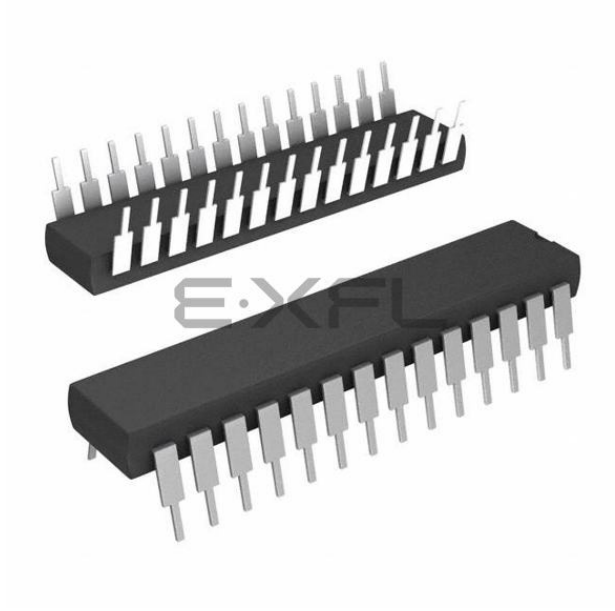


Welcome to E-XFL.COM

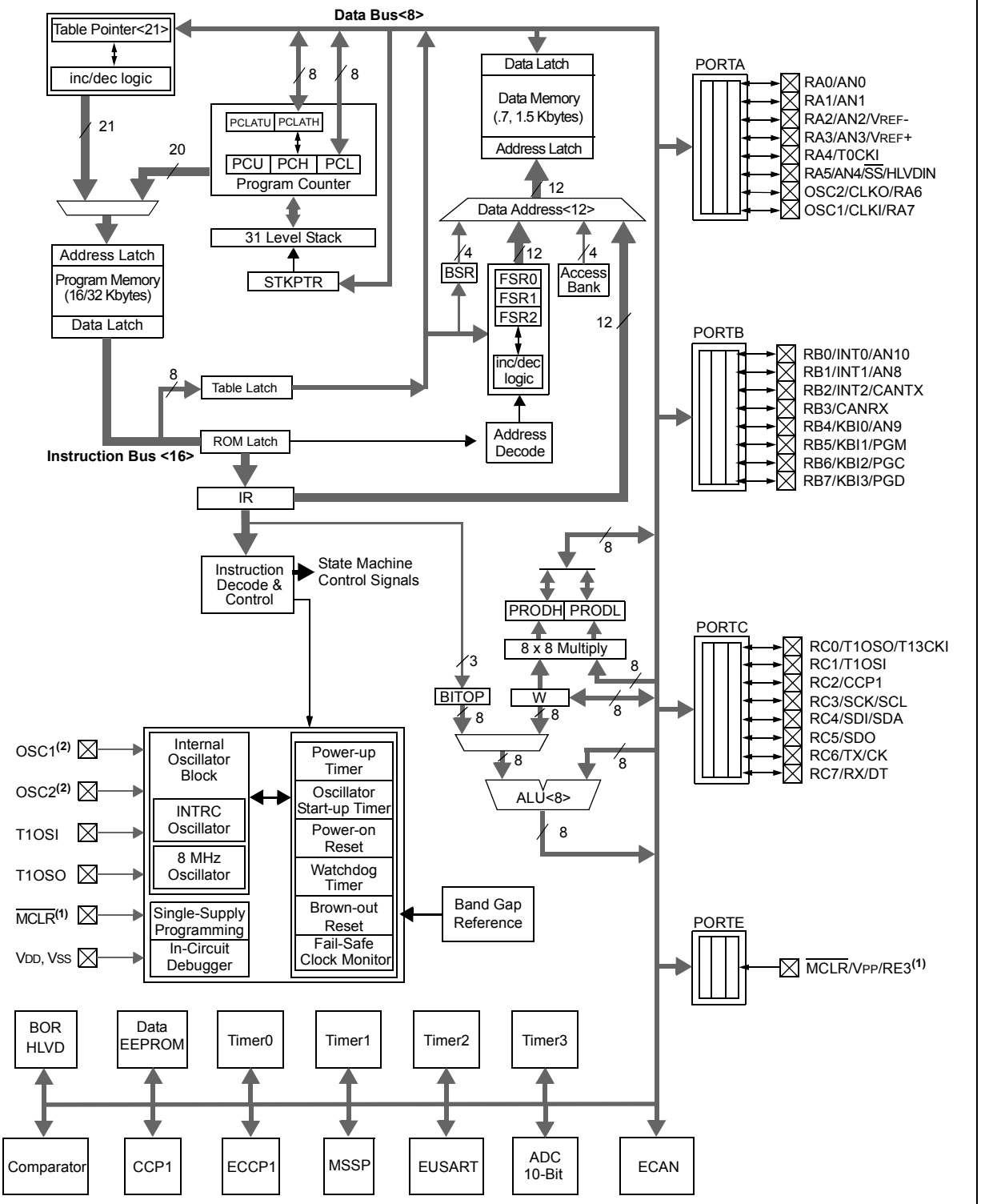
What is "Embedded - Microcontrollers"?

"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	25
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f2480-e-sp

FIGURE 1-1: PIC18F2480/2580 (28-PIN) BLOCK DIAGRAM



Note 1: RE3 is multiplexed with $\overline{\text{MCLR}}$ and is only available when the $\overline{\text{MCLR}}$ Resets are disabled.

2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 3.0 "Oscillator Configurations"** for additional information.

3.0 OSCILLATOR CONFIGURATIONS

3.1 Oscillator Types

PIC18F2480/2580/4480/4580 devices can be operated in ten different oscillator modes. The user can program the Configuration bits, FOSC<3:0>, in Configuration Register 1H to select one of these ten modes:

1. LP Low-Power Crystal
2. XT Crystal/Resonator
3. HS High-Speed Crystal/Resonator
4. HSPLL High-Speed Crystal/Resonator with PLL Enabled
5. RC External Resistor/Capacitor with FOSC/4 Output on RA6
6. RCIO External Resistor/Capacitor with I/O on RA6
7. INTIO1 Internal Oscillator with FOSC/4 Output on RA6 and I/O on RA7
8. INTIO2 Internal Oscillator with I/O on RA6 and RA7
9. EC External Clock with FOSC/4 Output
10. ECIO External Clock with I/O on RA6

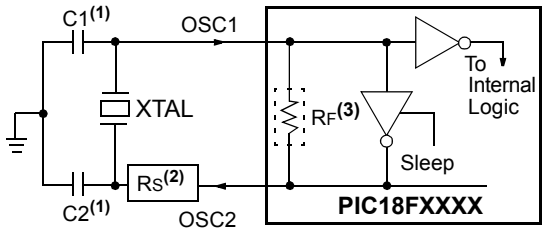
3.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 3-1 shows the pin connections.

The oscillator design requires the use of a parallel resonant crystal.

Note: Use of a series resonant crystal may give a frequency out of the crystal manufacturer's specifications.

FIGURE 3-1: CRYSTAL/CERAMIC RESONATOR OPERATION (XT, LP, HS OR HSPLL CONFIGURATION)



Note 1: See Table 3-1 and Table 3-2 for initial values of C1 and C2.
2: A series resistor (RS) may be required for AT strip cut crystals.
3: RF varies with the oscillator mode chosen.

TABLE 3-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

Typical Capacitor Values Used:			
Mode	**Freq**	**OSC1**	**OSC2**
XT	455 kHz 2.0 MHz 4.0 MHz	56 pF 47 pF 33 pF	56 pF 47 pF 33 pF
HS	8.0 MHz 16.0 MHz	27 pF 22 pF	27 pF 22 pF

Capacitor values are for design guidance only.

These capacitors were tested with the resonators listed below for basic start-up and operation. **These values are not optimized**.

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes on page 30 for additional information.

Resonators Used:	
455 kHz	4.0 MHz
2.0 MHz	8.0 MHz
16.0 MHz	

Note: When using resonators with frequencies above 3.5 MHz, the use of HS mode, rather than XT mode, is recommended. HS mode may be used at any VDD for which the controller is rated. If HS is selected, it is possible that the gain of the oscillator will overdrive the resonator. Therefore, a series resistor should be placed between the OSC2 pin and the resonator. As a good starting point, the recommended value of RS is 330Ω.

3.7.1 OSCILLATOR CONTROL REGISTER

The OSCCON register (Register 3-2) controls several aspects of the device clock's operation, both in full-power operation and in power-managed modes.

The System Clock Select bits, SCS<1:0>, select the clock source. The available clock sources are the primary clock (defined by the FOSC<3:0> Configuration bits), the secondary clock (Timer1 oscillator) and the internal oscillator block. The clock source changes immediately after one or more of the bits is written to, following a brief clock transition interval. The SCS bits are cleared on all forms of Reset.

The Internal Oscillator Frequency Select bits, IRCF<2:0>, select the frequency output of the internal oscillator block to drive the device clock. The choices are the INTRC source, the INTOSC source (8 MHz) or one of the frequencies derived from the INTOSC postscaler (31 kHz to 4 MHz). If the internal oscillator block is supplying the device clock, changing the states of these bits will have an immediate change on the internal oscillator's output. On device Resets, the default output frequency of the internal oscillator block is set at 1 MHz.

When an output frequency of 31 kHz is selected (IRCF<2:0> = 000), users may choose which internal oscillator acts as the source. This is done with the INTSRC bit in the OSCTUNE register (OSCTUNE<7>). Setting this bit selects INTOSC as a 31.25 kHz clock source by enabling the divide-by-256 output of the INTOSC postscaler. Clearing INTSRC selects INTRC (nominally 31 kHz) as the clock source.

This option allows users to select the tunable and more precise INTOSC as a clock source, while maintaining power savings with a very low clock speed. Regardless of the setting of INTSRC, INTRC always remains the clock source for features such as the Watchdog Timer and the Fail-Safe Clock Monitor.

The OSTS, IOFS and T1RUN bits indicate which clock source is currently providing the device clock. The OSTS bit indicates that the Oscillator Start-up Timer (OST) has timed out and the primary clock is providing the device clock in primary clock modes. The IOFS bit indicates when the internal oscillator block has stabilized and is providing the device clock in RC Clock modes. The T1RUN bit (T1CON<6>) indicates when the Timer1 oscillator is providing the device clock in secondary clock modes. In power-managed modes, only one of these three bits will be set at any time. If none of these bits are set, the INTRC is providing the clock or the internal oscillator block has just started and is not yet stable.

The IDLEN bit determines if the device goes into Sleep mode or one of the Idle modes when the SLEEP instruction is executed.

The use of the flag and control bits in the OSCCON register is discussed in more detail in **Section 4.0 "Power-Managed Modes"**.

Note 1:	The Timer1 oscillator must be enabled to select the secondary clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 Control register (T1CON<3>). If the Timer1 oscillator is not enabled, then any attempt to select a secondary clock source when executing a SLEEP instruction will be ignored.
2:	It is recommended that the Timer1 oscillator be operating and stable before executing the SLEEP instruction, or a very long delay may occur while the Timer1 oscillator starts.

3.7.2 OSCILLATOR TRANSITIONS

PIC18F2480/2580/4480/4580 devices contain circuitry to prevent clock "glitches" when switching between clock sources. A short pause in the device clock occurs during the clock switch. The length of this pause is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Clock transitions are discussed in greater detail in **Section 4.1.2 "Entering Power-Managed Modes"**.

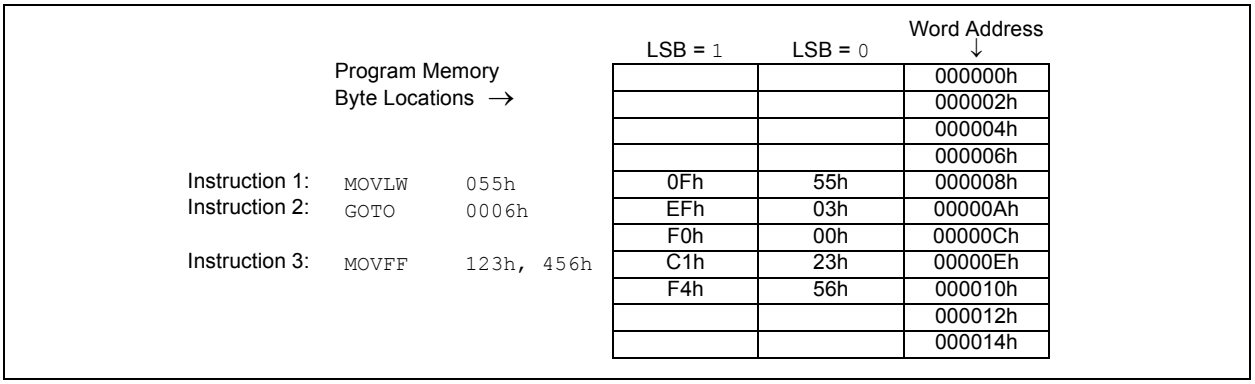
6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSB = 0). To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see **Section 6.1.1 "Program Counter"**).

Figure 6-4 shows an example of how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 6-4 shows how the instruction, GOTO 0006h, is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 26.0 "Instruction Set Summary"** provides further details of the instruction set.

FIGURE 6-4: INSTRUCTIONS IN PROGRAM MEMORY



6.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four, two-word instructions: CALL, MOVFF, GOTO and LSFR. In all cases, the second word of the instructions always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the 4 MSbs of an instruction specifies a special form of NOP. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a NOP is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 6-4 shows how this works.

Note:	See **Section 6.5 "Program Memory and the Extended Instruction Set"** for information on two-word instructions in the extended instruction set.

EXAMPLE 6-4: TWO-WORD INSTRUCTIONS

CASE 1:			
Object Code	**Source Code**		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; No, skip this word
1111 0100 0101 0110			; Execute this word as a NOP
0010 0100 0000 0000	ADDWF	REG3	; continue code
CASE 2:			
Object Code	**Source Code**		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; Yes, execute this word
1111 0100 0101 0110			; 2nd word of instruction
0010 0100 0000 0000	ADDWF	REG3	; continue code

6.4 Data Addressing Modes

Note: The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 6.6 "Data Memory and the Extended Instruction Set"** for more information.

While the program memory can be addressed in only one way – through the program counter – information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in **Section 6.6.1 "Indexed Addressing with Literal Offset"**.

6.4.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include SLEEP, RESET and DAW.

Other instructions work in a similar way but require an additional explicit argument in the opcode. This is known as Literal Addressing mode because they require some literal value as an argument. Examples include ADDLW and MOVLW which, respectively, add or move a literal value to the W register. Other examples include CALL and GOTO, which include a 20-bit program memory address.

6.4.2 DIRECT ADDRESSING

Direct Addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 6.3.3 "General Purpose Register File"**) or a location in the Access Bank (**Section 6.3.2 "Access Bank"**) as the data source for the instruction.

The Access RAM bit 'a' determines how the address is interpreted. When 'a' is '1', the contents of the BSR (**Section 6.3.1 "Bank Select Register (BSR)"**) are used with the address to determine the complete 12-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as MOVFF, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation's results is determined by the destination bit, 'd'. When 'd' is '1', the results are stored back in the source register, overwriting its original contents. When 'd' is '0', the results are stored in the W register. Instructions without the 'd' argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

6.4.3 INDIRECT ADDRESSING

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations to be read or written to. Since the FSRs are themselves located in RAM as Special File Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for Indirect Addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in Example 6-5.

EXAMPLE 6-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR   FSR0, 100h  ;
NEXT    CLRF   POSTINC0    ; Clear INDF
                           ; register then
                           ; inc pointer
        BTFSS  FSR0H,1     ; All done with
                           ; Bank1?
        BRA    NEXT        ; NO, clear next
CONTINUE                   ; YES, continue
```

7.4 Erasing Flash Program Memory

The minimum erase block is 32 words or 64 bytes. Only through the use of an external programmer, or through ICSP control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

When initiating an erase sequence from the micro-controller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:5> point to the block being erased. TBLPTR<4:0> are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

For protection, the write initiate sequence for EECON2 must be used.

A long write is necessary for erasing the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

7.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory location is:

1. Load Table Pointer register with address of row being erased.
2. Set the EECON1 register for the erase operation:
   - set EEPGD bit to point to program memory;
   - clear the CFGS bit to access program memory;
   - set WREN bit to enable writes;
   - set FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit. This will begin the row erase cycle.
7. The CPU will stall for duration of the erase (about 2 ms using internal timer).
8. Re-enable interrupts.

EXAMPLE 7-2: ERASING A FLASH PROGRAM MEMORY ROW

```
                MOVLW   CODE_ADDR_UPPER   ; load TBLPTR with the base
                MOVWF   TBLPTRU           ; address of the memory block
                MOVLW   CODE_ADDR_HIGH
                MOVWF   TBLPTRH
                MOVLW   CODE_ADDR_LOW
                MOVWF   TBLPTRL
    ERASE_ROW
                BSF     EECON1, EEPGD     ; point to Flash program memory
                BCF     EECON1, CFGS      ; access Flash program memory
                BSF     EECON1, WREN      ; enable write to memory
                BSF     EECON1, FREE      ; enable Row Erase operation
                BCF     INTCON, GIE       ; disable interrupts
Required        MOVLW   55h
Sequence        MOVWF   EECON2            ; write 55h
                MOVLW   0AAh
                MOVWF   EECON2            ; write 0AAh
                BSF     EECON1, WR        ; start erase (CPU stall)
                BSF     INTCON, GIE       ; re-enable interrupts
```

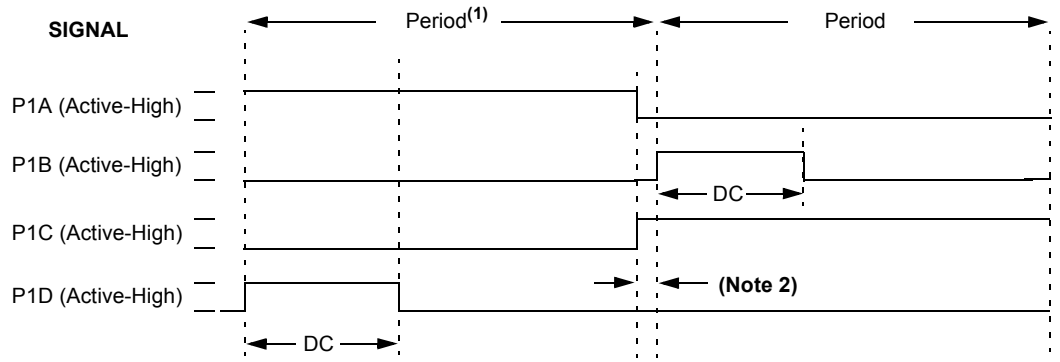
TABLE 11-1: PORTA I/O SUMMARY

Pin Name	Function	I/O	TRIS	Buffer	Description
RA0/AN0/CVREF	RA0	OUT	0	DIG	LATA<0> data output.
		IN	1	TTL	PORTA<0> data input.
	AN0	IN	1	ANA	A/D Input Channel 0. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
	CVREF(1)	OUT	x	ANA	Comparator voltage reference analog output. Enabling this analog output overrides the digital I/O (read as clear – low level).
RA1/AN1	RA1	OUT	0	DIG	LATA<1> data output.
		IN	1	TTL	PORTA<1> data input.
	AN1	IN	1	ANA	A/D Input Channel 1. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
RA2/AN2/VREF-	RA2	OUT	0	DIG	LATA<2> data output.
		IN	1	TTL	PORTA<2> data input.
	AN2	IN	1	ANA	A/D Input Channel 2. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
	VREF-	IN	1	ANA	A/D and comparator negative voltage analog input.
RA3/AN3/VREF+	RA3	OUT	0	DIG	LATA<3> data output.
		IN	1	TTL	PORTA<3> data input.
	AN3	IN	1	ANA	A/D Input Channel 3. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
	VREF+	IN	1	ANA	A/D and comparator positive voltage analog input.
RA4/T0CKI	RA4	OUT	0	DIG	LATA<4> data output.
		IN	1	TTL	PORTA<4> data input.
	T0CKI	IN	1	ST	Timer0 clock input.
RA5/AN4/$\overline{SS}$/HLVDIN	RA5	OUT	0	DIG	LATA<5> data output.
		IN	1	TTL	PORTA<5> data input.
	AN4	IN	1	ANA	A/D Input Channel 4. Enabled on POR; this analog input overrides the digital input (read as clear – low level).
	$\overline{SS}$	IN	1	TTL	Slave select input for MSSP.
	HLVDIN	IN	1	ANA	High/Low-Voltage Detect external trip point input.
OSC2/CLKO/RA6	OSC2	OUT	x	ANA	Output connection; selected by FOSC<3:0> Configuration bits. Enabling OSC2 overrides digital I/O.
	CLKO	OUT	x	DIG	Output connection; selected by FOSC<3:0> Configuration bits. Enabling CLKO overrides digital I/O (FOSC/4).
	RA6	OUT	0	DIG	LATA<6> data output.
		IN	1	TTL	PORTA<6> data input.
OSC1/CLKI/RA7	OSC1	IN	x	ANA	Main oscillator input connection determined by FOSC<3:0> Configuration bits. Enabling OSC1 overrides digital I/O.
	CLKI	IN	x	ANA	Main clock input connection determined by FOSC<3:0> Configuration bits. Enabling CLKI overrides digital I/O.
	RA7	OUT	0	DIG	LATA<7> data output.
		IN	1	TTL	PORTA<7> data input.

Legend: OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input

Note 1: Available on 40/44-pin devices only.

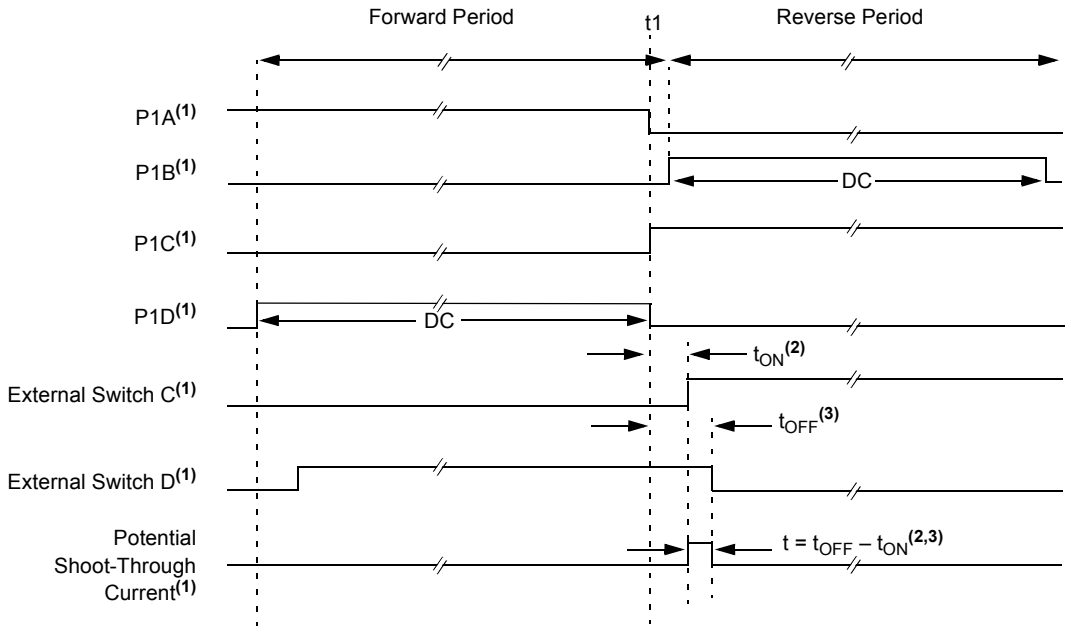
FIGURE 17-8: PWM DIRECTION CHANGE



Note 1: The direction bit in the CCP1 Control register (CCP1CON<7>) is written any time during the PWM cycle.

2: When changing directions, the P1A and P1C signals switch before the end of the current PWM cycle at intervals of 4 TOSC, 16 TOSC or 64 TOSC, depending on the Timer2 prescaler value. The modulated P1B and P1D signals are inactive at this time.

FIGURE 17-9: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



Note 1: All signals are shown as active-high.

2: t_{ON} is the turn-on delay of power switch QC and its driver.

3: t_{OFF} is the turn-off delay of power switch QD and its driver.

17.4.7.1 Auto-Shutdown and Auto-Restart

The auto-shutdown feature can be configured to allow automatic restarts of the module following a shutdown event. This is enabled by setting the PRSEN bit of the ECCP1DEL register (ECCP1DEL<7>).

In Shutdown mode with PRSEN = 1 (Figure 17-10), the ECCPASE bit will remain set for as long as the cause of the shutdown continues. When the shutdown condition clears, the ECCP1ASE bit is cleared. If PRSEN = 0 (Figure 17-11), once a shutdown condition occurs, the ECCPASE bit will remain set until it is cleared by firmware. Once ECCPASE is cleared, the Enhanced PWM will resume at the beginning of the next PWM period.

Note:	Writing to the ECCPASE bit is disabled while a shutdown condition is active.

Independent of the PRSEN bit setting, if the auto-shutdown source is one of the comparators, the shutdown condition is a level. The ECCPASE bit cannot be cleared as long as the cause of the shutdown persists.

The Auto-Shutdown mode can be forced by writing a '1' to the ECCPASE bit.

17.4.8 START-UP CONSIDERATIONS

When the ECCP module is used in the PWM mode, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins. When the microcontroller is released from Reset, all of the I/O pins are in the high-impedance state. The external circuits must keep the power switch devices in the off state until the microcontroller drives the I/O pins with the proper signal levels, or activates the PWM output(s).

The CCP1M<1:0> bits (ECCP1CON<1:0>) allow the user to choose whether the PWM output signals are active-high or active-low for each pair of PWM output pins (P1A/P1C and P1B/P1D). The PWM output polarities must be selected before the PWM pins are configured as outputs. Changing the polarity configuration while the PWM pins are configured as outputs is not recommended, since it may result in damage to the application circuits.

The P1A, P1B, P1C and P1D output latches may not be in the proper states when the PWM module is initialized. Enabling the PWM pins for output at the same time as the ECCP module may cause damage to the application circuit. The ECCP module must be enabled in the proper output mode and complete a full PWM cycle before configuring the PWM pins as outputs. The completion of a full PWM cycle is indicated by the TMR2IF bit being set as the second PWM period begins.

FIGURE 17-10: PWM AUTO-SHUTDOWN (PRSEN = 1, AUTO-RESTART ENABLED)

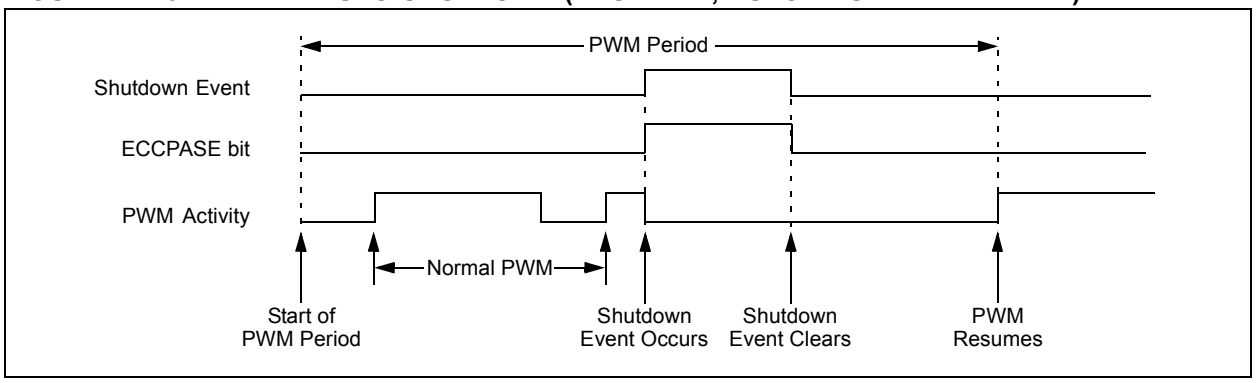


FIGURE 17-11: PWM AUTO-SHUTDOWN (PRSEN = 0, AUTO-RESTART DISABLED)

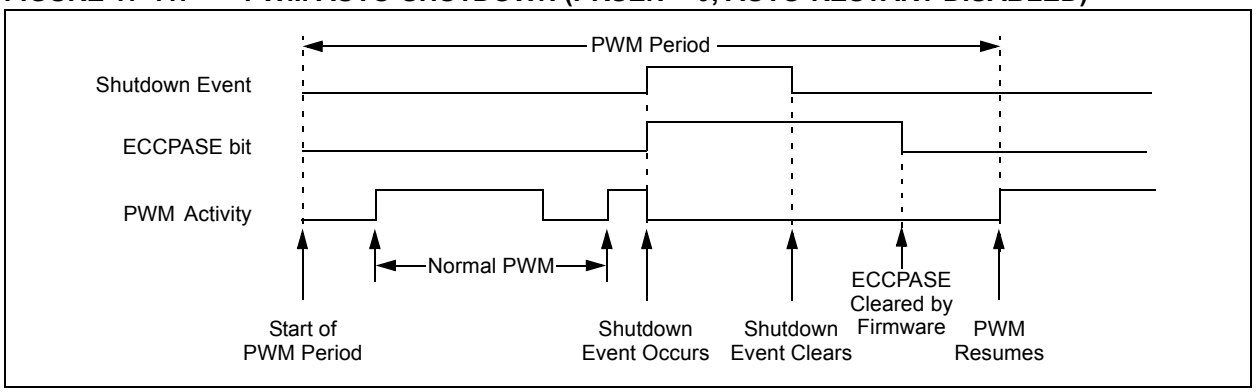


FIGURE 18-27: BUS COLLISION DURING START CONDITION (SCL = 0)

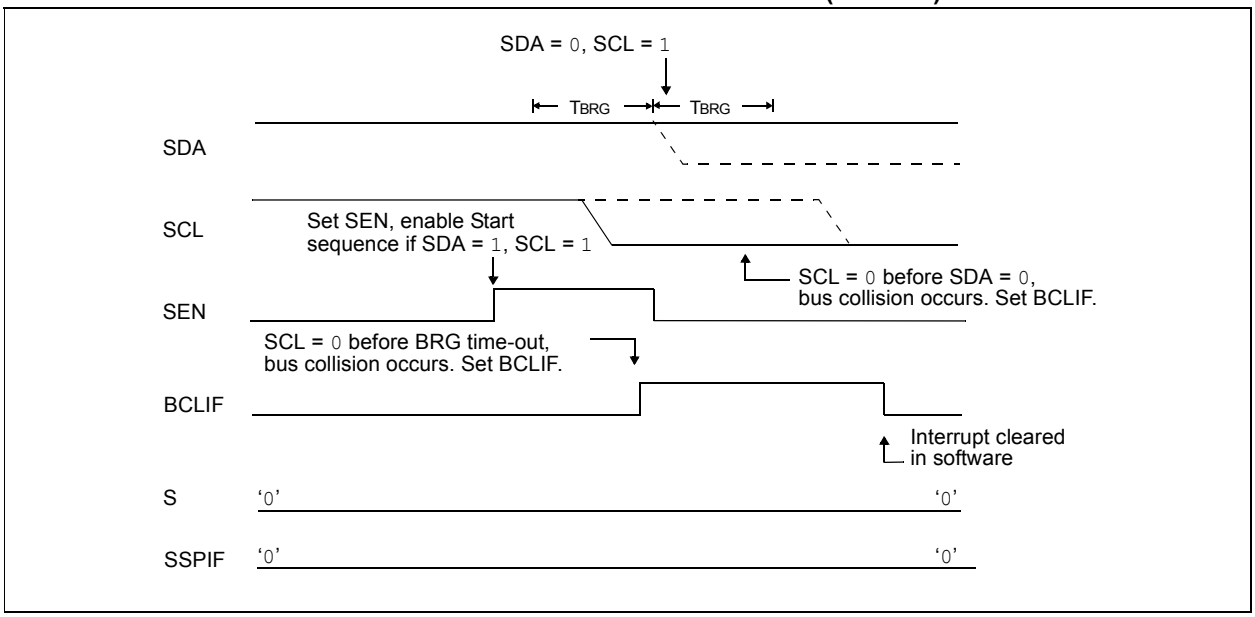


FIGURE 18-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION

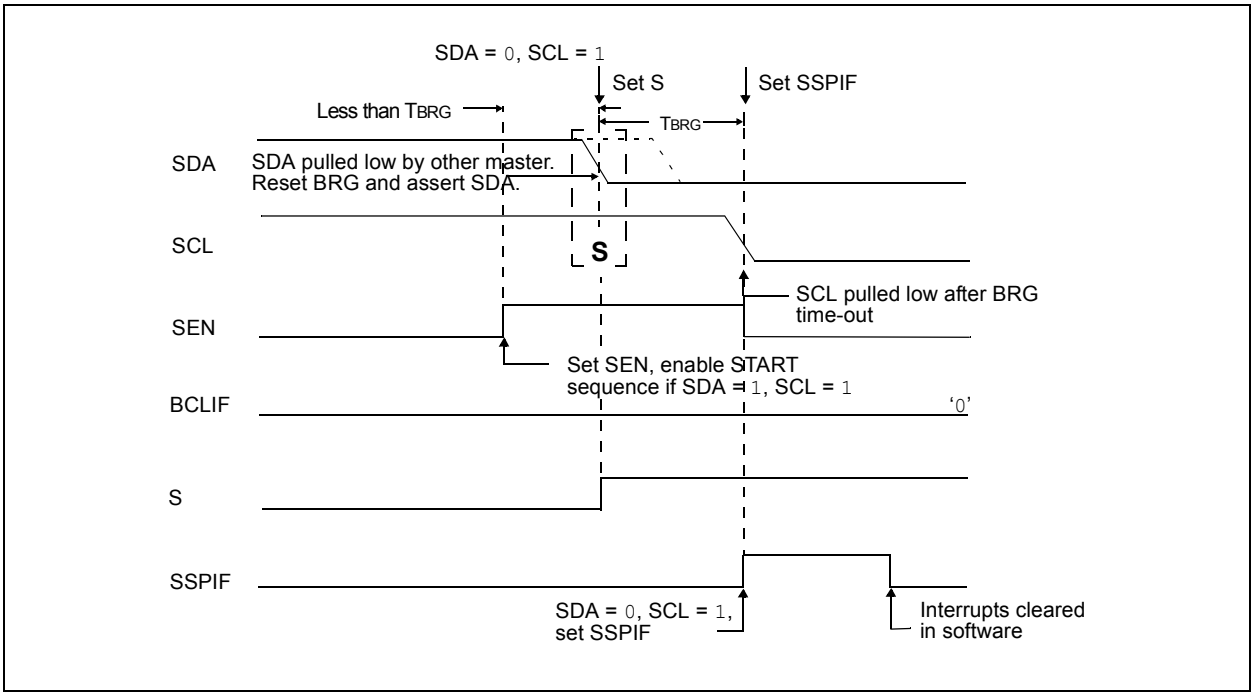
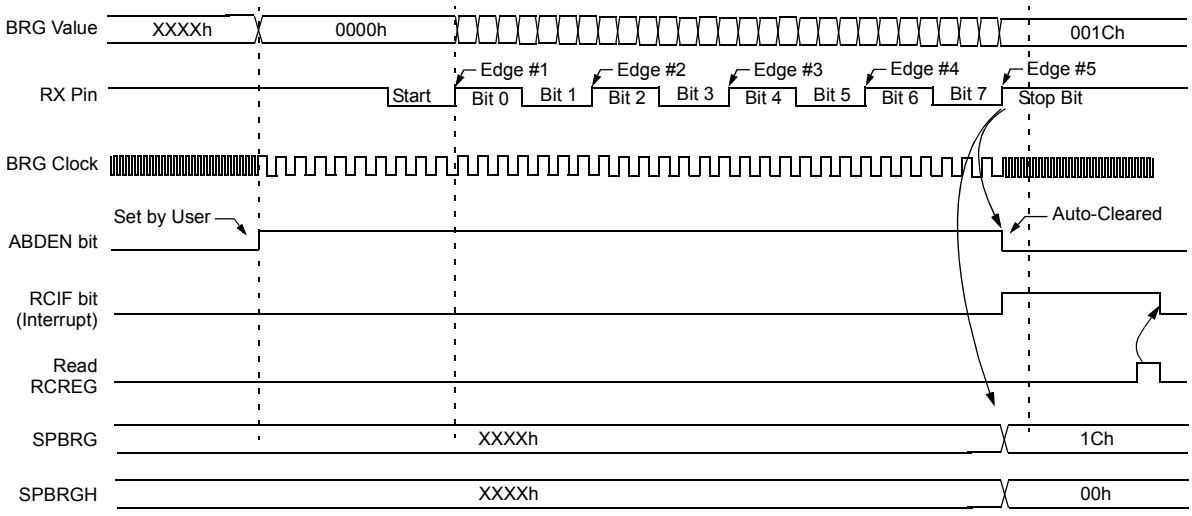
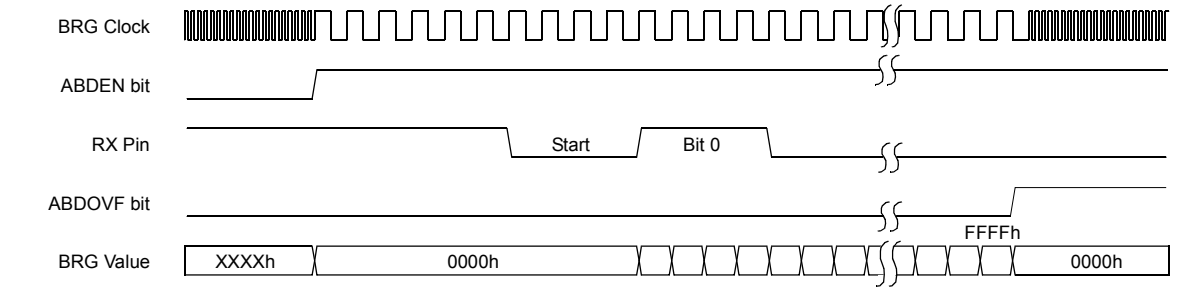


FIGURE 19-1: AUTOMATIC BAUD RATE CALCULATION



Note: The ABD sequence requires the EUSART module to be configured in Asynchronous mode and WUE = 0.

FIGURE 19-2: BRG OVERFLOW SEQUENCE



REGISTER 24-34: BnDLC: TX/RX BUFFER n DATA LENGTH CODE REGISTERS IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 0](1)

U-0	R-x	R-x	R-x	R-x	R-x	R-x	R-x
—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0
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 **Unimplemented:** Read as '0'

bit 6 **RXRTR:** Receiver Remote Transmission Request bit
1 = This is a remote transmission request
0 = This is not a remote transmission request

bit 5 **RB1:** Reserved bit 1
Reserved by CAN Spec and read as '0'.

bit 4 **RB0:** Reserved bit 0
Reserved by CAN Spec and read as '0'.

bit 3-0 **DLC<3:0>:** Data Length Code bits
1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = Reserved
1011 = Reserved
1010 = Reserved
1001 = Reserved
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

Note 1: These registers are available in Mode 1 and 2 only.

REGISTER 24-39: RXFnEIDH: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15](1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
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 7-0 **EID<15:8>:** Extended Identifier Filter bits

Note 1: Registers, RXF6EIDH:RXF15EIDH, are available in Mode 1 and 2 only.

REGISTER 24-40: RXFnEIDL: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 15](1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
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 7-0 **EID<7:0>:** Extended Identifier Filter bits

Note 1: Registers, RXF6EIDL:RXF15EIDL, are available in Mode 1 and 2 only.

REGISTER 24-41: RXMnSIDH: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
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 7-0 **SID<10:3>:** Standard Identifier Mask bits or Extended Identifier Mask bits (EID<28:21>)

REGISTER 24-48: MSEL0: MASK SELECT REGISTER 0(1)

R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
FIL3_1	FIL3_0	FIL2_1	FIL2_0	FIL1_1	FIL1_0	FIL0_1	FIL0_0
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 **FIL3_<1:0>:** Filter 3 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL2_<1:0>:** Filter 2 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL1_<1:0>:** Filter 1 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL0_<1:0>:** Filter 0 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

24.13 Bit Timing Configuration Registers

The Baud Rate Control registers (BRGCON1, BRGCON2, BRGCON3) control the bit timing for the CAN bus interface. These registers can only be modified when the PIC18F2480/2580/4480/4580 devices are in Configuration mode.

24.13.1 BRGCON1

The BRP bits control the baud rate prescaler. The SJW<1:0> bits select the synchronization jump width in terms of multiples of TQ.

24.13.2 BRGCON2

The PRSEG bits set the length of the propagation segment in terms of TQ. The SEG1PH bits set the length of Phase Segment 1 in TQ. The SAM bit controls how many times the RXCAN pin is sampled. Setting this bit to a '1' causes the bus to be sampled three times: twice at TQ/2 before the sample point and once at the normal sample point (which is at the end of Phase Segment 1). The value of the bus is determined to be the value read during at least two of the samples. If the SAM bit is set to a '0', then the RXCAN pin is sampled only once at the sample point. The SEG2PHTS bit controls how the length of Phase Segment 2 is determined. If this bit is set to a '1', then the length of Phase Segment 2 is determined by the SEG2PH bits of BRGCON3. If the SEG2PHTS bit is set to a '0', then the length of Phase Segment 2 is the greater of Phase Segment 1 and the information processing time (which is fixed at 2 TQ for the PIC18F2480/2580/4480/4580).

24.13.3 BRGCON3

The PHSEG2<2:0> bits set the length (in TQ) of Phase Segment 2 if the SEG2PHTS bit is set to a '1'. If the SEG2PHTS bit is set to a '0', then the PHSEG2<2:0> bits have no effect.

24.14 Error Detection

The CAN protocol provides sophisticated error detection mechanisms. The following errors can be detected.

24.14.1 CRC ERROR

With the Cyclic Redundancy Check (CRC), the transmitter calculates special check bits for the bit sequence, from the start of a frame until the end of the data field. This CRC sequence is transmitted in the CRC field. The receiving node also calculates the CRC sequence using the same formula and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an error frame is generated. The message is repeated.

24.14.2 ACKNOWLEDGE ERROR

In the Acknowledge field of a message, the transmitter checks if the Acknowledge slot (which was sent out as a recessive bit) contains a dominant bit. If not, no other node has received the frame correctly. An Acknowledge error has occurred, an error frame is generated and the message will have to be repeated.

24.14.3 FORM ERROR

If a node detects a dominant bit in one of the four segments, including End-Of-Frame (EOF), interframe space, Acknowledge delimiter or CRC delimiter, then a form error has occurred and an error frame is generated. The message is repeated.

24.14.4 BIT ERROR

A bit error occurs if a transmitter sends a dominant bit and detects a recessive bit, or if it sends a recessive bit and detects a dominant bit, when monitoring the actual bus level and comparing it to the just transmitted bit. In the case where the transmitter sends a recessive bit and a dominant bit is detected during the arbitration field and the Acknowledge slot, no bit error is generated because normal arbitration is occurring.

24.14.5 STUFF BIT ERROR

If, between the Start-Of-Frame (SOF) and the CRC delimiter, six consecutive bits with the same polarity are detected, the bit stuffing rule has been violated. A stuff bit error occurs and an error frame is generated. The message is repeated.

24.14.6 ERROR STATES

Detected errors are made public to all other nodes via error frames. The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states; "error-active", "error-passive" or "bus-off", according to the value of the internal error counters. The error-active state is the usual state where the bus node can transmit messages and activate error frames (made of dominant bits) without any restrictions. In the error-passive state, messages and passive error frames (made of recessive bits) may be transmitted. The bus-off state makes it temporarily impossible for the node to participate in the bus communication. During this state, messages can neither be received nor transmitted.

24.14.7 ERROR MODES AND ERROR COUNTERS

The PIC18F2480/2580/4480/4580 devices contain two error counters: the Receive Error Counter (RXERRCNT) and the Transmit Error Counter (TXERRCNT). The values of both counters can be read by the MCU. These counters are incremented or decremented in accordance with the CAN bus specification.

BNOV **Branch if Not Overflow**

Syntax: BNOV n

Operands: -128 ≤ n ≤ 127

Operation: if Overflow bit is '0',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0101	nnnn	nnnn

Description: If the Overflow bit is '0', 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 BNOV Jump`

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

BNZ **Branch if Not Zero**

Syntax: BNZ n

Operands: -128 ≤ n ≤ 127

Operation: if Zero bit is '0',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0001	nnnn	nnnn

Description: If the Zero bit is '0', 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 BNZ Jump`

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

POP **Pop Top of Return Stack**

Syntax: POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding:

0000	0000	0000	0110

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack.

This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	POP TOS value	No operation

Example:

```
POP
GOTO    NEW
```

Before Instruction
TOS = 0031A2h
Stack (1 level down) = 014332h

After Instruction
TOS = 014332h
PC = NEW

PUSH **Push Top of Return Stack**

Syntax: PUSH

Operands: None

Operation: (PC + 2) → TOS

Status Affected: None

Encoding:

0000	0000	0000	0101

Description: The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack.

This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	PUSH PC + 2 onto return stack	No operation	No operation

Example:

```
PUSH
```

Before Instruction
TOS = 345Ah
PC = 0124h

After Instruction
PC = 0126h
TOS = 0126h
Stack (1 level down) = 345Ah

RETFIE **Return from Interrupt**

Syntax: RETFIE {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL;
if s = 1,
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding:

0000	0000	0001	000s

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low-priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers occurs.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	POP PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example: RETFIE 1

After Interrupt
PC = TOS
W = WS
BSR = BSRS
STATUS = STATUSS
GIE/GIEH, PEIE/GIEL = 1

RETLW **Return Literal to W**

Syntax: RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W,
(TOS) → PC,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	1100	kkkk	kkkk

Description: W is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	POP PC from stack, Write to W
No operation	No operation	No operation	No operation

Example:

```
    CALL TABLE   ; W contains table
                 ; offset value
                 ; W now has
                 ; table value
   :
TABLE
    ADDWF PCL    ; W = offset
    RETLW k0     ; Begin table
    RETLW k1     ;
   :
   :
    RETLW kn     ; End of table
```

Before Instruction
W = 07h
After Instruction
W = value of kn

F

G

H

I

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support
- Development Systems Information Line

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://support.microchip.com