



Welcome to [E-XFL.COM](https://www.e-xfl.com)

What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "[Embedded - Microcontrollers](#)"

Details

Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	32MHz
Connectivity	I ² C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	17
Program Memory Size	14KB (8K x 14)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1829-e-so

2.0 ENHANCED MID-RANGE CPU

This family of devices contain an enhanced mid-range 8-bit CPU core. The CPU has 49 instructions. Interrupt capability includes automatic context saving. The hardware stack is 16 levels deep and has Overflow and Underflow Reset capability. Direct, Indirect, and Relative Addressing modes are available. Two File Select Registers (FSRs) provide the ability to read program and data memory.

- Automatic Interrupt Context Saving
- 16-level Stack with Overflow and Underflow
- File Select Registers
- Instruction Set

2.1 Automatic Interrupt Context Saving

During interrupts, certain registers are automatically saved in shadow registers and restored when returning from the interrupt. This saves stack space and user code. See **Section 8.5 “Automatic Context Saving”**, for more information.

2.2 16-level Stack with Overflow and Underflow

These devices have an external stack memory 15 bits wide and 16 words deep. A Stack Overflow or Underflow will set the appropriate bit (STKOVF or STKUNF) in the PCON register, and if enabled will cause a software Reset. See **Section 3.4 “Stack”** for more details.

2.3 File Select Registers

There are two 16-bit File Select Registers (FSR). FSRs can access all file registers and program memory, which allows one Data Pointer for all memory. When an FSR points to program memory, there is one additional instruction cycle in instructions using INDF to allow the data to be fetched. General purpose memory can now also be addressed linearly, providing the ability to access contiguous data larger than 80 bytes. There are also new instructions to support the FSRs. See **Section 3.5 “Indirect Addressing”** for more details.

2.4 Instruction Set

There are 49 instructions for the enhanced mid-range CPU to support the features of the CPU. See **Section 29.0 “Instruction Set Summary”** for more details.

PIC16(L)F1825/9

TABLE 3-8: 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	Value on all other Resets	
Bank 0												
000h ⁽¹⁾	INDF0	Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register)								xxxx xxxx	xxxx xxxx	
001h ⁽¹⁾	INDF1	Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register)								xxxx xxxx	xxxx xxxx	
002h ⁽¹⁾	PCL	Program Counter (PC) Least Significant Byte								0000 0000	0000 0000	
003h ⁽¹⁾	STATUS	—	—	—	\overline{TO}	\overline{PD}	Z	DC	C	---1 1000	---q quuu	
004h ⁽¹⁾	FSR0L	Indirect Data Memory Address 0 Low Pointer								0000 0000	uuuu uuuu	
005h ⁽¹⁾	FSR0H	Indirect Data Memory Address 0 High Pointer								0000 0000	0000 0000	
006h ⁽¹⁾	FSR1L	Indirect Data Memory Address 1 Low Pointer								0000 0000	uuuu uuuu	
007h ⁽¹⁾	FSR1H	Indirect Data Memory Address 1 High Pointer								0000 0000	0000 0000	
008h ⁽¹⁾	BSR	—	—	—	BSR<4:0>					---0 0000	---0 0000	
009h ⁽¹⁾	WREG	Working Register								0000 0000	uuuu uuuu	
00Ah ⁽¹⁾	PCLATH	—	Write Buffer for the upper 7 bits of the Program Counter								-000 0000	-000 0000
00Bh ⁽¹⁾	INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	0000 0000	0000 0000	
00Ch	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--xx xxxx	--xx xxxx	
00Dh	PORTB ⁽²⁾	RB7	RB6	RB5	RB4	—	—	—	—	xxxx ----	xxxx ----	
00Eh	PORTC	RC7 ⁽²⁾	RC6 ⁽²⁾	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	xxxx xxxx	
00Fh	—	Unimplemented								—	—	
010h	—	Unimplemented								—	—	
011h	PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000	
012h	PIR2	OSFIF	C2IF	C1IF	EEIF	BCL1IF	—	—	CCP2IF	0000 0--0	0000 0--0	
013h	PIR3	—	—	CCP4IF	CCP3IF	TMR6IF	—	TMR4IF	—	--00 0-0-	--00 0-0-	
014h	PIR4 ⁽²⁾	—	—	—	—	—	—	BCL2IF	SSP2IF	---- --00	---- --00	
015h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu	
016h	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu	
017h	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu	
018h	T1CON	TMR1CS1	TMR1CS0	T1CKPS<1:0>		T1OSCEN	T1SYNC	—	TMR1ON	0000 00-0	uuuu uu-u	
019h	T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/ DONE	T1GVAL	T1GSS<1:0>		0000 0x00	uuuu uxuu	
01Ah	TMR2	Timer2 Module Register								0000 0000	0000 0000	
01Bh	PR2	Timer2 Period Register								1111 1111	1111 1111	
01Ch	T2CON	—	T2OUTPS<3:0>				TMR2ON	T2CKPS<1:0>		-000 0000	-000 0000	
01Dh	—	Unimplemented								—	—	
01Eh	CPSCON0	CPSON	CPSRM	—	—	CPSRNG<1:0>		CPSOUT	T0XCS	00-- 0000	00-- 0000	
01Fh	CPSCON1	—	—	—	—	CPSCH<3:0>				---- 0000	---- 0000	

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

Note 1: These registers can be addressed from any bank.

2: PIC16(L)F1829 only.

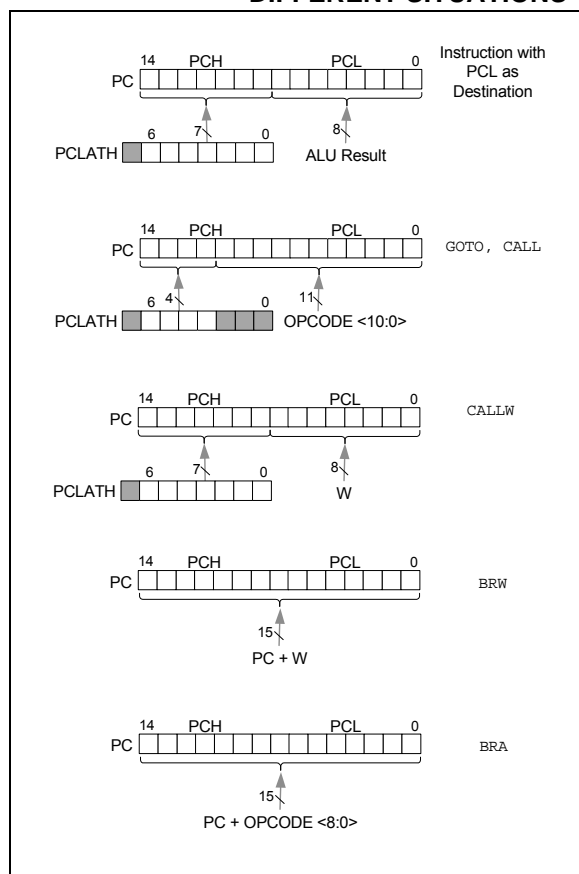
3: PIC16(L)F1825 only.

4: Unimplemented, read as '1'.

3.3 PCL and PCLATH

The Program Counter (PC) is 15 bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<14:8>) is not directly readable or writable and comes from PCLATH. On any Reset, the PC is cleared. Figure 3-3 shows the five situations for the loading of the PC.

FIGURE 3-3: LOADING OF PC IN DIFFERENT SITUATIONS



3.3.1 MODIFYING PCL

Executing any instruction with the PCL register as the destination simultaneously causes the Program Counter PC<14:8> bits (PCH) to be replaced by the contents of the PCLATH register. This allows the entire contents of the program counter to be changed by writing the desired upper seven bits to the PCLATH register. When the lower eight bits are written to the PCL register, all 15 bits of the program counter will change to the values contained in the PCLATH register and those being written to the PCL register.

3.3.2 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When performing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to the Application Note AN556, "Implementing a Table Read" (DS00556).

3.3.3 COMPUTED FUNCTION CALLS

A computed function CALL allows programs to maintain tables of functions and provide another way to execute state machines or look-up tables. When performing a table read using a computed function CALL, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block).

If using the CALL instruction, the PCH<2:0> and PCL registers are loaded with the operand of the CALL instruction. PCH<6:3> is loaded with PCLATH<6:3>.

The CALLW instruction enables computed calls by combining PCLATH and W to form the destination address. A computed CALLW is accomplished by loading the W register with the desired address and executing CALLW. The PCL register is loaded with the value of W and PCH is loaded with PCLATH.

3.3.4 BRANCHING

The branching instructions add an offset to the PC. This allows relocatable code and code that crosses page boundaries. There are two forms of branching, BRW and BRA. The PC will have incremented to fetch the next instruction in both cases. When using either branching instruction, a PCL memory boundary may be crossed.

If using BRW, load the W register with the desired unsigned address and execute BRW. The entire PC will be loaded with the address PC + 1 + W.

If using BRA, the entire PC will be loaded with PC + 1 +, the signed value of the operand of the BRA instruction.

FIGURE 3-5: ACCESSING THE STACK EXAMPLE 2

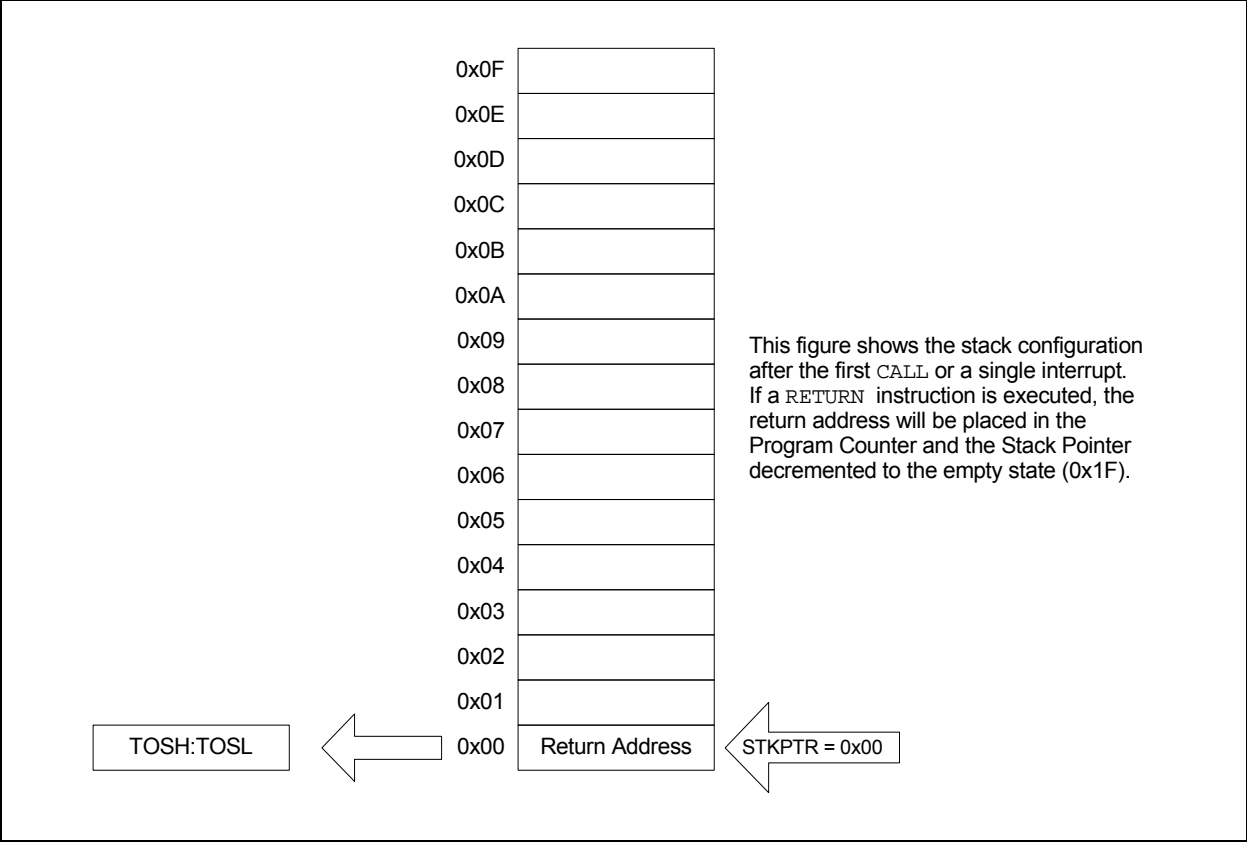
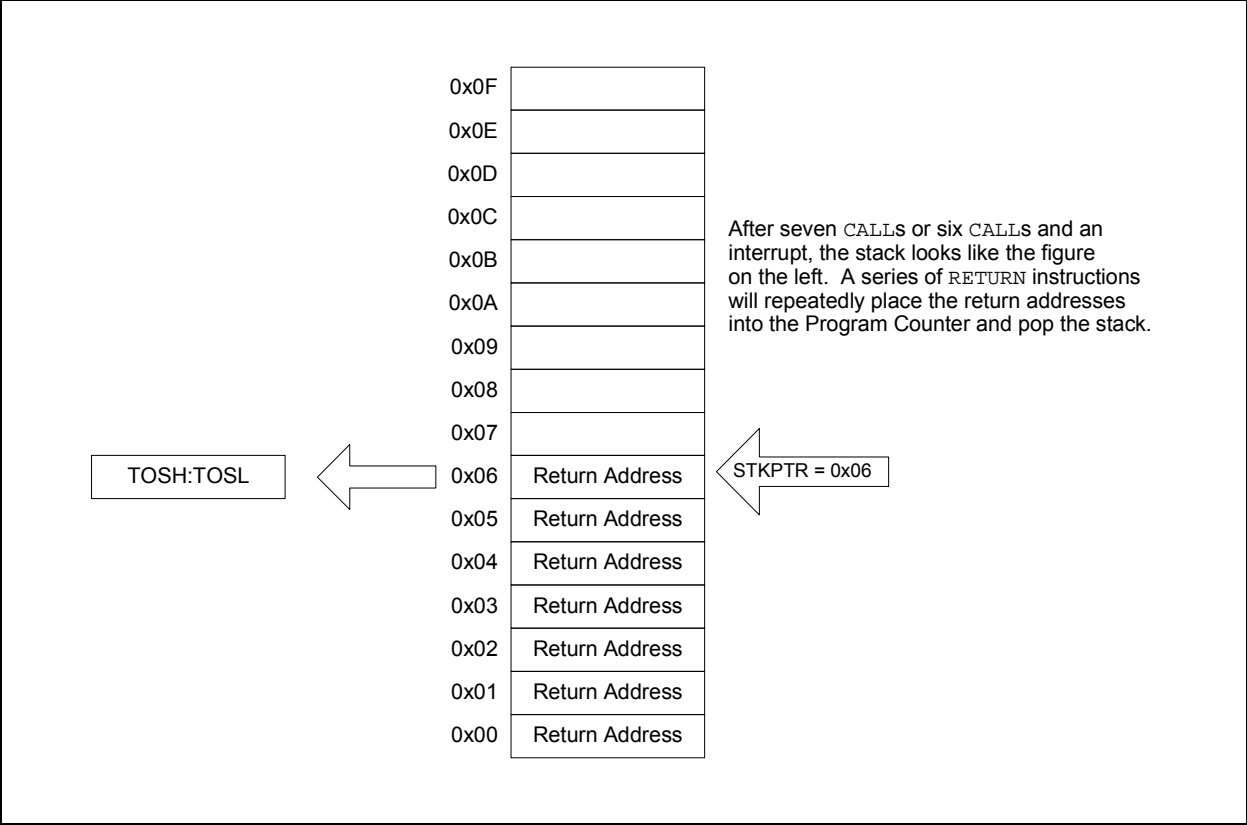


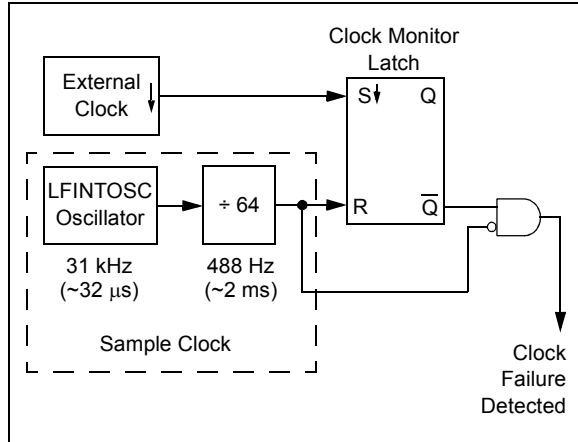
FIGURE 3-6: ACCESSING THE STACK EXAMPLE 3



5.5 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM can detect oscillator failure any time after the Oscillator Start-up Timer (OST) has expired. The FSCM is enabled by setting the FCMEN bit in the Configuration Word 1. The FSCM is applicable to all external Oscillator modes (LP, XT, HS, EC, Timer1 Oscillator and RC).

FIGURE 5-9: FSCM BLOCK DIAGRAM



5.5.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64. See Figure 5-9. Inside the fail detector block is a latch. The external clock sets the latch on each falling edge of the external clock. The sample clock clears the latch on each rising edge of the sample clock. A failure is detected when an entire half-cycle of the sample clock elapses before the external clock goes low.

5.5.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM switches the device clock to an internal clock source and sets the bit flag OSFIF of the PIR2 register. Setting this flag will generate an interrupt if the OSFIE bit of the PIE2 register is also set. The device firmware can then take steps to mitigate the problems that may arise from a failed clock. The system clock will continue to be sourced from the internal clock source until the device firmware successfully restarts the external oscillator and switches back to external operation.

The internal clock source chosen by the FSCM is determined by the IRCF<3:0> bits of the OSCCON register. This allows the internal oscillator to be configured before a failure occurs.

5.5.3 FAIL-SAFE CONDITION CLEARING

The Fail-Safe condition is cleared after a Reset, executing a `SLEEP` instruction or changing the SCS bits of the OSCCON register. When the SCS bits are changed, the OST is restarted. While the OST is running, the device continues to operate from the INTOSC selected in OSCCON. When the OST times out, the Fail-Safe condition is cleared after successfully switching to the external clock source. The OSFIF bit should be cleared prior to switching to the external clock source. If the Fail-Safe condition still exists, the OSFIF flag will again become set by hardware.

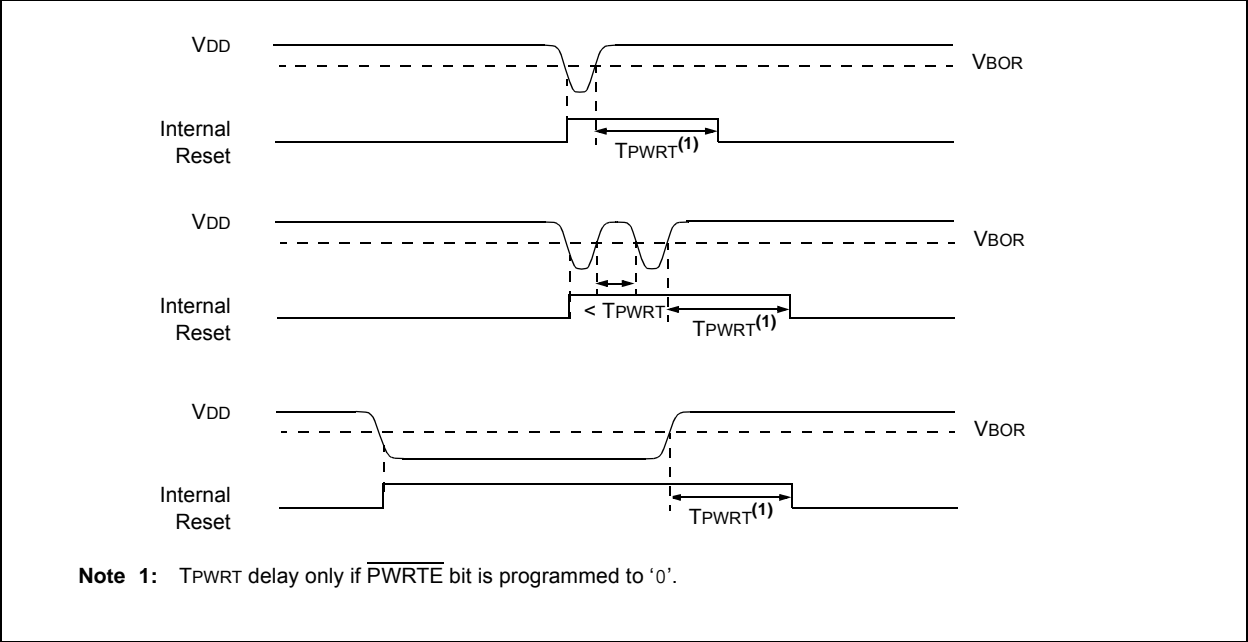
5.5.4 RESET OR WAKE-UP FROM SLEEP

The FSCM is designed to detect an oscillator failure after the Oscillator Start-up Timer (OST) has expired. The OST is used after waking up from Sleep and after any type of Reset. The OST is not used with the EC or RC Clock modes so that the FSCM will be active as soon as the Reset or wake-up has completed. When the FSCM is enabled, the Two-Speed Start-up is also enabled. Therefore, the device will always be executing code while the OST is operating.

Note: Due to the wide range of oscillator start-up times, the Fail-Safe circuit is not active during oscillator start-up (i.e., after exiting Reset or Sleep). After an appropriate amount of time, the user should check the Status bits in the OSCSTAT register to verify the oscillator start-up and that the system clock switchover has successfully completed.

PIC16(L)F1825/9

FIGURE 7-2: BROWN-OUT SITUATIONS



REGISTER 7-1: BORCON: BROWN-OUT RESET CONTROL REGISTER

R/W-1/u	U-0	U-0	U-0	U-0	U-0	U-0	R-q/u
SBOREN	—	—	—	—	—	—	BORRDY
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	q = Value depends on condition

- bit 7 **SBOREN:** Software Brown-out Reset Enable bit
If $\text{BOREN} \langle 1:0 \rangle$ in Configuration Word 1 $\neq 01$:
SBOREN is read/write, but has no effect on the BOR.
If $\text{BOREN} \langle 1:0 \rangle$ in Configuration Word 1 $= 01$:
1 = BOR Enabled
0 = BOR Disabled
- bit 6-1 **Unimplemented:** Read as '0'
- bit 0 **BORRDY:** Brown-out Reset Circuit Ready Status bit
1 = The Brown-out Reset circuit is active
0 = The Brown-out Reset circuit is inactive

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 **PD** bit. If the **PD** bit is set, the **SLEEP** instruction was executed as a **NOP**.

FIGURE 9-1: WAKE-UP FROM SLEEP THROUGH INTERRUPT

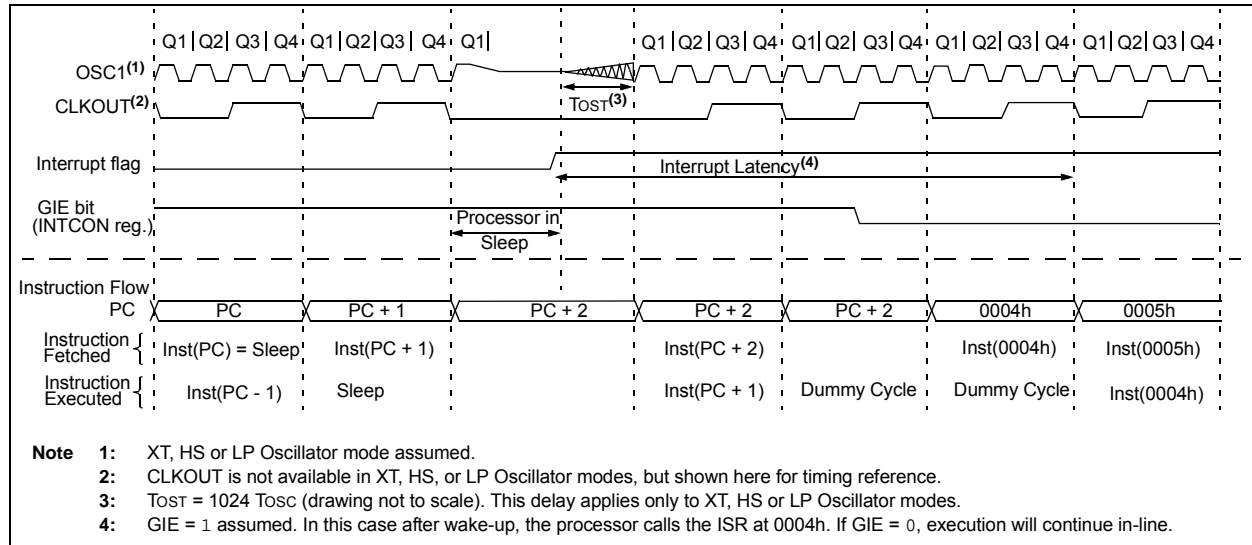


TABLE 9-1: SUMMARY OF REGISTERS ASSOCIATED WITH POWER-DOWN MODE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE	PEIE	TMR0IE	INTE	IOCFIE	TMR0IF	INTF	IOCF	87
IOCAF	—	—	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0	138
IOCAN	—	—	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0	138
IOCAP	—	—	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0	137
IOCBF ⁽¹⁾	IOCBF7	IOCBF6	IOCBF5	IOCBF4	—	—	—	—	139
IOCBN ⁽¹⁾	IOCBN7	IOCBN6	IOCBN5	IOCBN4	—	—	—	—	139
IOCBP ⁽¹⁾	IOCBP7	IOCBP6	IOCBP5	IOCBP4	—	—	—	—	138
PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	88
PIE2	OSFIE	C2IE	C1IE	EEIE	BCL1IE	—	—	CCP2IE	89
PIE4 ⁽¹⁾	—	—	—	—	—	—	BCL2IE	SSP2IE	91
PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	92
PIR2	OSFIF	C2IF	C1IF	EEIF	BCL1IF	—	—	CCP2IF	93
PIR4 ⁽¹⁾	—	—	—	—	—	—	BCL2IF	SSP2IF	95
STATUS	—	—	—	\overline{TO}	\overline{PD}	Z	DC	C	22
WDTCON	—	—	WDTPS<4:0>					SWDTEN	100

Legend: — Unimplemented, read as '0'. Shaded cells are not used in Power-Down mode.

Note 1: PIC16(L)F1829 only.

12.2.2 PORTA FUNCTIONS AND OUTPUT PRIORITIES

Each PORTA pin is multiplexed with other functions. The pins, their combined functions and their output priorities are briefly described here. For additional information, refer to the appropriate section in this data sheet.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the lowest number in the following lists.

Analog input functions, such as ADC, comparator and Cap Sense inputs, are not shown in the priority lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown below.

TABLE 12-2: PORTA OUTPUT PRIORITY

Pin Name	Function Priority ⁽¹⁾
RA0	ICSPDAT ICDDAT DACOUT
RA1	ICSPCLK ICDCLK RX/DT ⁽²⁾
RA2	SRQ C1OUT CCP3
RA3	None (input only)
RA4	CLKOUT T1OSO CLKR SDO1 P2B ⁽²⁾
RA5	SDO2 (PIC16(L)F1829 only) CCP2 ⁽²⁾ /P2A ⁽²⁾

- Note** 1: Priority listed from highest to lowest.
 2: Pin function is selectable via the APFCON0 or APFCON1 register.

PIC16(L)F1825/9

TABLE 13-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELA	—	—	—	ANSA4	—	ANSA2	ANSA1	ANSA0	123
ANSELB ⁽¹⁾	—	—	ANSB5	ANSB4	—	—	—	—	129
INLVLA	—	—	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0	124
INLVLB ⁽¹⁾	INLVLB7	INLVLB6	INLVLB5	INLVLB4	—	—	—	—	129
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	87
IOCAF	—	—	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0	138
IOCAN	—	—	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0	138
IOCAP	—	—	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0	137
IOCBF ⁽¹⁾	IOCBF7	IOCBF6	IOCBF5	IOCBF4	—	—	—	—	139
IOCBN ⁽¹⁾	IOCBN7	IOCBN6	IOCBN5	IOCBN4	—	—	—	—	139
IOCBP ⁽¹⁾	IOCBP7	IOCBP6	IOCBP5	IOCBP4	—	—	—	—	138
TRISA	—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	122
TRISB ⁽¹⁾	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	128

Legend: — Unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.

Note 1: PIC16(L)F1829 only.

PIC16(L)F1825/9

20.2 Option and Timer0 Control Register

REGISTER 20-1: OPTION_REG: OPTION REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
WPUEN	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>		
bit 7							bit 0

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
u = Bit is unchanged x = Bit is unknown -n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set '0' = Bit is cleared

- bit 7 **WPUEN:** Weak Pull-up Enable bit
1 = All weak pull-ups are disabled (except $\overline{\text{MCLR}}$, if it is enabled)
0 = Weak pull-ups are enabled by individual WPUx latch values
- bit 6 **INTEDG:** Interrupt Edge Select bit
1 = Interrupt on rising edge of INT pin
0 = Interrupt on falling edge of INT pin
- bit 5 **TMR0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (Fosc/4)
- bit 4 **TMR0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin
- bit 3 **PSA:** Prescaler Assignment bit
1 = Prescaler is not assigned to the Timer0 module
0 = Prescaler is assigned to the Timer0 module
- bit 2-0 **PS<2:0>:** Prescaler Rate Select bits

Bit Value	Timer0 Rate
000	1 : 2
001	1 : 4
010	1 : 8
011	1 : 16
100	1 : 32
101	1 : 64
110	1 : 128
111	1 : 256

TABLE 20-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMERO

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CPSCON0	CPSON	CPSRM	—	—	CPSRNG<1:0>		CPSOUT	T0xCS	315
FVRCON	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR<1:0>		ADFVR<1:0>		142
INLVLA	—	—	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0	124
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	87
OPTION_REG	WPUEN	INTEDG	TMR0CS	TMR0SE	PSA	PS<2:0>			176
TMR0	Timer0 Module Register								174*
TRISA	—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	122

Legend: — Unimplemented location, read as '0'. Shaded cells are not used by the Timer0 module.

* Page provides register information.

24.3 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides power to a load by switching quickly between fully on and fully off states. The PWM signal resembles a square wave where the high portion of the signal is considered the on state and the low portion of the signal is considered the off state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of on and off time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the on time to the off time and is expressed in percentages, where 0% is fully off and 100% is fully on. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied.

Figure 24-3 shows a typical waveform of the PWM signal.

24.3.1 STANDARD PWM OPERATION

The standard PWM function described in this section is available and identical for CCP modules ECCP1, ECCP2, CCP3 and CCP4.

The standard PWM mode generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to 10 bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- PRx registers
- TxCON registers
- CCPRxL registers
- CCPxCON registers

Figure 24-4 shows a simplified block diagram of PWM operation.

Note 1: The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

2: Clearing the CCPxCON register will relinquish control of the CCPx pin.

FIGURE 24-3: CCP PWM OUTPUT SIGNAL

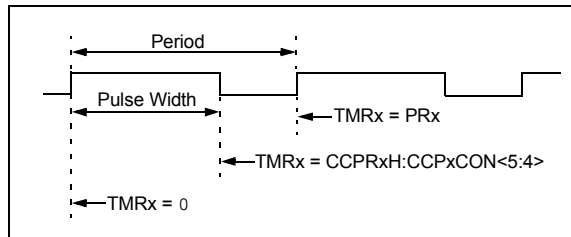


FIGURE 24-4: SIMPLIFIED PWM BLOCK DIAGRAM

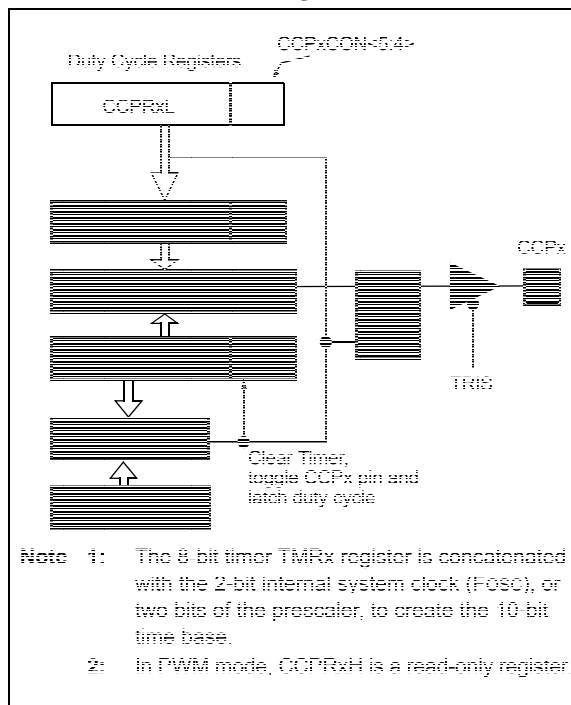
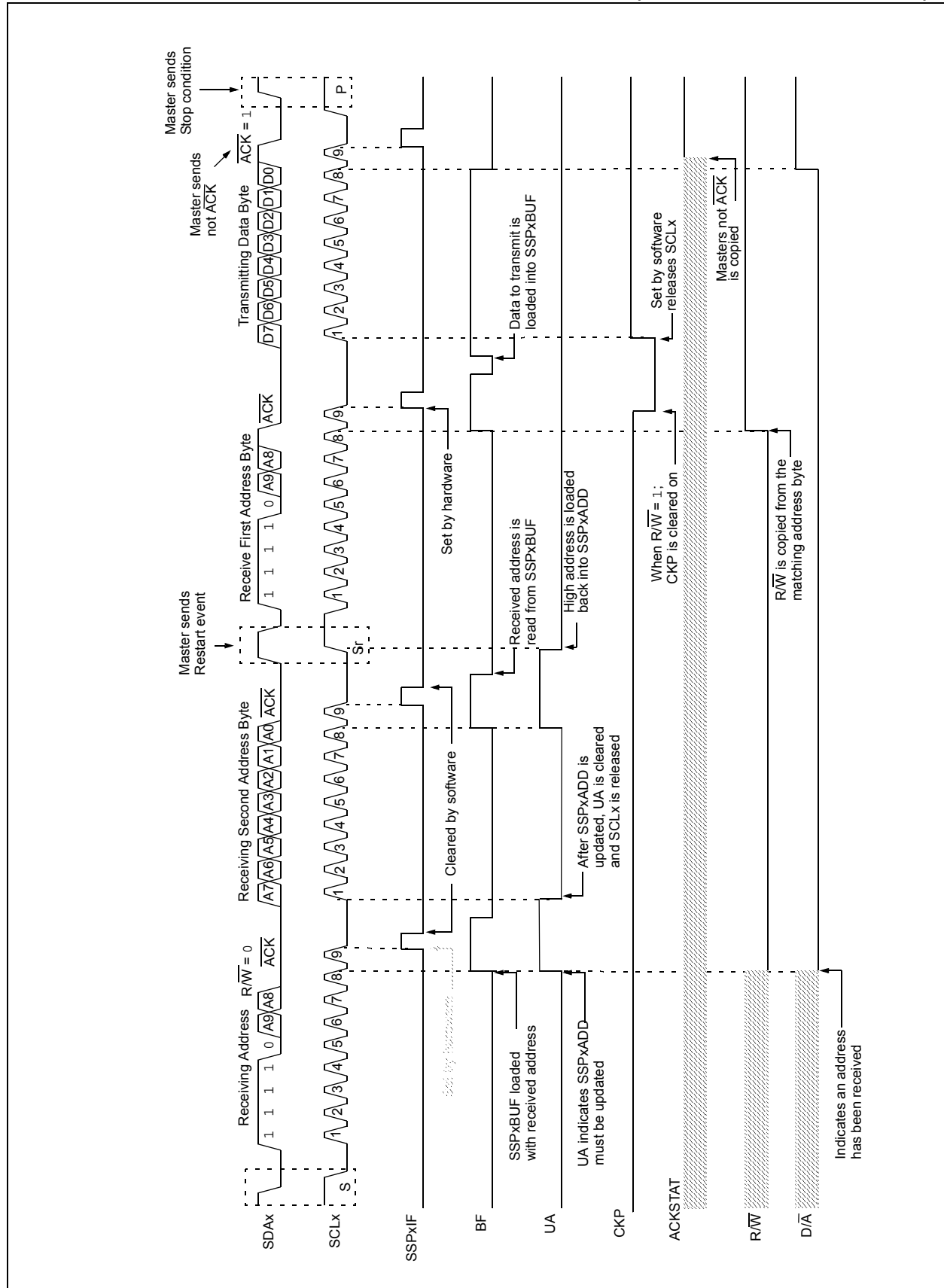


FIGURE 25-22: I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)



25.6.10 SLEEP OPERATION

While in Sleep mode, the I²C slave module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSPx interrupt is enabled).

25.6.11 EFFECTS OF A RESET

A Reset disables the MSSPx module and terminates the current transfer.

25.6.12 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSPx module is disabled. Control of the I²C bus may be taken when the P bit of the SSPxSTAT register is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the SSPx interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDAx line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLxIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

25.6.13 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDAx pin, arbitration takes place when the master outputs a '1' on SDAx, by letting SDAx float high and another master asserts a '0'. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a '1' and the data sampled on the SDAx pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF and reset the I²C port to its Idle state (Figure 25-32).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDAx and SCLx lines are deasserted and the SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I²C bus is free, the user can resume communication by asserting a Start condition.

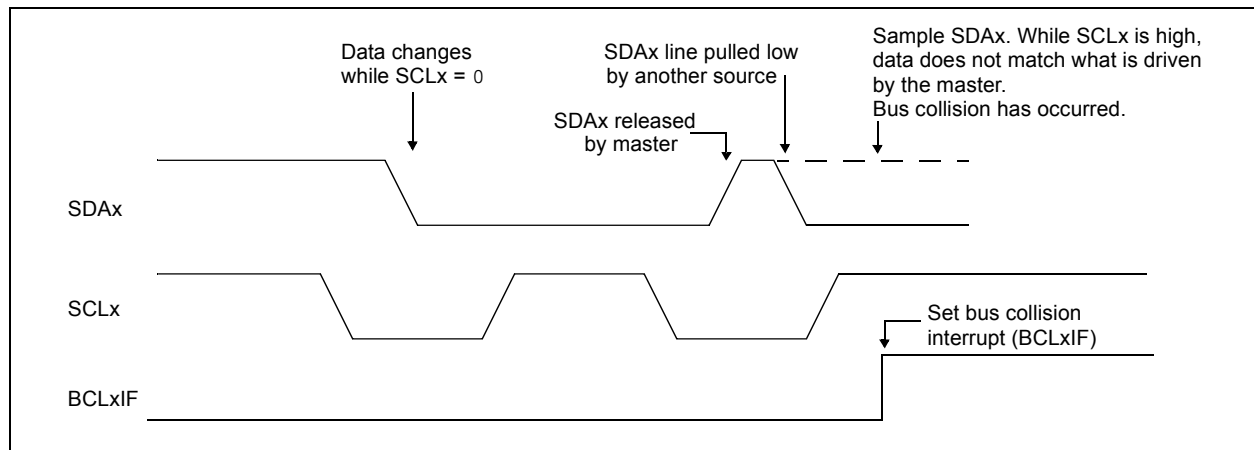
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDAx and SCLx lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I²C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDAx and SCLx pins. If a Stop condition occurs, the SSPxIF bit will be set.

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

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

FIGURE 25-32: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



26.4.1.5 Synchronous Master Reception

Data is received at the RX/DT pin. The RX/DT pin output driver is automatically disabled when the EUSART is configured for synchronous master receive operation.

In Synchronous mode, reception is enabled by setting either the Single Receive Enable bit (SREN of the RCSTA register) or the Continuous Receive Enable bit (CREN of the RCSTA register).

When SREN is set and CREN is clear, only as many clock cycles are generated as there are data bits in a single character. The SREN bit is automatically cleared at the completion of one character. When CREN is set, clocks are continuously generated until CREN is cleared. If CREN is cleared in the middle of a character the CK clock stops immediately and the partial character is discarded. If SREN and CREN are both set, then SREN is cleared at the completion of the first character and CREN takes precedence.

To initiate reception, set either SREN or CREN. Data is sampled at the RX/DT pin on the trailing edge of the TX/CK clock pin and is shifted into the Receive Shift Register (RSR). When a complete character is received into the RSR, the RCIF bit is set and the character is automatically transferred to the two character receive FIFO. The Least Significant eight bits of the top character in the receive FIFO are available in RCREG. The RCIF bit remains set as long as there are unread characters in the receive FIFO.

Note: If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

26.4.1.6 Slave Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a slave receives the clock on the TX/CK line. The TX/CK pin output driver is automatically disabled when the device is configured for synchronous slave transmit or receive operation. Serial data bits change on the leading edge to ensure they are valid at the trailing edge of each clock. One data bit is transferred for each clock cycle. Only as many clock cycles should be received as there are data bits.

Note: If the device is configured as a slave and the TX/CK function is on an analog pin, the corresponding ANSEL bit must be cleared.

26.4.1.7 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before RCREG is read to access the FIFO. When this happens the OERR bit of the RCSTA register is set. Previous data in the FIFO will not be overwritten. The two characters in the FIFO buffer can be read, however, no additional characters will be received until the error is cleared. The OERR bit can only be cleared by clearing the overrun condition. If the overrun error occurred when the SREN bit is set and CREN is clear then the error is cleared by reading RCREG. If the overrun occurred when the CREN bit is set then the error condition is cleared by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

26.4.1.8 Receiving 9-bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTA register is set the EUSART will shift 9-bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth, and Most Significant, data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the eight Least Significant bits from the RCREG.

26.4.1.9 Synchronous Master Reception Setup:

1. Initialize the SPBRGH, SPBRGL register pair for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Clear the ANSEL bit for the RX pin (if applicable).
3. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
4. Ensure bits CREN and SREN are clear.
5. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
6. If 9-bit reception is desired, set bit RX9.
7. Start reception by setting the SREN bit or for continuous reception, set the CREN bit.
8. Interrupt flag bit RCIF will be set when reception of a character is complete. An interrupt will be generated if the enable bit RCIE was set.
9. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
10. Read the 8-bit received data by reading the RCREG register.
11. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

PIC16(L)F1825/9

26.4.2 SYNCHRONOUS SLAVE MODE

The following bits are used to configure the EUSART for synchronous slave operation:

- SYNC = 1
- CSRC = 0
- SREN = 0 (for transmit); SREN = 1 (for receive)
- CREN = 0 (for transmit); CREN = 1 (for receive)
- SPEN = 1

Setting the SYNC bit of the TXSTA register configures the device for synchronous operation. Clearing the CSRC bit of the TXSTA register configures the device as a slave. Clearing the SREN and CREN bits of the RCSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTA register enables the EUSART.

26.4.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see **Section 27.4.1.3 “Synchronous Master Transmission”**), except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

1. The first character will immediately transfer to the TSR register and transmit.
2. The second word will remain in TXREG register.
3. The TXIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXREG register will transfer the second character to the TSR and the TXIF bit will now be set.
5. If the PEIE and TXIE bits are set, the interrupt will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will call the Interrupt Service Routine.

26.4.2.2 Synchronous Slave Transmission Setup:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for the CK pin (if applicable).
3. Clear the CREN and SREN bits.
4. If interrupts are desired, set the TXIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. If 9-bit transmission is desired, set the TX9 bit.
6. Enable transmission by setting the TXEN bit.
7. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
8. Start transmission by writing the Least Significant eight bits to the TXREG register.

TABLE 26-9: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
APFCON0	RXDTSEL	SDO1SEL ⁽¹⁾	SS1SEL ⁽¹⁾	—	T1GSEL	TXCKSEL	—	—	118
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	292
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	87
PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	88
PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	92
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	291
TXREG	EUSART Transmit Data Register								283*
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	290

Legend: — Unimplemented location, read as ‘0’. Shaded cells are not used for Synchronous Slave Transmission.

* Page provides register information.

Note 1: PIC16(L)F1825 only.

PIC16(L)F1825/9

29.0 INSTRUCTION SET SUMMARY

Each PIC16 instruction is a 14-bit word containing the operation code (opcode) and all required operands. The opcodes are broken into three broad categories.

- Byte Oriented
- Bit Oriented
- Literal and Control

The literal and control category contains the most varied instruction word format.

Table 29-3 lists the instructions recognized by the MPASM™ assembler.

All instructions are executed within a single instruction cycle, with the following exceptions, which may take two or three cycles:

- Subroutine takes two cycles (CALL, CALLW)
- Returns from interrupts or subroutines take two cycles (RETURN, RETLW, RETFIE)
- Program branching takes two cycles (GOTO, BRA, BRW, BTFSS, BTFSC, DECFSZ, INCSFZ)
- One additional instruction cycle will be used when any instruction references an indirect file register and the file select register is pointing to program memory.

One instruction cycle consists of four oscillator cycles; for an oscillator frequency of 4 MHz, this gives a nominal instruction execution rate of 1 MHz.

All instruction examples use the format '0xhh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

29.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the instruction, or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

TABLE 29-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
n	FSR or INDF number. (0-1)
mm	Pre-post increment-decrement mode selection

TABLE 29-2: ABBREVIATION DESCRIPTIONS

Field	Description
PC	Program Counter
\overline{TO}	Time-out bit
C	Carry bit
DC	Digit carry bit
Z	Zero bit
\overline{PD}	Power-down bit

TABLE 30-2: OSCILLATOR PARAMETERS

Standard Operating Conditions (unless otherwise stated) Operating Temperature $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$								
Param No.	Sym.	Characteristic	Freq. Tolerance	Min.	Typ†	Max.	Units	Conditions
OS08	HFOSC	Internal Calibrated HFINTOSC Frequency ⁽¹⁾	$\pm 2\%$	—	16.0	—	MHz	$0^{\circ}\text{C} \leq T_A \leq +60^{\circ}\text{C}$, $V_{DD} \geq 2.5\text{V}$
			$\pm 3\%$	—	16.0	—	MHz	$60^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $V_{DD} \geq 2.5\text{V}$
			$\pm 5\%$	—	16.0	—	MHz	$-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$
OS08A	MFOSC	Internal Calibrated MFINTOSC Frequency ⁽¹⁾	$\pm 2\%$	—	500	—	kHz	$0^{\circ}\text{C} \leq T_A \leq +60^{\circ}\text{C}$, $V_{DD} \geq 2.5\text{V}$
			$\pm 3\%$	—	500	—	kHz	$60^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$, $V_{DD} \geq 2.5\text{V}$
			$\pm 5\%$	—	500	—	kHz	$-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$
OS09	LFOSC	Internal LFINTOSC Frequency	$\pm 25\%$	—	31	—	kHz	$-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$
OS10*	Tiosc ST	HFINTOSC Wake-up from Sleep Start-up Time	—	—	5	8	μs	
		MFINTOSC Wake-up from Sleep Start-up Time	—	—	20	30	μs	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: To ensure these oscillator frequency tolerances, V_{DD} and V_{SS} must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

TABLE 30-3: PLL CLOCK TIMING SPECIFICATIONS ($V_{DD} = 2.7\text{V TO } 5.5\text{V}$)

Param No.	Sym.	Characteristic	Min.	Typ†	Max.	Units	Conditions
F10	FOSC	Oscillator Frequency Range	4	—	8	MHz	
F11	FSYS	On-Chip VCO System Frequency	16	—	32	MHz	
F12	TRC	PLL Start-up Time (Lock Time)	—	—	2	ms	
F13*	ΔCLK	CLKOUT Stability (Jitter)	-0.25%	—	+0.25%	%	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 30-7: CLKOUT AND I/O TIMING

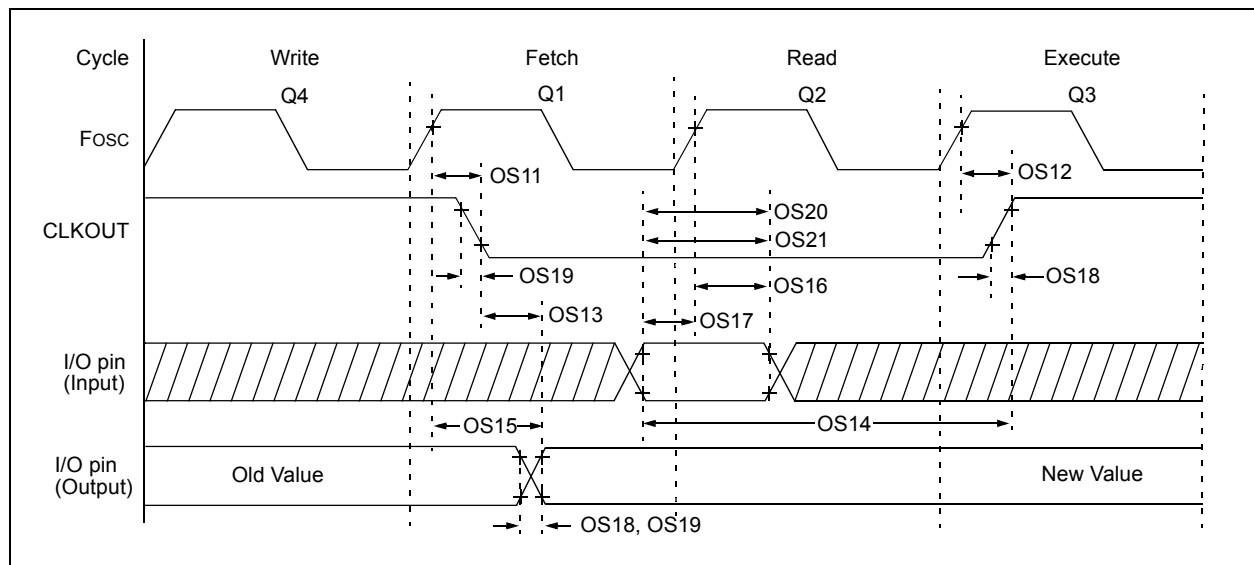


FIGURE 31-13: I_{DD} , MFINTOSC MODE ($F_{osc} = 500$ kHz), PIC16LF1825/9 ONLY

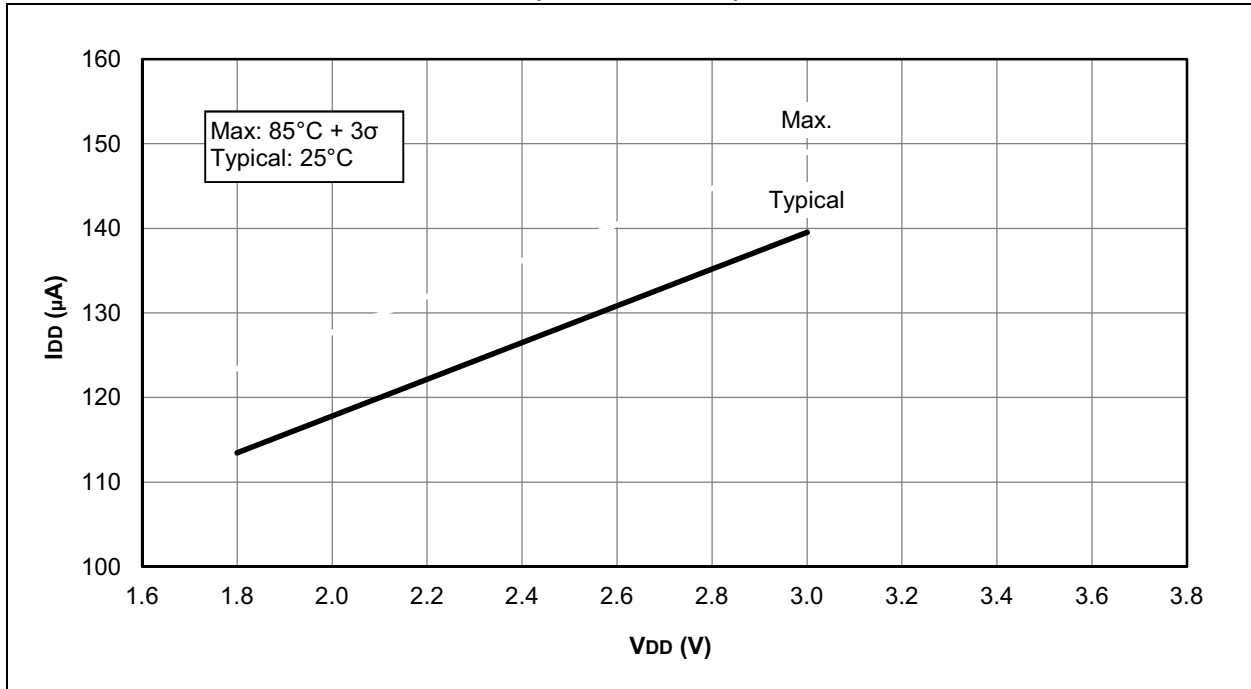
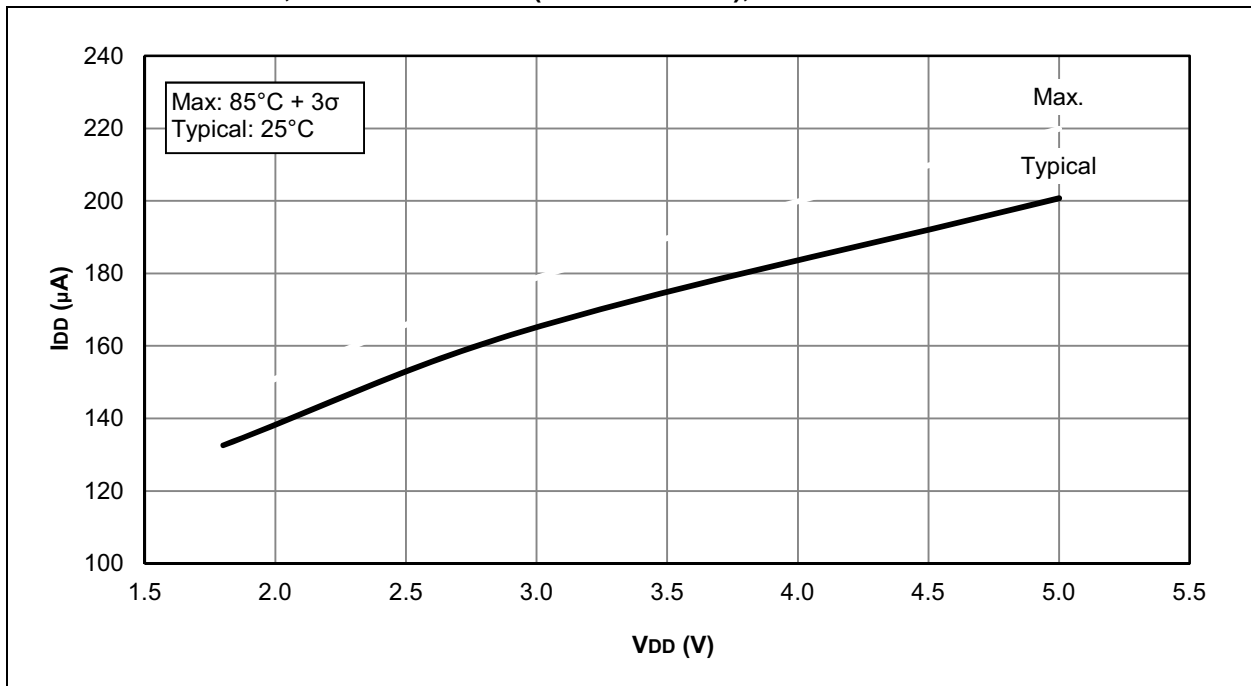


FIGURE 31-14: I_{DD} , MFINTOSC MODE ($F_{osc} = 500$ kHz), PIC16F1825/9 ONLY



32.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
 - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
 - MPLAB XC Compiler
 - MPASM™ Assembler
 - MPLINK™ Object Linker/
MPLIB™ Object Librarian
 - MPLAB Assembler/Linker/Librarian for
Various Device Families
- Simulators
 - MPLAB X SIM Software Simulator
- Emulators
 - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
 - MPLAB ICD 3
 - PICKit™ 3
- Device Programmers
 - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards,
Evaluation Kits and Starter Kits
- Third-party development tools

32.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

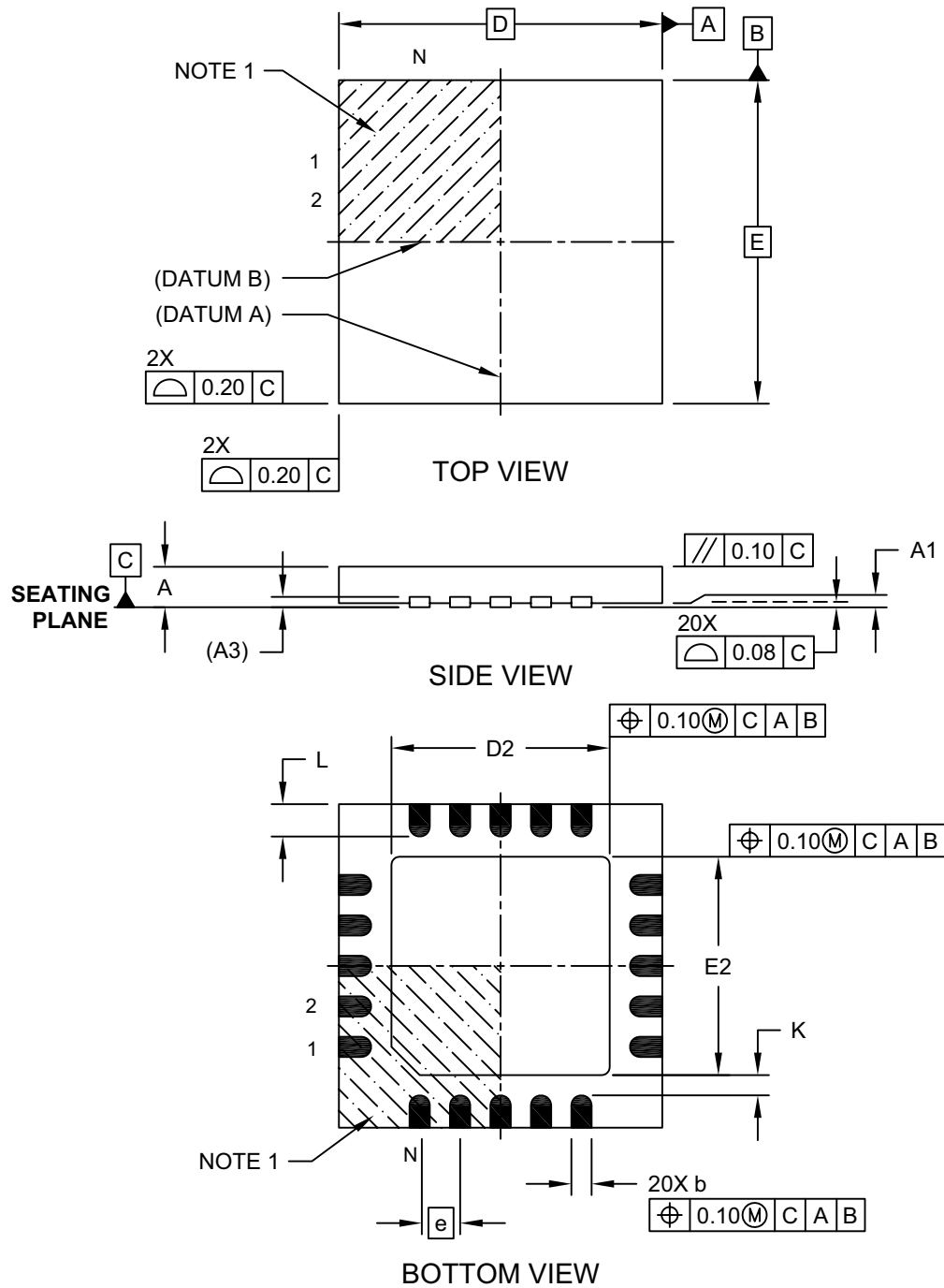
- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

20-Lead Ultra Thin Plastic Quad Flat, No Lead Package (GZ) - 4x4x0.5 mm Body [UQFN]

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



Microchip Technology Drawing C04-255A Sheet 1 of 2