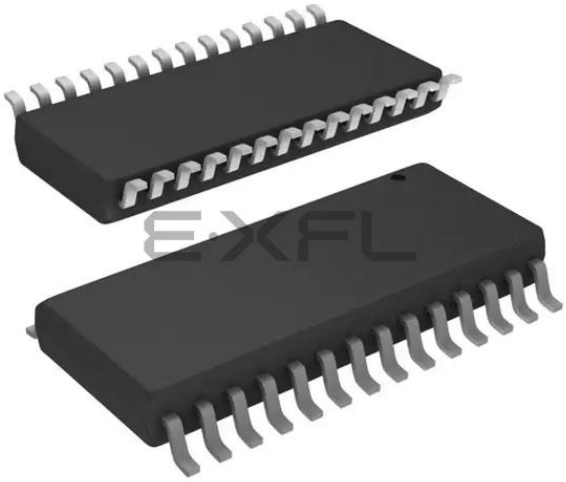


Welcome to E-XFL.COM

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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	22
Program Memory Size	32KB (16K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 5x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°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/pic18c252t-e-so

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at **www.microchip.com/cn** to receive the most current information on all of our products.

TABLE 1-3: PIC18C4X2 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number DIP	PLCC	TQFP	Pin Type	Buffer Type	Description
MCLR/VPP	1	2	18			Master clear (input) or programming voltage (input).
MCLR				I	ST	Master Clear (Reset) input. This pin is an active low RESET to the device.
VPP				P		Programming voltage input.
NC	—			—	—	These pins should be left unconnected.
OSC1/CLKI	13	14	30			Oscillator crystal or external clock input.
OSC1				I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise.
CLKI				I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKIN, OSC2/CLKOUT pins.)
OSC2/CLKO/RA6	14	15	31			Oscillator crystal output.
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	—	In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6				I/O	TTL	General Purpose I/O pin.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19			
RA0				I/O	TTL	Digital I/O.
AN0				I	Analog	Analog input 0.
RA1/AN1	3	4	20			
RA1				I/O	TTL	Digital I/O.
AN1				I	Analog	Analog input 1.
RA2/AN2/VREF-	4	5	21			
RA2				I/O	TTL	Digital I/O.
AN2				I	Analog	Analog input 2.
VREF-				I	Analog	A/D Reference Voltage (Low) input.
RA3/AN3/VREF+	5	6	22			
RA3				I/O	TTL	Digital I/O.
AN3				I	Analog	Analog input 3.
VREF+				I	Analog	A/D Reference Voltage (High) input.
RA4/T0CKI	6	7	23			
RA4				I/O	ST/OD	Digital I/O. Open drain when configured as output.
T0CKI				I	ST	Timer0 external clock input.
RA5/AN4/SS/LVDIN	7	8	24			
RA5				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
SS				I	ST	SPI Slave Select input.
LVDIN				I	Analog	Low Voltage Detect Input.
RA6						See the OSC2/CLKO/RA6 pin.

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

NOTES:

TABLE 4-1: SPECIAL FUNCTION REGISTER MAP

FFFh	TOSU	FDFh	INDF2(3)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(3)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(3)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(3)	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2(3)	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE(2)
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD(2)
FF4h	PRODH	FD4h	—	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0(3)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0(3)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0(3)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(2)
FECh	PREINC0(3)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(2)
FEBh	PLUSW0(3)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	—	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	—	F88h	—
FE7h	INDF1(3)	FC7h	SSPSTAT	FA7h	—	F87h	—
FE6h	POSTINC1(3)	FC6h	SSPCON1	FA6h	—	F86h	—
FE5h	POSTDEC1(3)	FC5h	SSPCON2	FA5h	—	F85h	—
FE4h	PREINC1(3)	FC4h	ADRESH	FA4h	—	F84h	PORTE(2)
FE3h	PLUSW1(3)	FC3h	ADRESL	FA3h	—	F83h	PORTD(2)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

Note 1: Unimplemented registers are read as ’0’.
2: This register is not available on PIC18C2X2 devices.
3: This is not a physical register.

4.13.1 RCON REGISTER

The Reset Control (RCON) register contains flag bits that allow differentiation between the sources of a device RESET. These flags include the $\overline{\text{TO}}$, $\overline{\text{PD}}$, $\overline{\text{POR}}$, $\overline{\text{BOR}}$ and $\overline{\text{RI}}$ bits. This register is readable and writable.

> **Note 1:** If the BOREN configuration bit is set (Brown-out Reset enabled), the $\overline{\text{BOR}}$ bit is '1' on a Power-on Reset. After a Brown-out Reset has occurred, the $\overline{\text{BOR}}$ bit will be clear and must be set by firmware to indicate the occurrence of the next Brown-out Reset.
> If the BOREN configuration bit is clear (Brown-out Reset disabled), $\overline{\text{BOR}}$ is unknown after Power-on Reset and Brown-out Reset conditions.
>
> **2:** It is recommended that the $\overline{\text{POR}}$ bit be set after a Power-on Reset has been detected, so that subsequent Power-on Resets may be detected.

REGISTER 4-3: RCON REGISTER

R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0
IPEN	LWRT	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (16CXXX compatibility mode)

bit 6 **LWRT:** Long Write Enable bit
1 = Enable `TBLWT` to internal program memory
Once this bit is set, it can only be cleared by a POR or $\overline{\text{MCLR}}$ Reset.
0 = Disable `TBLWT` to internal program memory; `TBLWT` only to external program memory

bit 5 **Unimplemented:** Read as '0'

bit 4 **$\overline{\text{RI}}$:** `RESET` Instruction Flag bit
1 = The `RESET` instruction was not executed
0 = The `RESET` instruction was executed causing a device RESET
(must be set in software after a Brown-out Reset occurs)

bit 3 **$\overline{\text{TO}}$:** Watchdog Time-out Flag bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 2 **$\overline{\text{PD}}$:** Power-down Detection Flag bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 1 **$\overline{\text{POR}}$:** Power-on Reset Status bit
1 = A Power-on Reset has not occurred
0 = A Power-on Reset occurred
(must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{\text{BOR}}$:** Brown-out Reset Status bit
1 = A Brown-out Reset has not 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 reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

7.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Priority Registers (IPR1, IPR2). The operation of the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

REGISTER 7-8: PERIPHERAL INTERRUPT PRIORITY REGISTER 1 (IPR1)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

bit 7 **PSPIP:** Parallel Slave Port Read/Write Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **ADIP**: A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP**: USART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP**: USART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSPIP**: Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP**: CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP**: TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP**: TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

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

14.2 Control Registers

The MSSP module has three associated registers. These include a status register (SSPSTAT) and two control registers (SSPCON1 and SSPCON2).

REGISTER 14-1: SSPSTAT: MSSP STATUS REGISTER

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Sample bit

SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time

SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode

In I^2C Master or Slave mode:
1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for high speed mode (400 kHz)

bit 6 **CKE:** SPI Clock Edge Select bit

CKP = 0:
1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK

CKP = 1:
1 = Data transmitted on falling edge of SCK
0 = Data transmitted on rising edge of SCK

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit (I^2C mode only)
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** STOP bit
(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a STOP bit has been detected last (this bit is '0' on RESET)
0 = STOP bit was not detected last

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

14.3 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) - RC5/SDO
- Serial Data In (SDI) - RC4/SDI/SDA
- Serial Clock (SCK) - RC3/SCK/SCL/LVOIN

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

- Slave Select ($\overline{SS}$) - RA5/$\overline{SS}$/AN4

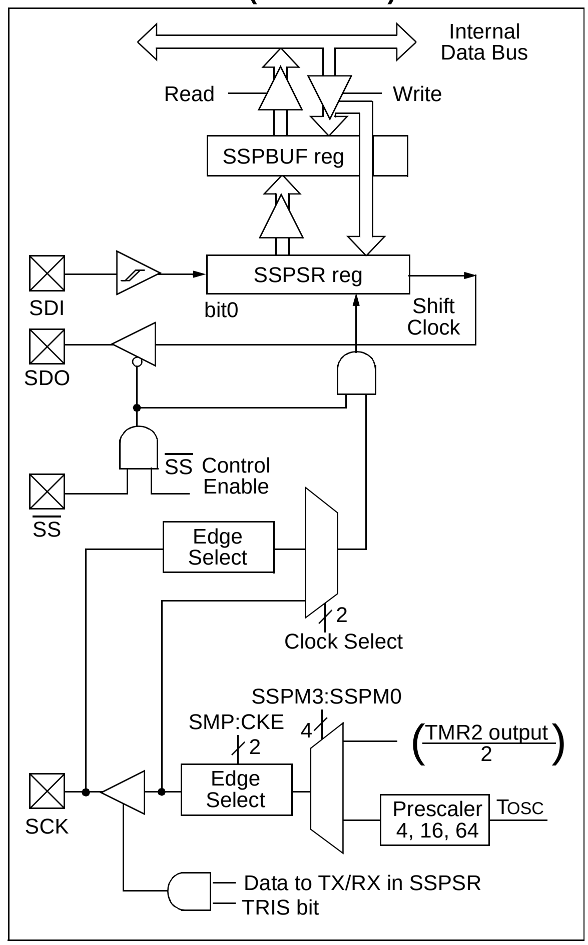
14.3.1 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<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 14-1 shows the block diagram of the MSSP module, when in SPI mode.

FIGURE 14-1: MSSP BLOCK DIAGRAM (SPI MODE)



The MSSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR, until the received data is ready. Once the 8-bits of data have been received, that byte is moved to the SSPBUF register. Then the buffer full detect bit, BF (SSPSTAT<0>), and the interrupt flag bit, SSPIF, are set. This double buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored, and the write collision detect bit, WCOL (SSPCON1<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

TABLE 15-5: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)

BAUD RATE (K)	FOSC = 40 MHz			FOSC = 20 MHz			FOSC = 16 MHz			FOSC = 10 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
9.6	9.77	-1.70	255	9.615	+0.16	129	9.615	+0.16	103	9.615	+0.16	64
19.2	19.23	-0.16	129	19.230	+0.16	64	19.230	+0.16	51	18.939	-1.36	32
38.4	38.46	-0.16	64	37.878	-1.36	32	38.461	+0.16	25	39.062	+1.7	15
57.6	58.14	-0.93	42	56.818	-1.36	21	58.823	+2.12	16	56.818	-1.36	10
115.2	113.64	+1.38	21	113.63	-1.36	10	111.11	-3.55	8	125	+8.51	4
250	250.00	0	9	250	0	4	250	0	3	NA	—	—
625	625.00	0	3	625	0	1	NA	—	—	625	0	0
1250	1250.00	0	1	1250	0	0	NA	—	—	NA	—	—

BAUD RATE (K)	FOSC = 7.16MHz			FOSC = 5.068 MHz			FOSC = 4 MHz			FOSC = 3.579545 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
9.6	9.520	-0.83	46	9.6	0	32	NA	—	—	9.727	+1.32	22
19.2	19.454	+1.32	22	18.645	-2.94	16	1.202	+0.17	207	18.643	-2.90	11
38.4	37.286	-2.90	11	39.6	+3.12	7	2.403	+0.13	103	37.286	-2.90	5
57.6	55.930	-2.90	7	52.8	-8.33	5	9.615	+0.16	25	55.930	-2.90	3
115.2	111.860	-2.90	3	105.6	-8.33	2	19.231	+0.16	12	111.86	-2.90	1
250	NA	—	—	NA	—	—	NA	—	—	223.72	-10.51	0
625	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1250	NA	—	—	NA	—	—	NA	—	—	NA	—	—

BAUD RATE (K)	FOSC = 1 MHz			FOSC = 32.768 kHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
9.6	8.928	-6.99	6	NA	—	—
19.2	20.833	+8.51	2	NA	—	—
38.4	31.25	-18.61	1	NA	—	—
57.6	62.5	+8.51	0	NA	—	—
115.2	NA	—	—	NA	—	—
250	NA	—	—	NA	—	—
625	NA	—	—	NA	—	—
1250	NA	—	—	NA	—	—

15.3 USART Synchronous Master Mode

In Synchronous Master mode, the data is transmitted in a half-duplex manner, (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit SYNC (TXSTA<4>). In addition, enable bit SPEN (RCSTA<7>) is set in order to configure the RC6/TX/CK and RC7/RX/DT I/O pins to CK (clock) and DT (data) lines, respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting bit CSRC (TXSTA<7>).

15.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 15-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer register TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once the TXREG register transfers the data to the TSR register (occurs in one T_{CYCLE}), the TXREG is empty and interrupt bit TXIF (PIR1<4>) is set. The interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set, regardless of the state of enable bit TXIE, and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. TRMT is a read only bit, which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory, so it is not available to the user.

To set up a Synchronous Master Transmission:

1. Initialize the SPBRG register for the appropriate baud rate (Section 15.1).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN, and CSRC.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.

TABLE 15-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'.
Shaded cells are not used for Synchronous Master Transmission.

Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18C2X2 devices. Always maintain these bits clear.

17.2 Operation

Depending on the power source for the device voltage, the voltage normally decreases relatively slowly. This means that the LVD module does not need to be constantly operating. To decrease the current requirements, the LVD circuitry only needs to be enabled for short periods, where the voltage is checked. After doing the check, the LVD module may be disabled.

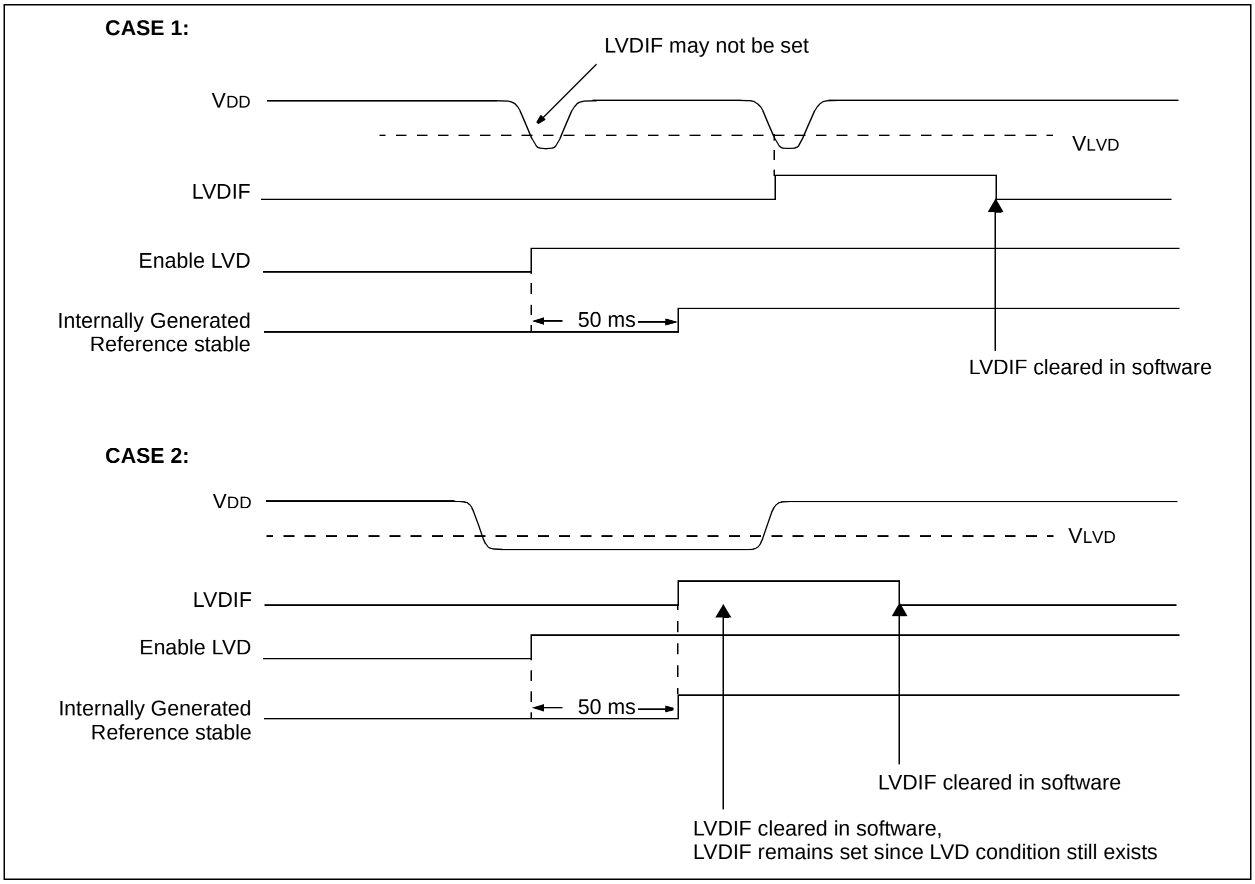
Each time that the LVD module is enabled, the circuitry requires some time to stabilize. After the circuitry has stabilized, all status flags may be cleared. The module will then indicate the proper state of the system.

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

1. Write the value to the LVDL3:LVDL0 bits (LVDCON register), which selects the desired LVD Trip Point.
2. Ensure that LVD interrupts are disabled (the LVDIE bit is cleared, or the GIE bit is cleared).
3. Enable the LVD module (set the LVDEN bit in the LVDCON register).
4. Wait for the LVD module to stabilize (the IRVST bit to become set).
5. Clear the LVD interrupt flag, which may have falsely become set until the LVD module has stabilized (clear the LVDIF bit).
6. Enable the LVD interrupt (set the LVDIE and the GIE bits).

Figure 17-4 shows typical waveforms that the LVD module may be used to detect.

FIGURE 17-4: LOW VOLTAGE DETECT WAVEFORMS



18.3 Power-down Mode (SLEEP)

Power-down mode is entered by executing a `SLEEP` instruction.

If enabled, the Watchdog Timer will be cleared, but keeps running, the $\overline{\text{PD}}$ bit (RCON<3>) is cleared, the $\overline{\text{TO}}$ (RCON<4>) bit is set, and the oscillator driver is turned off. The I/O ports maintain the status they had before the `SLEEP` instruction was executed (driving high, low, or hi-impedance).

For lowest current consumption in this mode, place all I/O pins at either V_{DD} or V_{SS}, ensure no external circuitry is drawing current from the I/O pin, power-down the A/D and disable external clocks. Pull all I/O pins that are hi-impedance inputs, high or low externally, to avoid switching currents caused by floating inputs. The T0CKI input should also be at V_{DD} or V_{SS} for lowest current consumption. The contribution from on-chip pull-ups on PORTB should be considered.

The $\overline{\text{MCLR}}$ pin must be at a logic high level (V_{IHMC}).

18.3.1 WAKE-UP FROM SLEEP

The device can wake up from SLEEP through one of the following events:

1. External RESET input on $\overline{\text{MCLR}}$ pin.
2. Watchdog Timer Wake-up (if WDT was enabled).
3. Interrupt from INT pin, RB port change, or a Peripheral Interrupt.

The following peripheral interrupts can wake the device from SLEEP:

1. PSP read or write.
2. TMR1 interrupt. Timer1 must be operating as an asynchronous counter.
3. TMR3 interrupt. Timer3 must be operating as an asynchronous counter.
4. CCP capture mode interrupt.
5. Special event trigger (Timer1 in Asynchronous mode using an external clock).
6. MSSP (START/STOP) bit detect interrupt.
7. MSSP transmit or receive in Slave mode (SPI/I^{2}C).
8. USART RX or TX (Synchronous Slave mode).
9. A/D conversion (when A/D clock source is RC).

Other peripherals cannot generate interrupts, since during SLEEP, no on-chip clocks are present.

External $\overline{\text{MCLR}}$ Reset will cause a device RESET. All other events are considered a continuation of program execution and will cause a "wake-up". The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits in the RCON register can be used to determine the cause of the device RESET. The $\overline{\text{PD}}$ bit, which is set on power-up, is cleared when `SLEEP` is invoked. The $\overline{\text{TO}}$ bit is cleared, if a WDT time-out occurred (and caused wake-up).

When the `SLEEP` instruction is being executed, the next instruction (PC + 2) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the `SLEEP` instruction. If the GIE bit is set (enabled), the device executes the instruction after the `SLEEP` instruction and then branches to the interrupt address. In cases where the execution of the instruction following SLEEP is not desirable, the user should have a `NOP` after the `SLEEP` instruction.

18.3.2 WAKE-UP USING INTERRUPTS

When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If an interrupt condition (interrupt flag bit and interrupt enable bits are set) occurs **before** the execution of a `SLEEP` instruction, the `SLEEP` instruction will complete as a `NOP`. Therefore, the WDT and WDT postscaler will not be cleared, the $\overline{\text{TO}}$ bit will not be set and $\overline{\text{PD}}$ bits will not be cleared.
- If the interrupt condition occurs **during or after** the execution of a `SLEEP` instruction, the device will immediately wake up from SLEEP. The `SLEEP` instruction will be completely executed before the wake-up. Therefore, the WDT and WDT postscaler will be cleared, the $\overline{\text{TO}}$ bit will be set and the $\overline{\text{PD}}$ bit will be cleared.

Even if the flag bits were checked before executing a `SLEEP` instruction, it may be possible for flag bits to become set before the `SLEEP` instruction completes. To determine whether a `SLEEP` instruction executed, test the $\overline{\text{PD}}$ bit. If the $\overline{\text{PD}}$ bit is set, the `SLEEP` instruction was executed as a `NOP`.

To ensure that the WDT is cleared, a `CLRWDT` instruction should be executed before a `SLEEP` instruction.

COMF — Complement f

Syntax:	[*label*] COMF f [,d [,a]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$(\overline{f}) \rightarrow$ dest
Status Affected:	N,Z
Encoding:	`0001` `11da` `ffff` `ffff`
Description:	The contents of register 'f' are complemented. If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `COMF REG, 0, 0`

Before Instruction

```
REG   =  0x13
```

After Instruction

```
REG   =  0x13
WREG  =  0xEC
```

CPFSEQ — Compare f with WREG, skip if f = WREG

Syntax:	[*label*] CPFSEQ f [,a]
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	(f) – (WREG), skip if (f) = (WREG) (unsigned comparison)
Status Affected:	None
Encoding:	`0110` `001a` `ffff` `ffff`
Description:	Compares the contents of data memory location 'f' to the contents of WREG by performing an unsigned subtraction. If 'f' = WREG, then the fetched instruction is discarded and a `NOP` is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

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

If skip and followed by 2-word instruction:

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

Example:

```
HERE     CPFSEQ REG, 0
NEQUAL   :
EQUAL    :
```

Before Instruction

```
PC Address  =  HERE
WREG        =  ?
REG         =  ?
```

After Instruction

```
If REG   =  WREG;
   PC    =  Address (EQUAL)
If REG   ≠  WREG;
   PC    =  Address (NEQUAL)
```

MOVLW **Move literal to WREG**

Syntax:	[*label*] MOVLW k
Operands:	$0 \le k \le 255$
Operation:	k → WREG
Status Affected:	None
Encoding:	`0000` `1110` `kkkk` `kkkk`
Description:	The eight-bit literal 'k' is loaded into WREG.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: `MOVLW 0x5A`

After Instruction

```
WREG = 0x5A
```

MOVWF **Move WREG to f**

Syntax:	[*label*] MOVWF f [,a]
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	(WREG) → f
Status Affected:	None
Encoding:	`0110` `111a` `ffff` `ffff`
Description:	Move data from WREG to register 'f'. Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `MOVWF REG, 0`

Before Instruction

```
WREG = 0x4F
REG  = 0xFF
```

After Instruction

```
WREG = 0x4F
REG  = 0x4F
```

SUBLW — Subtract WREG from literal

Syntax:	[*label*] SUBLW k
Operands:	$0 \leq k \leq 255$
Operation:	k – (WREG) → WREG
Status Affected:	N,OV, C, DC, Z
Encoding:	`0000` `1000` `kkkk` `kkkk`
Description:	WREG is subtracted from the eight-bit literal 'k'. The result is placed in WREG.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example 1: `SUBLW 0x02`

Before Instruction

```
WREG   =  1
C      =  ?
```

After Instruction

```
WREG   =  1
C      =  1      ; result is positive
Z      =  0
N      =  0
```

Example 2: `SUBLW 0x02`

Before Instruction

```
WREG   =  2
C      =  ?
```

After Instruction

```
WREG   =  0
C      =  1      ; result is zero
Z      =  1
N      =  0
```

Example 3: `SUBLW 0x02`

Before Instruction

```
WREG   =  3
C      =  ?
```

After Instruction

```
WREG   =  FF   ; (2's complement)
C      =  0    ; result is negative
Z      =  0
N      =  1
```

SUBWF — Subtract WREG from f

Syntax:	[*label*] SUBWF f [,d [,a]
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – (WREG) → dest
Status Affected:	N,OV, C, DC, Z
Encoding:	`0101` `11da` `ffff` `ffff`
Description:	Subtract WREG from register 'f' (2's complement method). If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example 1: `SUBWF REG, 1, 0`

Before Instruction

```
REG    =  3
WREG   =  2
C      =  ?
```

After Instruction

```
REG    =  1
WREG   =  2
C      =  1      ; result is positive
Z      =  0
N      =  0
```

Example 2: `SUBWF REG, 0, 0`

Before Instruction

```
REG    =  2
WREG   =  2
C      =  ?
```

After Instruction

```
REG    =  2
WREG   =  0
C      =  1      ; result is zero
Z      =  1
N      =  0
```

Example 3: `SUBWF REG, 1, 0`

Before Instruction

```
REG    =  1
WREG   =  2
C      =  ?
```

After Instruction

```
REG    =  FFh    ;(2's complement)
WREG   =  2
C      =  0      ; result is negative
Z      =  0
N      =  1
```

TBLWT **Table Write**

Syntax: [*label*] TBLWT (*; *+; *-; +*)

Operands: None

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

Status Affected: None

Encoding:

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

Description: This instruction is used to program the contents of Program Memory (P.M.). The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2 Mbyte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

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

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

Words: 1

Cycles: 2 (many if long write is to on-chip EPROM program memory)

Q Cycle Activity:

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

TBLWT **Table Write (Continued)**

Example 1: `TBLWT *+;`

Before Instruction

```
TABLAT              =   0x55
TBLPTR              =   0x00A356
MEMORY(0x00A356)    =   0xFF
```

After Instructions (table write completion)

```
TABLAT              =   0x55
TBLPTR              =   0x00A357
MEMORY(0x00A356)    =   0x55
```

Example 2: `TBLWT +*;`

Before Instruction

```
TABLAT              =   0x34
TBLPTR              =   0x01389A
MEMORY(0x01389A)    =   0xFF
MEMORY(0x01389B)    =   0xFF
```

After Instruction (table write completion)

```
TABLAT              =   0x34
TBLPTR              =   0x01389B
MEMORY(0x01389A)    =   0xFF
MEMORY(0x01389B)    =   0x34
```

TSTFSZ — Test f, skip if 0

Syntax:	[*label*] TSTFSZ f [,a]
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	skip if f = 0
Status Affected:	None
Encoding:	`0110` `011a` `ffff` `ffff`
Description:	If 'f' = 0, the next instruction, fetched during the current instruction execution, is discarded and a `NOP` is executed, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

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

If skip and followed by 2-word instruction:

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

Example:

```
HERE    TSTFSZ  CNT, 1
NZERO     :
ZERO    :
```

Before Instruction

```
PC = Address(HERE)
```

After Instruction

```
If CNT    =   0x00,
    PC    =   Address (ZERO)
If CNT    ≠   0x00,
    PC    =   Address (NZERO)
```

XORLW — Exclusive OR literal with WREG

Syntax:	[*label*] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(WREG) .XOR. k → WREG
Status Affected:	N,Z
Encoding:	`0000` `1010` `kkkk` `kkkk`
Description:	The contents of WREG are XORed with the 8-bit literal 'k'. The result is placed in WREG.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to WREG

Example: XORLW 0xAF

Before Instruction

```
WREG   =   0xB5
```

After Instruction

```
WREG   =   0x1A
```

TABLE 20-1: DEVELOPMENT TOOLS FROM MICROCHIP

Device	Software Tools: MPLAB® Integrated Development Environment	Software Tools: MPLAB® C17 C Compiler	Software Tools: MPLAB® C18 C Compiler	Software Tools: MPASM™ Assembler/ MPLINK™ Object Linker	Emulators: MPLAB® ICE In-Circuit Emulator	Emulators: ICEPIC™ In-Circuit Emulator	Debugger: MPLAB® ICD In-Circuit Debugger	Programmers: PICSTART® Plus Entry Level Development Programmer	Programmers: PRO MATE® II Universal Device Programmer
PIC12CXXX	✓			✓	✓	✓		✓	✓
PIC14000	✓			✓	✓			✓	✓
PIC16C5X	✓			✓	✓	✓		✓	✓
PIC16C6X	✓			✓	✓	✓	✓*	✓	✓
PIC16CXXX	✓			✓	✓	✓		✓	✓
PIC16F62X	✓			✓	✓**			✓**	✓**
PIC16C7X	✓			✓	✓	✓	✓*	✓	✓
PIC16C7XX	✓			✓	✓	✓		✓	✓
PIC16C8X	✓			✓	✓	✓		✓	✓
PIC16F8XX	✓			✓	✓		✓	✓	✓
PIC16C9XX	✓			✓	✓	✓		✓	✓
PIC17C4X	✓	✓		✓	✓			✓	✓
PIC17C7XX	✓	✓		✓	✓			✓	✓
PIC18CXX2	✓		✓	✓	✓			✓	✓
24CXX/ 25CXX/ 93CXX				✓					✓
HCSXXX				✓					✓
MCRFXXX									
MCP2510									

Device	PICDEM™ 1 Demonstration Board	PICDEM™ 2 Demonstration Board	PICDEM™ 3 Demonstration Board	PICDEM™ 14A Demonstration Board	PICDEM™ 17 Demonstration Board	KEELOQ® Evaluation Kit	KEELOQ® Transponder Kit	microID™ Programmer's Kit	125 kHz microID™ Developer's Kit	125 kHz Anticollision microID™ Developer's Kit	13.56 MHz Anticollision microID™ Developer's Kit	MCP2510 CAN Developer's Kit
PIC12CXXX												
PIC14000				✓								
PIC16C5X	✓											
PIC16C6X		✓†										
PIC16CXXX	✓											
PIC16F62X												
PIC16C7X	✓†	✓†										
PIC16C7XX												
PIC16C8X	✓											
PIC16F8XX												
PIC16C9XX			✓									
PIC17C4X	✓											
PIC17C7XX					✓							
PIC18CXX2		✓										
24CXX/ 25CXX/ 93CXX												
HCSXXX						✓	✓					
MCRFXXX								✓	✓	✓	✓	
MCP2510												✓

* Contact the Microchip Technology Inc. web site at www.microchip.com for information on how to use the MPLAB® ICD In-Circuit Debugger (DV164001) with PIC16C62, 63, 64, 65, 72, 73, 74, 76, 77.

** Contact Microchip Technology Inc. for availability date.

† Development tool is available on select devices.

FIGURE 22-11: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R (RC MODE, C = 100 pF, 25°C)

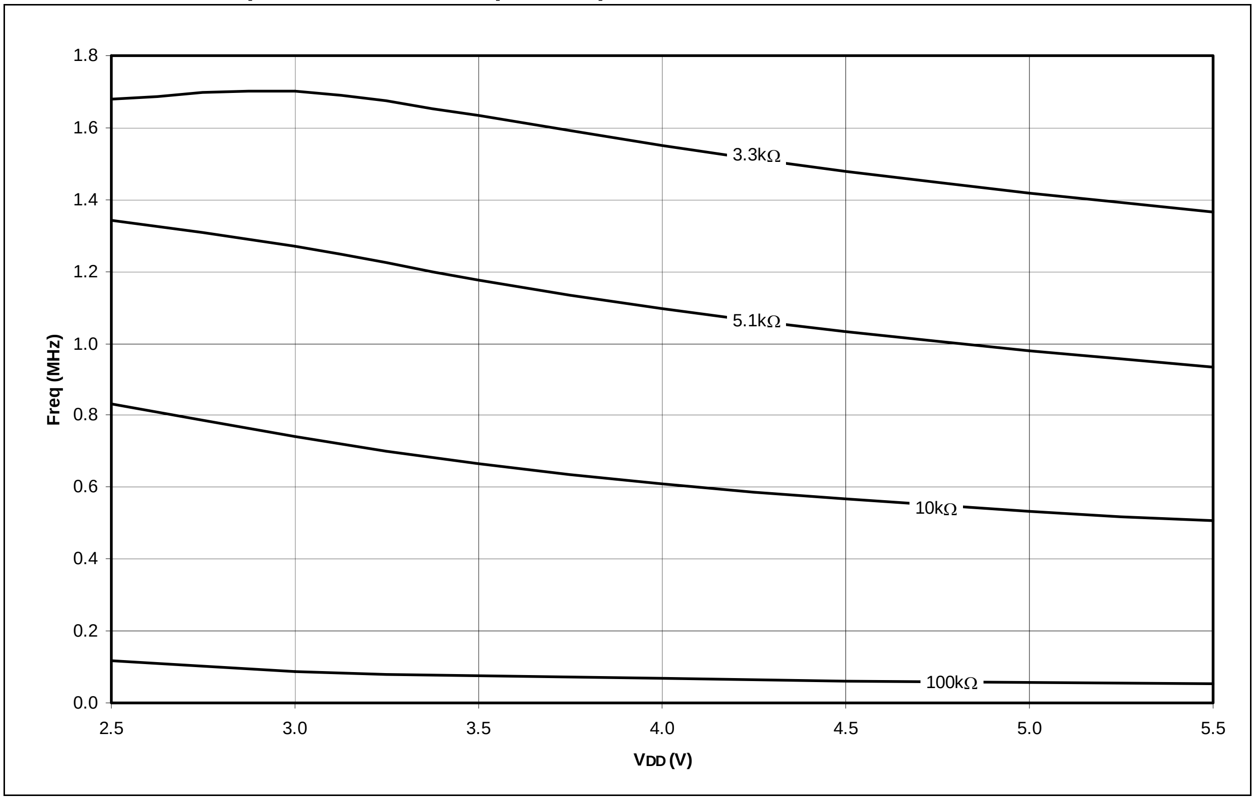
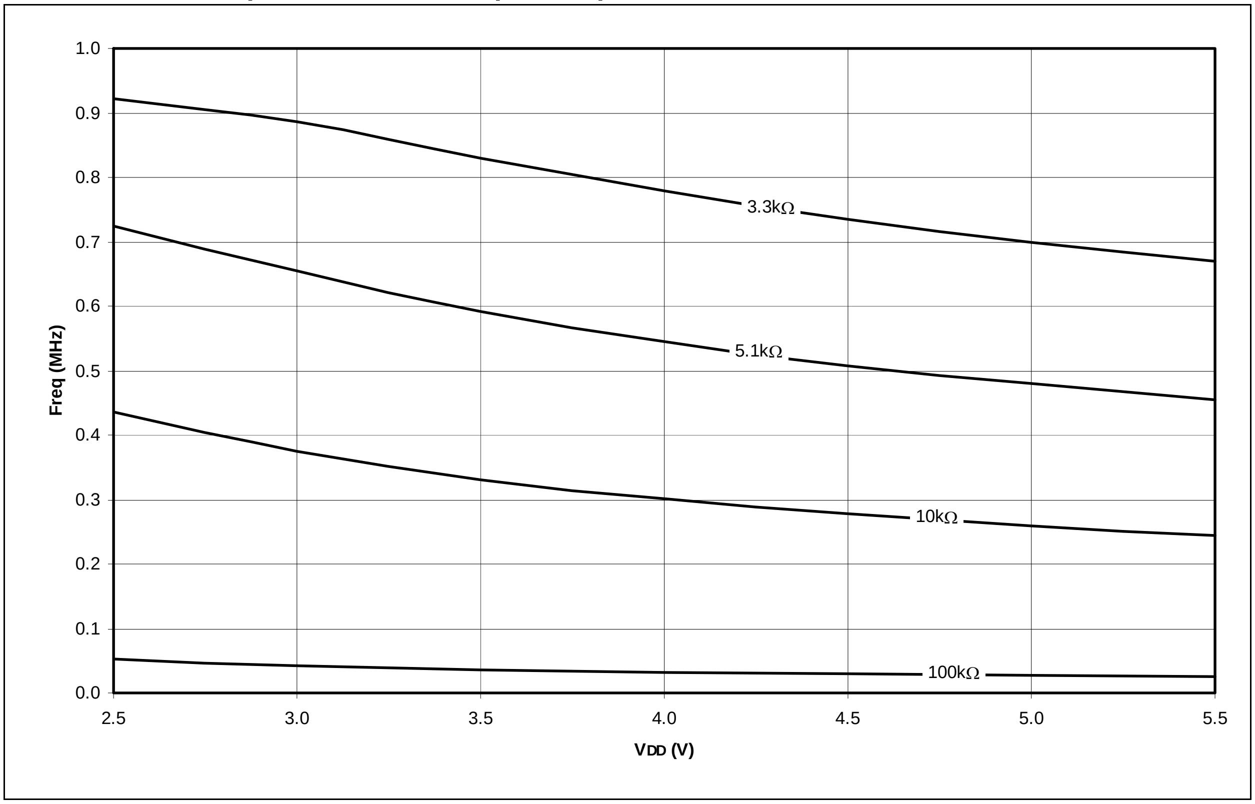
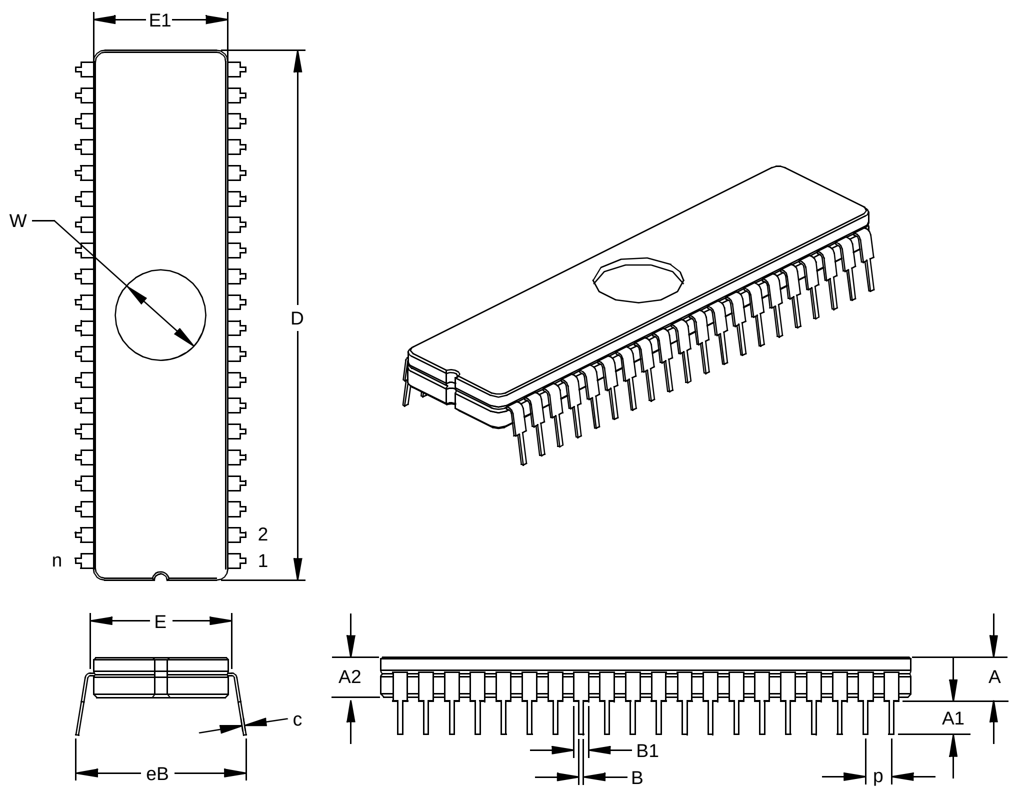


FIGURE 22-12: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R (RC MODE, C = 300 pF, 25°C)



40-Lead Ceramic Dual In-line with Window (JW) – 600 mil (CERDIP)

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



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		40			40	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.185	.205	.225	4.70	5.21	5.72
Ceramic Package Height	A2	.155	.160	.165	3.94	4.06	4.19
Standoff	A1	.030	.045	.060	0.76	1.14	1.52
Shoulder to Shoulder Width	E	.595	.600	.625	15.11	15.24	15.88
Ceramic Pkg. Width	E1	.514	.520	.526	13.06	13.21	13.36
Overall Length	D	2.040	2.050	2.060	51.82	52.07	52.32
Tip to Seating Plane	L	.135	.140	.145	3.43	3.56	3.68
Lead Thickness	c	.008	.011	.014	0.20	0.28	0.36
Upper Lead Width	B1	.050	.053	.055	1.27	1.33	1.40
Lower Lead Width	B	.016	.020	.023	0.41	0.51	0.58
Overall Row Spacing §	eB	.610	.660	.710	15.49	16.76	18.03
Window Diameter	W	.340	.350	.360	8.64	8.89	9.14

* Controlling Parameter
§ Significant Characteristic
JEDEC Equivalent: MO-103
Drawing No. C04-014