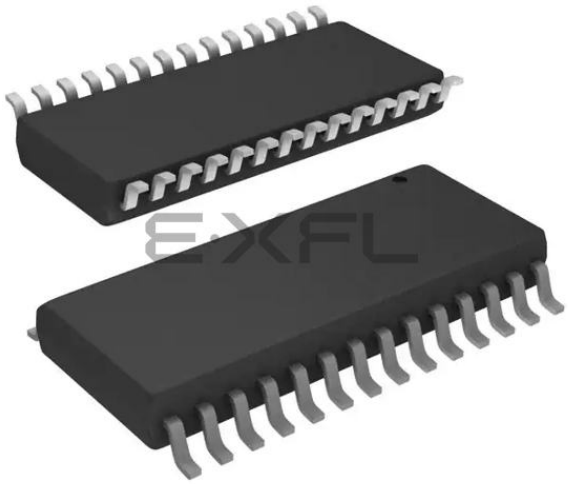


Welcome to E-XFL.COM

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	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
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-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf25k42-e-so

4.2.4 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21-bit wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by any operation that reads PCL. This is useful for computed offsets to the PC (see **Section 4.3.2.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by two to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

4.2.5 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack (TOS) Special File Registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL, CALLW or RCALL instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits in the PCON0 register indicate if the stack has overflowed or underflowed.

4.2.5.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, holds the contents of the stack location pointed to by the STKPTR register (Figure 4-1). This allows users to implement a software stack, if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the Global Interrupt Enable (GIE) bits while accessing the stack to prevent inadvertent stack corruption.

TABLE 4-4: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F24/25K42 DEVICES BANK 62

3EFFh	ADCLK	3EDFh	ADLTHH	3EBFh	CM1PCH	3E9Fh	—	3E7Fh	—	3E5Fh	—	3E3Fh	—	3E1Fh	—
3EFEh	ADACT	3EDEh	ADLTHL	3EBEh	CM1NCH	3E9Eh	DAC1CON0	3E7Eh	—	3E5Eh	—	3E3Eh	—	3E1Eh	—
3EFDh	ADREF	3EDDh	—	3EBDh	CM1CON1	3E9Dh	—	3E7Dh	—	3E5Dh	—	3E3Dh	—	3E1Dh	—
3EFCh	ADSTAT	3EDCh	—	3EBCh	CM1CON0	3E9Ch	DAC1CON1	3E7Ch	—	3E5Ch	—	3E3Ch	—	3E1Ch	—
3EFBh	ADCON3	3EDBh	—	3EBBh	CM2PCH	3E9Bh	—	3E7Bh	—	3E5Bh	—	3E3Bh	—	3E1Bh	—
3EFAh	ADCON2	3EDAh	—	3EBAh	CM2NCH	3E9Ah	—	3E7Ah	—	3E5Ah	—	3E3Ah	—	3E1Ah	—
3EF9h	ADCON1	3ED9h	—	3EB9h	CM2CON1	3E99h	—	3E79h	—	3E59h	—	3E39h	—	3E19h	—
3EF8h	ADCON0	3ED8h	—	3EB8h	CM2CON0	3E98h	—	3E78h	—	3E58h	—	3E38h	—	3E18h	—
3EF7h	ADPREH	3ED7h	ADCP	3EB7h	—	3E97h	—	3E77h	—	3E57h	—	3E37h	—	3E17h	—
3EF6h	ADPREL	3ED6h	—	3EB6h	—	3E96h	—	3E76h	—	3E56h	—	3E36h	—	3E16h	—
3EF5h	ADCAP	3ED5h	—	3EB5h	—	3E95h	—	3E75h	—	3E55h	—	3E35h	—	3E15h	—
3EF4h	ADACQH	3ED4h	—	3EB4h	—	3E94h	—	3E74h	—	3E54h	—	3E34h	—	3E14h	—
3EF3h	ADACQL	3ED3h	—	3EB3h	—	3E93h	—	3E73h	—	3E53h	—	3E33h	—	3E13h	—
2EF2h	—	3ED2h	—	3EB2h	—	3E92h	—	3E72h	—	3E52h	—	3E32h	—	3E12h	—
3EF1h	ADPCH	3ED1h	—	3EB1h	—	3E91h	—	3E71h	—	3E51h	—	3E31h	—	3E11h	—
3EF0h	ADRESH	3ED0h	—	3EB0h	—	3E90h	—	3E70h	—	3E50h	—	3E30h	—	3E10h	—
3EEFh	ADRESL	3ECFh	—	3EAFh	—	3E8Fh	—	3E6Fh	—	3E4Fh	—	3E2Fh	—	3E0Fh	—
3EEEh	ADPREVH	3ECEh	—	3EAEh	—	3E8Eh	—	3E6Eh	—	3E4Eh	—	3E2Eh	—	3E0Eh	—
3EEDh	ADPREVL	3ECDh	—	3EADh	—	3E8Dh	—	3E6Dh	—	3E4Dh	—	3E2Dh	—	3E0Dh	—
3EECh	ADRPT	3ECCh	—	3EACh	—	3E8Ch	—	3E6Ch	—	3E4Ch	—	3E2Ch	—	3E0Ch	—
3EEBh	ADCNT	3ECBh	—	3EABh	—	3E8Bh	—	3E6Bh	—	3E4Bh	—	3E2Bh	—	3E0Bh	—
3EEAh	ADACCU	3ECAh	HLVDCON1	3EAAh	—	3E8Ah	—	3E6Ah	—	3E4Ah	—	3E2Ah	—	3E0Ah	—
3EE9h	ADACCH	3EC9h	HLVDCON0	3EA9h	—	3E89h	—	3E69h	—	3E49h	—	3E29h	—	3E09h	—
3EE8h	ADACCL	3EC8h	—	3EA8h	—	3E88h	—	3E68h	—	3E48h	—	3E28h	—	3E08h	—
3EE7h	ADFLTRH	3EC7h	—	3EA7h	—	3E87h	—	3E67h	—	3E47h	—	3E27h	—	3E07h	—
3EE6h	ADFLTRL	3EC6h	—	3EA6h	—	3E86h	—	3E66h	—	3E46h	—	3E26h	—	3E06h	—
3EE5h	ADSTPTH	3EC5h	—	3EA5h	—	3E85h	—	3E65h	—	3E45h	—	3E25h	—	3E05h	—
3EE4h	ADSTPTL	3EC4h	—	3EA4h	—	3E84h	—	3E64h	—	3E44h	—	3E24h	—	3E04h	—
3EE3h	ADERRH	3EC3h	ZCDCON	3EA3h	—	3E83h	—	3E63h	—	3E43h	—	3E23h	—	3E03h	—
3EE2h	ADERRL	3EC2h	—	3EA2h	—	3E82h	—	3E62h	—	3E42h	—	3E22h	—	3E02h	—
3EE1h	ADUTHH	3EC1h	FVRCON	3EA1h	—	3E81h	—	3E61h	—	3E41h	—	3E21h	—	3E01h	—
3EE0h	ADUTHL	3EC0h	CMOUT	3EA0h	—	3E80h	—	3E60h	—	3E40h	—	3E20h	—	3E00h	—

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

TABLE 4-11: REGISTER FILE SUMMARY FOR PIC18(L)F24/25K42 DEVICES (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
3DE0h	U2UIR	WUIF	ABDIF	—	—	—	ABDIE	—	—	00---0--
3DDFh	U2FIFO	TXWRE	STPMD	TXBE	TXBF	RXIDL	XON	RXBE	RXBF	00101110
3DDEh	U2BRGH	BRGH								00000000
3DDDh	U2BRGL	BRGL								00000000
3DDCh	U2CON2	RUNOVF	RXPOL	STP		—	TXPOL	FLO		0000-000
3DDBh	U2CON1	ON	—	—	WUE	RXBIMD	—	BRKOVR	SENDB	0--00-00
3DDAh	U2CON0	BRGS	ABDEN	TXEN	RXEN	MODE				0000-000
3DD9h	—	Unimplemented								—
3DD8h	U2P3L	P3L								xxxxxxxx
3DD7h	—	Unimplemented								—
3DD6h	U2P2L	P2L								00000000
3DD5h	—	Unimplemented								—
3DD4h	U2P1L	P1L								00000000
3DD3h	—	Unimplemented								—
3DD2h	U2TXB	TXB								00000000
3DD1h	—	Unimplemented								—
3DD0h	U2RXB	RXB								00000000
3DCFh - 3D7Dh	—	Unimplemented								—
3D7Ch	I2C1BTO	BTO								-----000
3D7Bh	I2C1CLK	CLK								----0000
3D7Ah	I2C1PIE	CNTIE	ACKTIE	—	WRIE	ADRIE	PCIE	RSCIE	SCIE	00000000
3D79h	I2C1PIR	CNTIF	ACKTIF	—	WRIF	ADRIF	PCIF	RSCIF	SCIF	00000000
3D78h	I2C1STAT1	TXWE	—	TXBE	—	RXRE	CLRBF	—	RXBF	00100000
3D77h	I2C1STAT0	BFRE	SMA	MMA	R	D	—	—	—	00000000
3D76h	I2C1ERR	—	BTOIF	BCLIF	NACKIF	—	BTOIE	BCLIE	NACKIE	00000000
3D75h	I2C1CON2	ACNT	GCEN	FME	ABD	SDAHT		BFRET		00000000
3D74h	I2C1CON1	ACKCNT	ACKDT	ACKSTAT	ACKT	—	RXO	TXU	CSD	00000000
3D73h	I2C1CON0	EN	RSEN	S	CSTR	MDR	MODE			00000000
3D72h	I2C1ADR3	ADR							—	11111110
3D71h	I2C1ADR2	ADR								11111111
3D70h	I2C1ADR1	ADR							—	11111110
3D6Fh	I2C1ADR0	ADR								11111111
3D6Eh	I2C1ADB1	ADB								xxxxxxxx
3D6Dh	I2C1ADB0	ADB								xxxxxxxx
3D6Ch	I2C1CNT	CNT								xxxxxxxx
3D6Bh	I2C1TXB	TXB								xxxxxxxx
3D6Ah	I2C1RXB	RXB								xxxxxxxx
3D69h - 3D67h	—	Unimplemented								—
3D66h	I2C2BTO	BTO								-----000
3D65h	I2C2CLK	CLK								----0000
3D64h	I2C2PIE	CNTIE	ACKTIE	—	WRIE	ADRIE	PCIE	RSCIE	SCIE	00000000
3D63h	I2C2PIR	CNTIF	ACKTIF	—	WRIF	ADRIF	PCIF	RSCIF	SCIF	00000000
3D62h	I2C2STAT1	TXWE	—	—	—	RXRE	CLRBF	—	RXBF	00100000
3D61h	I2C2STAT0	BFRE	—	MMA	—	D	—	—	—	00000000
3D60h	I2C2ERR	—	BTOIF	BCLIF	NACKIF	—	BTOIE	BCLIE	NACKIE	00000000
3D5Fh	I2C2CON2	ACNT	GCEN	FME	ABD	SDAHT		BFRET		00000000
3D5Eh	I2C2CON1	ACKCNT	ACKDT	ACKSTAT	ACKT	—	RXO	TXU	CSD	00000000

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not present in LF devices.

7.0 DEVICE CONFIGURATION INFORMATION

The Device Configuration Information (DCI) is a dedicated region in the Program memory space mapped from 3FFF00h to 3FFF09h. The data stored in these locations is ready-only and cannot be erased or modified.

Refer to Table 7-1: Device Configuration Information for the complete DCI table address and description. The DCI holds information about the device which is useful for programming and Bootloader applications. These locations are read-only and cannot be erased or modified.

TABLE 7-1: DEVICE CONFIGURATION INFORMATION

ADDRESS	Name	DESCRIPTION	VALUE	UNITS
3FFF00h-3FFF01h	ERSIZ	Erase Row Size	32	Words
3FFF02h-3FFF03h	WLSIZ	Number of write latches	64	
3FFF04h-3FFF05h	URSIZ	Number of User Rows	See Table 7-2	Rows
3FFF06h-3FFF07h	EESIZ	EE Data memory size	256	Bytes
3FFF08h-3FFF09h	PCNT	Pin Count	28	Pins

TABLE 7-2: MEMORY SIZE AND NUMBER OF USER ROWS

Part Name	Memory size	Number of user rows
PIC18(L)F24K42	8K	256
PIC18(L)F25K42	16K	512

7.1 DIA and DCI Access

The DIA and DCI addresses are read-only and cannot be erased or modified. See **Section 15.2 "Device Information Area, Device Configuration Area, User ID, Device ID and Configuration Word Access"** for more information on accessing these memory locations.

Development tools, such as device programmers and debuggers, may be used to read the DIA and DCI regions, similar to the Device ID and Revision ID.

FIGURE 8-2: LPBOR, BOR, POR RELATIONSHIP

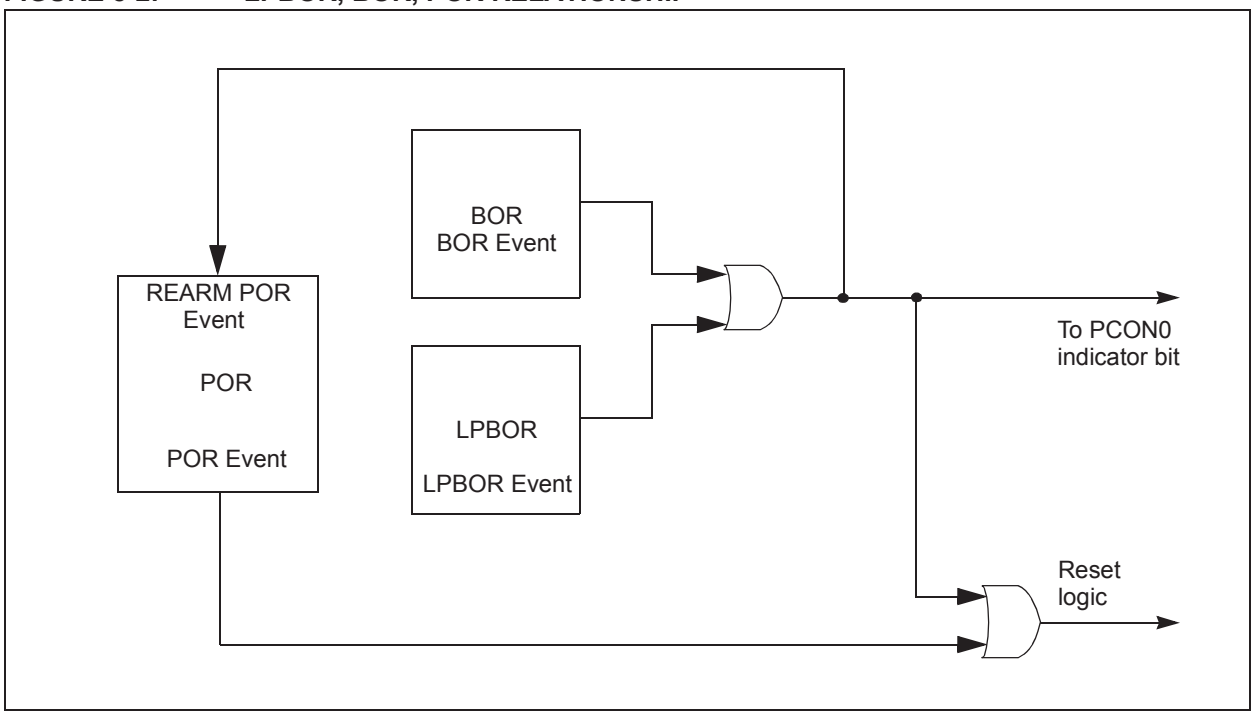
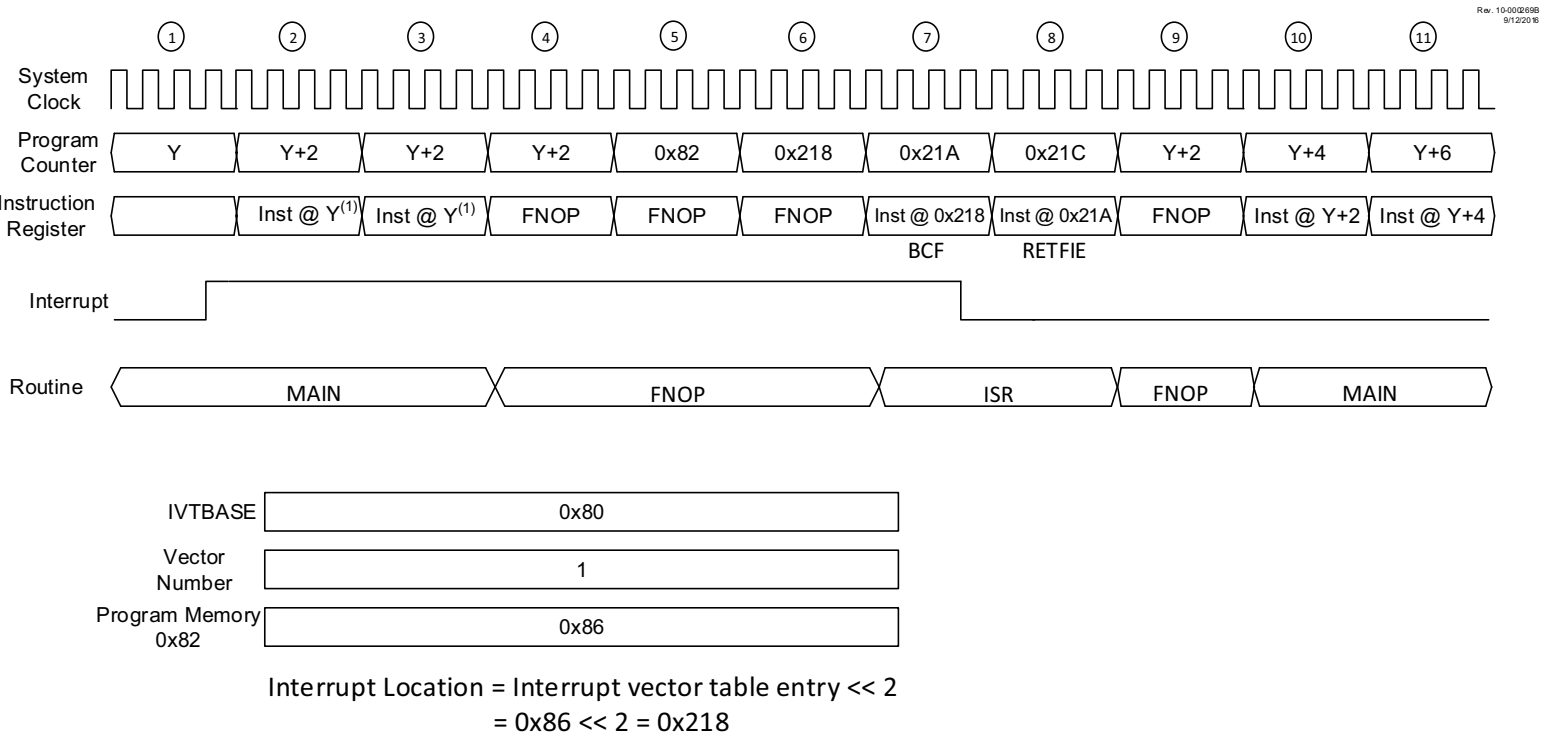


FIGURE 11-8: INTERRUPT TIMING DIAGRAM - TWO WORD INSTRUCTION



Note 1: Instruction @ Y is a Two-cycle instruction.

REGISTER 11-36: IVTBASEU: INTERRUPT VECTOR TABLE BASE ADDRESS UPPER REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	BASE<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 **BASE<20:16>**: Interrupt Vector Table Base Address bits

REGISTER 11-37: IVTBASEH: INTERRUPT VECTOR TABLE BASE 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
BASE<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 **BASE<15:8>**: Interrupt Vector Table Base Address bits

REGISTER 11-38: IVTBASEL: INTERRUPT VECTOR TABLE BASE ADDRESS LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-1/1	R/W-0/0	R/W-0/0	R/W-0/0
BASE<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 **BASE<7:0>**: Interrupt Vector Table Base Address bits

17.0 DIRECT MEMORY ACCESS (DMA)

17.1 Introduction

The Direct Memory Access (DMA) module is designed to service data transfers between different memory regions directly without intervention from the CPU. By eliminating the need for CPU-intensive management of handling interrupts intended for data transfers, the CPU now can spend more time on other tasks.

PIC18(L)F2XK42 family has two DMA modules which can be independently programmed to transfer data between different memory locations, move different data sizes, and use a wide range of hardware triggers to initiate transfers. The two DMA registers can even be programmed to work together, in order to carry out more complex data transfers without CPU overhead.

Key features of the DMA module include:

- Support access to the following memory regions:
  - GPR and SFR space (R/W)
  - Program Flash Memory (R only)
  - Data EEPROM Memory (R only)
- Programmable priority between the DMA and CPU Operations. Refer to **Section 3.1 "System Arbitration"** for details.
- Programmable Source and Destination address modes
  - Fixed address
  - Post-increment address
  - Post-decrement address
- Programmable Source and Destination sizes
- Source and destination pointer register, dynamically updated and reloadable
- Source and destination count register, dynamically updated and reloadable
- Programmable auto-stop based on Source or Destination counter
- Software triggered transfers
- Multiple user selectable sources for hardware triggered transfers
- Multiple user selectable sources for aborting DMA transfers

17.2 DMA Registers

The operation of the DMA module has the following registers:

- Control registers (DMAxCON0, DMAxCON1)
- Data buffer register (DMAxBUF)
- Source Start Address Register (DMAxSSAU:H:L)
- Source Pointer Register (DMAxSPTRU:H:L)
- Source Message Size Register (DMAxSSZH:L)
- Source Count Register (DMAxSCNTH:L)
- Destination Start Address Register (DMAxDSAH:L)
- Destination Pointer Register (DMAxDPTRH:L)
- Destination Message Size Register (DMAxDSZH:L)
- Destination Count Register (DMAxDCNTH:L)
- Start Interrupt Request Source Register (DMAxSIRQ)
- Abort Interrupt Request Source Register (DMAxAIRQ)

These registers are detailed in **Section ""**.

REGISTER 24-6: TxHLT: TIMERx HARDWARE LIMIT CONTROL 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
PSYNC	CKPOL	CKSYNC	MODE<4:0>				
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 **PSYNC:** Timerx Prescaler Synchronization Enable bit(1, 2)
1 = TxTMR Prescaler Output is synchronized to Fosc/4
0 = TxTMR Prescaler Output is not synchronized to Fosc/4

bit 6 **CKPOL:** Timerx Clock Polarity Selection bit(3)
1 = Falling edge of input clock clocks timer/prescaler
0 = Rising edge of input clock clocks timer/prescaler

bit 5 **CKSYNC:** Timerx Clock Synchronization Enable bit(4, 5)
1 = ON register bit is synchronized to T2TMR_clk input
0 = ON register bit is not synchronized to T2TMR_clk input

bit 4-0 **MODE<4:0>:** Timerx Control Mode Selection bits(6, 7)
See Table 24-1 for all operating modes.

Note 1: Setting this bit ensures that reading TxTMR will return a valid data value.
2: When this bit is '1', Timer2 cannot operate in Sleep mode.
3: CKPOL should not be changed while ON = 1.
4: Setting this bit ensures glitch-free operation when the ON is enabled or disabled.
5: When this bit is set then the timer operation will be delayed by two TxTMR input clocks after the ON bit is set.
6: Unless otherwise indicated, all modes start upon ON = 1 and stop upon ON = 0 (stops occur without affecting the value of TxTMR).
7: When TxTMR = TxPR, the next clock clears TxTMR, regardless of the operating mode.

FIGURE 28-12: DEAD-BAND OPERATION, CWGxDBR = 0x01, CWGxDBF = 0x02

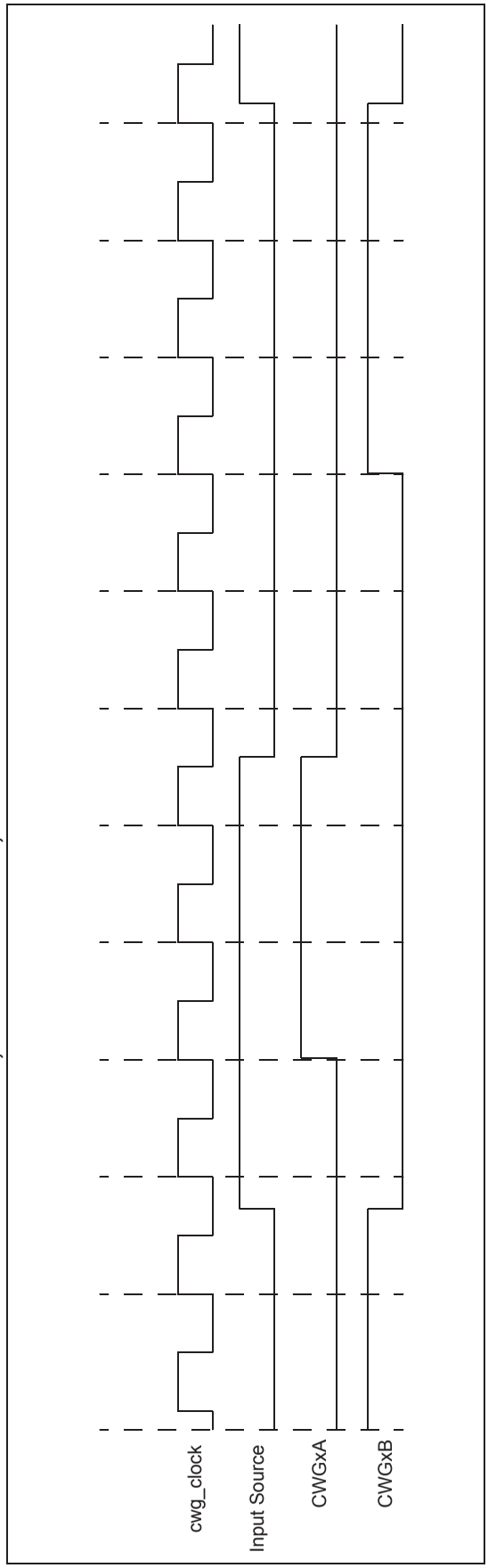
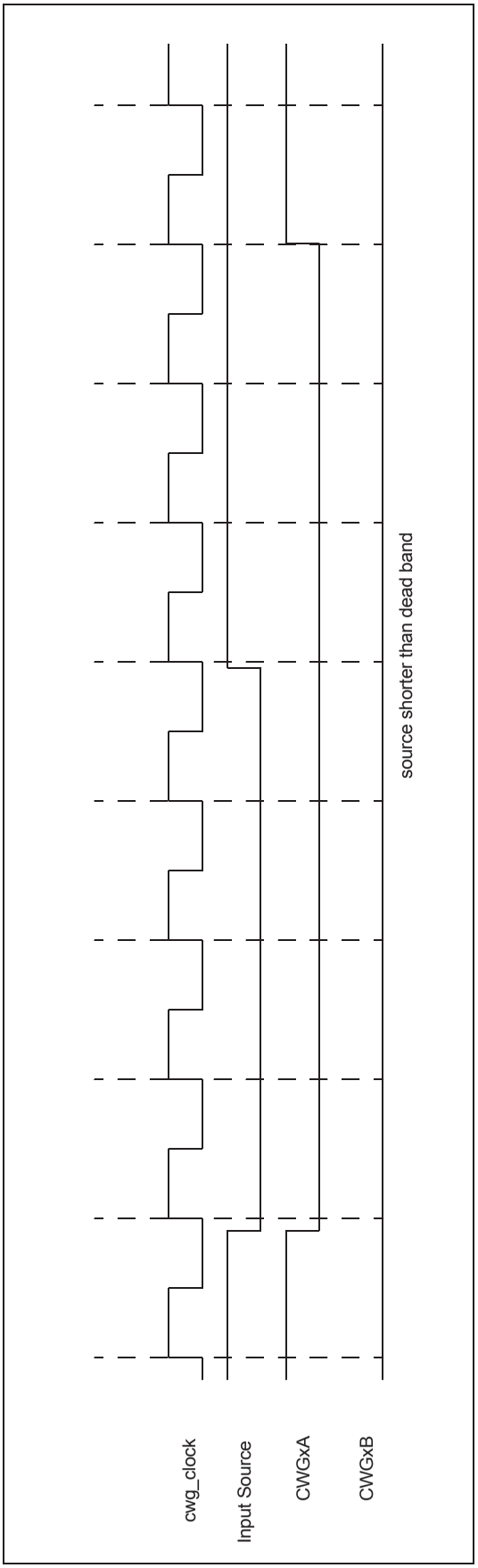


FIGURE 28-13: DEAD-BAND OPERATION, CWGxDBR = 0x03, CWGxDBF = 0x06, SOURCE SHORTER THAN DEAD BAND



32.0 DATA SIGNAL MODULATOR (DSM) MODULE

The Data Signal Modulator (DSM) is a peripheral which allows the user to mix a data stream, also known as a modulator signal, with a carrier signal to produce a modulated output.

Both the carrier and the modulator signals are supplied to the DSM module either internally, from the output of a peripheral, or externally through an input pin.

The modulated output signal is generated by performing a logical “AND” operation of both the carrier and modulator signals and then provided to the MDOUT pin.

The carrier signal is comprised of two distinct and separate signals. A carrier high (CARH) signal and a carrier low (CARL) signal. During the time in which the modulator (MOD) signal is in a logic high state, the DSM mixes the carrier high signal with the modulator signal. When the modulator signal is in a logic low state, the DSM mixes the carrier low signal with the modulator signal.

Using this method, the DSM can generate the following types of Key Modulation schemes:

- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)
- On-Off Keying (OOK)

Additionally, the following features are provided within the DSM module:

- Carrier Synchronization
- Carrier Source Polarity Select
- Programmable Modulator Data
- Modulated Output Polarity Select
- Peripheral Module Disable, which provides the ability to place the DSM module in the lowest power consumption mode

Figure 32-1 shows a Simplified Block Diagram of the Data Signal Modulator peripheral.

REGISTER 33-14: UxP2H: UART PARAMETER 2 HIGH REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0
—	—	—	—	—	—	—	P2<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 **P2<8>:** Most Significant Bit of Parameter 2
DMX mode:
Most Significant bit of first address of receive block
DALI mode:
Most Significant bit of number of half-bit periods of idle time in Forward Frame detection threshold
Other modes:
Not used

REGISTER 33-15: UxP2L: UART PARAMETER 2 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
P2<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 **P2<7:0>:** Least Significant Bits of Parameter 2
DMX mode:
Least Significant Byte of first address of receive block
LIN Slave mode:
Number of data bytes to transmit
DALI mode:
Least Significant Byte of number of half-bit periods of idle time in Forward Frame detection threshold
Asynchronous Address mode:
Receiver address
Other modes:
Not used

35.4.3.5 Slave Transmission (10-bit Addressing Mode)

This section describes the sequence of events for the I^2C module configured as an I^2C slave in 10-bit Addressing mode and is transmitting data. Figure 35-12 is used as a visual reference for this description.

1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
2. Master transmits high address byte with R/W = 0.
3. The received high address is compared with the values in I2CxADR1 and I2CxADR3 registers.
4. If high address matches; R/W is copied to R/W bit, D/A bit is cleared, high address data is copied to I2CxADB1. If the address does not match; module becomes idle.
5. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set. I2CxIF is set.
6. Slave software can read high address from I2CxADB1 and set/clear ACKDT before releasing SCL.
7. ACKDT value is copied out to SDA for ACK pulse. SCL line is released by clearing CSTR.
8. Master sends ninth SCL pulse for ACK
9. Slave can force a NACK at this point due to previous error not being cleared. E.g. Receive buffer overflow or transmit buffer underflow errors. In these cases the Slave hardware forces a NACK and the module becomes idle.
10. Master transmits low address data byte
11. If the low address matches; SMA is set, ADRIF is set, R/W is copied to R/W bit, D/A bit is cleared, low address data is copied to I2CxADB0, and ACTDT is copied to SDA. If the address does not match; module becomes idle.
12. If address hold interrupt is enabled, the CSTR bit is set as mentioned in step 6. Slave software can read low address byte from I2CxADB0 register and change ACKDT value before releasing SCL.
13. Master sends 9th SCL pulse for ACK
14. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set.
15. Slave software can read address from I2CxADB0 and I2CxADB1 registers and change the value of ACKDT before releasing SCL by clearing CSTR.
16. Master asserts Restart condition (cannot be Start) on the bus. Restart Condition Interrupt Flag (RSCIF) is set. If the Restart Condition Interrupt is enabled, generic interrupt I2CxIF is set
17. Master transmits high address byte with R/W = 1.
18. If SMA = 1, and if high address matches; R/W is copied to R/W bit, D/A bit is cleared, high address data is copied to I2CxADB1, and ACTDT is output to SDA. If the address does not match or SMA = 0; module become idle.
19. If ADRIE = 1, CSTR is set. I2CIF is set. Slave software can read address from I2CxADB0/1 and set/clear ACKDT. The ACKDT value is copied out to SDA. SCL is released by clearing CSTR bit.
20. If TXBE = 1 and I2CCNT!= 0 (I2CTXIF = 1), CSTR is set. Slave software must load data into I2CxTXB to release SCL.
21. Master sends SCL pulse for ACK. If I2CCNT = 0, CNTIF is set.
22. If NACK; NACKIF is set, slave goes idle.
23. If ACKTIE = 1, CSTR is set, I2CIF is set. Slave software can read address from I2CxADB0/1 before releasing SCL by clearing CSTR.
24. Master sends eight SCL pulses to clock out data.
25. Go to step 20.

REGISTER 35-16: I2CxADB0 – I²C ADDRESS DATA BUFFER 0 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
ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
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	HS = Hardware set HC = Hardware clear

bit 7-0

MODE<2:0> = 00x
- **ADB<7:1>:** Address Data byte
  - Received matching 7-bit slave address data
- **R/$\overline{W}$:** Read/not-Write Data bit
  - Received read/write value from 7-bit address byte

MODE<2:0> = 01x
- **ADB<7:0>:** Address Data byte
  - Received matching lower 8-bits of 10-bit slave address data

MODE<2:0> = 100
- Unused in this mode; bit state is a don't care

MODE<2:0> = 101
- **ADB<7:0>:** Low Address Data byte
  - Low 10-bit address value copied to transmit shift register

MODE<2:0> = 11x
- **ADB<7:1>:** Address Data byte
  - Received matching 7-bit slave address
- **R/$\overline{W}$:** Read/not-Write Data bit
  - Received read/write value received 7-bit slave address byte

Note 1: This register is read only except in master, 10-bit Address mode (MODE<2:0> = 101).

38.6.5 BURST AVERAGE MODE

The Burst Average mode (ADMD = 011) acts the same as the Average mode in most respects. The one way it differs is that it continuously retriggers ADC sampling until the CNT value is greater than or equal to RPT, even if Continuous Sampling mode (see **Section 38.6.8 "Continuous Sampling mode"**) is not enabled. This allows for a threshold comparison on the average of a short burst of ADC samples.

38.6.6 LOW-PASS FILTER MODE

The Low-pass Filter mode (ADMD = 100) acts similarly to the Average mode in how it handles samples (accumulates samples until CNT value greater than or equal to RPT, then triggers threshold comparison), but instead of a simple average, it performs a low-pass filter operation on all of the samples, reducing the effect of high-frequency noise on the average, then performs a threshold comparison on the results. (see Table 38-2 for a more detailed description of the mathematical operation). In this mode, the ADCRS bits determine the cut-off frequency of the low-pass filter (as demonstrated by Table 38-3).

38.6.7 THRESHOLD COMPARISON

At the end of each computation:

- The conversion results are latched and held stable at the end-of-conversion.
- The error is calculated based on a difference calculation which is selected by the ADCALC<2:0> bits in the ADCON3 register. The value can be one of the following calculations (see Register 38-4 for more details):
  - The first derivative of single measurements
  - The CVD result in CVD mode
  - The current result vs. a setpoint
  - The current result vs. the filtered/average result
  - The first derivative of the filtered/average value
  - Filtered/average value vs. a setpoint
- The result of the calculation (ERR) is compared to the upper and lower thresholds, UTH<ADUTHH:ADUTHL> and LTH<ADLTHH:ADLTHL> registers, to set the ADUTHR and ADLTHR flag bits. The threshold logic is selected by ADTMD<2:0> bits in the ADCON3 register. The threshold trigger option can be one of the following:
  - Never interrupt
  - Error is less than lower threshold
  - Error is greater than or equal to lower threshold
  - Error is between thresholds (inclusive)
  - Error is outside of thresholds
  - Error is less than or equal to upper threshold
  - Error is greater than upper threshold
  - Always interrupt regardless of threshold test results
  - If the threshold condition is met, the threshold interrupt flag ADTIF is set.

Note 1: The threshold tests are signed operations.

2: If ADAOV is set, a threshold interrupt is signaled.

38.6.8 CONTINUOUS SAMPLING MODE

Setting the CONT bit in the ADCON0 register automatically retriggers a new conversion cycle after updating the ADACC register. The GO bit remains set and re-triggering occurs automatically.

If ADSOI = 1, a threshold interrupt condition will clear GO and the conversions will stop.

38.6.9 DOUBLE SAMPLE CONVERSION

Double sampling is enabled by setting the ADDSEN bit of the ADCON1 register. When this bit is set, two conversions are required before the module will calculate threshold error (each conversion must still be triggered separately). The first conversion will set the ADMATH bit of the ADSTAT register and update ADACC, but will not calculate ERR or trigger ADTIF. When the second conversion completes, the first value is transferred to PREV (depending on the setting of ADPSIS) and the value of the second conversion is placed into ADRES. Only upon the completion of the second conversion is ERR calculated and ADTIF triggered (depending on the value of ADCALC).

REGISTER 38-9: ADPREL: ADC PRECHARGE TIME CONTROL REGISTER (LOW BYTE)

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
PRE<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 **PRE<7:0>**: Precharge Time Select bits
See Table 38-4.

REGISTER 38-10: ADPREH: ADC PRECHARGE TIME CONTROL REGISTER (HIGH BYTE)

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	PRE<12:8>				
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 7-5 **Unimplemented**: Read as '0'

bit 4-0 PRE<12:8>: Precharge Time Select bits
See Table 38-4.

Note: If PRE is not equal to '0', then ADACQ = b'00000000 means Acquisition time is 256 clocks of the selected ADC clock.

TABLE 38-4: PRECHARGE TIME

ADPRE	Precharge time
1 1111 1111 1111	8191 clocks of the selected ADC clock
1 1111 1111 1110	8190 clocks of the selected ADC clock
1 1111 1111 1101	8189 clocks of the selected ADC clock
...	...
0 0000 0000 0010	2 clocks of the selected ADC clock
0 0000 0000 0001	1 clock of the selected ADC clock
0 0000 0000 0000	Not included in the data conversion cycle

BTG **Bit Toggle f**

Syntax: BTG f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: $(\overline{f<b>}) \rightarrow f<b>$

Status Affected: None

Encoding:

0111	bbba	ffff	ffff

Description: Bit 'b' in data memory location 'f' is inverted.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 43.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
BTG     PORTC,  4, 0
```
Before Instruction:
PORTC = 0111 0101 [75h]
After Instruction:
PORTC = 0110 0101 [65h]

BOV **Branch if Overflow**

Syntax: BOV n

Operands: $-128 \le n \le 127$

Operation: if OVERFLOW bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0100	nnnn	nnnn

Description: If the OVERFLOW bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example:
```
HERE    BOV  Jump
```
Before Instruction
PC = address (HERE)
After Instruction
If OVERFLOW = 1;
PC = address (Jump)
If OVERFLOW = 0;
PC = address (HERE + 2)

45.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

45.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

45.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

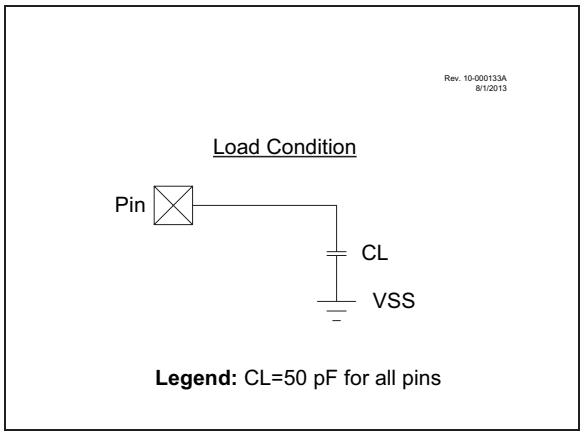
45.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

46.4 AC Characteristics

FIGURE 46-4: LOAD CONDITIONS



Legend: CL=50 pF for all pins

TABLE 46-21: SPI MODE REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
SP70*	TssL2scH, TssL2scL	$\overline{SS}$↓ to SCK↓ or SCK↑ input	2.25*Tcy	—	—	ns	
SP71*	TscH	SCK input high time (Slave mode)	Tcy + 20	—	—	ns	
SP72*	TscL	SCK input low time (Slave mode)	Tcy + 20	—	—	ns	
SP73*	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge	100	—	—	ns	
SP74*	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge	100	—	—	ns	
SP75*	TdoR	SDO data output rise time	—	10	25	ns	3.0V ≤ VDD ≤ 5.5V
			—	25	50	ns	1.8V ≤ VDD ≤ 5.5V
SP76*	TdoF	SDO data output fall time	—	10	25	ns	
SP77*	TssH2doZ	$\overline{SS}$↑ to SDO output high-impedance	10	—	50	ns	
SP78*	TscR	SCK output rise time (Master mode)	—	10	25	ns	3.0V ≤ VDD ≤ 5.5V
			—	25	50	ns	1.8V ≤ VDD ≤ 5.5V
SP79*	TscF	SCK output fall time (Master mode)	—	10	25	ns	
SP80*	TscH2doV, TscL2doV	SDO data output valid after SCK edge	—	—	50	ns	3.0V ≤ VDD ≤ 5.5V
			—	—	145	ns	1.8V ≤ VDD ≤ 5.5V
SP81*	TdoV2scH, TdoV2scL	SDO data output setup to SCK edge	1 Tcy	—	—	ns	
SP82*	TssL2doV	SDO data output valid after $\overline{SS}$↓ edge	—	—	50	ns	
SP83*	TscH2ssH, TscL2ssH	$\overline{SS}$ ↑ after SCK edge	1.5 Tcy + 40	—	—	ns	

* 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.