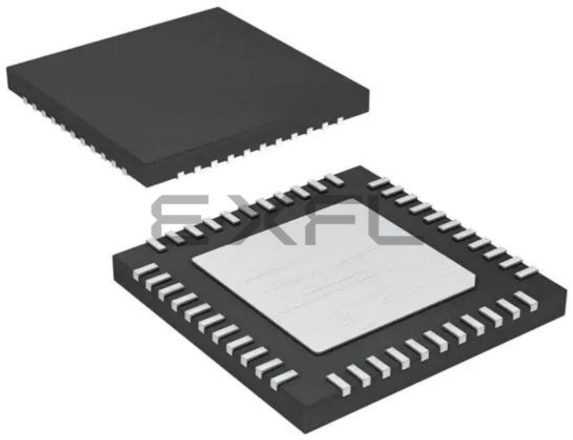


Welcome to E-XFL.COM

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	25MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f4580-e-ml

TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices				Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets, WDT Reset, `RESET` Instruction, Stack Resets	Wake-up via WDT or Interrupt
B0EIDL[6]	2480	2580	4480	4580	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
B0EIDH[6]	2480	2580	4480	4580	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
B0SIDL[6]	2480	2580	4480	4580	`xxxx x-xx`	`uuuu u-uu`	`uuuu u-uu`
B0SIDH[6]	2480	2580	4480	4580	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
B0CON[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
TXBIE[6]	2480	2580	4480	4580	`---0 00--`	`---u uu--`	`---u uu--`
BIE0[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
BSEL0[6]	2480	2580	4480	4580	`0000 00--`	`0000 00--`	`uuuu uu--`
MSEL3[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
MSEL2[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
MSEL1[6]	2480	2580	4480	4580	`0000 0101`	`0000 0101`	`uuuu uuuu`
MSEL0[6]	2480	2580	4480	4580	`0101 0000`	`0101 0000`	`uuuu uuuu`
SDFLC[6]	2480	2580	4480	4580	`---0 0000`	`---0 0000`	`-u-- uuuu`
RXFCON1[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
RXFCON0[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
RXFBCON7[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
RXFBCON6[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
RXFBCON5[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
RXFBCON4[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
RXFBCON3[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
RXFBCON2[6]	2480	2580	4480	4580	`0001 0001`	`0001 0001`	`uuuu uuuu`
RXFBCON1[6]	2480	2580	4480	4580	`0001 0001`	`0001 0001`	`uuuu uuuu`
RXFBCON0[6]	2480	2580	4480	4580	`0000 0000`	`0000 0000`	`uuuu uuuu`
RXF15EIDL[6]	2480	2580	4480	4580	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXF15EIDH[6]	2480	2580	4480	4580	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXF15SIDL[6]	2480	2580	4480	4580	`xxx- x-xx`	`uuu- u-uu`	`uuu- u-uu`
RXF15SIDH[6]	2480	2580	4480	4580	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXF14EIDL[6]	2480	2580	4480	4580	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXF14EIDH[6]	2480	2580	4480	4580	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
RXF14SIDL[6]	2480	2580	4480	4580	`xxx- x-xx`	`uuu- u-uu`	`uuu- u-uu`
RXF14SIDH[6]	2480	2580	4480	4580	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`

Legend: `u` = unchanged, `x` = unknown, `-` = unimplemented bit, read as '`0`', `q` = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 5-3 for Reset value for specific condition.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '`0`'.

6: This register reads all '`0`'s until ECAN™ technology is set up in Mode 1 or Mode 2.

TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2480/2580/4480/4580 DEVICES (CONTINUED)

Address	Name	Address	Name	Address	Name	Address	Name
F7Fh	—	F5Fh	CANCON_RO0	F3Fh	CANCON_RO2	F1Fh	RXM1EIDL
F7Eh	—	F5Eh	CANSTAT_RO0	F3Eh	CANSTAT_RO2	F1Eh	RXM1EIDH
F7Dh	—	F5Dh	RXB1D7	F3Dh	TXB1D7	F1Dh	RXM1SIDL
F7Ch	—	F5Ch	RXB1D6	F3Ch	TXB1D6	F1Ch	RXM1SIDH
F7Bh	—	F5Bh	RXB1D5	F3Bh	TXB1D5	F1Bh	RXM0EIDL
F7Ah	—	F5Ah	RXB1D4	F3Ah	TXB1D4	F1Ah	RXM0EIDH
F79h	—	F59h	RXB1D3	F39h	TXB1D3	F19h	RXM0SIDL
F78h	—	F58h	RXB1D2	F38h	TXB1D2	F18h	RXM0SIDH
F77h	ECANCON	F57h	RXB1D1	F37h	TXB1D1	F17h	RXF5EIDL
F76h	TXERRCNT	F56h	RXB1D0	F36h	TXB1D0	F16h	RXF5EIDH
F75h	RXERRCNT	F55h	RXB1DLC	F35h	TXB1DLC	F15h	RXF5SIDL
F74h	COMSTAT	F54h	RXB1EIDL	F34h	TXB1EIDL	F14h	RXF5SIDH
F73h	CIOCON	F53h	RXB1EIDH	F33h	TXB1EIDH	F13h	RXF4EIDL
F72h	BRGCON3	F52h	RXB1SIDL	F32h	TXB1SIDL	F12h	RXF4EIDH
F71h	BRGCON2	F51h	RXB1SIDH	F31h	TXB1SIDH	F11h	RXF4SIDL
F70h	BRGCON1	F50h	RXB1CON	F30h	TXB1CON	F10h	RXF4SIDH
F6Fh	CANCON	F4Fh	CANCON_RO1	F2Fh	CANCON_RO3	F0Fh	RXF3EIDL
F6Eh	CANSTAT	F4Eh	CANSTAT_RO1	F2Eh	CANSTAT_RO3	F0Eh	RXF3EIDH
F6Dh	RXB0D7	F4DH	TXB0D7	F2Dh	TXB2D7	F0Dh	RXF3SIDL
F6Ch	RXB0D6	F4Ch	TXB0D6	F2Ch	TXB2D6	F0Ch	RXF3SIDH
F6Bh	RXB0D5	F4Bh	TXB0D5	F2Bh	TXB2D5	F0Bh	RXF2EIDL
F6Ah	RXB0D4	F4Ah	TXB0D4	F2Ah	TXB2D4	F0Ah	RXF2EIDH
F69h	RXB0D3	F49h	TXB0D3	F29h	TXB2D3	F09h	RXF2SIDL
F68h	RXB0D2	F48h	TXB0D2	F28h	TXB2D2	F08h	RXF2SIDH
F67h	RXB0D1	F47h	TXB0D1	F27h	TXB2D1	F07h	RXF1EIDL
F66h	RXB0D0	F46h	TXB0D0	F26h	TXB2D0	F06h	RXF1EIDH
F65h	RXB0DLC	F45h	TXB0DLC	F25h	TXB2DLC	F05h	RXF1SIDL
F64h	RXB0EIDL	F44h	TXB0EIDL	F24h	TXB2EIDL	F04h	RXF1SIDH
F63h	RXB0EIDH	F43h	TXB0EIDH	F23h	TXB2EIDH	F03h	RXF0EIDL
F62h	RXB0SIDL	F42h	TXB0SIDL	F22h	TXB2SIDL	F02h	RXF0EIDH
F61h	RXB0SIDH	F41h	TXB0SIDH	F21h	TXB2SIDH	F01h	RXF0SIDL
F60h	RXB0CON	F40h	TXB0CON	F20h	TXB2CON	F00h	RXF0SIDH

Note 1: Registers available only on PIC18F4X80 devices; otherwise, the registers read as '0'.
2: When any TX_ENn bit in RX_TX_SELn is set, then the corresponding bit in this register has transmit properties.
3: This is not a physical register.

6.6.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

The use of Indexed Literal Offset Addressing mode effectively changes how the lower half of Access RAM (00h to 7Fh) is mapped. Rather than containing just the contents of the bottom half of Bank 0, this mode maps the contents from Bank 0 and a user-defined "window" that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see **Section 6.3.2 "Access Bank"**). An example of Access Bank remapping in this addressing mode is shown in Figure 6-10.

Remapping of the Access Bank applies *only* to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit is '1') will continue to use Direct Addressing as before. Any indirect or indexed operation that explicitly uses any of the indirect file operands (including FSR2) will continue to operate as standard Indirect Addressing. Any instruction that uses the Access Bank, but includes a register address of greater than 05Fh, will use Direct Addressing and the normal Access Bank map.

6.6.4 BSR IN INDEXED LITERAL OFFSET MODE

Although the Access Bank is remapped when the extended instruction set is enabled, the operation of the BSR remains unchanged. Direct Addressing using the BSR to select the data memory bank operates in the same manner as previously described.

FIGURE 6-10: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING

Example Situation:

`ADDWF f, d, a`

FSR2H:FSR2L = 120h

Locations in the region from the FSR2 Pointer (120h) to the pointer plus 05Fh (17Fh) are mapped to the bottom of the Access RAM (000h-05Fh).

Special File Registers at F60h through FFFh are mapped to 60h through FFh, as usual.

Bank 0 addresses below 5Fh are not available in this mode. They can still be addressed by using the BSR.

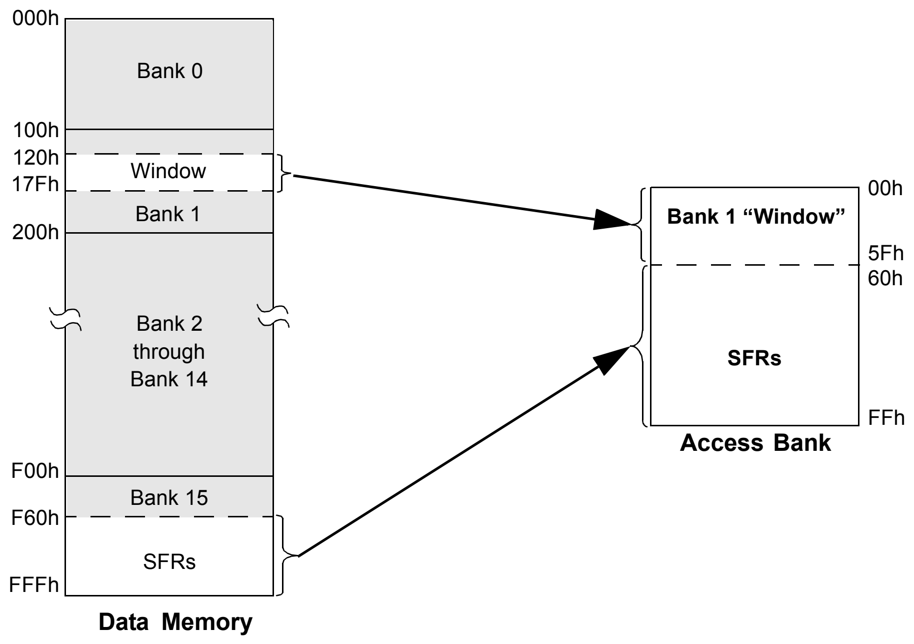
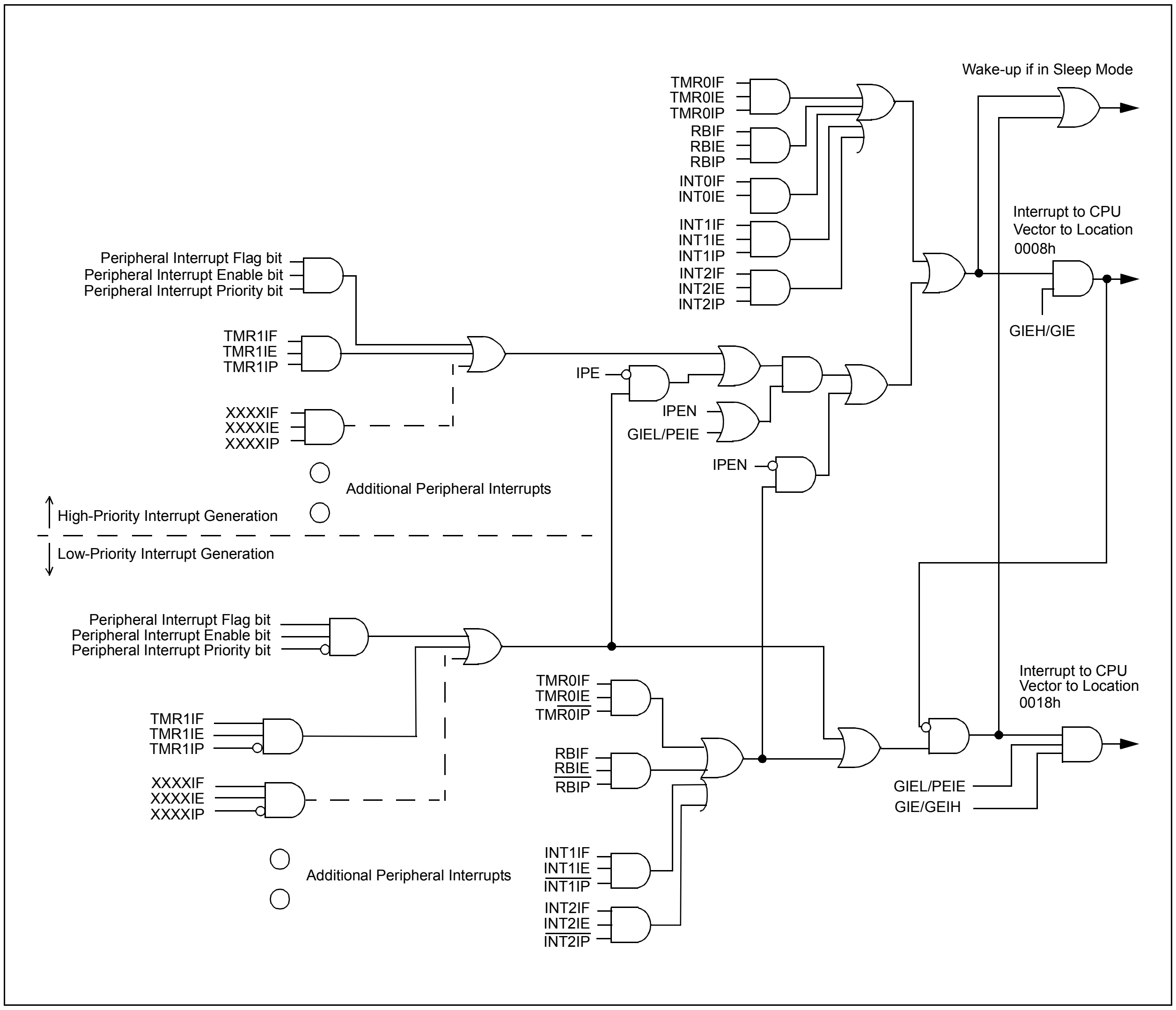


FIGURE 10-1: INTERRUPT LOGIC


TMR0IF
TMR0IE
TMR0IP
RBIF
RBIE
RBIP
INT0IF
INT0IE
INT1IF
INT1IE
INT1IP
INT2IF
INT2IE
INT2IP
Wake-up if in Sleep Mode
Interrupt to CPU Vector to Location 0008h
GIEH/GIE
Peripheral Interrupt Flag bit
Peripheral Interrupt Enable bit
Peripheral Interrupt Priority bit
TMR1IF
TMR1IE
TMR1IP
XXXXIF
XXXXIE
XXXXIP
IPE
IPEN
GIEL/PEIE
IPEN
Additional Peripheral Interrupts
High-Priority Interrupt Generation
Low-Priority Interrupt Generation
Peripheral Interrupt Flag bit
Peripheral Interrupt Enable bit
Peripheral Interrupt Priority bit
TMR1IF
TMR1IE
TMR1IP
XXXXIF
XXXXIE
XXXXIP
Additional Peripheral Interrupts
TMR0IF
TMR0IE
TMR0IP
RBIF
RBIE
RBIP
INT1IF
INT1IE
INT1IP
INT2IF
INT2IE
INT2IP
Interrupt to CPU Vector to Location 0018h
GIEL/PEIE
GIE/GEIH

REGISTER 10-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

Mode 0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IRXIE	WAKIE	ERRIE	TXB2IE	TXB1IE(1)	TXB0IE(1)	RXB1IE	RXB0IE

Mode 1,2	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IRXIE	WAKIE	ERRIE	TXBnIE	TXB1IE(1)	TXB0IE(1)	RXBnIE	FIFOWMIE(1)
	bit 7							bit 0

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

bit 7 **IRXIE:** CAN Invalid Received Message Interrupt Enable bit
1 = Enable invalid message received interrupt
0 = Disable invalid message received interrupt

bit 6 **WAKIE:** CAN bus Activity Wake-up Interrupt Enable bit
1 = Enable bus activity wake-up interrupt
0 = Disable bus activity wake-up interrupt

bit 5 **ERRIE:** CAN bus Error Interrupt Enable bit
1 = Enable CAN bus error interrupt
0 = Disable CAN bus error interrupt

bit 4 When CAN is in Mode 0:
TXB2IE: CAN Transmit Buffer 2 Interrupt Enable bit
1 = Enable Transmit Buffer 2 interrupt
0 = Disable Transmit Buffer 2 interrupt
When CAN is in Mode 1 or 2:
TXBnIE: CAN Transmit Buffer Interrupts Enable bit
1 = Enable transmit buffer interrupt; individual interrupt is enabled by TXBIE and BIE0
0 = Disable all transmit buffer interrupts

bit 3 **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 1 interrupt
0 = Disable Transmit Buffer 1 interrupt

bit 2 **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 0 interrupt
0 = Disable Transmit Buffer 0 interrupt

bit 1 When CAN is in Mode 0:
RXB1IE: CAN Receive Buffer 1 Interrupt Enable bit
1 = Enable Receive Buffer 1 interrupt
0 = Disable Receive Buffer 1 interrupt
When CAN is in Mode 1 or 2:
RXBnIE: CAN Receive Buffer Interrupts Enable bit
1 = Enable receive buffer interrupt; individual interrupt is enabled by BIE0
0 = Disable all receive buffer interrupts

bit 0 When CAN is in Mode 0:
RXB0IE: CAN Receive Buffer 0 Interrupt Enable bit
1 = Enable Receive Buffer 0 interrupt
0 = Disable Receive Buffer 0 interrupt
When CAN is in Mode 1:
Unimplemented: Read as ‘0’
When CAN is in Mode 2:
FIFOWMIE: FIFO Watermark Interrupt Enable bit(1)
1 = Enable FIFO watermark interrupt
0 = Disable FIFO watermark interrupt

Note 1: In CAN Mode 1 and 2, these bits are forced to ‘0’.

TABLE 11-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
PORTD(1)	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	58
LATD(1)	LATD Output Latch Register								58
TRISD(1)	PORTD Data Direction Register								58
TRISE(1)	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	58
ECCP1CON(1)	EPWM1M1	EPWM1M0	EDC1B1	EDC1B0	ECCP1M3	ECCP1M2	ECCP1M1	ECCP1M0	57

Legend: — = unimplemented, read as '0'. Shaded cells are not used by PORTD.

Note 1: These registers are available on PIC18F4X80 devices only.

15.2 Timer3 16-Bit Read/Write Mode

Timer3 can be configured for 16-bit reads and writes (see Figure 15-2). When the RD16 control bit (T3CON<7>) is set, the address for TMR3H is mapped to a buffer register for the high byte of Timer3. A read from TMR3L will load the contents of the high byte of Timer3 into the Timer3 High Byte Buffer register. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, has become invalid due to a rollover between reads.

A write to the high byte of Timer3 must also take place through the TMR3H Buffer register. The Timer3 high byte is updated with the contents of TMR3H when a write occurs to TMR3L. This allows a user to write all 16 bits to both the high and low bytes of Timer3 at once.

The high byte of Timer3 is not directly readable or writable in this mode. All reads and writes must take place through the Timer3 High Byte Buffer register.

Writes to TMR3H do not clear the Timer3 prescaler. The prescaler is only cleared on writes to TMR3L.

15.3 Using the Timer1 Oscillator as the Timer3 Clock Source

The Timer1 internal oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN (T1CON<3>) bit. To use it as the Timer3 clock source, the TMR3CS bit must also be set. As previously noted, this also configures Timer3 to increment on every rising edge of the oscillator source.

The Timer1 oscillator is described in **Section 13.0 "Timer1 Module"**.

15.4 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and overflows to 0000h. The Timer3 interrupt, if enabled, is generated on overflow and is latched in the interrupt flag bit, TMR3IF (PIR2<1>). This interrupt can be enabled or disabled by setting or clearing the Timer3 Interrupt Enable bit, TMR3IE (PIE2<1>).

15.5 Resetting Timer3 Using the CCP Special Event Trigger

If the ECCP1 module is configured to generate a special event trigger in Compare mode (ECCP1M<3:0> = `1011`), this signal will reset Timer3. It will also start an A/D conversion if the A/D module is enabled (see **Section 16.3.4 "Special Event Trigger"** for more information.).

The module must be configured as either a timer or synchronous counter to take advantage of this feature. When used this way, the ECCPR2H:ECCPR2L register pair effectively becomes a period register for Timer3.

If Timer3 is running in Asynchronous Counter mode, the Reset operation may not work.

In the event that a write to Timer3 coincides with a Special Event Trigger from a CCP module, the write will take precedence.

Note:	The special event triggers from the ECCP1 module will not set the TMR3IF interrupt flag bit (PIR1<0>).

TABLE 15-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	55
PIR2	OSCFIF	CMIF(2)	—	EEIF	BCLIF	HLVDIF	TMR3IF	ECCP1IF(2)	58
PIE2	OSCFIE	CMIE(2)	—	EEIE	BCLIE	HLVDIE	TMR3IE	ECCP1IE(2)	58
IPR2	OSCFIP	CMIP(2)	—	EEIP	BCLIP	HLVDIP	TMR3IP	ECCP1IP(2)	57
TMR3L	Timer3 Register Low Byte								57
TMR3H	Timer3 Register High Byte								57
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	56
T3CON	RD16	T3ECCP1(1)	T3CKPS1	T3CKPS0	T3CCP1(1)	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	57

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer3 module.
Note 1: These bits are available in PIC18F4X80 devices only.
2: These bits are available in PIC18F4X80 devices and reserved in PIC18F2X80 devices.

REGISTER 19-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

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

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don’t care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don’t care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-bit (RX9 = 0):
Don’t care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be an address/data bit or a parity bit and must be calculated by user firmware.

19.2.5 BREAK CHARACTER SEQUENCE

The Enhanced EUSART module has the capability of sending the special Break character sequences that are required by the LIN/J2602 bus standard. The Break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The Frame Break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN/J2602 specification).

Note that the data value written to the TXREG for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 19-10 for the timing of the Break character sequence.

19.2.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a Break, followed by an Auto-Baud Sync byte. This sequence is typical of a LIN/J2602 bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

19.2.6 RECEIVING A BREAK CHARACTER

The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 19.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABD bit once the TXIF interrupt is observed.

FIGURE 19-10: SEND BREAK CHARACTER SEQUENCE

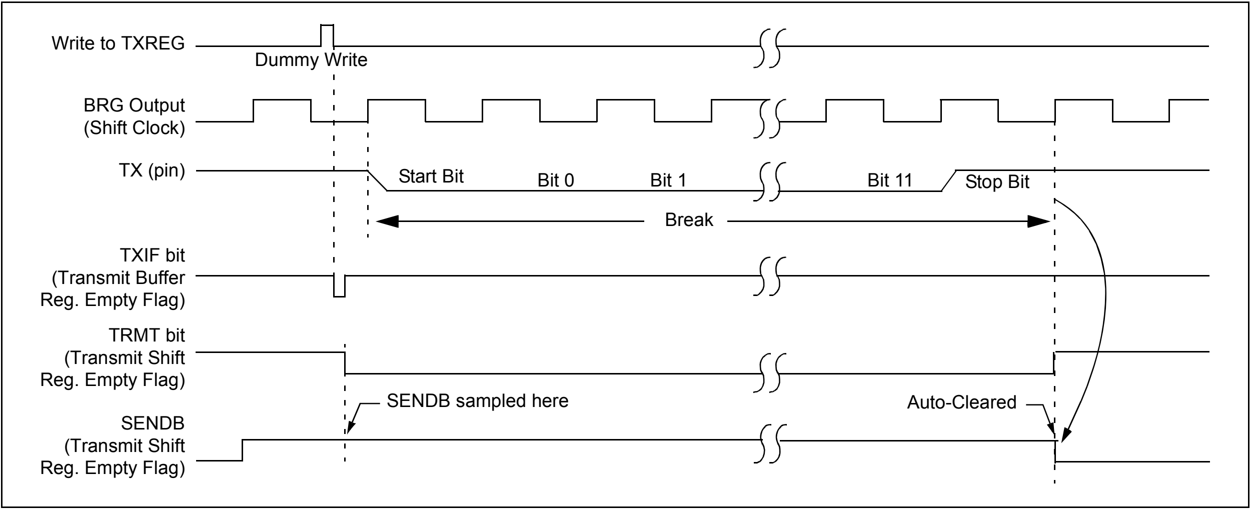


FIGURE 19-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

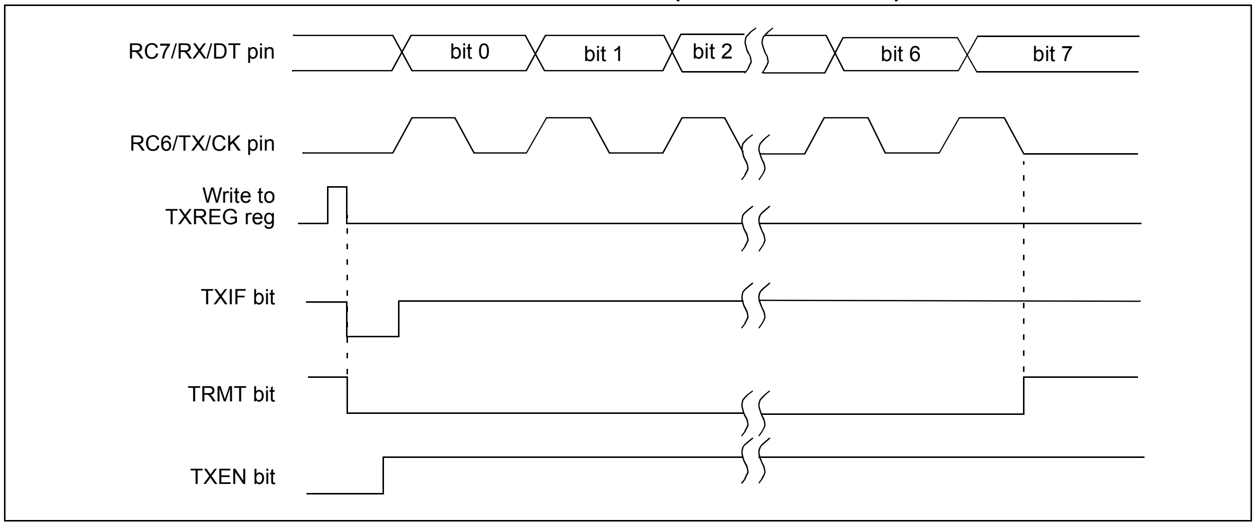


TABLE 19-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	55
PIR1	PSPIF[(1)]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	58
PIE1	PSPIE[(1)]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	58
IPR1	PSPIP[(1)]	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	58
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	57
TXREG	EUSART Transmit Register								57
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	57
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	57
SPBRGH	EUSART Baud Rate Generator Register, High Byte								57
SPBRG	EUSART Baud Rate Generator Register, Low Byte								57

Legend: — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

Note 1: Reserved in PIC18F2X80 devices; always maintain these bits clear.

24.2 CAN Module Registers

Note: Not all CAN registers are available in the Access Bank.

There are many control and data registers associated with the CAN module. For convenience, their descriptions have been grouped into the following sections:

- Control and Status Registers
- Dedicated Transmit Buffer Registers
- Dedicated Receive Buffer Registers
- Programmable TX/RX and Auto RTR Buffers
- Baud Rate Control Registers
- I/O Control Register
- Interrupt Status and Control Registers

Detailed descriptions of each register and their usage are described in the following sections.

24.2.1 CAN CONTROL AND STATUS REGISTERS

The registers described in this section control the overall operation of the CAN module and show its operational status.

REGISTER 24-54: BRGCON3: BAUD RATE CONTROL REGISTER 3

R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
WAKDIS	WAKFIL	—	—	—	SEG2PH2(1)	SEG2PH1(1)	SEG2PH0(1)
bit 7							bit 0

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

bit 7	**WAKDIS:** Wake-up Disable bit 1 = Disable CAN bus activity wake-up feature 0 = Enable CAN bus activity wake-up feature
bit 6	**WAKFIL:** Selects CAN bus Line Filter for Wake-up bit 1 = Use CAN bus line filter for wake-up 0 = CAN bus line filter is not used for wake-up
bit 5-3	**Unimplemented:** Read as ‘0’
bit 2-0	**SEG2PH<2:0>:** Phase Segment 2 Time Select bits(1) 111 = Phase Segment 2 time = 8 x TQ 110 = Phase Segment 2 time = 7 x TQ 101 = Phase Segment 2 time = 6 x TQ 100 = Phase Segment 2 time = 5 x TQ 011 = Phase Segment 2 time = 4 x TQ 010 = Phase Segment 2 time = 3 x TQ 001 = Phase Segment 2 time = 2 x TQ 000 = Phase Segment 2 time = 1 x TQ

Note 1: Ignored if SEG2PHTS bit (BRGCON2<7>) is ‘0’.

24.2.5 CAN MODULE I/O CONTROL REGISTER

This register controls the operation of the CAN module's I/O pins in relation to the rest of the microcontroller.

REGISTER 24-55: CIOCON: CAN I/O CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	—	ENDRHI[(1)]	CANCAP	—	—	—	—
bit 7							bit 0

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

bit 7-6	**Unimplemented:** Read as '0'
bit 5	**ENDRHI:** Enable Drive High bit[(1)] 1 = CANTX pin will drive VDD when recessive 0 = CANTX pin will be tri-state when recessive
bit 4	**CANCAP:** CAN Message Receive Capture Enable bit 1 = Enable CAN capture, CAN message receive signal replaces input on RC2/CCP1 0 = Disable CAN capture, RC2/CCP1 input to CCP1 module
bit 3-0	**Unimplemented:** Read as '0'

Note 1: Always set this bit when using a differential bus to avoid signal crosstalk in CANTX from other nearby pins.

REGISTER 25-4: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

R/P-1	U-0	U-0	U-0	U-0	R/P-0	R/P-1	U-0
MCLRE	—	—	—	—	LPT1OSC	PBADEN	—
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as ‘0’
-n = Value when device is unprogrammed		u = Unchanged from programmed state

bit 7	**MCLRE:** $\overline{\text{MCLR}}$ Pin Enable bit 1 = $\overline{\text{MCLR}}$ pin enabled; RE3 input pin disabled 0 = RE3 input pin enabled; $\overline{\text{MCLR}}$ disabled
bit 6-3	**Unimplemented:** Read as ‘0’
bit 2	**LPT1OSC:** Low-Power Timer1 Oscillator Enable bit 1 = Timer1 configured for low-power operation 0 = Timer1 configured for higher power operation
bit 1	**PBADEN:** PORTB A/D Enable bit (Affects ADCON1 Reset state. ADCON1 controls PORTB<4:0> pin configuration.) 1 = PORTB<4:0> pins are configured as analog input channels on Reset 0 = PORTB<4:0> pins are configured as digital I/O on Reset
bit 0	**Unimplemented:** Read as ‘0’

REGISTER 25-5: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/P-1	R/P-0	U-0	R/P-0	U-0	R/P-1	U-0	R/P-1
$\overline{\text{DEBUG}}$	XINST	—	BBSIZ	—	LVP	—	STVREN
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as ‘0’
-n = Value when device is unprogrammed		u = Unchanged from programmed state

bit 7	**$\overline{\text{DEBUG}}$:** Background Debugger Enable bit 1 = Background debugger disabled, RB6 and RB7 configured as general purpose I/O pins 0 = Background debugger enabled, RB6 and RB7 are dedicated to In-Circuit Debug
bit 6	**XINST:** Extended Instruction Set Enable bit 1 = Instruction set extension and Indexed Addressing mode enabled 0 = Instruction set extension and Indexed Addressing mode disabled (Legacy mode)
bit 5	**Unimplemented:** Read as ‘0’
bit 4	**BBSIZ:** Boot Block Size Select Bit 0 01 = 2K words (4 Kbytes) boot block 00 = 1K words (2 Kbytes) boot block
bit 3	**Unimplemented:** Read as ‘0’
bit 2	**LVP:** Single-Supply ICSP™ Enable bit 1 = Single-Supply ICSP enabled 0 = Single-Supply ICSP disabled
bit 1	**Unimplemented:** Read as ‘0’
bit 0	**STVREN:** Stack Full/Underflow Reset Enable bit 1 = Stack full/underflow will cause Reset 0 = Stack full/underflow will not cause Reset

BCF Bit Clear f

Syntax:	BCF f, b {,a}
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]
Operation:	0 → f<b>
Status Affected:	None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared.

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 ≤ 95 (5Fh). See **Section 26.2.3 "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: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

BN Branch if Negative

Syntax:	BN n
Operands:	-128 ≤ n ≤ 127
Operation:	if Negative bit is '1', (PC) + 2 + 2n → PC
Status Affected:	None

Encoding:

1110	0110	nnnn	nnnn

Description: If the Negative 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 two-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 BN Jump`

Before Instruction		
PC	=	address (HERE)
After Instruction		
If Negative	=	1;
PC	=	address (Jump)
If Negative	=	0;
PC	=	address (HERE + 2)

MULLW — Multiply Literal with W

Syntax:	MULLW k
Operands:	$0 \le k \le 255$
Operation:	(W) x k $\rightarrow$ PRODH:PRODL
Status Affected:	None
Encoding:	`0000` `1101` `kkkk` `kkkk`
Description:	An unsigned multiplication is carried out between the contents of W and the 8-bit literal ‘k’. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged. None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write registers PRODH: PRODL

Example: `MULLW 0C4h`

Before Instruction

W	=	E2h
PRODH	=	?
PRODL	=	?

After Instruction

W	=	E2h
PRODH	=	ADh
PRODL	=	08h

MULWF — Multiply W with f

Syntax:	MULWF f {,a}
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	(W) x (f) $\rightarrow$ PRODH:PRODL
Status Affected:	None
Encoding:	`0000` `001a` `ffff` `ffff`
Description:	An unsigned multiplication is carried out between the contents of W and the register file location ‘f’. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and ‘f’ are unchanged. None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. 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 26.2.3 “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 registers PRODH: PRODL

Example: `MULWF REG, 1`

Before Instruction

W	=	C4h
REG	=	B5h
PRODH	=	?
PRODL	=	?

After Instruction

W	=	C4h
REG	=	B5h
PRODH	=	8Ah
PRODL	=	94h

26.2.2 EXTENDED INSTRUCTION SET

ADDFSR **Add Literal to FSR**

Syntax:	ADDFSR f, k
Operands:	$0 \leq k \leq 63$ $f \in [\,0, 1, 2\,]$
Operation:	$FSR(f) + k \rightarrow FSR(f)$
Status Affected:	None
Encoding:	`1110` `1000` `ffkk` `kkkk`
Description:	The 6-bit literal ‘k’ is added to the contents of the FSR specified by ‘f’.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to FSR

Example: `ADDFSR 2, 23h`

Before Instruction
FSR2 = 03FFh

After Instruction
FSR2 = 0422h

ADDULNK **Add Literal to FSR2 and Return**

Syntax:	ADDULNK k
Operands:	$0 \leq k \leq 63$
Operation:	$FSR2 + k \rightarrow FSR2$, PC = (TOS)
Status Affected:	None
Encoding:	`1110` `1000` `11kk` `kkkk`
Description:	The 6-bit literal ‘k’ is added to the contents of FSR2. A `RETURN` is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a `NOP` is performed during the second cycle. This may be thought of as a special case of the `ADDFSR` instruction, where f = 3 (binary ‘`11`’); it operates only on FSR2.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to FSR
No Operation	No Operation	No Operation	No Operation

Example: `ADDULNK 23h`

Before Instruction
FSR2 = 03FFh
PC = 0100h
TOS = 02AFh

After Instruction
FSR2 = 0422h
PC = 02AFh
TOS = TOS – 1

Note:	All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).

27.0 DEVELOPMENT SUPPORT

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

- Integrated Development Environment
  - MPLAB® IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB C Compiler for Various Device Families
  - HI-TECH C for Various Device Families
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- Device Programmers
  - PICkit™ 2 Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits, and Starter Kits

27.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - In-Circuit Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either C or assembly)
- One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (C or assembly)
  - Mixed C and assembly
  - Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

FIGURE 28-18: MASTER SSP I²C™ BUS START/STOP BITS TIMING WAVEFORMS

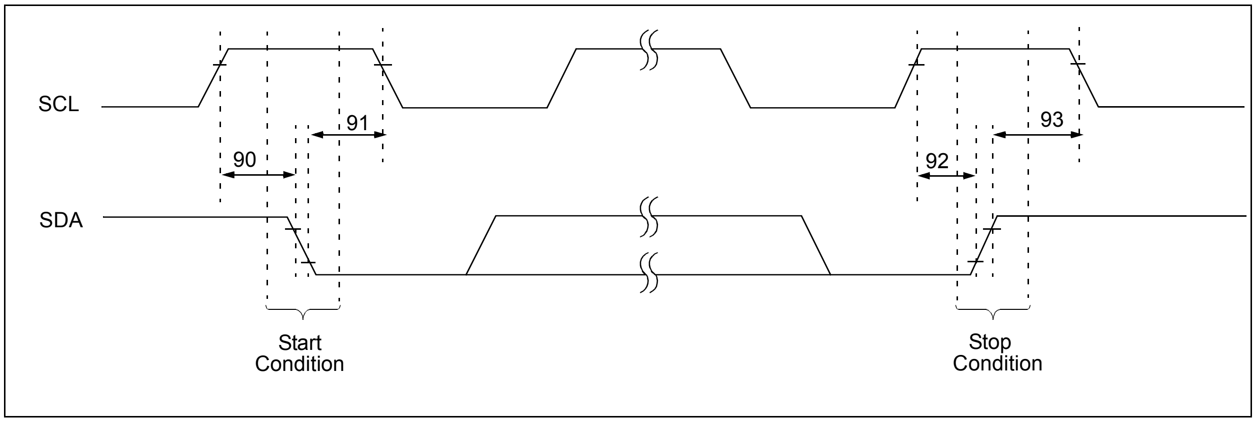
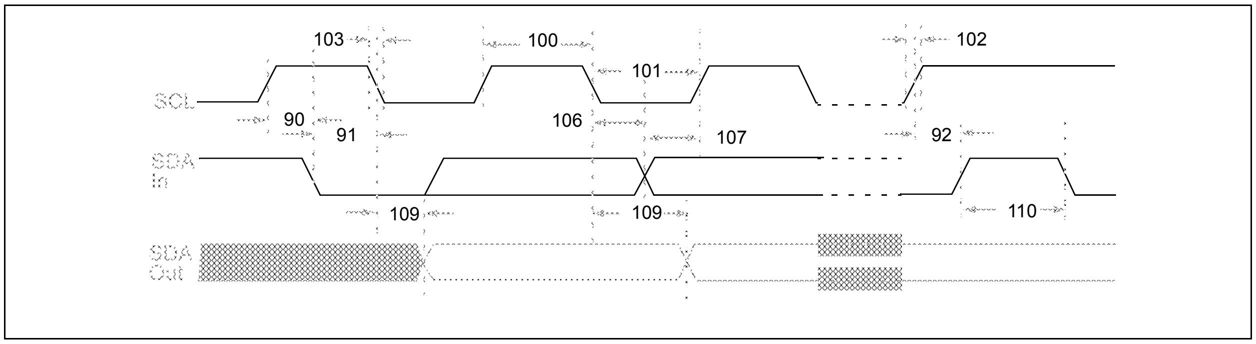


TABLE 28-20: MASTER SSP I²C™ BUS START/STOP BITS REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	Only relevant for Repeated Start condition
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		
93	THD:STO	Stop Condition Hold Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		

Note 1: Maximum pin capacitance = 10 pF for all I²C pins.

FIGURE 28-19: MASTER SSP I²C™ BUS DATA TIMING



- Memory Organization 67
  - Data Memory 73
  - Program Memory 67
- Memory Programming Requirements 435
- Microchip Internet Web Site 486
- Migration from Baseline to Enhanced Devices 472
- Migration from High-End to Enhanced Devices 473
- Migration from Mid-Range to Enhanced Devices 473
- MOVF 391
- MOVFF 392
- MOVLB 392
- MOVLW 393
- MOVSF 411
- MOVSS 412
- MOVWF 393
- MPLAB ASM30 Assembler, Linker, Librarian 418
- MPLAB Integrated Development Environment Software 417
- MPLAB PM3 Device Programmer 420
- MPLAB REAL ICE In-Circuit Emulator System 419
- MPLINK Object Linker/MPLIB Object Librarian 418
- MSSP
  - $\overline{\text{ACK}}$ Pulse 204, 205
  - Control Registers (general) 191
  - I^{2}C Mode. *See* I^{2}C Mode.
  - Module Overview 191
  - SPI Master/Slave Connection 195
  - SPI Mode. *See* SPI Mode.
  - SSPBUF 196
  - SSPSR 196
- MULLW 394
- MULWF 394

N

- NEGF 395
- NOP 395
- Normal Operation Mode 330

O

- Opcode Field Descriptions 368
- Oscillator Configuration 29
  - EC 29
  - ECIO 29
  - HS 29
  - HSPLL 29
  - Internal Oscillator Block 32
  - INTIO1 29
  - INTIO2 29
  - LP 29
  - RC 29
  - RCIO 29
  - XT 29
- Oscillator Selection 349
- Oscillator Start-up Timer (OST) 37, 51
- Oscillator Switching 34
- Oscillator Transitions 35
- Oscillator, Timer1 .... 155, 165
- Oscillator, Timer3 163

P

- Packaging Information 459
  - Details 461
  - Marking 459
- Parallel Slave Port (PSP) 143, 149
  - Associated Registers 150
  - $\overline{\text{CS}}$ (Chip Select) 149
  - PORTD 149
  - $\overline{\text{RD}}$ (Read Input) 149
  - Select (PSPMODE Bit) 143, 149
  - $\overline{\text{WR}}$ (Write Input) 149
- PIE Registers 127
- Pin Functions
  - $\overline{\text{MCLR}}$/VPP/RE3 .... 14, 18
  - OSC1/CLKI/RA7 .... 14, 18
  - OSC2/CLKO/RA6 .... 14, 18
  - RA0/AN0 15
  - RA0/AN0/CVREF 19
  - RA1/AN1 .... 15, 19
  - RA2/AN2/VREF- 15, 19
  - RA3/AN3/VREF+ 15, 19
  - RA4/T0CKI 15, 19
  - RA5/AN4/$\overline{\text{SS}}$/HLVDIN 15, 19
  - RB0/INT0/AN10 16
  - RB0/INT0/FLT0/AN10 20
  - RB1/INT1/AN8 .... 16, 20
  - RB2/INT2/CANTX 16, 20
  - RB3/CANRX 16, 20
  - RB4/KBI0/AN9 .... 16, 20
  - RB5/KBI1/PGM 16, 20
  - RB6/KBI2/PGC 16, 20
  - RB7/KBI3/PGD 16, 20
  - RC0/T1OSO/T13CKI 17, 21
  - RC1/T1OSI 17, 21
  - RC2/CCP1 .... 17, 21
  - RC3/SCK/SCL 17, 21
  - RC4/SDI/SDA 17, 21
  - RC5/SDO 17, 21
  - RC6/TX/CK 17, 21
  - RC7/RX/DT 17, 21
  - RD0/PSP0/C1IN+ 22
  - RD1/PSP1/C1IN- 22
  - RD2/PSP2/C2IN+ 22
  - RD3/PSP3/C2IN- 22
  - RD4/PSP4/ECCP1/P1A 22
  - RD5/PSP5/P1B 22
  - RD6/PSP6/P1C 22
  - RD7/PSP7/P1D 22
  - RE0/$\overline{\text{RD}}$/AN5 23
  - RE1/$\overline{\text{WR}}$/AN6/C1OUT 23
  - RE2/$\overline{\text{CS}}$/AN7/C2OUT 23
  - VDD 17, 23
  - VSS 17, 23
- Pinout I/O Descriptions
  - PIC18F2480/2580 14
  - PIC18F4480/4580 18
- PIR Registers 124
- PLL
  - Use with INTOSC 31
- PLL Frequency Multiplier 31
  - HSPLL Oscillator Mode 31
  - In INTOSC Modes 32
- PLL Lock Time-out 51
- POP 396
- POR. *See* Power-on Reset.
- PORTA
  - Associated Registers 137
  - I/O Summary 136
  - LATA Register 135
  - PORTA Register 135
  - TRISA Register 135