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	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, HLVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f26k42-e-ss

TABLE 4-4: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES BANK 63

3FFFh	TOSU	3FDFh	INDF2	3FBFh	LATF(3)	3F9Fh	T4PR	3F7Fh	CCP1CAP	3F5Fh	CCPTMRS1	3F3Fh	NCO1CLK	3F1Fh	SMT1CON1
3FFEh	TOSH	3FDEh	POSTINC2	3FBEh	LATE(2)	3F9Eh	T4TMR	3F7Eh	CCP1CON	3F5Eh	CCPTMRS0	3F3Eh	NCO1CON	3F1Eh	SMT1CON0
3FFDh	TOSL	3FDDh	POSTDEC2	3FBDh	LATD(2)	3F9Dh	T5CLK	3F7Dh	CCPR1H	3F5Dh	—	3F3Dh	NCO1INCU	3F1Dh	SMT1PRU
3FFCh	STKPTR	3FDCh	PRECIN2	3FBCh	LATC	3F9Ch	T5GATE	3F7Ch	CCPR1L	3F5Ch	—	3F3Ch	NCO1INCH	3F1Ch	SMT1PRH
3FFBh	PCLATU	3FDBh	PLUSW2	3FBBh	LATB	3F9Bh	T5GCON	3F7Bh	CCP2CAP	3F5Bh	—	3F3Bh	NCO1INCL	3F1Bh	SMT1PRL
3FFAh	PCLATH	3FDAh	FSR2H	3FBAh	LATA	3F9Ah	T5CON	3F7Ah	CCP2CON	3F5Ah	CWG1STR	3F3Ah	NCO1ACCU	3F1Ah	SMT1CPWU
3FF9h	PCL	3FD9h	FSR2L	3FB9h	T0CON1	3F99h	TMR5H	3F79h	CCPR2H	3F59h	CWG1AS1	3F39h	NCO1ACCH	3F19h	SMT1CPWH
3FF8h	TBLPRTU	3FD8h	STATUS	3FB8h	T0CON0	3F98h	TMR5L	3F78h	CCPR2L	3F58h	CWG1AS0	3F38h	NCO1ACCL	3F18h	SMT1CPWL
3FF7h	TBLPTRH	3FD7h	IVTBASEU	3FB7h	TMR0H	3F97h	T6RST	3F77h	CCP3CAP	3F57h	CWG1CON1	3F37h	—	3F17h	SMT1CPRU
3FF6h	TBLPTRL	3FD6h	IVTBASEH	3FB6h	TMR0L	3F96h	T6CLK	3F76h	CCP3CON	3F56h	CWG1CON0	3F36h	—	3F16h	SMT1CPRH
3FF5h	TABLAT	3FD5h	IVTBASEL	3FB5h	T1CLK	3F95h	T6HLT	3F75h	CCPR3H	3F55h	CWG1DBF	3F35h	—	3F15h	SMT1CPRL
3FF4h	PRODH	3FD4h	IVTLOCK	3FB4h	T1GATE	3F94h	T6CON	3F74h	CCPR3L	3F54h	CWG1DBR	3F34h	—	3F14h	SMT1TMRU
3FF3h	PRODL	3FD3h	INTCON1	3FB3h	T1GCON	3F93h	T6PR	3F73h	CCP4CAP	3F53h	CWG1ISM	3F33h	—	3F13h	SMT1TMRH
3FF2h	—	3FD2h	INTCON0	3FB2h	T1CON	3F92h	T6TMR	3F72h	CCP4CON	3F52h	CWG1CLK	3F32h	—	3F12h	SMT1TMRL
3FF1h	PCON1	3FD1h	—	3FB1h	TMR1H	3F91h	—	3F71h	CCPR4H	3F51h	CWG2STR	3F31h	—	3F11h	—
3FF0h	PCON0	3FD0h	—	3FB0h	TMR1L	3F90h	—	3F70h	CCPR4L	3F50h	CWG2AS1	3F30h	—	3F10h	—
3FEFh	INDF0	3FCFh	PORTF(3)	3FAFh	T2RST	3F8Fh	—	3F6Fh	—	3F4Fh	CWG2AS0	3F2Fh	—	3F0Fh	—
3FEEh	POSTINC0	3FCEh	PORTE	3FAEh	T2CLK	3F8Eh	—	3F6Eh	PWM5CON	3F4Eh	CWG2CON1	3F2Eh	—	3F0Eh	—
3FEDh	POSTDEC0	3FCDh	PORTD(2)	3FADh	T2HLT	3F8Dh	—	3F6Dh	PWM5DCH	3F4Dh	CWG2CON0	3F2Dh	—	3F0Dh	—
3FECh	PRECIN0	3FCCh	PORTC	3FACh	T2CON	3F8Ch	—	3F6Ch	PWM5DCL	3F4Ch	CWG2DBF	3F2Ch	—	3F0Ch	—
3FEBh	PLUSW0	3FCBh	PORTB	3FABh	T2PR	3F8Bh	—	3F6Bh	—	3F4Bh	CWG2DBR	3F2Bh	—	3F0Bh	—
3FEAh	FSR0H	3FCAh	PORTA	3FAAh	T2TMR	3F8Ah	—	3F6Ah	PWM6CON	3F4Ah	CWG2ISM	3F2Ah	—	3F0Ah	—
3FE9h	FSR0L	3FC9h	—	3FA9h	T3CLK	3F89h	—	3F69h	PWM6DCH	3F49h	CWG2CLK	3F29h	—	3F09h	—
3FE8h	WREG	3FC8h	—	3FA8h	T3GATE	3F88h	—	3F68h	PWM6DCL	3F48h	CWG3STR	3F28h	—	3F08h	—
3FE7h	INDF1	3FC7h	TRISF(3)	3FA7h	T3GCON	3F87h	—	3F67h	—	3F47h	CWG3AS1	3F27h	—	3F07h	—
3FE6h	POSTINC1	3FC6h	TRISE(2)	3FA6h	T3CON	3F86h	—	3F66h	PWM7CON	3F46h	CWG3AS0	3F26h	—	3F06h	—
3FE5h	POSTDEC1	3FC5h	TRISD(2)	3FA5h	TMR3H	3F85h	—	3F65h	PWM7DCH	3F45h	CWG3CON1	3F25h	—	3F05h	—
3FE4h	PRECIN1	3FC4h	TRISC	3FA4h	TMR3L	3F84h	—	3F64h	PWM7DCL	3F44h	CWG3CON0	3F24h	—	3F04h	—
3FE3h	PLUSW1	3FC3h	TRISB	3FA3h	T4RST	3F83h	—	3F63h	—	3F43h	CWG3DBF	3F23h	SMT1WIN	3F03h	—
3FE2h	FSR1H	3FC2h	TRISA	3FA2h	T4CLK	3F82h	—	3F62h	PWM8CON	3F42h	CWG3DBR	3F22h	SMT1SIG	3F02h	—
3FE1h	FSR1L	3FC1h	—	3FA1h	T4HLT	3F81h	—	3F61h	PWM8DCH	3F41h	CWG3ISM	3F21h	SMT1CLK	3F01h	—
3FE0h	BSR	3FC0h	—	3FA0h	T4CON	3F80h	—	3F60h	PWM8DCL	3F40h	CWG3CLK	3F20h	SMT1STAT	3F00h	—

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

Note 1: Unimplemented in LF devices.

2: Unimplemented in PIC18(L)F26/27K42.

3: Unimplemented in PIC18(L)F26/27/45/46/47K42.

6.1 Power-on Reset (POR)

The POR circuit holds the device in Reset until VDD has reached an acceptable level for minimum operation. Slow rising VDD, fast operating speeds or analog performance may require greater than minimum VDD. The PWRT, BOR or MCLR features can be used to extend the start-up period until all device operation conditions have been met.

6.2 Brown-out Reset (BOR)

The BOR circuit holds the device in Reset when VDD reaches a selectable minimum level. Between the POR and BOR, complete voltage range coverage for execution protection can be implemented.

The Brown-out Reset module has four operating modes controlled by the BOREN<1:0> bits in Configuration Words. The four operating modes are:

- BOR is always on
- BOR is off when in Sleep
- BOR is controlled by software
- BOR is always off

Refer to Table 6-1 for more information.

The Brown-out Reset voltage level is selectable by configuring the BORV<1:0> bits in Configuration Words.

A VDD noise rejection filter prevents the BOR from triggering on small events. If VDD falls below VBOR for a duration greater than parameter TBORDC, the device will reset. See Table 44-13 for more information.

6.2.1 BOR IS ALWAYS ON

When the BOREN bits of Configuration Words are programmed to '11', the BOR is always on. The device start-up will be delayed until the BOR is ready and VDD is higher than the BOR threshold.

BOR protection is active during Sleep. The BOR does not delay wake-up from Sleep.

6.2.2 BOR IS OFF IN SLEEP

When the BOREN bits of Configuration Words are programmed to '10', the BOR is on, except in Sleep. The device start-up will be delayed until the BOR is ready and VDD is higher than the BOR threshold.

BOR protection is not active during Sleep. The device wake-up will be delayed until the BOR is ready.

6.2.3 BOR CONTROLLED BY SOFTWARE

When the BOREN bits of Configuration Words are programmed to '01', the BOR is controlled by the SBOREN bit of the BORCON register. The device start-up is not delayed by the BOR ready condition or the VDD level.

BOR protection begins as soon as the BOR circuit is ready. The status of the BOR circuit is reflected in the BORRDY bit of the BORCON register.

BOR protection is unchanged by Sleep.

6.2.4 BOR AND BULK ERASE

BOR is forced ON during PFM Bulk Erase operations to make sure that a safe erase voltage is maintained for a successful erase cycle.

During Bulk Erase, the BOR is enabled at 2.45V for F and LF devices, even if it is configured to some other value. If VDD falls, the erase cycle will be aborted, but the device will not be reset.

9.2 Interrupt Vector Table (IVT)

The interrupt controller supports an Interrupt Vector Table (IVT) that contains the vector address location for each interrupt request source.

The Interrupt Vector Table (IVT) resides in program memory, starting at address location determined by the IVTBASE registers; refer to Register 9-36, Register 9-37 and Register 9-38 for details. The IVT contains 68 vectors, one for each source of interrupt. Each interrupt vector location contains the starting address of the associated Interrupt Service Routine (ISR).

The MVECEN bit in Configuration Word 2L controls the availability of the vector table.

9.2.1 INTERRUPT VECTOR TABLE BASE ADDRESS (IVTBASE)

The start address of the vector table is user programmable through the IVTBASE registers. The user must ensure the start address is such that it can encompass the entire vector table inside the program memory.

Each vector address is a 16-bit word (or two address locations on PIC18 devices). So for n interrupt sources, there are 2n address locations necessary to hold the table starting from IVTBASE as the first location. So the staring address of IVTBASE should be chosen such that the address range form IVTBASE to (IVTBASE +2n-1) can be encompassed inside the program flash memory.

For example, the K42 devices have the highest vector number: 81. So IVTBASE should be chosen such that (IVTBASE + 0xA1) is less than the last memory location in program flash memory.

A programmable vector table base address is useful in situations to switch between different sets of vector tables, depending on the application. It can also be used when the application program needs to update the existing vector table (vector address values).

Note:	It is required that the user assign an even address to the IVTBASE register for correct operation.

9.2.2 INTERRUPT VECTOR TABLE CONTENTS

MVECEN = 0

When MVECEN = 0, the address location pointed by the IVTBASE registers has a GOTO instruction for a high priority interrupt. Similarly, the corresponding low priority vector location also has a GOTO instruction, which is executed in case of a low priority interrupt.

MVECEN = 1

When MVECEN = 1, the value in the vector table of each interrupt, points to the address location of the first instruction of the interrupt service routine.

ISR Location = Interrupt Vector Table entry << 2.

9.2.3 INTERRUPT VECTOR TABLE (IVT) ADDRESS CALCULATION

MVECEN = 0

When the MVECEN bit in Configuration Word 2L (Register 5-3) is cleared, the address pointed by IVTBASE registers is used as the high priority interrupt vector address. The low priority interrupt vector address is offset eight instruction words from the address in IVTBASE registers.

For PIC18 devices the IVTBASE registers default to 00 0008h, the high priority interrupt vector address will be 00 0008h and the low priority interrupt vector address will be 00 0018h.

MVECEN = 1

Each interrupt has a unique vector number associated with it as defined in Table 9-2. This vector number is used for calculating the location of the interrupt vector for a particular interrupt source.

Interrupt Vector Address = IVTBASE + (2*Vector Number).

This calculated Interrupt Vector Address value is stored in the IVTAD<20:0> registers when an interrupt is received (Registers 9-39 through 9-41).

User-assigned software priority assigned using the IPRx registers does not affect address calculation and is only used to resolve concurrent interrupts.

If for any reason the address of the ISR could not be fetched from the vector table, it will cause the system to reset and clear the memory execution violation flag (MEMV bit) in PCON1 register (Register 6-3). This occurs due to any one of the following:

- The entry for the interrupt in the vector table lies outside the executable PFM area (SAF area is non-executable when SAFEN = 1).
- ISR pointed by the vector table lies outside the executable PFM area (SAF area is non-executable when SAFEN = 1).

REGISTER 9-39: IVTADU: INTERRUPT VECTOR TABLE ADDRESS UPPER REGISTER

U-0	U-0	U-0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
—	—	—	AD<20:16>				
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 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **AD<20:16>**: Interrupt Vector Table Address bits

REGISTER 9-40: IVTADH: INTERRUPT VECTOR TABLE ADDRESS HIGH REGISTER

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
AD<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	

bit 7-0 **AD<15:8>**: Interrupt Vector Table Address bits

REGISTER 9-41: IVTADL: INTERRUPT VECTOR TABLE ADDRESS LOW REGISTER

R-0/0	R-0/0	R-0/0	R-0/0	R-1/1	R-0/0	R-0/0	R-0/0
AD<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	

bit 7-0 **AD<7:0>**: Interrupt Vector Table Address bits

REGISTER 13-5: NVMDAT: DATA EEPROM MEMORY DATA

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
DAT<7:0>							
bit 7							bit 0

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

bit 7-0 **DAT<7:0>:** The value of the data memory word returned from NVMADR after a Read command, or the data written by a Write command.

TABLE 13-4: SUMMARY OF REGISTERS ASSOCIATED WITH NONVOLATILE MEMORY CONTROL

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
NVMCON1	REG<1:0>		—	FREE	WRERR	WREN	WR	RD	210
NVMCON2	Unlock Pattern								211
NVMADRL	NVMADR<7:0>								211
NVMADRH (1)	—	—	—	—	—	—	NVMADR<9:8>		211
NVMDAT	NVMDAT<7:0>								212

Legend: — = unimplemented, read as '0'. Shaded bits are not used during EEPROM access.

*Page provides register information.

Note 1: The NVMADRH register is not implemented on PIC18(L)F45/55K42.

14.2 CRC Functional Overview

The CRC module can be used to detect bit errors in the program memory using the built-in memory scanner or through user input RAM memory. The CRC module can accept up to a 16-bit polynomial with up to a 16-bit seed value. A CRC calculated check value (or checksum) will then be generated into the CRCACC<15:0> registers for user storage. The CRC module uses an XOR shift register implementation to perform the polynomial division required for the CRC calculation.

EXAMPLE 14-1: CRC EXAMPLE

Rev. 10-000206A
1/8/2014

CRC-16-ANSI

$x^{16} + x^{15} + x^2 + 1$ (17 bits)

Standard 16-bit representation = 0x8005

```
CRCXORH = 0b10000000
CRCXORL = 0b0000010-
```
(1)

Data Sequence:
0x55, 0x66, 0x77, 0x88

```
DLEN = 0b0111
PLEN = 0b1111
```

Data entered into the CRC:
SHIFTM = 0:

```
01010101 01100110 01110111 10001000
```

SHIFTM = 1:

```
10101010 01100110 11101110 00010001
```

Check Value (ACCM = 1):

SHIFTM = 0: 0x32D6

```
CRCACCH = 0b00110010
CRCACCL = 0b11010110
```

SHIFTM = 1: 0x6BA2

```
CRCACCH = 0b01101011
CRCACCL = 0b10100010
```

Note 1: Bit 0 is unimplemented. The LSb of any CRC polynomial is always '1' and will always be treated as a '1' by the CRC for calculating the CRC check value. This bit will be read in software as a '0'.

REGISTER 15-15: DMAxDSAH: DMAx DESTINATION START ADDRESS 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
DSA<15:8>							
bit 7							bit 0

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

bit 7-0 **DSA<15:8>:** Destination Start Address bits

REGISTER 15-16: DMAxDPTRL: DMAx DESTINATION POINTER LOW REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DPTR<7:0>							
bit 7							bit 0

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

bit 7-0 **DPTR<7:0>:** Current Destination Address Pointer

REGISTER 15-17: DMAxDPTRH: DMAx DESTINATION POINTER HIGH REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DPTR<15:8>							
bit 7							bit 0

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

bit 7-0 **DPTR<15:8>:** Current Destination Address Pointer

16.4 Register Definitions: Port Control

REGISTER 16-1: PORTx: PORTx REGISTER(1)

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
Rx7	Rx6	Rx5	Rx4	Rx3	Rx2	Rx1	Rx0
bit 7							bit 0

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

bit 7-0 **Rx<7:0>:** Rx7:Rx0 Port I/O Value bits
1 = Port pin is ≥ V_{IH}
0 = Port pin is ≤ V_{IL}

Note 1: Writes to PORTx are actually written to the corresponding LATx register. Reads from PORTx register return actual I/O pin values.

TABLE 16-2: PORT REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
PORTB	RB7(1)	RB6(1)	RB5	RB4	RB3	RB2	RB1	RB0
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
PORTD(3)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
PORTE	—	—	—	—	RE3(2)	RE2(3)	RE1(3)	RE0(3)
PORTF(4)	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0

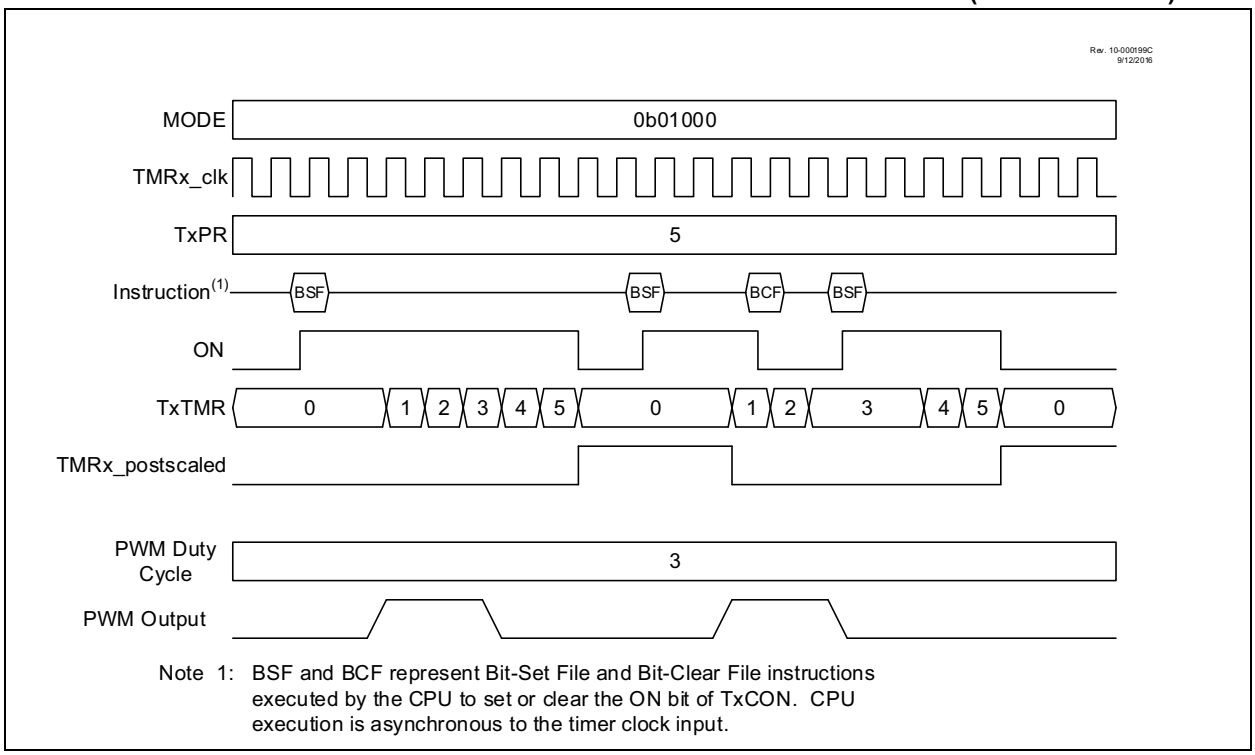
Note 1: Bits RB6 and RB7 read '1' while in Debug mode.
2: Bit PORTE3 is read-only, and will read '1' when MCLRE = 1 (Master Clear enabled).
3: Unimplemented in PIC18(L)F26/27K42.
4: Unimplemented in PIC18(L)F26/27/45/46/47K42.

22.5.5 SOFTWARE START ONE-SHOT MODE

In One-Shot mode, the timer resets and the ON bit is cleared when the timer value matches the T2PR period value. The ON bit must be set by software to start another timer cycle. Setting MODE<4:0> = 01000 selects One-Shot mode which is illustrated in Figure 22-8. In the example, ON is controlled by BSF and BCF instructions. In the first case, a BSF instruction sets ON and the counter runs to completion and clears ON. In the second case, a BSF instruction starts the cycle, BCF/BSF instructions turn the counter off and on during the cycle, and then it runs to completion.

When One-Shot mode is used in conjunction with the CCP PWM operation, the PWM pulse drive starts concurrent with setting the ON bit. Clearing the ON bit while the PWM drive is active will extend the PWM drive. The PWM drive will terminate when the timer value matches the CCPRx pulse width value. The PWM drive will remain off until software sets the ON bit to start another cycle. If software clears the ON bit after the CCPRx match but before the T2PR match then the PWM drive will be extended by the length of time the ON bit remains cleared. Another timing cycle can only be initiated by setting the ON bit after it has been cleared by a T2PR period count match.

FIGURE 22-8: SOFTWARE START ONE-SHOT MODE TIMING DIAGRAM (MODE = 01000)

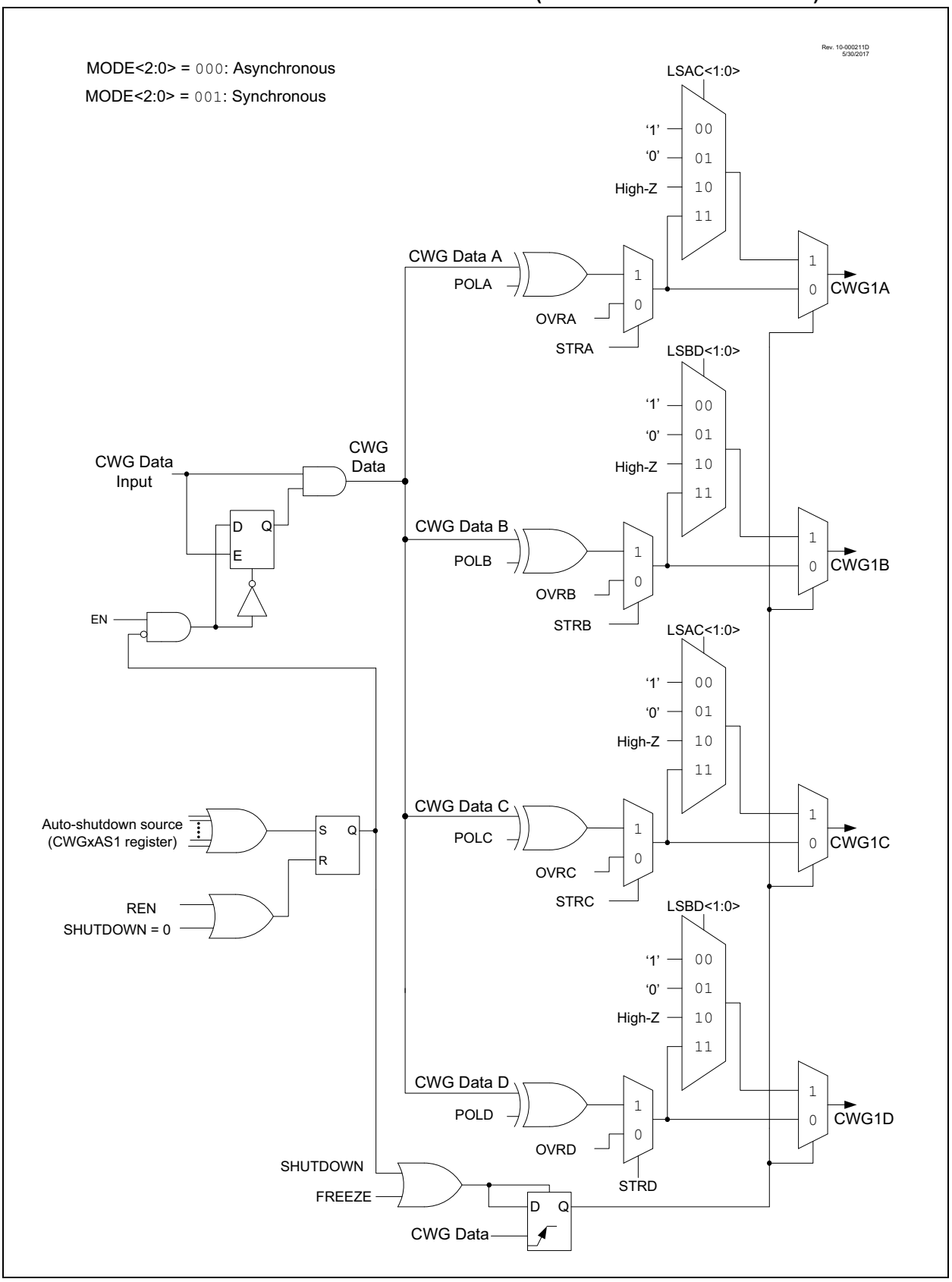


Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

25.6.4 HIGH AND LOW MEASURE MODE

This mode measures the high and low pulse time of the SMTSIGx relative to the SMT clock. It begins incrementing the SMT1TMR on a rising edge on the SMTSIGx input, then updates the SMT1CPW register with the value and resets the SMT1TMR on a falling edge, starting to increment again. Upon observing another rising edge, it updates the SMT1CPR register with its current value and once again resets the SMT1TMR value and begins incrementing again. See Figure 25-8 and Figure 25-9.

FIGURE 26-11: SIMPLIFIED CWG BLOCK DIAGRAM (OUTPUT STEERING MODES)



27.7 Register Definitions: CLC Control

REGISTER 27-1: CLCxCON: CONFIGURABLE LOGIC CELL CONTROL REGISTER

R/W-0/0	U-0	R-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
EN	—	OUT	INTP	INTN	MODE<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 **EN:** Configurable Logic Cell Enable bit
1 = Configurable logic cell is enabled and mixing input signals
0 = Configurable logic cell is disabled and has logic zero output

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **OUT:** Configurable Logic Cell Data Output bit
Read-only: logic cell output data, after LCPOL; sampled from CLCxOUT

bit 4 **INTP:** Configurable Logic Cell Positive Edge Going Interrupt Enable bit
1 = CLCxIF will be set when a rising edge occurs on CLCxOUT
0 = CLCxIF will not be set

bit 3 **INTN:** Configurable Logic Cell Negative Edge Going Interrupt Enable bit
1 = CLCxIF will be set when a falling edge occurs on CLCxOUT
0 = CLCxIF will not be set

bit 2-0 **MODE<2:0>:** Configurable Logic Cell Functional Mode bits
111 = Cell is 1-input transparent latch with S and R
110 = Cell is J-K flip-flop with R
101 = Cell is 2-input D flip-flop with R
100 = Cell is 1-input D flip-flop with S and R
011 = Cell is S-R latch
010 = Cell is 4-input AND
001 = Cell is OR-XOR
000 = Cell is AND-OR

The Master process is started by writing the PID to the UxP1L register when UxP2 is '0' and the UART is idle. The UxTXIF will not be set in this case. Only the six Least Significant bits of UxP1L are used in the PID transmission.

The two Most Significant bits of the transmitted PID are PID parity bits. PID<6> is the exclusive-or of PID bits 0,1,2,and 4. PID<7> is the inverse of the exclusive-or of PID bits 1,3,4,and 5.

The UART calculates and inserts these bits in the serial stream.

Writing UxP1L automatically clears the UxTXCHK and UxRXCHK registers and generates the Break, delimiter bit, Sync character (55h), and PID transmission portion of the transaction. The data portion of the transaction that follows, if there is one, is a Slave process. See **Section 31.5.2 "LIN Slave Mode"** for more details of that process. The Master receives it's own PID when RXEN is set. Software performs the Slave process corresponding to the PID that was sent and received. Attempting to write UxP1L before an active master process is complete will not succeed. Instead, the TXWRE bit will be set.

31.5.2 LIN SLAVE MODE

LIN Slave mode is configured by the following settings:

- MODE<3:0> = 1011
- TXEN = 1
- RXEN = 1
- UxP2 = Number of data bytes to transmit
- UxP3 = Number of data bytes to receive
- UxBRGH:L = Value to achieve default baud rate
- TXPOL = 0 (for high Idle state)
- STP = desired Stop bits selection
- C0EN = desired checksum mode
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1

The Slave process starts upon detecting a Break on the RX pin. The Break clears the UxTXCHK, UxRXCHK, UxP2, and UxP3 registers. At the end of the Break, the auto-baud circuity is activated and the baud rate is automatically set using the Sync character following the Break. The character following the Sync character is received as the PID code and is saved in the receive FIFO. The UART computes the two PID parity bits from the six Least Significant bits of the PID. If either parity bit does not match the corresponding bit of the received PID code, the PERIF flag is set and saved at the same FIFO location as the PID code. The UxRXIF bit is set indicating that the PID is available.

Software retrieves the PID by reading the UxRXB register and determines the Slave process to execute from that. The checksum method, number of data bytes, and whether to send or receive data, is defined by software according to the PID code.

31.5.2.1 LIN Slave Receiver

When the Slave process is a receiver, the software performs the following tasks:

- UxP3 register is written with a value equal to the number of data bytes to receive.
- C0EN bit is set or cleared to select the appropriate checksum. This must be completed before the Start bit of the checksum byte is received.
- Each byte of the process response is read from UxRXB when UxRXIF is set.

The UART updates the checksum on each received byte. When the last data byte is received, the computed checksum total is stored in the UxRXCHK register. The next received byte is saved in the receive FIFO and added with the value in UxRXCHK. The result of this addition is not accessible. However, if the result is not all '1's, the CERIF bit in the UxERRIR is set. The CERIF flag persists until cleared by software. Software needs to read UxRXB to remove the checksum byte from the FIFO, but the byte can be discarded if not needed for any other purpose.

After the checksum is received, the UART ignores all activity on the RX pin until a Break starts the next transaction.

31.5.2.2 LIN Slave Transmitter

When the Slave process is a transmitter, software performs the following tasks in the order shown:

- UxP2 register is written with a value equal to the number of bytes to transmit. This will enable TXIF flag which is disabled when UxP2 is '0'.
- C0EN bit is set or cleared to select the appropriate checksum
- Inter-byte delay is performed
- Each byte of the process response is written to UxTXB when UxTXIF is set

The UART accumulates the checksum as each byte is written to UxTXB. After the last byte is written, the UART stores the calculated checksum in the UxTXCHK register and transmits the inverted result as the last byte in the response.

The TXIF flag is disabled when UxP2 bytes have been written. Any writes to UxTXB that exceed the UxP2 count will be ignored and set the TXWRE flag in the UxFIFO register.

REGISTER 31-12: UxP1H: UART PARAMETER 1 HIGH REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	P1<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	

bit 7-6 **Unimplemented:** Read as '0'

bit 0 **P1<8>:** Most Significant Bit of Parameter 1

DMX mode:
Most Significant bit of number of bytes to transmit between Start Code and automatic Break generation

DALI Control Device mode:
Most Significant bit of idle time delay after which a Forward Frame is sent. Measured in half-bit periods

DALI Control Gear mode:
Most Significant bit of delay between the end of a Forward Frame and the start of the Back Frame Measured in half-bit periods

Other modes:
Not used

REGISTER 31-13: UxP1L: UART PARAMETER 1 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
P1<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	

bit 7-0 **P1<7:0>:** Least Significant Bits of Parameter 1

DMX mode:
Least Significant Byte of number of bytes to transmit between Start Code and automatic Break generation

DALI Control Device mode:
Least Significant Byte of idle time delay after which a Forward Frame is sent. Measured in half-bit periods

DALI Control Gear mode:
Least Significant Byte of delay between the end of a Forward Frame and the start of the Back Frame Measured in half-bit periods

LIN mode:
PID to transmit (Only Least Significant 6 bits used)

Asynchronous Address mode:
Address to transmit (9th transmit bit automatically set to '1')

Other modes:
Not used

36.5.2 PRECHARGE CONTROL

The precharge stage is an optional period of time that brings the external channel and internal sample and hold capacitor to known voltage levels. Precharge is enabled by writing a non-zero value to the ADPRE register. This stage is initiated when an ADC conversion begins, either from setting the GO bit, a special event trigger, or a conversion restart from the computation functionality. If the ADPRE register is cleared when an ADC conversion begins, this stage is skipped.

During the precharge time, CHOLD is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either VDD or VSS, depending on the value of the PPOL bit of ADCON1. At the same time, the port pin logic of the selected analog channel is overridden to drive a digital high or low out, in order to precharge the outer portion of the ADC's sample path, which includes the external sensor. The output polarity of this override is also determined by the PPOL bit of ADCON1. The amount of time that this charging receives is controlled by the ADPRE register.

Note 1:	The external charging overrides the TRIS setting of the respective I/O pin.
2:	If there is a device attached to this pin, Precharge should not be used.

36.5.3 ACQUISITION CONTROL

The Acquisition stage is an optional time for the voltage on the internal sample and hold capacitor to charge or discharge from the selected analog channel.This acquisition time is controlled by the ADACQ register. If PRE = 0, acquisition starts at the beginning of conversion. When PRE = 1, the acquisition stage begins when precharge ends.

At the start of the acquisition stage, the port pin logic of the selected analog channel is overridden to turn off the digital high/low output drivers so they do not affect the final result of the charge averaging. Also, the selected ADC channel is connected to CHOLD. This allows charge averaging to proceed between the precharged channel and the CHOLD capacitor.

Note:	When PRE! = 0, acquisition time cannot be '0'. In this case, setting ADACQ to '0' will set a maximum acquisition time (8191 ADC clock cycles). When precharge is disabled, setting ADACQ to '0' will disable hardware acquisition time control.

36.5.4 GUARD RING OUTPUTS

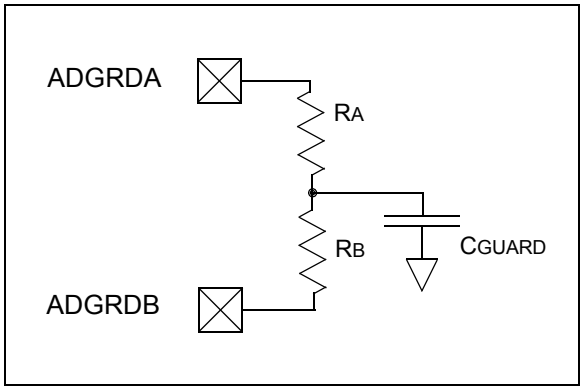
Figure 36-8 shows a typical guard ring circuit. CGUARD represents the capacitance of the guard ring trace placed on the PCB board. The user selects values for RA and RB that will create a voltage profile on CGUARD, which will match the selected acquisition channel.

The purpose of the guard ring is to generate a signal in phase with the CVD sensing signal to minimize the effects of the parasitic capacitance on sensing electrodes. It also can be used as a mutual drive for mutual capacitive sensing. For more information about active guard and mutual drive, see Application Note AN1478, *"mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider"* (DS01478).

The ADC has two guard ring drive outputs, ADGRDA and ADGRDB. These outputs can be routed through PPS controls to I/O pins (see **Section 17.0 "Peripheral Pin Select (PPS) Module"** for details) and the polarity of these outputs are controlled by the ADGPOL and ADIPEN bits of ADCON1.

At the start of the first precharge stage, both outputs are set to match the ADGPOL bit of ADCON1. Once the acquisition stage begins, ADGRDA changes polarity, while ADGRDB remains unchanged. When performing a double sample conversion, setting the ADIPEN bit of ADCON1 causes both guard ring outputs to transition to the opposite polarity of ADGPOL at the start of the second precharge stage, and ADGRDA toggles again for the second acquisition. For more information on the timing of the guard ring output, refer to Figure 36-8 and Figure 36-9.

FIGURE 36-8: GUARD RING CIRCUIT



37.1 Output Voltage Selection

The DAC has 32 voltage level ranges. The 32 levels are set with the DATA<4:0> bits of the DAC1CON1 register.

The DAC output voltage can be determined by using Equation 37-1.

37.2 Ratiometric Output Level

The DAC output value is derived using a resistor ladder with each end of the ladder tied to a positive and negative voltage reference input source. If the voltage of either input source fluctuates, a similar fluctuation will result in the DAC output value.

The value of the individual resistors within the ladder can be found in Table 44-18.

37.3 DAC Voltage Reference Output

The unbuffered DAC voltage can be output to the DAC1OUTn pin(s) by setting the respective DACOEn bit(s) of the DAC1CON0 register. Selecting the DAC reference voltage for output on either DAC1OUTn pin automatically overrides the digital output buffer, the weak pull-up and digital input threshold detector functions of that pin.

Reading the DAC1OUTn pin when it has been configured for DAC reference voltage output will always return a '0'.

Note:	The unbuffered DAC output (DAC1OUTn) is not intended to drive an external load.

37.4 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Windowed Watchdog Timer Time-out, the contents of the DAC1CON0 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

37.5 Effects of a Reset

A device Reset affects the following:

- DAC1 is disabled.
- DAC1 output voltage is removed from the DAC1OUTn pin(s).
- The DAC1R<4:0> range select bits are cleared.

EQUATION 37-1: DAC OUTPUT VOLTAGE

IF DACEN = 1

$$DACx\_output = \left((V_{REF+} - V_{REF-}) \times \frac{DATA[4:0]}{2^5} \right) + V_{REF-}$$

Note: See the DAC1CON0 register for the available $V_{SOURCE+}$ and $V_{SOURCE-}$ selections.

TABLE 41-2: INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
LITERAL INSTRUCTIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f_n, k	Load FSR(f_n) with a 14-bit literal (k)	2	1110	1110	00ff	kkkk	None	
				1111	00kk	kkkk	kkkk		
ADDFSR	f_n, k	Add FSR(f_n) with (k)	1	1110	1000	ffkk	kkkk	None	
SUBFSR	f_n, k	Subtract (k) from FSR(f_n)	1	1110	1001	ffkk	kkkk	None	
MOVLB	k	Move literal to BSR<5:0>	1	0000	0001	00kk	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY – PROGRAM MEMORY INSTRUCTIONS									
TBLRD*		Table Read	2 - 5	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2 - 5	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Note 1: If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a NOP.

2: Some instructions are multi word instructions. The second/third words of these instructions will be decoded as a NOP, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

3: f_s and f_d do not cover the full memory range. 2 MSBs of bank selection are forced to ‘b00 to limit the range of these instructions to lower 4k addressing space.

41.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18(L)F2x/4x/5xK42 family of devices. This includes the MPLAB C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
3989h	IPR9	—	—	—	—	CLC3IP	CWG3IP	CCP3IP	TMR6IP	165
3988h	IPR8	TMR5GIP	TMR5IP	—	—	—	—	—	—	164
3987h	IPR7	—	—	INT2IP	CLC2IP	CWG2IP	-	CCP2IP	TMR4IP	164
3986h	IPR6	TMR3GIP	TMR3IP	U2IP	U2EIP	U2TXIP	U2RXIP	I2C2EIP	I2C2IP	163
3985h	IPR5	I2C2TXIP	I2C2RXIP	DMA2AIP	DMA2ORIP	DMA2DCN-TIP	DMA2SCN-TIP	C2IP	INT1IP	162
3984h	IPR4	CLC1IP	CWG1IP	NCO1IP	—	CCP1IP	TMR2IP	TMR1GIP	TMR1IP	161
3983h	IPR3	TMR0IP	U1IP	U1EIP	U1TXIP	U1RXIP	I2C1EIP	I2C1IP	I2C1TXIP	160
3982h	IPR2	I2C1RXIP	SPI1IP	SPI1TXIP	SPI1RXIP	DMA1AIP	DMA1ORIP	DMA1DCN-TIP	DMA1SCNTIP	159
3981h	IPR1	SMT1PWAIP	SMT1PRAIP	SMT1IP	C1IP	ADTIP	ADIP	ZCDIP	INT0IP	158
3980h	IPR0	IOCIP	CRCIP	SCANIP	NVMIP	CSWIP	OSFIP	HLVDIP	SWIP	157
397Fh - 397Eh	—	Unimplemented								
397Dh	SCANTRIG	—	—	—	—	TSEL				226
397Ch	SCANCON0	EN	TRIGEN	SGO	—	—	MREG	BURSTMD	BUSY	222
397Bh	SCANHADRU	—	—	HADR						224
397Ah	SCANHADRH	HADR								225
3979h	SCANHADRL	HADR								225
3978h	SCANLADRU	—	—	LADR						223
3977h	SCANLADRH	LADR								223
3976h	SCANLADRL	LADR								224
3975h - 396Ah	—	Unimplemented								
3969h	CRCCON1	DLEN				PLEN				218
3968h	CRCCON0	EN	CRCGO	BUSY	ACCM	—	—	SHIFTM	FULL	218
3967h	CRCXORH	X15	X14	X13	X12	X11	X10	X9	X8	221
3966h	CRCXORL	X7	X6	X5	X4	X3	X2	X1	—	221
3965h	CRCSHIFTH	SHFT15	SHFT14	SHFT13	SHFT12	SHFT11	SHFT10	SHFT9	SHFT8	220
3964h	CRCSHIFTL	SHFT7	SHFT6	SHFT5	SHFT4	SHFT3	SHFT2	SHFT1	SHFT0	220
3963h	CRCACCH	ACC15	ACC14	ACC13	ACC12	ACC11	ACC10	ACC9	ACC8	219
3962h	CRCACCL	ACC7	ACC6	ACC5	ACC4	ACC3	ACC2	ACC1	ACC0	220
3961h	CRCDATH	DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8	219
3960h	CRCDATL	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0	219
395Fh	WDTTMR	WDTTMR					STATE	PSCNT		185
395Eh	WDTPSH	PSCNT								184
395Dh	WDTPSL	PSCNT								184
395Ch	WDTCON1	—	CS			—	WINDOW			183
395Bh	WDTCON0	—	—	PS					SEN	182
395Ah - 38A0h	—	Unimplemented								
389Fh	IVTADU	AD								167
389Eh	IVTADH	AD								167
389Dh	IVTADL	AD								167
389Ch - 3891h	—	Unimplemented								
3890h	PRODH_SHAD	PRODH								125
388Fh	PRODL_SHAD	PRODL								125
388Eh	FSR2H_SHAD	—	—	FSR2H						125

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
3: Unimplemented on PIC18(L)F26/27/45/46/47K42 devices.
4: Unimplemented in PIC18(L)F45/55K42.

43.0 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

43.1 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