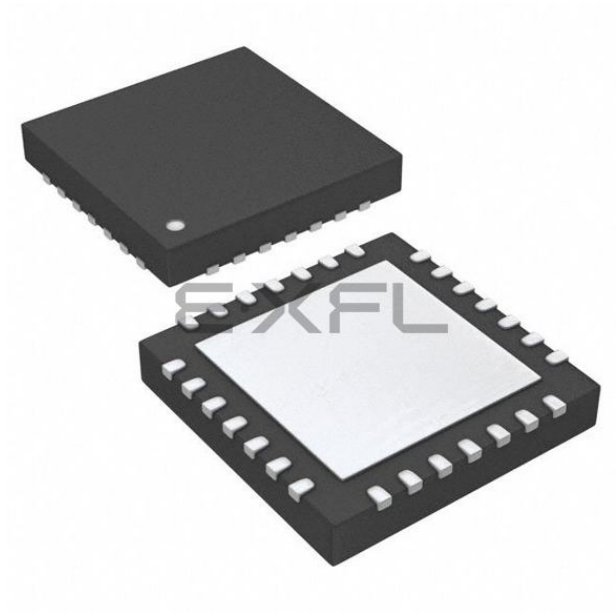


Welcome to E-XFL.COM

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	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K 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 ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-UFQFN Exposed Pad
Supplier Device Package	28-UQFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f24k42t-i-mv

11.3.2 NATURAL ORDER (HARDWARE) PRIORITY

When more than one interrupt with the same user specified priority level are requested, the priority conflict is resolved by using a method called “Natural Order Priority”. Natural order priority is a fixed priority scheme that is based on the Interrupt Vector Table. Table 11-2 shows the natural order priority and the interrupt vector number assigned for each source.

TABLE 11-2: INTERRUPT VECTOR PRIORITY TABLE

Vector Number	Interrupt Source	Vector Number	Interrupt Source
0	Software Interrupt	42	DMA2SCNT
1	HLVD	43	DMA2DCNT
2	OSF	44	DMA2OR
3	CSW	45	DMA2A
4	NVM	46	I2C2RX
5	SCAN	47	I2C2TX
6	CRC	48	I2C2
7	IOC	49	I2C2E
8	INT0	50	U2RX
9	ZCD	51	U2TX
10	AD	52	U2E
11	ADT	53	U2
12	C1	54	TMR3
13	SMT1	55	TMR3G
14	SMT1PRA	56	TMR4
15	SMT1PWA	57	CCP2
16	DMA1SCNT	58	—
17	DMA1DCNT	59	CWG2
18	DMA1OR	60	CLC2
19	DMA1A	61	INT2
20	SPI1RX	62	—
21	SPI1TX	63	—
22	SPI1	64	—
23	I2C1RX	65	—
24	I2C1TX	66	—
25	I2C1	67	—
26	I2C1E	68	—
27	U1RX	69	—
28	U1TX	70	TMR5
29	U1E	71	TMR5G
30	U1	72	TMR6
31	TMR0	73	CCP3
32	TMR1	74	CWG3
33	TMR1G	75	CLC3
34	TMR2	76	—
35	CCP1	77	—
36	—	78	—
37	NCO	79	—
38	CWG1	80	CCP4
39	CLC1	81	CLC4
40	INT1		
41	C2		

The natural order priority scheme has vector interrupt 0 as the highest priority and vector interrupt 81 as the lowest priority.

For example, when two concurrently occurring interrupt sources that are both designated high priority using the IPRx register will be resolved using the natural order priority (i.e., the interrupt with a lower corresponding vector number will preempt the interrupt with the higher vector number).

The ability for the user to assign every interrupt source to high or low priority levels means that the user program can give an interrupt with a low natural order priority a higher overall priority level.

11.4 Interrupt Operation

All pending interrupts are indicated by the flag bit being equal to a ‘1’ in the PIRx register. All pending interrupts are resolved using the priority scheme explained in **Section 11.3 “Interrupt Priority”**.

Once the interrupt source to be serviced is resolved, the program execution vectors to the resolved interrupt vector addresses, as explained in **Section 11.2 “Interrupt Vector Table (IVT)”**. The vector number is also stored in the WREG register. Most of the flag bits are required to be cleared by the application software, but in some cases, device hardware clears the interrupt automatically. Some flag bits are read-only in the PIRx registers, these flags are a summary of the source interrupts and the corresponding interrupt flags of the source must be cleared.

A valid interrupt can be either a high or low priority interrupt when in main routine or a high priority interrupt when in low priority Interrupt Service Routine. Depending on order of interrupt requests received and their relative timing, the CPU will be in the state of execution indicated by the STAT bits of the INTCON1 register (Register 11-2).

The State machine shown in Figure 11-1 and the subsequent sections detail the execution of interrupts when received in different orders.

Note:	The state of GIEH/L is not changed by the hardware when servicing an interrupt. The internal state machine is used to keep track of execution states. These bits can be manipulated in the user code resulting in transferring execution to the main routine and ignoring existing interrupts.

REGISTER 15-2: NVMCON2: NONVOLATILE MEMORY CONTROL 2 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMCON2<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 **NVMCON2<7:0>:**
Refer to **Section 15.1.4 "NVM Unlock Sequence"**.

Note 1: This register always reads zeros, regardless of data written.

Register 15-3: NVMADRL: Data EEPROM Memory Address Low

R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0	R/W-x/0
ADR<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 **ADR<7:0>:** EEPROM Read Address bits

REGISTER 15-4: NVMADRH: DATA EEPROM MEMORY ADDRESS HIGH(1)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-x/u	R/W-x/u
—	—	—	—	—	—	ADR<9:8>	
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 7-2 **Unimplemented:** Read as '0'
bit 1-0 **ADR<9:8>:** EEPROM Read Address bits

Note 1: The NVMADRH register is not implemented on PIC18(L)F24/25K42.

REGISTER 16-11: SCANCON0: SCANNER ACCESS CONTROL REGISTER 0

R/W-0/0	R/W-0/0	R/W/HC-0/0	U-0	U-0	R/W-0/0	R/W-0/0	R-0/0
EN	TRIGEN	SGO	—	—	MREG	BURSTMD	BUSY
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	HC = Bit is cleared by hardware

bit 7 **EN:** Scanner Enable bit(1)
1 = Scanner is enabled
0 = Scanner is disabled

bit 6 **TRIGEN:** Scanner Trigger Enable bit(2)
1 = Scanner trigger is enabled
0 = Scanner trigger is disabled
Refer Table 16-2.

bit 5 **SGO:** Scanner GO bit(3, 4)
1 = When the CRC is ready, the Memory region set by the MREG bit will be accessed and data is passed to the CRC peripheral.
0 = Scanner operations will not occur

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **MREG:** Scanner Memory Region Select bit(2)
1 = Scanner address points to Data EEPROM
0 = Scanner address points to Program Flash Memory

bit 1 **BURSTMD:** Scanner Burst Mode bit
1 = Memory access request to the CPU Arbiter is always true
0 = Memory access request to the CPU Arbiter is dependent on the CRC request and Trigger
Refer Table 16-2.

bit 0 **BUSY:** Scanner Busy Indicator bit
1 = Scanner cycle is in process
0 = Scanner cycle is compete (or never started)

Note 1: Setting EN = 1 (SCANCON0 register) does not affect any other register content.
2: Scanner trigger selection can be set using the SCANTRIG register.
3: This bit can be cleared in software. It is cleared in hardware when LADR>HADR (and a data cycle is not occurring) or when CRCGO = 0 (CRCCON0 register).
4: CRCEN and CRCGO bits (CRCCON0 register) must be set before setting the SGO bit.

REGISTER 17-6: DMAxSSAU – DMAx SOURCE START ADDRESS UPPER REGISTER

U-0	U-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
—	—	SSA<21:16>					
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 **SSA<21:16>:** Source Start Address bits

REGISTER 17-7: DMAxSPTRL – DMAx SOURCE POINTER LOW REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SPTR<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 15-0 **SPTR<7:0>:** Current Source Address Pointer

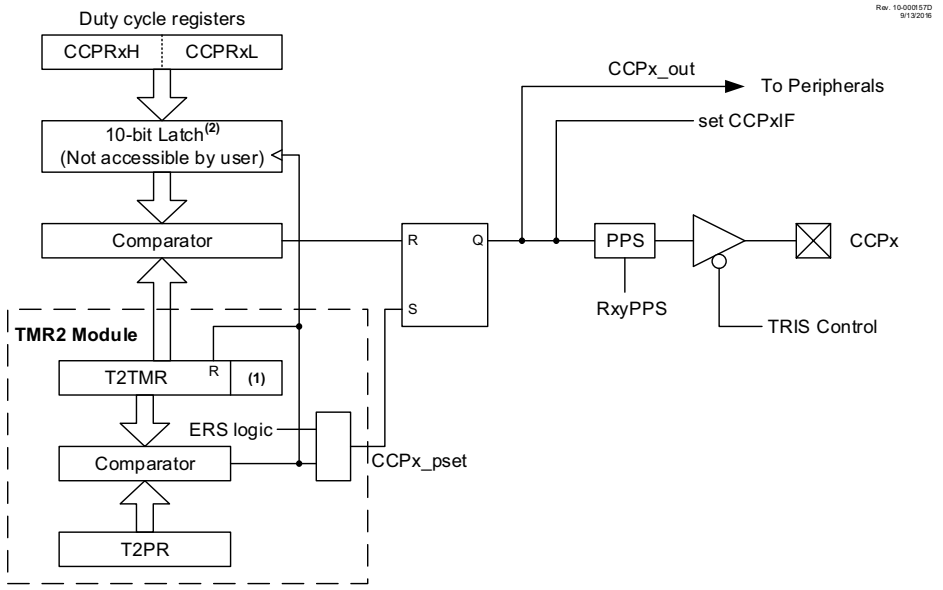
REGISTER 17-8: DMAxSPTRH – DMAx SOURCE POINTER HIGH REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SPTR<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 5-0 **SPTR<15:8>:** Current Source Address Pointer

FIGURE 25-4: SIMPLIFIED PWM BLOCK DIAGRAM



Notes: 1. 8-bit timer is concatenated with two bits generated by Fosc or two bits of the internal prescaler to create 10-bit time-base.
2. The alignment of the 10 bits from the CCPR register is determined by the CCPxFMT bit.

TABLE 25-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 20 MHz)

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescale	16	4	1	1	1	1
T2PR Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	6.6

TABLE 25-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 8 MHz)

PWM Frequency	1.22 kHz	4.90 kHz	19.61 kHz	76.92 kHz	153.85 kHz	200.0 kHz
Timer Prescale	16	4	1	1	1	1
T2PR Value	0x65	0x65	0x65	0x19	0x0C	0x09
Maximum Resolution (bits)	8	8	8	6	5	5

25.4.7 OPERATION IN SLEEP MODE

In Sleep mode, the T2TMR register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, T2TMR will continue from its previous state.

25.4.8 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency. Any changes in the system clock frequency will result in changes to the PWM frequency. See **Section 9.0 "Oscillator Module (with Fail-Safe Clock Monitor)"** for additional details.

25.4.9 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

27.6.2 GATED TIMER MODE

Gated Timer mode uses the SMTSIGx input to control whether or not the SMT1TMR will increment. Upon a falling edge of the external signal, the SMT1CPW register will update to the current value of the SMT1TMR. Example waveforms for both repeated and single acquisitions are provided in Figure 27-4 and Figure 27-5.

REGISTER 29-2: CLCxPOL: SIGNAL POLARITY CONTROL REGISTER

R/W-0/0	U-0	U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
POL	—	—	—	G4POL	G3POL	G2POL	G1POL
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 **POL:** CLCxOUT Output Polarity Control bit
1 = The output of the logic cell is inverted
0 = The output of the logic cell is not inverted

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **G4POL:** Gate 3 Output Polarity Control bit
1 = The output of gate 3 is inverted when applied to the logic cell
0 = The output of gate 3 is not inverted

bit 2 **G3POL:** Gate 2 Output Polarity Control bit
1 = The output of gate 2 is inverted when applied to the logic cell
0 = The output of gate 2 is not inverted

bit 1 **G2POL:** Gate 1 Output Polarity Control bit
1 = The output of gate 1 is inverted when applied to the logic cell
0 = The output of gate 1 is not inverted

bit 0 **G1POL:** Gate 0 Output Polarity Control bit
1 = The output of gate 0 is inverted when applied to the logic cell
0 = The output of gate 0 is not inverted

REGISTER 33-2: UxCON1: UART CONTROL REGISTER 1

R/W-0/0	U-0	U-0	R/W/HC-0/0	R/W-0/0	U-0	R/W-0/0	R/W/HC-0/0
ON	—	—	WUE	RXBIMD	—	BRKOVR	SENDB
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	HC = Hardware clear

bit 7 **ON:** Serial Port Enable bit
- 1 = Serial port enabled
- 0 = Serial port disabled (held in Reset)

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **WUE:** Wake-up Enable bit
- 1 = Receiver is waiting for falling RX input edge which will set the UxIF bit. Cleared by hardware on wake event. Also requires UxIE bit of PIEx to enable wake
- 0 = Receiver operates normally

bit 3 **RXBIMD:** Receive Break Interrupt Mode Select bit
- 1 = Set RXBKIF immediately when RX in has been low for the minimum Break time
- 0 = Set RXBKIF on rising RX input after RX in has been low for the minimum Break time

bit 2 **Unimplemented:** Read as '0'

bit 1 **BRKOVR:** Send Break Software Override bit
- 1 = TX output is forced to non-idle state
- 0 = TX output is driven by transmit shift register

bit 0 **SENDB:** Send Break Control bit(1)
- 1 = Output Break upon UxTXB write. Written byte follows Break. Bit is cleared by hardware.
- 0 = Break transmission completed or disabled

Note 1: This bit is read-only in LIN, DMX, and DALI modes.

TABLE 33-4: SUMMARY OF REGISTERS ASSOCIATED WITH THE UART

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
UxCON0	BRGS	ABDEN	TXEN	RXEN	MODE<3:0>				500
UxCON1	ON	—	—	WUE	RXBIMD	—	BRKOVR	SENDB	501
UxCON2	RUNOVF	RXPOL	STP<1:0>		C0EN	TXPOL	FLO<1:0>		502
UxERRIR	TXMTIF	PERIF	ABDOVF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF	503
UxERRIE	TXMTIE	PERIE	ABDOVE	CERIE	FERIE;	RXBKIE	RXFOIE	TXCIE	504
UxUIR	WUIF	ABDIF	—	—	—	ABDIE	—	—	505
UxFIFO	TXWRE	STPMD	TXBE	TXBF	RXIDL	XON	RXBE	RXBF	506
UxBRGL	BRG<7:0>								507
UxBRGH	BRG<15:8>								507
UxRXB	RXB<7:0>								508
UxTXB	TXB<7:0>								508
UxP1H	—	—	—	—	—	—	—	P1<8>	509
UxP1L	P1<7:0>								509
UxP2H	—	—	—	—	—	—	—	P2<8>	510
UxP2L	P2<7:0>								510
UxP3H	—	—	—	—	—	—	—	P3<8>	511
UxP3L	P3<7:0>								511
UxTXCHK	TXCHK<7:0>								512
UxRXCHK	RXCHK<7:0>								512

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the UART module.

34.3.3 TRANSMIT AND RECEIVE FIFOS

The transmission and reception of data from the SPI module is handled by two FIFOs, one for reception and one for transmission (addressed by the SFRs SPIxRXB and SPIxTXB, respectively.). The TXFIFO is written by software and is read by the SPI module to shift the data onto the SDO pin. The RXFIFO is written by the SPI module as it shifts in the data from the SDI pin and is read by software. Setting the CLRBF bit of SPIxSTATUS resets the occupancy for both FIFOs, emptying both buffers. The FIFOs are also reset by disabling the SPI module.

Note:	TXFIFO occupancy and RXFIFO occupancy simply refer to the number of bytes that are currently being stored in each FIFO. These values are used in this chapter to illustrate the function of these FIFOs and are not directly accessible through software.

The SPIxRXB register addresses the receive FIFO and is read-only. Reading from this register will read from the first FIFO location that was written to by hardware and decrease the RXFIFO occupancy. If the FIFO is empty, reading from this register will instead return a value of zero and set the RXRE (Receive Buffer Read Error) bit of the SPIxSTATUS register. The RXRE bit must then be cleared in software in order to properly reflect the status of the read error. When RXFIFO is full, the RXBF bit of the SPIxSTATUS register will be set. When the device receives data on the SDI pin, the receive FIFO may be written to by hardware and the occupancy increased, depending on the mode and receiver settings, as summarized in Table 34-1.

The SPIxTXB register addresses the transmit FIFO and is write-only. Writing to the register will write to the first empty FIFO location and increase the occupancy. If the FIFO is full, writing to this register will not affect the data and will set the TXWE bit of the SPIxSTATUS register. When the TXFIFO is empty, the TXBE bit of SPIxSTATUS will be set. When a data transfer occurs, data may be read from the first FIFO location written to and the occupancy decreases, depending on mode and transmitter settings, as summarized in Table 34-1 and **Section 34.6.1 "Slave Mode Transmit options"**.

34.3.4 LSB VS. MSB-FIRST OPERATION

Typically, SPI communication is output Most-Significant bit first, but some devices/buses may not conform to this standard. In this case, the LSBF bit may be used to alter the order in which bits are shifted out during the data exchange. In both Master and Slave mode, the LSBF bit of SPIxCON0 controls if data is shifted MSb or LSb first. Clearing the bit (default) configures the data to transfer MSb first, which is traditional SPI operation, while setting the bit configures the data to transfer LSb first.

34.3.5 INPUT AND OUTPUT POLARITY BITS

SPIxCON1 has three bits that control the polarity of the SPI inputs and outputs. The SDIP bit controls the polarity of the SDI input, the SDOP bit controls the polarity of the SDO output, and the SSP bit controls the polarity of both the slave $\overline{SS}$ input and the master SS output. For all three bits, when the bit is clear, the input or output is active-high, and when the bit is set, the input or output is active-low. When the EN bit of SPIxCON0 is cleared, SS(out) and SCK(out) both revert to the inactive state dictated by their polarity bits. The SDO output state when the EN bit of SPIxCON0 is cleared is determined by several factors.

- When the associated TRIS bit for the SDO pin is cleared, and the SPI goes Idle after a transmission, the SDO output will remain at the last bit level. The SDO pin will revert to the Idle state if EN is cleared.
- When the associated TRIS bit for the SDO pin is set, behavior varies in Slave and Master mode.
  - In Slave mode, the SDO pin tri-states when:
  - Slave Select is inactive,
  - the EN bit of SPIxCON0 is cleared, or when
  - the TXR bit of SPIxCON2 is cleared.
  - In Master mode, the SDO pin tri-states when TXR = `0`. When TXR = `1` and the SPI goes Idle after a transmission, the SDO output will remain at the last bit level. The SDO pin will revert to the Idle state if EN is cleared.

There are four main operations based on the direction of the data being shared during I²C communication.

- Master Transmit (master is transmitting data to a slave)
- Master Receive (master is receiving data from a slave)
- Slave Transmit (slave is transmitting data to a master)
- Slave Receive (slave is receiving data from the master)

To begin any I²C communication, the master device sends out a Start bit followed by the address byte of the slave it intends to communicate with. This is followed by a single Read/Write bit, which determines whether the master intends to transmit to or receive data from the slave device.

If the requested slave exists on the bus, it will respond with an Acknowledge bit, otherwise known as an ACK. The master then continues to shift data in or out of the slave until it terminates the message with a Stop.

Further details about the I²C module are discussed in Section 35.3, I2C Mode Operation.

35.3 I²C Mode Operation

All I²C communication is 8-bit data and 1-bit acknowledge and shifted out MSb first. The user can control the interaction between the software and the module using several control registers and interrupt flags. Two pins, SDA and SCL, are exercised by the module to communicate with other external I²C devices.

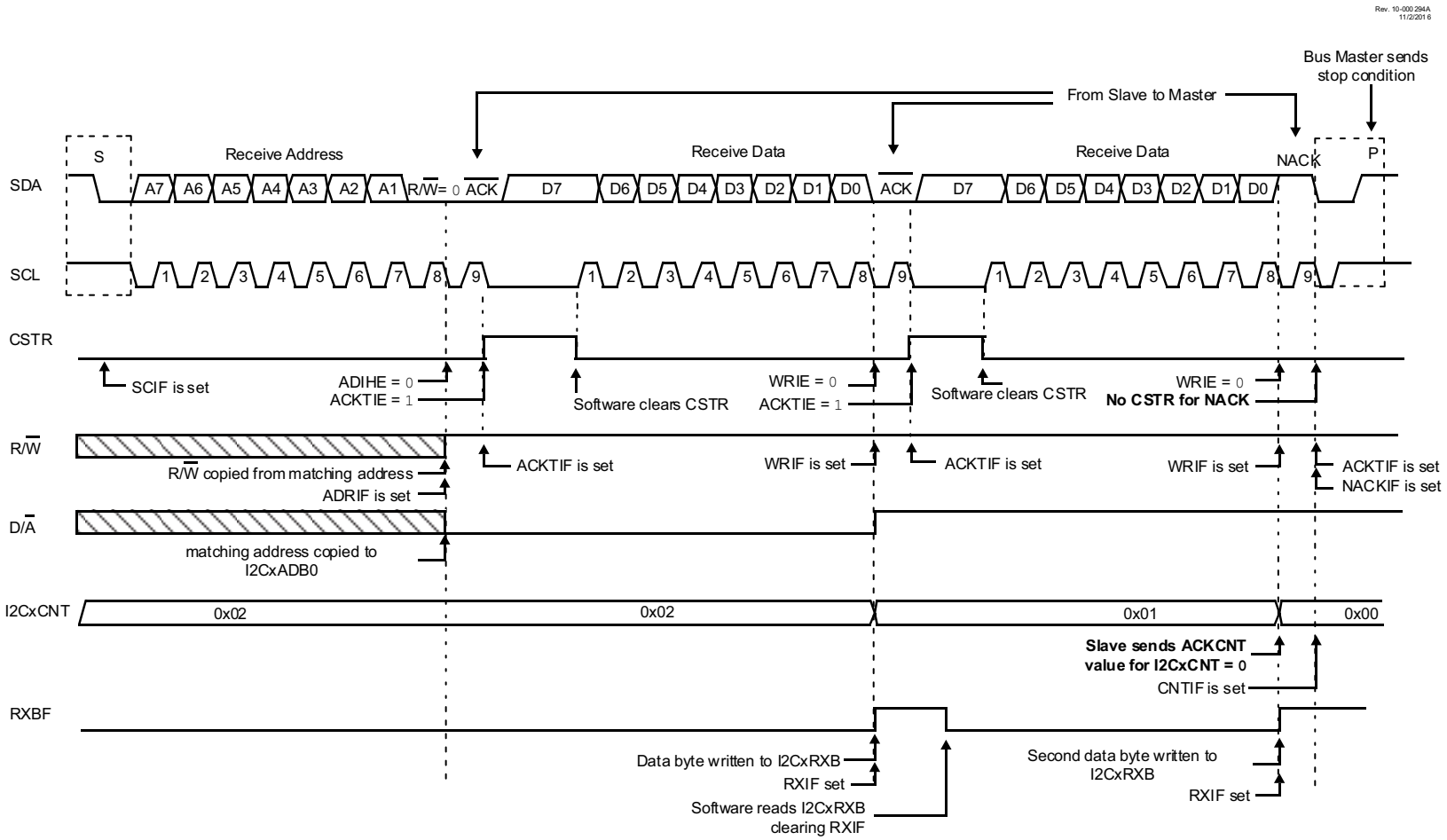
35.3.1 DEFINITION OF I²C TERMINOLOGY

The I²C communication protocol terminologies are defined for reference below in Table 35-1. These terminologies are used throughout this document. Table 35-1 has been adapted from the Phillips I²C specification.

TABLE 35-1: I²C BUS TERMS

TERM	Description
Transmitter	The device which shifts data out onto the bus
Receiver	The device which shifts data in from the bus
Master	The device that initiates a transfer, generates clock signals and terminates a transfer
Slave	The device addressed by the master
Multi-master	A bus with more than one device that can initiate data transfers
Arbitration	Procedure to ensure that only one master at a time controls the bus. Winning arbitration ensures that the message is not corrupted
Synchronization	Procedure to synchronize the clocks of two or more devices on the bus.
Idle	No master is controlling the bus, and both SDA and SCL lines are high
Active	Any time one or more master devices are controlling the bus
Addressed Slave	Slave device that has received a matching address and is actively being clocked by a master
Matching Address	Address byte that is clocked into a slave that matches the value stored in I2CxADR
Write Request	Slave receives a matching address with R/W bit clear and is ready to clock in data
Read Request	Master sends an address byte with the R/W bit set, indicating that it wishes to clock data out of the Slave. This data is the next and all following bytes until a Restart or Stop.
Clock Stretching	When a device on the bus holds SCL low to stall communication
Bus Collision	Any time the SDA line is sampled low by the module while it is outputting and expected high state.
Bus Timeout	Any time the I2CBTOISM input transitions high, the I²C module is reset and the module goes Idle.

FIGURE 35-7: I²C SLAVE, 7-BIT ADDRESS, RECEPTION WITH I2CxCNT (ACKTIE = 1, ADRIE = 0, WRIE = 0)



38.2.6 AUTO-CONVERSION TRIGGER

The Auto-conversion Trigger allows periodic ADC measurements without software intervention. When a rising edge of the selected source occurs, the GO bit is set by hardware.

The Auto-conversion Trigger source is selected by the ADACT register.

Using the Auto-conversion Trigger does not assure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met. See Register 38-33 for auto-conversion sources.

38.2.7 ADC CONVERSION PROCEDURE (BASIC MODE)

This is an example procedure for using the ADC to perform an Analog-to-Digital conversion:

1. Configure Port:
   - Disable pin output driver (Refer to the TRISx register)
   - Configure pin as analog (Refer to the ANSELx register)
2. Configure the ADC module:
   - Select ADC conversion clock
   - Select voltage reference
   - Select ADC input channel
   - Precharge and acquisition
   - Turn on ADC module
3. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable global interrupt (GIEL bit)**(1)**
4. If ADACQ = 0, software must wait the required acquisition time**(2)**.
5. Start conversion by setting the GO bit.
6. Wait for ADC conversion to complete by one of the following:
   - Polling the GO bit
   - Polling the ADIF bit
   - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result.
8. Clear the ADC interrupt flag (required if interrupt is enabled).

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

2: Refer to **Section 38.3 "ADC Acquisition Requirements"**.

EXAMPLE 38-1: ADC CONVERSION

```
/*This code block configures the ADC
for polling, VDD and VSS references, FRC
oscillator and AN0 input.
Conversion start & polling for completion
are included.
 */
void main() {
    //System Initialize
    initializeSystem();

    //Setup ADC
    ADCON0bits.FM = 1; //right justify
    ADCON0bits.CS = 1; //FRC Clock
    ADPCH = 0x00; //RA0 is Analog channel
    TRISAbits.TRISA0 = 1; //Set RA0 to input
    ANSELAbits.ANSELA0 = 1; //Set RA0 to analog
    ADCON0bits.ON = 1; //Turn ADC On

    while (1) {
        ADCON0bits.GO = 1; //Start conversion
        while (ADCON0bits.GO); //Wait for conversion done
        resultHigh = ADRESH; //Read result
        resultLow = ADRESL; //Read result
    }
}
```

REGISTER 38-20: ADRESH: ADC RESULT REGISTER HIGH, FM = 1

U-0	U-0	U-0	U-0	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	—	—	ADRES<11:8>			
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 7-4 **Reserved**

bit 3-0 **ADRES<11:8>**: ADC Sample Result bits. Upper four bits of 12-bit conversion result.

REGISTER 38-21: ADRESL: ADC RESULT REGISTER LOW, FM = 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
ADRES<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 **ADRES<7:0>**: ADC Result Register bits. Lower eight bits of 12-bit conversion result.

REGISTER 38-29: ADERRH: ADC SETPOINT ERROR REGISTER HIGH

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ERR<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 **ERR<15:8>**: ADC Setpoint Error MSB. Upper byte of ADC Setpoint Error. Setpoint Error calculation is determined by ADCALC bits of ADCON3, see Register 31-1 for more details.

REGISTER 38-30: ADERRL: ADC SETPOINT ERROR LOW BYTE REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ERR<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 **ERR<7:0>**: ADC Setpoint Error LSB. Lower byte of ADC Setpoint Error calculation is determined by ADCALC bits of ADCON3, see Register 31-1 for more details.

REGISTER 38-31: ADLTHH: ADC LOWER THRESHOLD HIGH BYTE 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
LTH<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 **LTH<15:8>**: ADC Lower Threshold MSB. LTH and UTH are compared with ERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

40.3 Comparator Hysteresis

A selectable amount of separation voltage can be added to the input pins of each comparator to provide a hysteresis function to the overall operation. Hysteresis is enabled by setting the HYS bit of the CMxCON0 register.

See Comparator Specifications in Table 46-15 for more information.

40.3.1 COMPARATOR OUTPUT SYNCHRONIZATION

The output from a comparator can be synchronized with Timer1 by setting the SYNC bit of the CMxCON0 register.

Once enabled, the comparator output is latched on the falling edge of the Timer1 source clock. If a prescaler is used, the CxOUT bit is synchronized with the timer, so that the software sees no ambiguity due to timing. See the Comparator Block Diagram (Figure 40-2) and the Timer1 Block Diagram (Figure 23-1) for more information.

40.4 Comparator Interrupt

An interrupt can be generated for every rising or falling edge of the comparator output.

When either edge detector is triggered and its associated enable bit is set (INTP and/or INTN bits of the CMxCON1 register), the Corresponding Interrupt Flag bit (CxIF bit of the respective PIR register) will be set.

To enable the interrupt, you must set the following bits:

- EN bit of the CMxCON0 register
- CxIE bit of the respective PIE register
- INTP bit of the CMxCON1 register (for a rising edge detection)
- INTN bit of the CMxCON1 register (for a falling edge detection)
- GIE bit of the INTCON0 register

The associated interrupt flag bit, CxIF bit of the respective PIR register, must be cleared in software. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

Note:	Although a comparator is disabled, an interrupt can be generated by changing the output polarity with the POL bit of the CMxCON0 register, or by switching the comparator on or off with the EN bit of the CMxCON0 register.

40.5 Comparator Positive Input Selection

Configuring the PCH<2:0> bits of the CMxPCH register directs an internal voltage reference or an analog pin to the non-inverting input of the comparator:

- CxIN0+, CxIN1+ analog pin
- DAC output
- FVR (Fixed Voltage Reference)
- VSS (Ground)

See **Section 36.0 "Fixed Voltage Reference (FVR)"** for more information on the Fixed Voltage Reference module.

See **Section 39.0 "5-Bit Digital-to-Analog Converter (DAC) Module"** for more information on the DAC input signal.

Any time the comparator is disabled (EN = 0), all comparator inputs are disabled.

40.6 Comparator Negative Input Selection

The NCH<2:0> bits of the CMxNCH register direct an analog input pin and internal reference voltage or analog ground to the inverting input of the comparator:

- CxIN0-, CxIN1-, CxIN2-, CxIN3- analog pin
- FVR (Fixed Voltage Reference)
- Analog Ground

Note:	To use CxINy+ and CxINy- pins as analog input, the appropriate bits must be set in the ANSEL register and the corresponding TRIS bits must also be set to disable the output drivers.

43.0 INSTRUCTION SET SUMMARY

PIC18(L)F2x/4xK42 devices incorporate the standard set of PIC18 core instructions, as well as an extended set of instructions, for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

43.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC® MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are four instructions that require two-program memory locations and two that require three-program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18 instruction set summary in Table 43-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 43-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The destination of the result (specified by ‘d’)
3. The accessed memory (specified by ‘a’)

The file register designator ‘f’ specifies which file register is to be used by the instruction. The destination designator ‘d’ specifies where the result of the operation is to be placed. If ‘d’ is zero, the result is placed in the WREG register. If ‘d’ is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The bit in the file register (specified by ‘b’)
3. The accessed memory (specified by ‘a’)

The bit field designator ‘b’ selects the number of the bit affected by the operation, while the file register designator ‘f’ represents the number of the file in which the bit is located.

The literal instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by ‘k’)
- The desired FSR register to load the literal value into (specified by ‘f’)
- No operand required (specified by ‘—’)

The control instructions may use some of the following operands:

- A program memory address (specified by ‘n’)
- The mode of the CALL or RETURN instructions (specified by ‘s’)
- The mode of the table read and table write instructions (specified by ‘m’)
- No operand required (specified by ‘—’)

All instructions are a single word, except for four double-word instructions. These instructions were made double-word to contain the required information in 32 bits. In the second word, the four MSbs are ‘1’s. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μs. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 μs. Two-word branch instructions (if true) would take 3 μs.

Figure 43-1 shows the general formats that the instructions can have. All examples use the convention ‘nnh’ to represent a hexadecimal number.

The Instruction Set Summary, shown in Table 43-2, lists the standard instructions recognized by the Microchip Assembler (MPASM™).

Section 43.1.1 “Standard Instruction Set” provides a description of each instruction.

TABLE 44-1: 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	Register on page
3F5Eh	CCPTMRS0	C4TSEL		C3TSEL		C2TSEL		C1TSEL		362
3F5Dh - 3F5Bh	—	Unimplemented								
3F5Ah	CWG1STR	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA	429
3F59h	CWG1AS1	—	AS6E	AS5E	AS4E	AS3E	AS2E	AS1E	AS0E	433
3F58h	CWG1AS0	SHUTDOWN	REN	LSBD		LSAC		—	—	430
3F57h	CWG1CON1	—	—	IN	—	POLD	POLC	POLB	POLA	428
3F56h	CWG1CON0	EN	LD	—	—	—	MODE			427
3F55h	CWG1DBF	—	—	DBF						434
3F54h	CWG1DBR	—	—	DBR						434
3F53h	CWG1ISM	—	—	—	—	IS				430
3F52h	CWG1CLK	—	—	—	—	—	—	—	CS	429
3F51h	CWG2STR	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA	429
3F50h	CWG2AS1	—	AS6E	AS5E	AS4E	AS3E	AS2E	AS1E	AS0E	433
3F4Fh	CWG2AS0	SHUTDOWN	REN	LSBD		LSAC		—	—	430
3F4Eh	CWG2CON1	—	—	IN	—	POLD	POLC	POLB	POLA	428
3F4Dh	CWG2CON0	EN	LD	—	—	—	MODE			427
3F4Ch	CWG2DBF	—	—	DBF						434
3F4Bh	CWG2DBR	—	—	DBR						434
3F4Ah	CWG2ISM	—	—	—	—	IS				430
3F49h	CWG2CLK	—	—	—	—	—	—	—	CS	429
3F48h	CWG3STR	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA	429
3F47h	CWG3AS1	—	AS6E	AS5E	AS4E	AS3E	AS2E	AS1E	AS0E	433
3F46h	CWG3AS0	SHUTDOWN	REN	LSBD		LSAC		—	—	430
3F45h	CWG3CON1	—	—	IN	—	POLD	POLC	POLB	POLA	428
3F44h	CWG3CON0	EN	LD	—	—	—	MODE			427
3F43h	CWG3DBF	—	—	DBF						434
3F42h	CWG3DBR	—	—	DBR						434
3F41h	CWG3ISM	—	—	—	—	IS				430
3F40h	CWG3CLK	—	—	—	—	—	—	—	CS	429
3F3Fh	NCO1CLK	PWS			—	CKS				457
3F3Eh	NCO1CON	EN	—	OUT	POL	—	—	—	PFM	456
3F3Dh	NCO1INCU	INC								460
3F3Ch	NCO1INCH	INC								459
3F3Bh	NCO1INCL	INC								459
3F3Ah	NCO1ACCU	ACC								459
3F39h	NCO1ACCH	ACC								458
3F38h	NCO1ACCL	ACC								458
3F37h - 3F24h	—	Unimplemented								
3F23h	SMT1WIN	—	—	—	WSEL					401
3F22h	SMT1SIG	—	—	—	SSEL					402
3F21h	SMT1CLK	—	—	—	—	—	CSEL			400
3F20h	SMT1STAT	CPRUP	CPWUP	RST	—	—	TS	WS	AS	399
3F1Fh	SMT1CON1	GO	REPEAT	—	—	MODE				398
3F1Eh	SMT1CON0	EN	—	STP	WPOL	SPOL	CPOL	PS		397
3F1Dh	SMT1PRU	PR								406
3F1Ch	SMT1PRH	PR								406
3F1Bh	SMT1PRL	PR								406

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

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