS2ppS
2. TppS
3. TCC:ST (I^2C specifications only)
4. Ts (I^2C specifications only)

T			
F	Frequency	T	Time

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKOUT	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

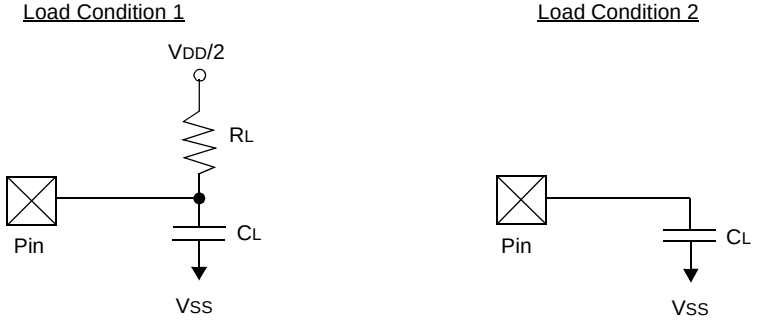
Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedance)	V	Valid
L	Low	Z	Hi-impedance
I^2C only			
AA	output access	High	High
BUF	Bus free	Low	Low

TCC:ST (I^2C specifications only)

CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	STOP condition
STA	START condition		

FIGURE 15-5: LOAD CONDITIONS

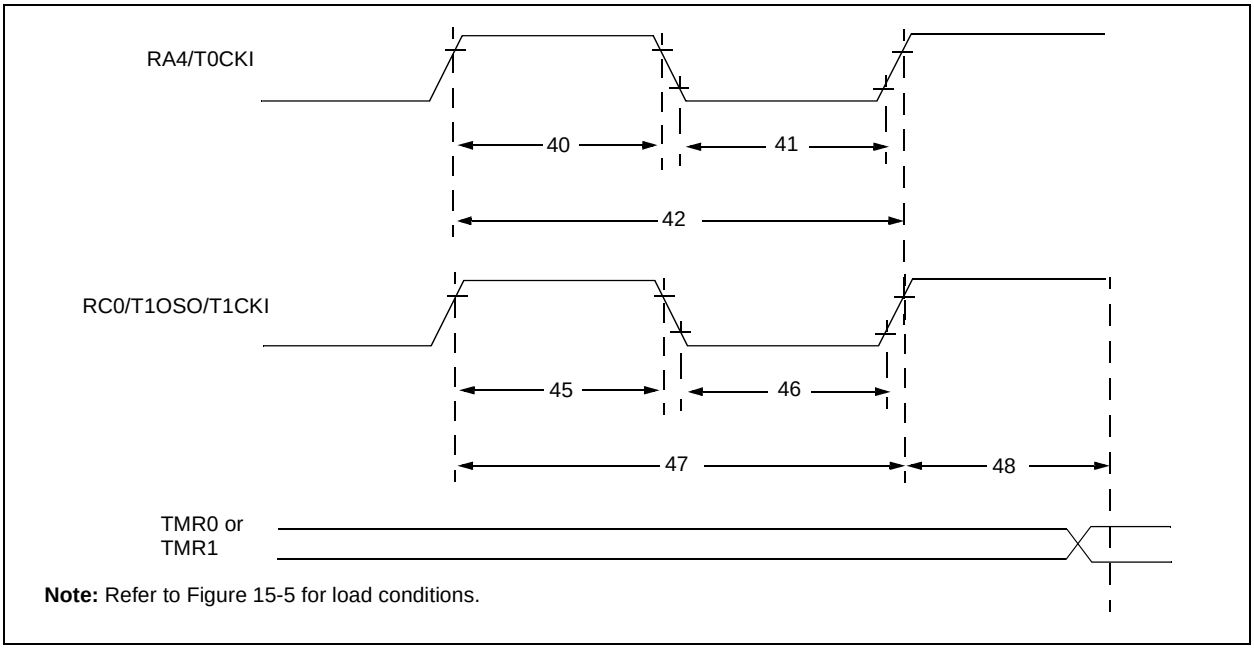


R_L = 464 Ω

C_L = 50 pF for all pins except OSC2, but including PORTD and PORTE outputs as ports,
15 pF for OSC2 output

Note: PORTD and PORTE are not implemented on PIC16F873/876 devices.

FIGURE 15-10: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS



Note: Refer to Figure 15-5 for load conditions.

TABLE 15-4: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param No.	Symbol	Characteristic			Min	Typ†	Max	Units	Conditions
40*	Tt0H	T0CKI High Pulse Width		No Prescaler	0.5TCY + 20	—	—	ns	Must also meet parameter 42
				With Prescaler	10	—	—	ns	
41*	Tt0L	T0CKI Low Pulse Width		No Prescaler	0.5TCY + 20	—	—	ns	Must also meet parameter 42
				With Prescaler	10	—	—	ns	
42*	Tt0P	T0CKI Period		No Prescaler	TCY + 40	—	—	ns	
				With Prescaler	Greater of: 20 or TCY + 40 / N	—	—	ns	N = prescale value (2, 4,..., 256)
45*	Tt1H	T1CKI High Time	Synchronous, Prescaler = 1		0.5TCY + 20	—	—	ns	Must also meet parameter 47
			Synchronous, Prescaler = 2,4,8	Standard(**F**)	15	—	—	ns	
				Extended(**LF**)	25	—	—	ns	
			Asynchronous	Standard(**F**)	30	—	—	ns	
				Extended(**LF**)	50	—	—	ns	
46*	Tt1L	T1CKI Low Time	Synchronous, Prescaler = 1		0.5TCY + 20	—	—	ns	Must also meet parameter 47
			Synchronous, Prescaler = 2,4,8	Standard(**F**)	15	—	—	ns	
				Extended(**LF**)	25	—	—	ns	
			Asynchronous	Standard(**F**)	30	—	—	ns	
				Extended(**LF**)	50	—	—	ns	
47*	Tt1P	T1CKI input period	Synchronous	Standard(**F**)	Greater of: 30 OR TCY + 40 / N	—	—	ns	N = prescale value (1, 2, 4, 8)
				Extended(**LF**)	Greater of: 50 OR TCY + 40 / N				N = prescale value (1, 2, 4, 8)
			Asynchronous	Standard(**F**)	60	—	—	ns	
				Extended(**LF**)	100	—	—	ns	
	Ft1	Timer1 oscillator input frequency range (oscillator enabled by setting bit T1OSCEN)			DC	—	200	kHz	
48	TCKEZtmr1	Delay from external clock edge to timer increment			2TOSC	—	7TOSC	—	

* These parameters are characterized but not tested.

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

FIGURE 15-13: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)

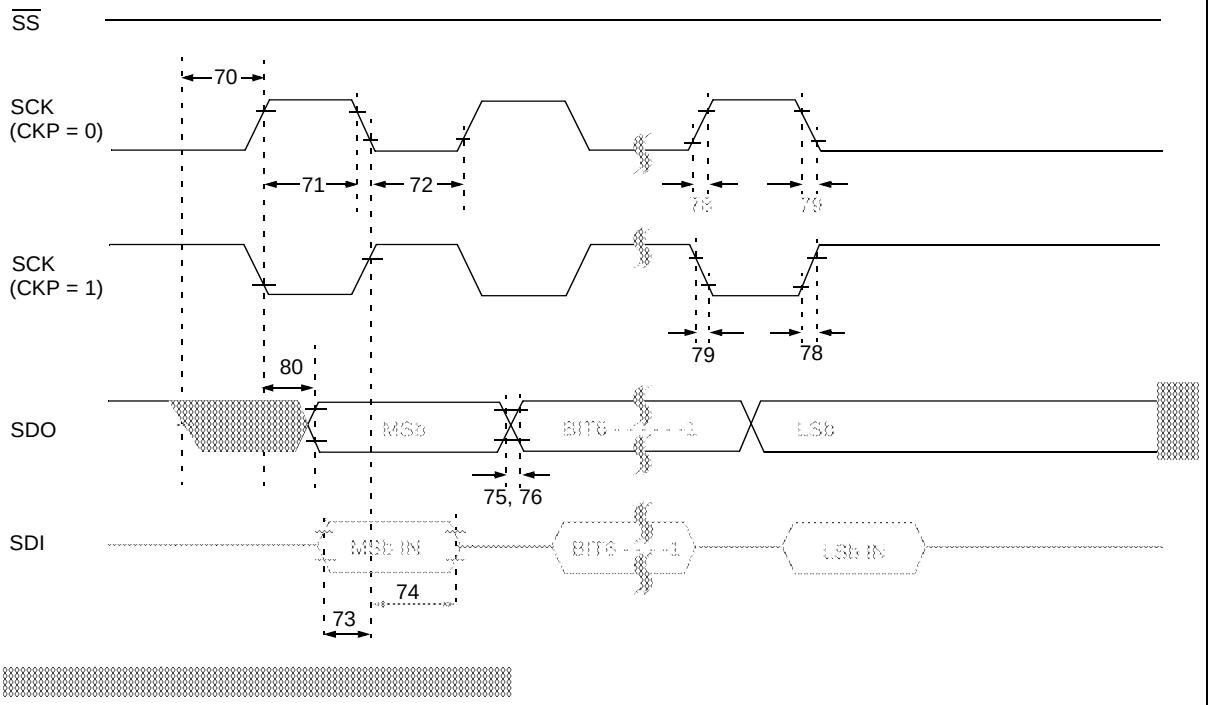
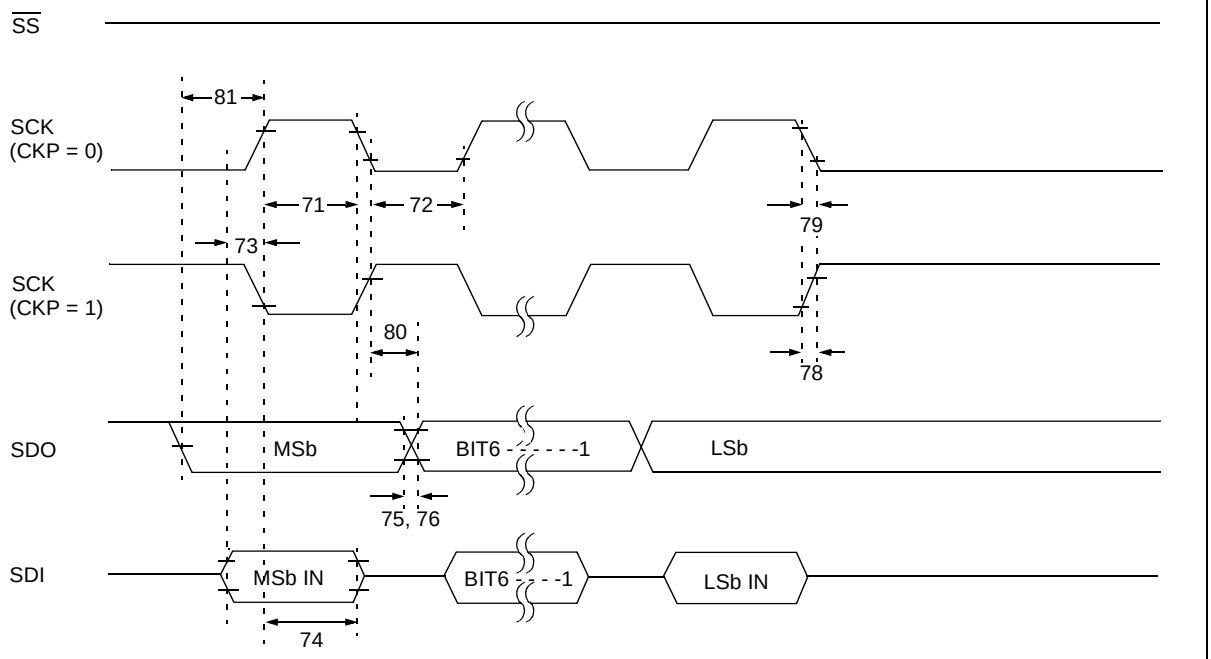


FIGURE 15-14: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)

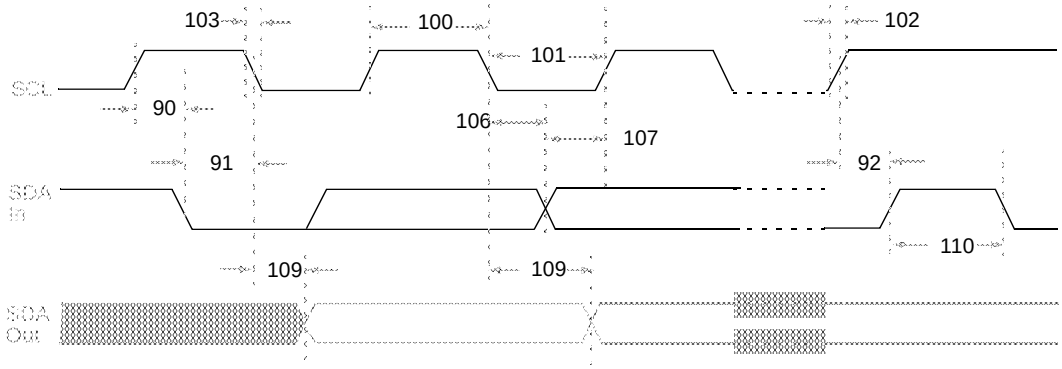


Note: Refer to Figure 15-5 for load conditions.

TABLE 15-8: I²C BUS START/STOP BITS REQUIREMENTS

Parameter No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
90	Tsu:sta	START condition Setup time	100 kHz mode	4700	—	—	ns	Only relevant for Repeated START condition
			400 kHz mode	600	—	—		
91	Thd:sta	START condition Hold time	100 kHz mode	4000	—	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—	—		
92	Tsu:sto	STOP condition Setup time	100 kHz mode	4700	—	—	ns	
			400 kHz mode	600	—	—		
93	Thd:sto	STOP condition Hold time	100 kHz mode	4000	—	—	ns	
			400 kHz mode	600	—	—		

FIGURE 15-18: I²C BUS DATA TIMING



Note: Refer to Figure 15-5 for load conditions.

FIGURE 16-9: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R (RC MODE, C = 300 pF, 25°C)

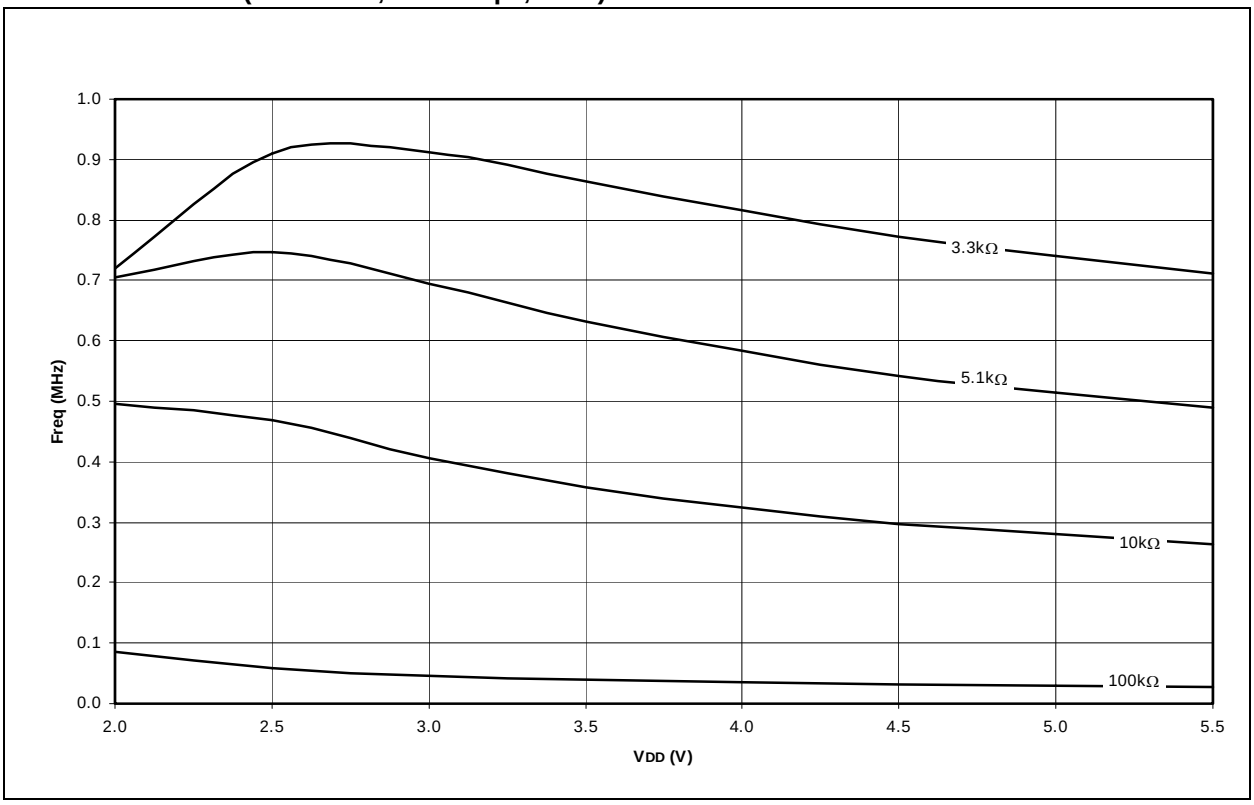


FIGURE 16-10: IPD vs. VDD (SLEEP MODE, ALL PERIPHERALS DISABLED)

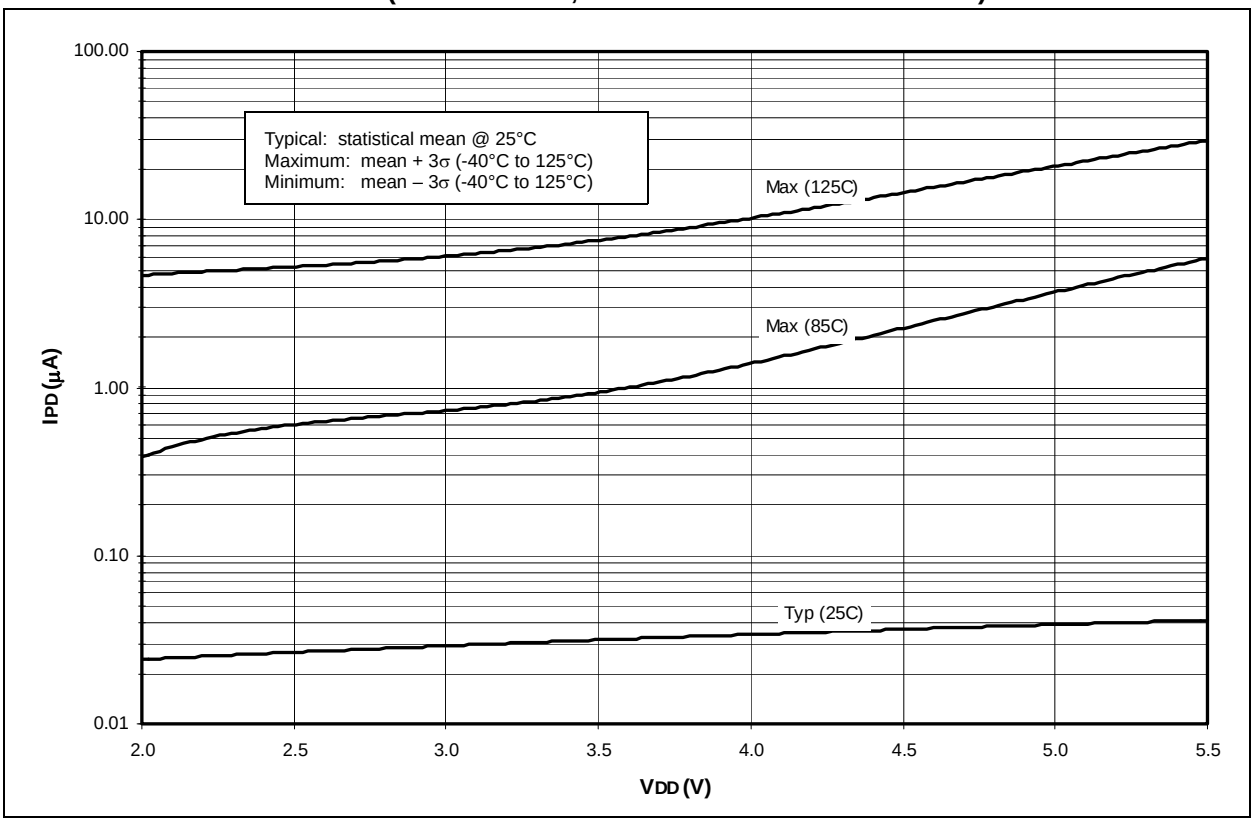


FIGURE 16-19: TYPICAL, MINIMUM AND MAXIMUM VOL vs. IOL (VDD=3V, -40°C TO 125°C)

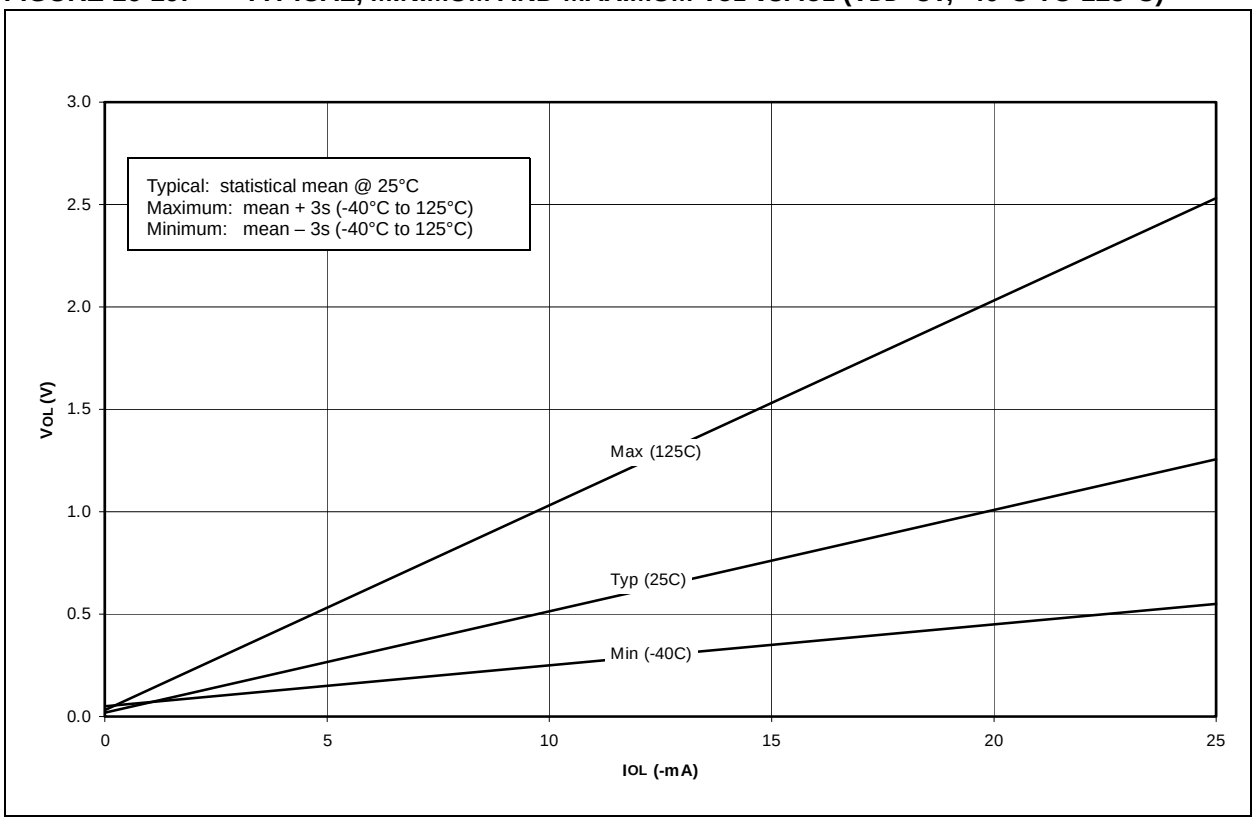


FIGURE 16-20: MINIMUM AND MAXIMUM VIN vs. VDD, (TTL INPUT, -40°C TO 125°C)

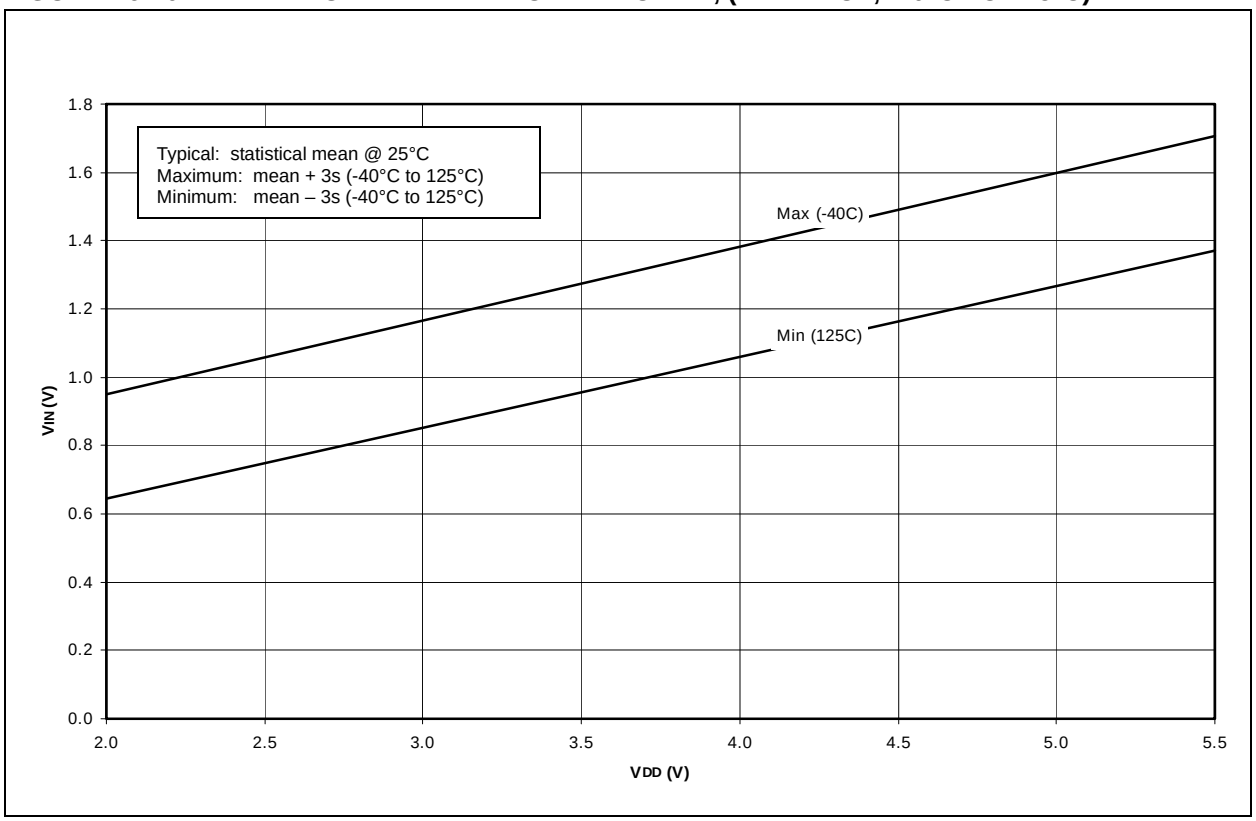


FIGURE 16-21: MINIMUM AND MAXIMUM VIN vs. VDD (ST INPUT, -40°C TO 125°C)

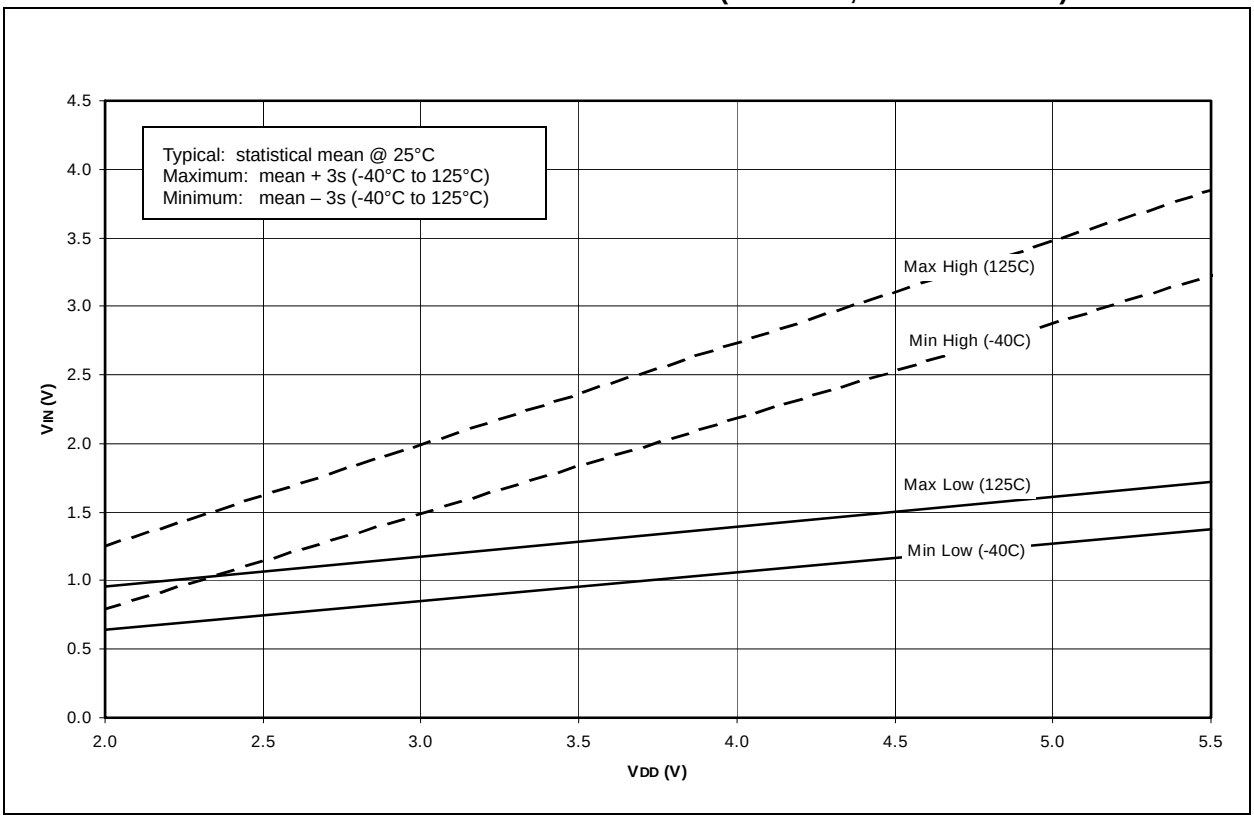
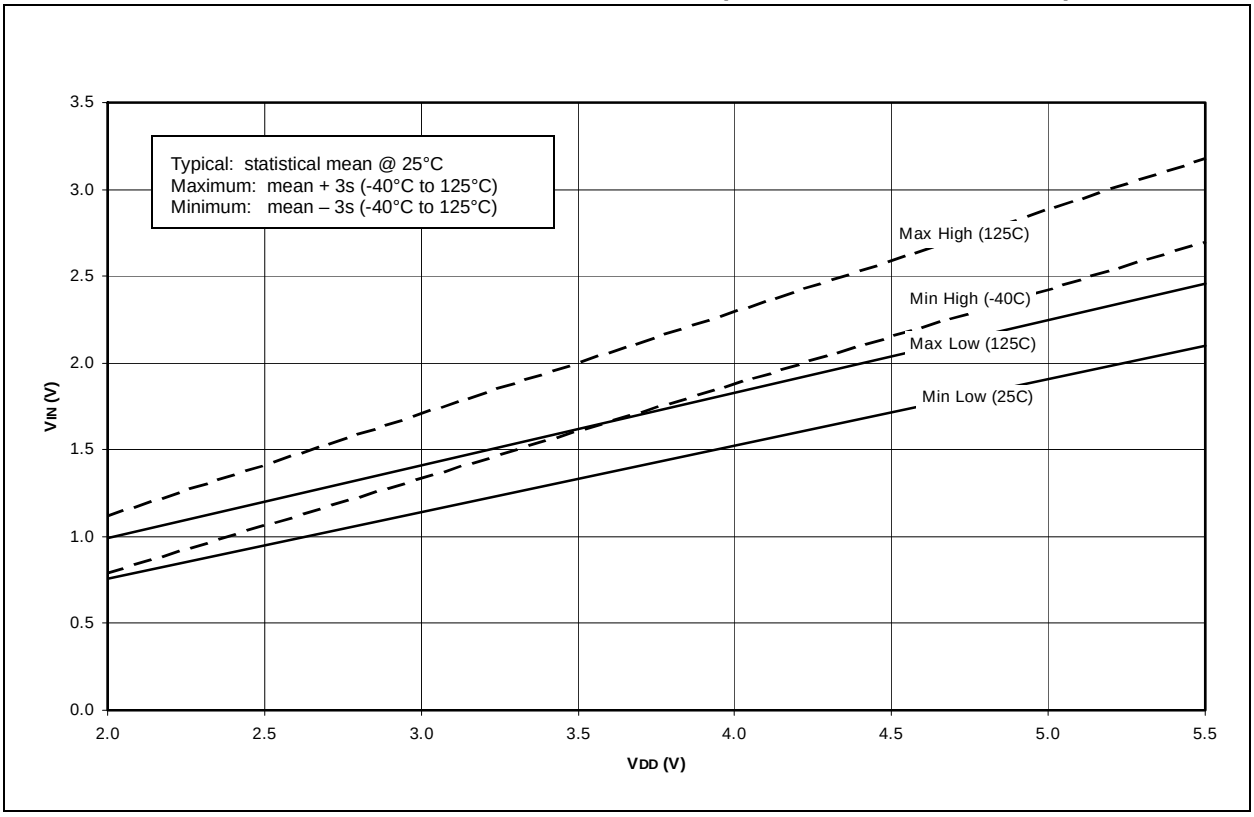


FIGURE 16-22: MINIMUM AND MAXIMUM VIN vs. VDD (I^2C INPUT, -40°C TO 125°C)



NOTES: