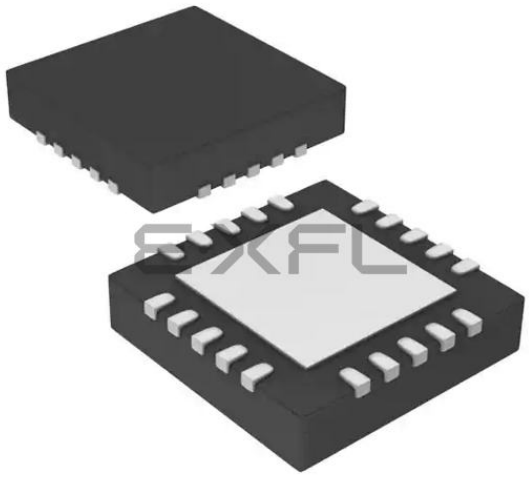


Welcome to E-XFL.COM

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	32MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	18
Program Memory Size	14KB (8K x 14)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 12x10b; D/A 1x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-VFQFN Exposed Pad
Supplier Device Package	20-QFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1619t-i-ml

TABLE 3-3: PIC16(L)F1619 MEMORY MAP, BANK 0-7

BANK 0		BANK 1		BANK 2		BANK 3		BANK 4		BANK 5		BANK 6		BANK 7	
000h	Core Registers (Table 3-1)	080h	Core Registers (Table 3-1)	100h	Core Registers (Table 3-1)	180h	Core Registers (Table 3-1)	200h	Core Registers (Table 3-1)	280h	Core Registers (Table 3-1)	300h	Core Registers (Table 3-1)	380h	Core Registers (Table 3-1)
00Bh		08Bh		10Bh		18Bh		20Bh		28Bh		30Bh		38Bh	
00Ch	PORTA	08Ch	TRISA	10Ch	LATA	18Ch	ANSELA	20Ch	WPUA	28Ch	ODCONA	30Ch	SLRCONA	38Ch	INLVLA
00Dh	PORTB	08Dh	TRISB	10Dh	LATB	18Dh	ANSELB	20Eh	WPUB	28Eh	ODCONB	30Eh	SLRCONB	38Eh	INLVLB
00Eh	PORTC	08Eh	TRISC	10Eh	LATC	18Eh	ANSELC	20Eh	WPUC	28Eh	ODCONC	30Eh	SLRCONC	38Eh	INLVLC
00Fh	—	08Fh	—	10Fh	—	18Fh	—	20Fh	—	28Fh	—	30Fh	—	38Fh	—
010h	PIR1	090h	PIE1	110h	—	190h	—	210h	—	290h	—	310h	—	390h	—
011h	PIR2	091h	PIE2	111h	CM1CON0	191h	PMADRL	211h	SSP1BUF	291h	CCP1RL	311h	—	391h	IOCAP
012h	PIR3	092h	PIE3	112h	CM1CON1	192h	PMADRH	212h	SSP1ADD	292h	CCP1RH	312h	—	392h	IOCAN
013h	PIR4	093h	PIE4	113h	CM2CON0	193h	PMDATL	213h	SSP1MSK	293h	CCP1CON	313h	—	393h	IOCAF
014h	PIR5	094h	PIE5	114h	CM2CON1	194h	PMDATH	214h	SSP1STAT	294h	CCP1CAP	314h	—	394h	IOCBP
015h	TMR0	095h	OPTION_REG	115h	CMOUT	195h	PMCON1	215h	SSP1CON	295h	—	315h	—	395h	IOCBN
016h	TMR1L	096h	PCON	116h	BORCON	196h	PMCON2	216h	SSP1CON2	296h	—	316h	—	396h	IOCBF
017h	TMR1H	097h	—	117h	FVRCON	197h	VREGCON	217h	SSP1CON3	297h	—	317h	—	397h	IOCCP
018h	T1CON	098h	OSCTUNE	118h	DAC1CON0	198h	—	218h	—	298h	CCP2RL	318h	—	398h	IOCCN
019h	T1GCON	099h	OSCCON	119h	DAC1CON1	199h	RC1REG	219h	—	299h	CCP2RH	319h	—	399h	IOCCF
01Ah	TMR2	09Ah	OSCSTAT	11Ah	—	19Ah	TX1REG	21Ah	—	29Ah	CCP2CON	31Ah	—	39Ah	—
01Bh	PR2	09Bh	ADRESL	11Bh	—	19Bh	SP1BRGL	21Bh	—	29Bh	CCP2CAP	31Bh	—	39Bh	—
01Ch	T2CON	09Ch	ADRESH	11Ch	ZCD1CON	19Ch	SP1BRGH	21Ch	—	29Ch	—	31Ch	—	39Ch	—
01Dh	T2HLT	09Dh	ADCON0	11Dh	—	19Dh	RC1STA	21Dh	—	29Dh	—	31Dh	—	39Dh	—
01Eh	T2CLKCON	09Eh	ADCON1	11Eh	—	19Eh	TX1STA	21Eh	—	29Eh	CCPTMRS	31Eh	—	39Eh	—
01Fh	T2RST	09Fh	ADCON2	11Fh	—	19Fh	BAUD1CON	21Fh	—	29Fh	—	31Fh	—	39Fh	—
020h	General Purpose Register 96 Bytes	0A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h	General Purpose Register 80 Bytes	220h	General Purpose Register 80 Bytes	2A0h	General Purpose Register 80 Bytes	320h	General Purpose Register 80 Bytes	3A0h	General Purpose Register 80 Bytes
07Fh		0EFh		16Fh		1EFh		26Fh		2EFh		36Fh		3EFh	
		0F0h	Common RAM (Accesses 70h – 7Fh)	170h	Common RAM (Accesses 70h – 7Fh)	1F0h	Common RAM (Accesses 70h – 7Fh)	270h	Common RAM (Accesses 70h – 7Fh)	2F0h	Common RAM (Accesses 70h – 7Fh)	370h	Common RAM (Accesses 70h – 7Fh)	3F0h	Common RAM (Accesses 70h – 7Fh)
		0FFh		17Fh		1FFh		27Fh		2FFh		37Fh		3FFh	

Legend: = Unimplemented data memory locations, read as '0'.

TABLE 3-14: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Addr	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 11											
58Ch	PID1SELT	SET<7:0>								xxxx xxxx	xxxx xxxx
58Dh	PID1SETH	SET<15:8>								xxxx xxxx	xxxx xxxx
58Eh	PID1INL	IN<7:0>								0000 0000	0000 0000
58Fh	PID1INH	IN<15:8>								0000 0000	0000 0000
590h	PID1K1L	K1<7:0>								xxxx xxxx	xxxx xxxx
591h	PID1K1H	K1<15:8>								xxxx xxxx	xxxx xxxx
592h	PID1K2L	K2<7:0>								xxxx xxxx	xxxx xxxx
593h	PID1K2H	K2<15:8>								xxxx xxxx	xxxx xxxx
594h	PID1K3L	K3<7:0>								xxxx xxxx	xxxx xxxx
595h	PID1K3H	K3<15:8>								xxxx xxxx	xxxx xxxx
596h	PID1OUTLL	OUT<7:0>								0000 0000	0000 0000
597h	PID1OUTLH	OUT<15:8>								0000 0000	0000 0000
598h	PID1OUTHL	OUT<23:16>								0000 0000	0000 0000
599h	PID1OUTHH	OUT<31:24>								0000 0000	0000 0000
59Ah	PID1OUTU	—	—	—	—	OUT<35:32>				---- 0000	---- 0000
59Bh	PID1Z1L	Z1<7:0>								0000 0000	0000 0000
59Ch	PID1Z1H	Z1<15:8>								0000 0000	0000 0000
59Dh	PID1Z1U	—	—	—	—	—	—	—	Z116	---- ---0	---- ---0
59Eh	—	Unimplemented								—	—
59Fh	—	Unimplemented								—	—

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

Note 1: PIC16F1615/9 only.

2: Unimplemented, read as '1'.

3: PIC16(L)F1615 only.

4: PIC16(L)F1619 only.

TABLE 3-14: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Addr	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
Banks 28 (Continued)											
E24h	RXPPS(3)	—	—	—	RXPPS<4:0>					---1 0101	---1 0101
E24h	RXPPS(4)	—	—	—	RXPPS<4:0>					---0 1101	---0 1101
E25h	CKPPS(3)	—	—	—	CKPPS<4:0>					---1 0100	---1 0100
E25h	CKPPS(4)	—	—	—	CKPPS<4:0>					---0 1111	---0 1111
E26h	SMT1SIGPPS	—	—	—	SMT1SIGPPS<4:0>					---0 0100	---0 0100
E27h	SMT1WINPPS	—	—	—	SMT1WINPPS<4:0>					---0 0101	---0 0101
E28h	CLCIN0PPS	—	—	—	CLCIN0PPS<4:0>					---1 0011	---1 0011
E29h	CLCIN1PPS	—	—	—	CLCIN1PPS<4:0>					---1 0100	---1 0100
E2Ah	CLCIN2PPS	—	—	—	CLCIN2PPS<4:0>					---1 0001	---1 0001
E2Bh	CLCIN3PPS	—	—	—	CLCIN3PPS<4:0>					---0 0101	---0 0101
E2Ch	SMT2SIGPPS	—	—	—	SMT2SIGPPS<4:0>					---1 0001	---1 0001
E2Dh	SMT2WINPPS	—	—	—	SMT2WINPPS<4:0>					---0 0011	---0 0011
E2Eh	ATCC3PPS	—	—	—	ATCC3PPS<4:0>					---1 0101	---1 0101
E2Fh to E6Fh	—	Unimplemented								—	—

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

Note 1: PIC16F1615/9 only.
2: Unimplemented, read as '1'.
3: PIC16(L)F1615 only.
4: PIC16(L)F1619 only.

3.5 Stack

All devices have a 16-level x 15-bit wide hardware stack (refer to Figures 3-4 through 3-7). The stack space is not part of either program or data space. The PC is PUSHed onto the stack when CALL or CALLW instructions are executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer if the STVREN bit is programmed to '0' (Configuration Words). This means that after the stack has been PUSHed sixteen times, the seventeenth PUSH overwrites the value that was stored from the first PUSH. The eighteenth PUSH overwrites the second PUSH (and so on). The STKOVF and STKUNF flag bits will be set on an Overflow/Underflow, regardless of whether the Reset is enabled.

Note 1: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, CALLW, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

3.5.1 ACCESSING THE STACK

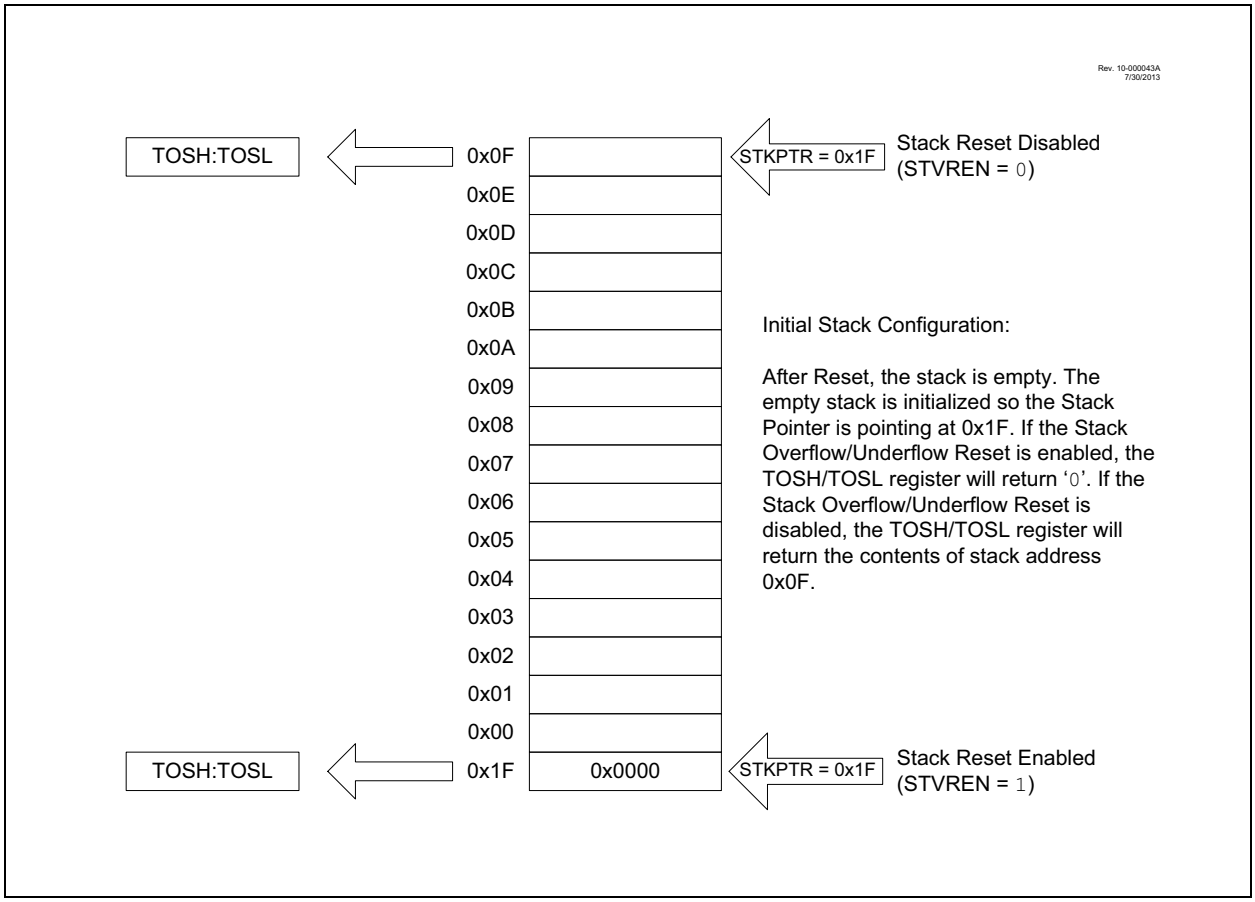
The stack is available through the TOSH, TOSL and STKPTR registers. STKPTR is the current value of the Stack Pointer. TOSH:TOSL register pair points to the TOP of the stack. Both registers are read/writable. TOS is split into TOSH and TOSL due to the 15-bit size of the PC. To access the stack, adjust the value of STKPTR, which will position TOSH:TOSL, then read/write to TOSH:TOSL. STKPTR is five bits to allow detection of overflow and underflow.

Note: Care should be taken when modifying the STKPTR while interrupts are enabled.

During normal program operation, CALL, CALLW and Interrupts will increment STKPTR while RETLW, RETURN, and RETFIE will decrement STKPTR. At any time STKPTR can be inspected to see how much stack is left. The STKPTR always points at the currently used place on the stack. Therefore, a CALL or CALLW will increment the STKPTR and then write the PC, and a return will unload the PC and then decrement the STKPTR.

Reference Figure 3-4 through Figure 3-7 for examples of accessing the stack.

FIGURE 3-4: ACCESSING THE STACK EXAMPLE 1



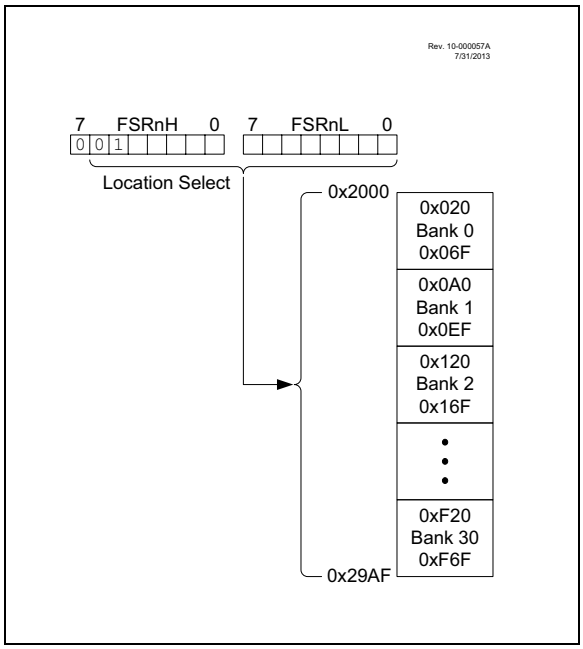
3.6.2 LINEAR DATA MEMORY

The linear data memory is the region from FSR address 0x2000 to FSR address 0x29AF. This region is a virtual region that points back to the 80-byte blocks of GPR memory in all the banks.

Unimplemented memory reads as 0x00. Use of the linear data memory region allows buffers to be larger than 80 bytes because incrementing the FSR beyond one bank will go directly to the GPR memory of the next bank.

The 16 bytes of common memory are not included in the linear data memory region.

FIGURE 3-10: LINEAR DATA MEMORY MAP



3.6.3 PROGRAM FLASH MEMORY

To make constant data access easier, the entire program Flash memory is mapped to the upper half of the FSR address space. When the MSb of FSRnH is set, the lower 15 bits are the address in program memory which will be accessed through INDF. Only the lower eight bits of each memory location is accessible via INDF. Writing to the program Flash memory cannot be accomplished via the FSR/INDF interface. All instructions that access program Flash memory via the FSR/INDF interface will require one additional instruction cycle to complete.

FIGURE 3-11: PROGRAM FLASH MEMORY MAP

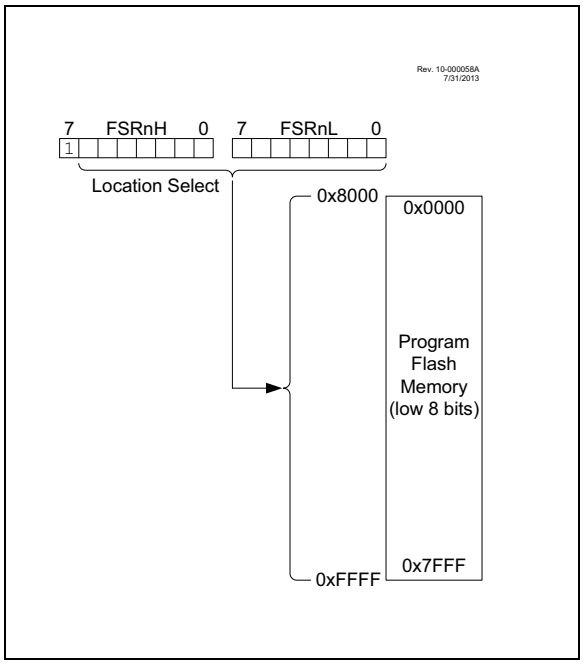
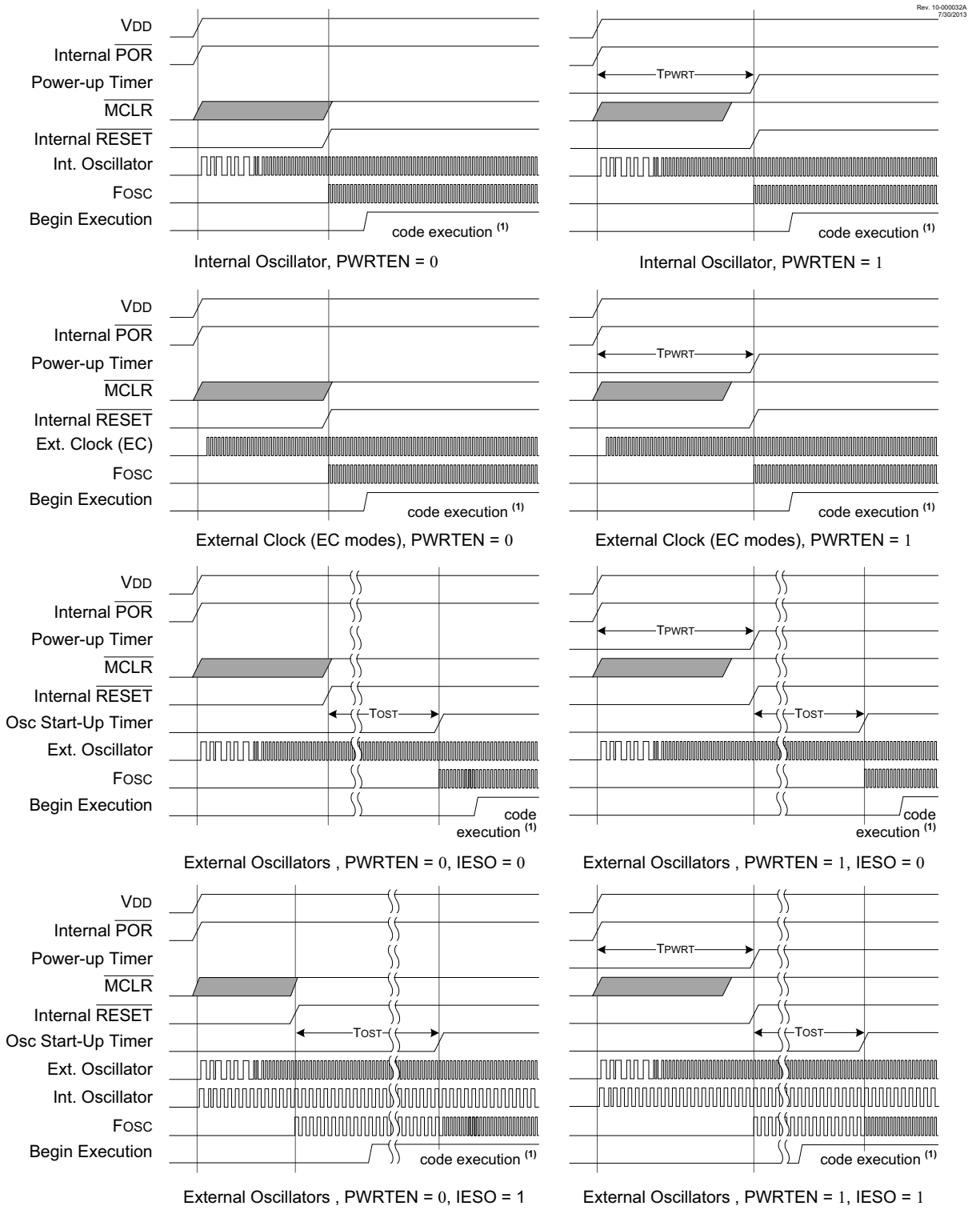


FIGURE 6-3: RESET START-UP SEQUENCE



Note 1: Code execution begins 10 FOSC cycles after the FOSC clock is released.

REGISTER 9-3: WDTPSL: WDT PRESCALE SELECT LOW BYTE REGISTER (READ ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
PSCNT<7:0>(1)							
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 7-0 **PSCNT<7:0>**: Prescale Select Low Byte bits(1)

Note 1: The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should be read during normal operation.

REGISTER 9-4: WDTPSH: WDT PRESCALE SELECT HIGH BYTE REGISTER (READ ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
PSCNT<15:8>(1)							
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 7-0 **PSCNT<15:8>**: Prescale Select High Byte bits(1)

Note 1: The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should be read during normal operation.

REGISTER 9-5: WDTTMR: WDT TIMER REGISTER (READ ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
WDTTMR<3:0>					STATE	PSCNT<17:16>(1)	
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 7-3 **WDTTMR<4:0>**: Watchdog Timer Value

bit 2 **STATE**: WDT Armed Status bit
1 = WDT is armed
0 = WDT is not armed

bit 1-0 **PSCNT<17:16>**: Prescale Select Upper Byte bits(1)

Note 1: The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should be read during normal operation.

11.7 Configuring the CRC

The following steps illustrate how to properly configure the CRC.

1. Determine if the automatic Program Memory scan will be used with the Scanner or manual calculation through the SFR interface and perform the actions specified in **Section11.4 "CRC Data Sources"**, depending on which decision was made.
2. If desired, seed a starting CRC value into the CRCACCH/L registers.
3. Program the CRCXORH/L registers with the desired generator polynomial.
4. Program the DLEN<3:0> bits of the CRCCON1 register with the length of the data word - 1 (refer to Example 11-1). This determines how many times the shifter will shift into the accumulator for each data word.
5. Program the PLEN<3:0> bits of the CRCCON1 register with the length of the polynomial - 2 (refer to Example 11-1).
6. Determine whether shifting in trailing zeros is desired and set the ACCM bit of CRCCON0 register appropriately.
7. Likewise, determine whether the MSb or LSb should be shifted first and write the SHIFTM bit of CRCCON0 register appropriately.
8. Write the CRCGO bit of the CRCCON0 register to begin the shifting process.

9a. If manual SFR entry is used, monitor the FULL bit of CRCCON0 register. When FULL = 0, another word of data can be written to the CRCDATH/L registers, keeping in mind that CRCDATH should be written first if the data has >8 bits, as the shifter will begin upon the CRCDATL register being written.

9b. If the scanner is used, the scanner will automatically stuff words into the CRCDATH/L registers as needed, as long as the SCANGO bit is set.

10a. If using the Flash memory scanner, monitor the SCANIF (or the SCANGO bit) for the scanner to finish pushing information into the CRCDATA registers. After the scanner is completed, monitor the CRCIF (or the BUSY bit) to determine that the CRC has been completed and the check value can be read from the CRCACC registers. If both the interrupt flags are set (or both BUSY and SCANGO bits are cleared), the completed CRC calculation can be read from the CRCACCH/L registers.

10b. If manual entry is used, monitor the CRCIF (or BUSY bit) to determine when the CRCACC registers will hold the check value.

11.8 Program Memory Scan Configuration

If desired, the Program Memory Scan module may be used in conjunction with the CRC module to perform a CRC calculation over a range of program memory addresses. In order to set up the Scanner to work with the CRC you need to perform the following steps:

1. Set the EN bit to enable the module. This can be performed at any point preceding the setting of the SCANGO bit, but if it gets disabled, all internal states of the Scanner are reset (registers are unaffected).
2. Choose which memory access mode is to be used (see **Section11.10 "Scanning Modes"**) and set the MODE bits of the SCANCON0 register appropriately.
3. Based on the memory access mode, set the INTM bits of the SCANCON0 register to the appropriate interrupt mode (see **Section11.10.5 "Interrupt Interaction"**)
4. Set the SCANLADRL/H and SCANHADRL/H registers with the beginning and ending locations in memory that are to be scanned.
5. Begin the scan by setting the SCANGO bit in the SCANCON0 register. The scanner will wait (CRCGO must be set) for the signal from the CRC that it is ready for the first Flash memory location, then begin loading data into the CRC. It will continue to do so until it either hits the configured end address or an address that is unimplemented on the device, at which point the SCANGO bit will clear, Scanner functions will cease, and the SCANIF interrupt will be triggered. Alternately, the SCANGO bit can be cleared in software if desired.

11.9 Scanner Interrupt

The scanner will trigger an interrupt when the SCANGO bit transitions from 1 to 0. The SCANIF interrupt flag of PIR4 is set when the last memory location is reached and the data is entered into the CRCDATA registers. The SCANIF bit can only be cleared in software. The SCAN interrupt enable is the SCANIE bit of the PIE4 register.

11.10 Scanning Modes

The memory scanner can scan in four modes: Burst, Peek, Concurrent, and Triggered. These modes are controlled by the MODE bits of the SCANCON0 register. The four modes are summarized in Table 11-1.

11.10.1 BURST MODE

When MODE = 01, the scanner is in Burst mode. In Burst mode, CPU operation is stalled beginning with the operation after the one that sets the SCANGO bit, and the scan begins, using the instruction clock to execute.

FIGURE 14-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTA EXAMPLE)

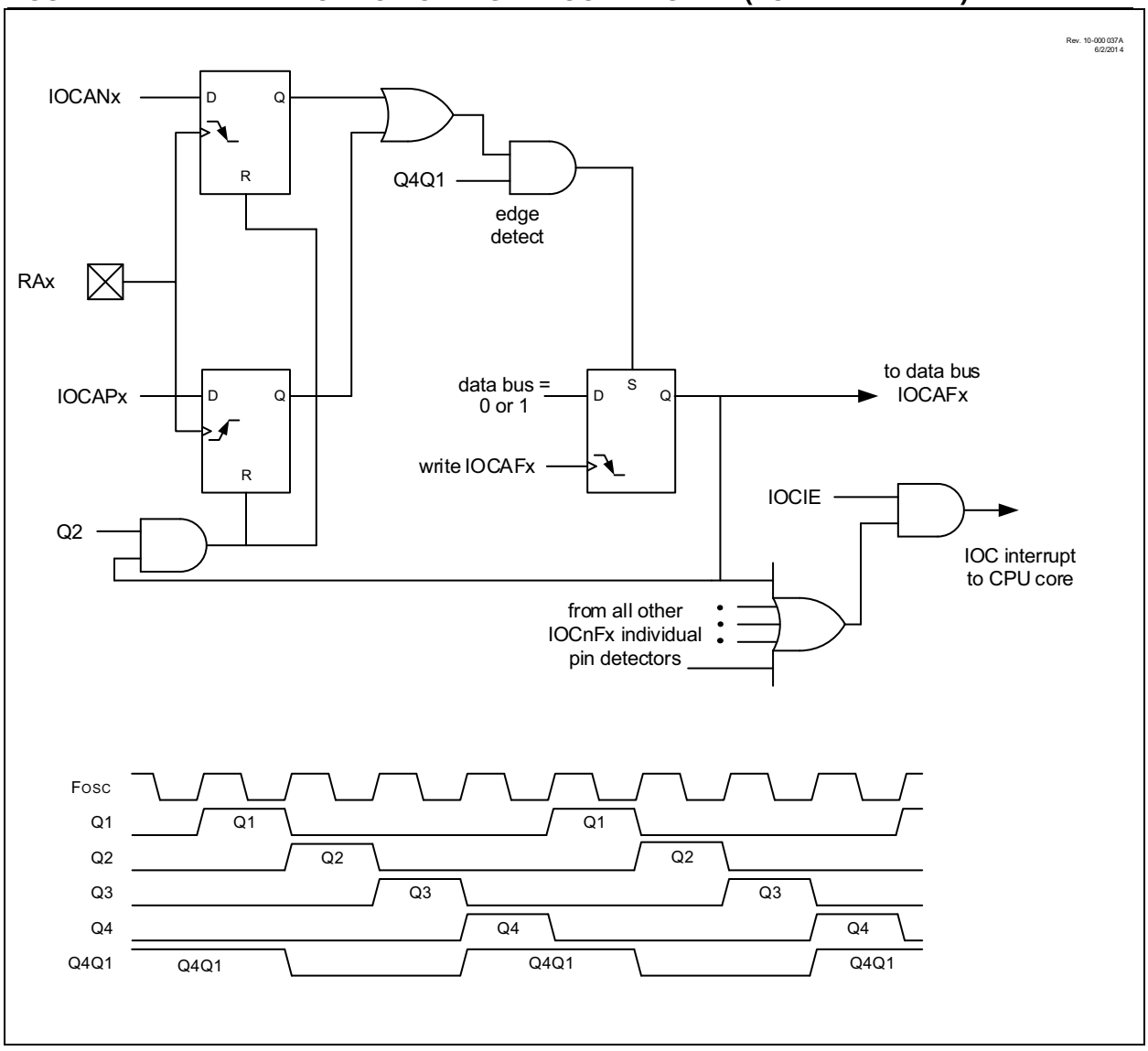


TABLE 22-5: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1

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	160
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	105
PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	106
PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	111
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Count								235*
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Count								235*
TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Count								235*
TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Count								235*
TMR5H	Holding Register for the Most Significant Byte of the 16-bit TMR5 Count								235*
TMR5L	Holding Register for the Least Significant Byte of the 16-bit TMR5 Count								235*
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	159
T1CON	TMR1CS<1:0>		T1CKPS<1:0>		—	$\overline{\text{T1SYNC}}$	—	TMR1ON	239
T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/$\overline{\text{DONE}}$	T1GVAL	T1GSS<1:0>		240
T3CON	TMR3CS<1:0>		T3CKPS<1:0>		—	T3SYNC	—	TMR3ON	239
T3GCON	TMR3GE	T3GPOL	T3GTM	T3GSPM	T3GGO/$\overline{\text{DONE}}$	T3GVAL	T3GSS<1:0>		240
T5CON	TMR5CS<1:0>		T5CKPS<1:0>		—	T5SYNC	—	TMR5ON	239
T5GCON	TMR5GE	T5GPOL	T5GTM	T5GSPM	T5GGO/$\overline{\text{DONE}}$	T5GVAL	T5GSS<1:0>		240

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by the Timer1 module.
* Page provides register information.
Note 1: Unimplemented, read as '1'.

REGISTER 23-4: TxRST: TIMERx EXTERNAL RESET SIGNAL SELECTION REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	RSEL<3:0>			
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	

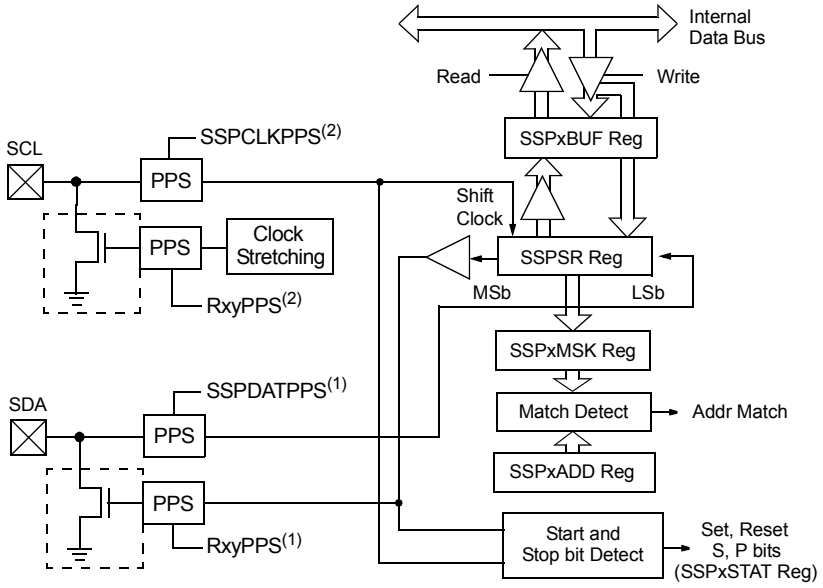
bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **RSEL<4:0>:** TimerX External Reset Signal Source Selection bits
See Table 23-4.

TABLE 23-4: EXTERNAL RESET SOURCES

RSEL<4:0>	Timer2	Timer4	Timer6
1111	Reserved	Reserved	Reserved
1110	PWM4_out	PWM4_out	PWM4_out
1101	PWM3_out	PWM3_out	PWM3_out
1100	LC4_out	LC4_out	LC4_out
1011	LC3_out	LC3_out	LC3_out
1010	LC2_out	LC2_out	LC2_out
1001	LC1_out	LC1_out	LC1_out
1000	ZCD1_out	ZCD1_out	ZCD1_out
0111	TMR6_postscaled	TMR6_postscaled	Reserved
0110	TMR4_postscaled	Reserved	TMR4_postscaled
0101	Reserved	TMR2_postscaled	TMR2_postscaled
0100	CCP2_out	CCP2_out	CCP2_out
0011	CCP1_out	CCP1_out	CCP1_out
0010	C2OUT_sync	C2OUT_sync	C2OUT_sync
0001	C1OUT_sync	C1OUT_sync	C1OUT_sync
0000	Pin selected by T2INPPS	Pin selected by T2INPPS	Pin selected by T2INPPS

FIGURE 24-3: MSSP BLOCK DIAGRAM (I^2C SLAVE MODE)



Note 1: SDA pin selections must be the same for input and output.

2: SCL pin selections must be the same for input and output.

26.4 CCP/PWM Clock Selection

The PIC16(L)F1615/9 allows each individual CCP and PWM module to select the timer source that controls the module. Each module has an independent selection.

As there are up to three 8-bit timers with auto-reload (Timer2/4/6), PWM mode on the CCP and PWM modules can use any of these timers.

The CCPTMRS register is used to select which timer is used.

26.4.1 USING THE TMR2/4/6 WITH THE CCP MODULE

This device has a new version of the TMR2 module that has many new modes, which allow for greater customization and control of the PWM signals than older parts. Refer to **Section23.5 "Operation Examples"** for examples of PWM signal generation using the different modes of Timer2. The CCP operation requires that the timer used as the PWM time base has the FOSC/4 clock source selected.

26.4.2 PWM PERIOD

The PWM period is specified by the PR2/4/6 register of Timer2/4/6. The PWM period can be calculated using the formula of Equation 26-1.

EQUATION 26-1: PWM PERIOD

$$PWM\ Period = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

Note 1: TOSC = 1/FOSC

When TMR2/4/6 is equal to its respective PR2/4/6 register, the following three events occur on the next increment cycle:

- TMR2/4/6 is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is latched from the CCPRxH:CCPRxL pair into the internal 10-bit latch.

Note:	The Timer postscaler (see Figure) is not used in the determination of the PWM frequency.

26.4.3 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to two registers: the CCPRxH:CCPRxL register pair. Where the particular bits go is determined by the FMT bit of the CCPxCON register. If FMT = 0, the two Most Significant bits of the duty cycle value should be written to bits <1:0> of CCPRxH register and the remaining eight bits to the CCPRxL register. If FMT = 1, the Least Significant two bits of the duty cycle should be written to bits <7:6> of the CCPRxL register and the Most Significant eight bits to the CCPRxH register. This is illustrated in Figure 26-4. These bits can be written at any time. The duty cycle value is not latched into the internal latch until after the period completes (i.e., a match between PR2/4/6 and TMR2/4/6 registers occurs).

Equation 26-2 is used to calculate the PWM pulse width. Equation 26-3 is used to calculate the PWM duty cycle ratio.

EQUATION 26-2: PULSE WIDTH

$$Pulse\ Width = CCPRxH{:}CCPRxL \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

EQUATION 26-3: DUTY CYCLE RATIO

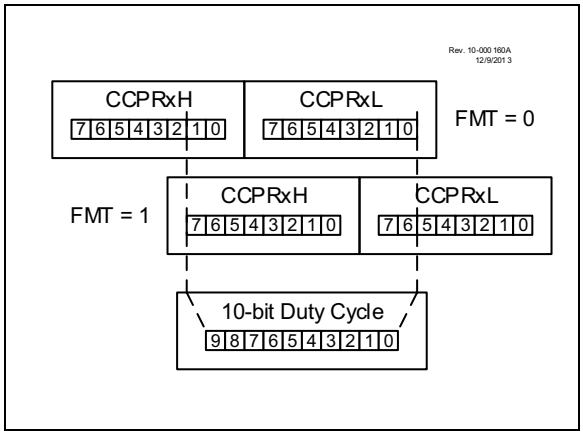
$$Duty\ Cycle\ Ratio = \frac{(CCPRxH{:}CCPRxL)}{4(PRx + 1)}$$

The PWM duty cycle registers are double buffered for glitchless PWM operation.

The 8-bit timer TMR2/4/6 register is concatenated with either the 2-bit internal system clock (FOSC), or two bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timer2/4/6 prescaler is set to 1:1.

When the 10-bit time base matches the internal buffer register, then the CCPx pin is cleared (see Figure).

FIGURE 26-4: CCPx DUTY-CYCLE ALIGNMENT



26.4.4 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

26.5 Register Definitions: CCP Control

REGISTER 26-1: CCPxCON: CCPx CONTROL REGISTER

R/W-0/0	U/U-0/0	R-x	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	OUT	FMT	MODE<3:0>			
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 Reset
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7 **EN:** CCPx Module Enable bit
1 = CCPx is enabled
0 = CCPx is disabled

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **OUT:** CCPx Output Data bit (read-only)

bit 4 **FMT:** CCPW (Pulse-Width) Alignment bit
If MODE = PWM Mode
1 = Left-aligned format, CCPRxH <7> is the MSb of the PWM duty cycle
0 = Right-aligned format, CCPRxL<0> is the LSb of the PWM duty cycle

bit 3-0 **MODE<3:0>:** CCPx Mode Selection bit
11xx = PWM mode

1011 = Compare mode: Pulse output, clear TMR1
1010 = Compare mode: Pulse output (0 - 1 - 0)
1001 = Compare mode: clear output on compare match
1000 = Compare mode: set output on compare match

0111 = Capture mode: every 16th rising edge
0110 = Capture mode: every 4th rising edge
0101 = Capture mode: every rising edge
0100 = Capture mode: every falling edge

0011 = Capture mode: every rising or falling edge
0010 = Compare mode: toggle output on match
0001 = Compare mode: Toggle output and clear TMR1 on match
0000 = Capture/Compare/PWM off (resets CCPx module) (reserved for backwards compatibility)

FIGURE 28-6: DEAD-BAND OPERATION CWGXDBR = 0X01, CWGXDBF = 0X02

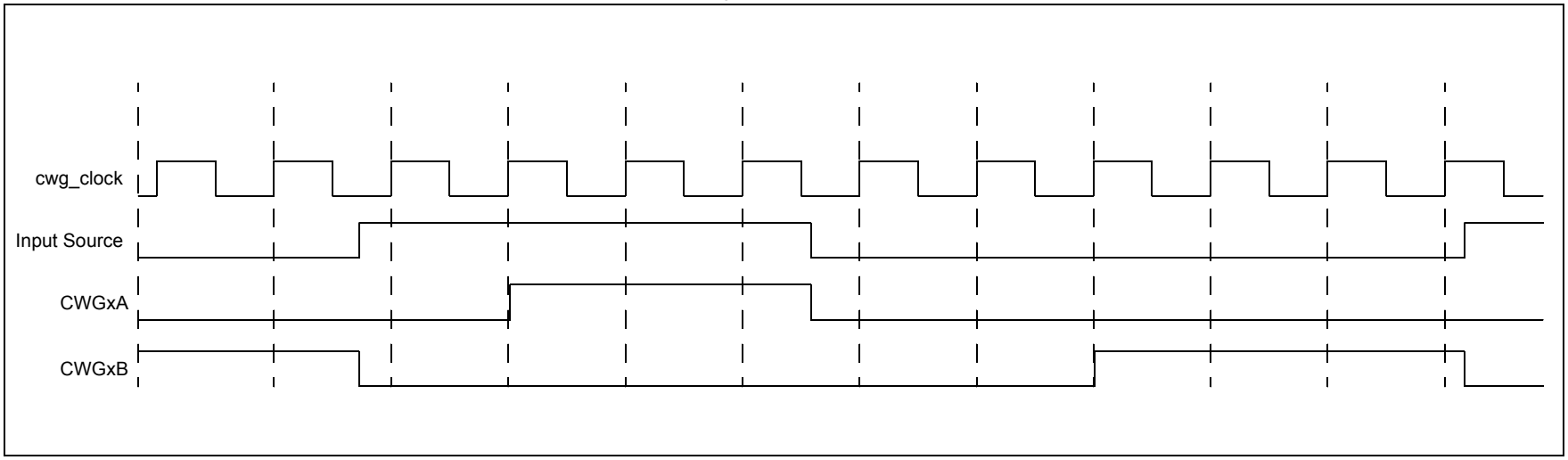
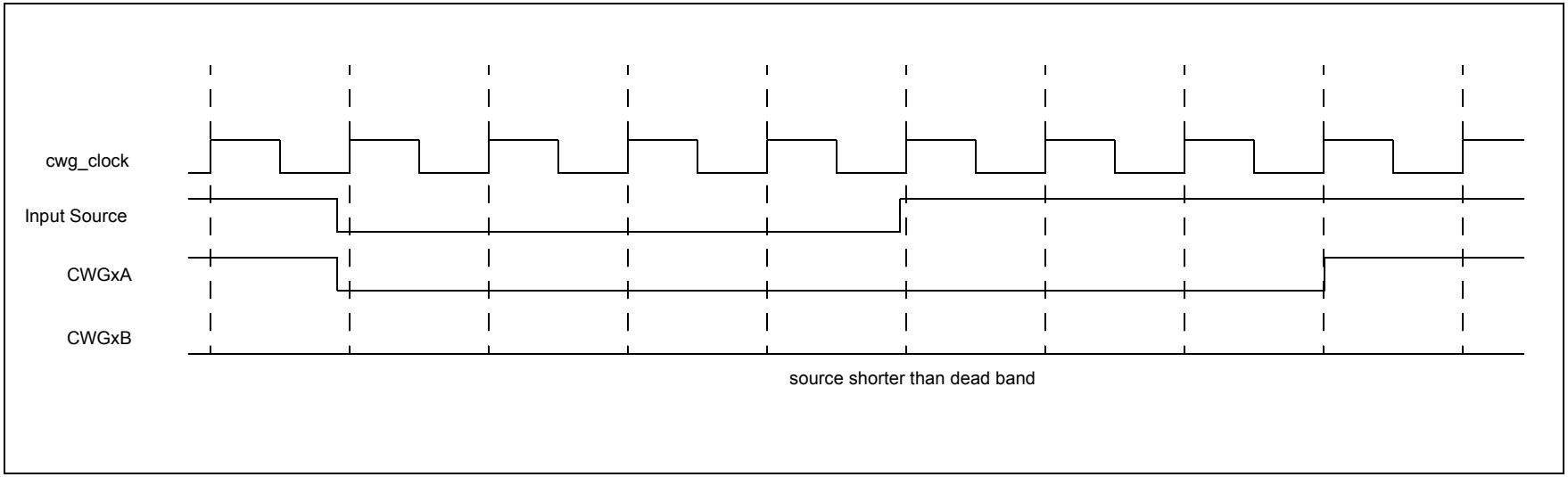


FIGURE 28-7: DEAD-BAND OPERATION, CWGXDBR = 0X03, CWGXDBF = 0X04, SOURCE SHORTER THAN DEAD BAND



REGISTER 29-7: CLCxGLS0: GATE 1 LOGIC SELECT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
LCxG1D4T	LCxG1D4N	LCxG1D3T	LCxG1D3N	LCxG1D2T	LCxG1D2N	LCxG1D1T	LCxG1D1N
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 **LCxG1D4T:** Gate 1 Data 4 True (non-inverted) bit
1 = lcxd4T is gated into lcxg1
0 = lcxd4T is not gated into lcxg1

bit 6 **LCxG1D4N:** Gate 1 Data 4 Negated (inverted) bit
1 = lcxd4N is gated into lcxg1
0 = lcxd4N is not gated into lcxg1

bit 5 **LCxG1D3T:** Gate 1 Data 3 True (non-inverted) bit
1 = lcxd3T is gated into lcxg1
0 = lcxd3T is not gated into lcxg1

bit 4 **LCxG1D3N:** Gate 1 Data 3 Negated (inverted) bit
1 = lcxd3N is gated into lcxg1
0 = lcxd3N is not gated into lcxg1

bit 3 **LCxG1D2T:** Gate 1 Data 2 True (non-inverted) bit
1 = lcxd2T is gated into lcxg1
0 = lcxd2T is not gated into lcxg1

bit 2 **LCxG1D2N:** Gate 1 Data 2 Negated (inverted) bit
1 = lcxd2N is gated into lcxg1
0 = lcxd2N is not gated into lcxg1

bit 1 **LCxG1D1T:** Gate 1 Data 1 True (non-inverted) bit
1 = lcxd1T is gated into lcxg1
0 = lcxd1T is not gated into lcxg1

bit 0 **LCxG1D1N:** Gate 1 Data 1 Negated (inverted) bit
1 = lcxd1N is gated into lcxg1
0 = lcxd1N is not gated into lcxg1

REGISTER 30-4: SMTxCLK: SMT CLOCK SELECTION REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	CSEL<2:0>		
bit 7							bit 0

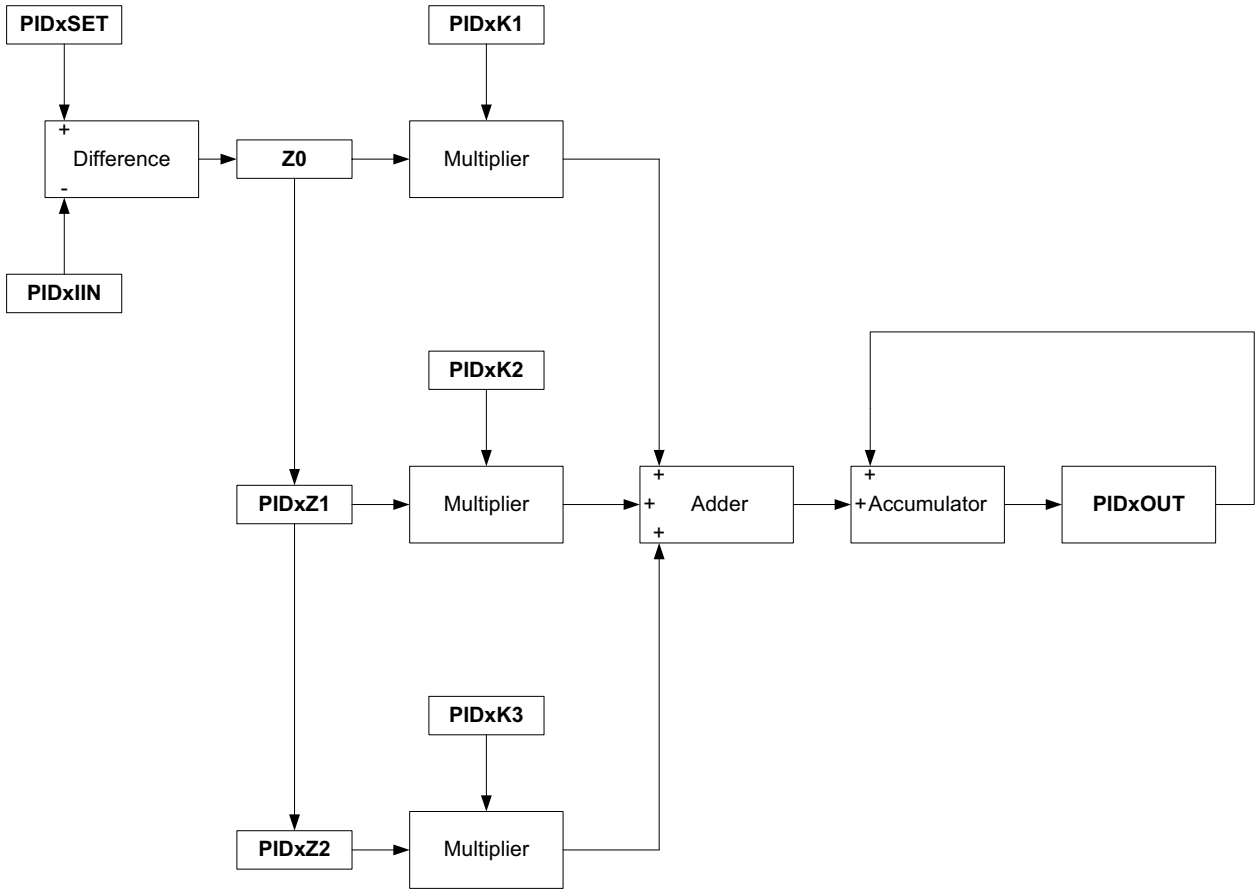
Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	q = Value depends on condition

bit 7-3 **Unimplemented**: Read as ‘0’

bit 2-0 **CSEL<2:0>:** SMT Clock Selection bits

111 = Reserved
110 = AT1_perclk
101 = MFINTOSC
100 = MFINTOSC/16
011 = LFINTOSC
010 = HFINTOSC 16 MHz
001 = FOSC/4
000 = FOSC

FIGURE 32-1: PID MODULE BASIC DATA FLOW BLOCK DIAGRAM, PID MODES



Note 1: After the results of PIDxZ2 are multiplied by PIDxK3 and the result is added to the accumulator, the current value from PIDxZ1 is loaded into PIDxZ2. The same is true for PIDxZ1 and the current SET-IN value.

REGISTER 32-17: PIDxZ1U: PID Z1 UPPER REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	Z116
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 7-1 **Unimplemented:** Read as ‘0’

bit 0 **Z116:** Bit 16 of Z1. In PID mode, Z1 is the value of the error (IN minus SET) from the previous iteration of the PID control loop.

REGISTER 32-18: PIDxZ1H: PID Z1 HIGH REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
Z1<15:8>							
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 7-0 **Z1<15:8>:** Bits <15:8> of Z1. In PID mode, Z1 is the value of the error (IN minus SET) from the previous iteration of the PID control loop.

REGISTER 32-19: PIDxZ1L: PID Z1 LOW REGISTER

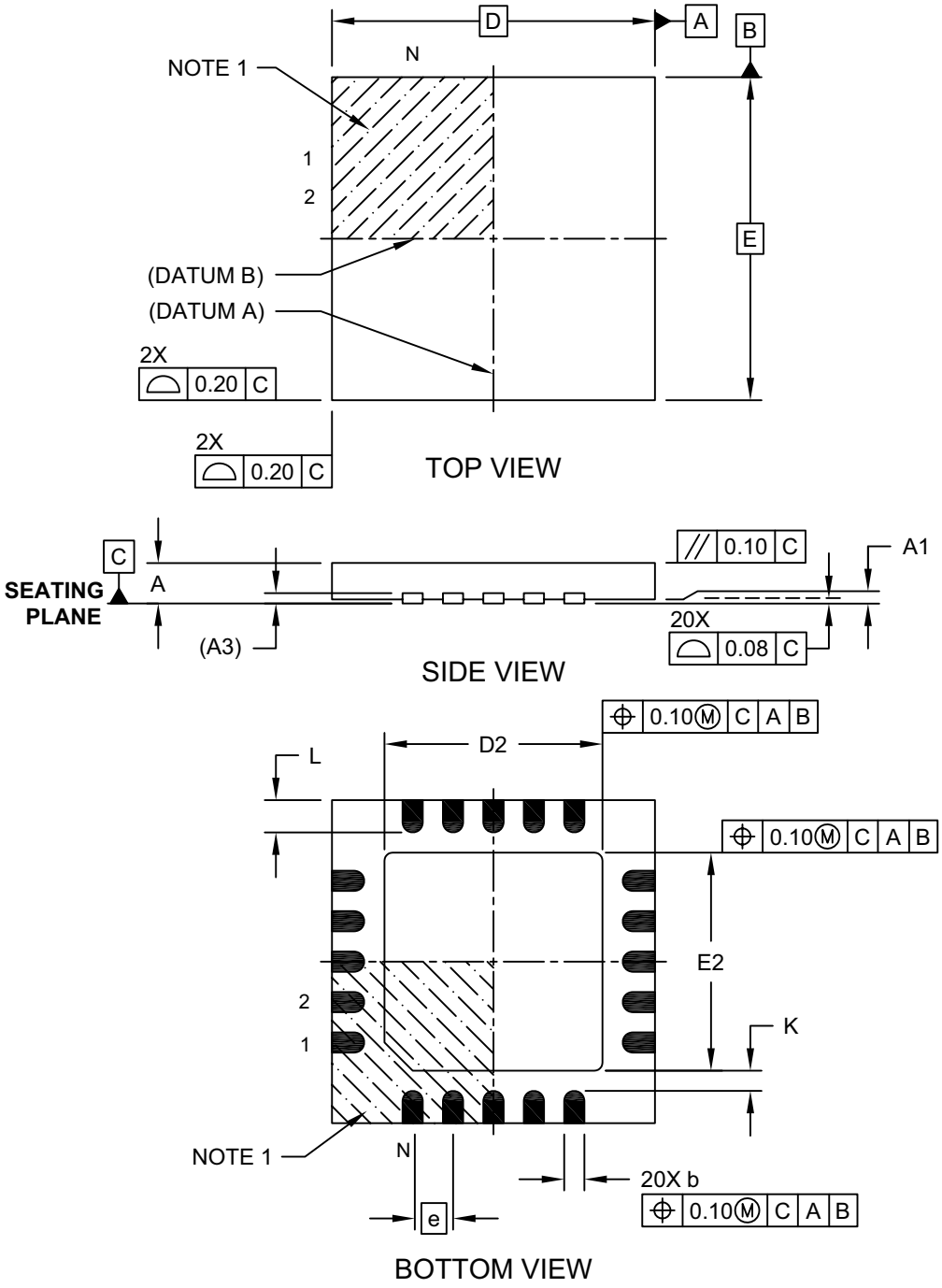
R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
Z1<7:0>							
bit 7							bit 0

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	q = Value depends on condition

bit 7-0 **Z1<7:0>:** Bits <7:0> of Z1. In PID mode, Z1 is the value of the error (IN minus SET) from the previous iteration of the PID control loop.

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