Welcome to E-XFL.COM

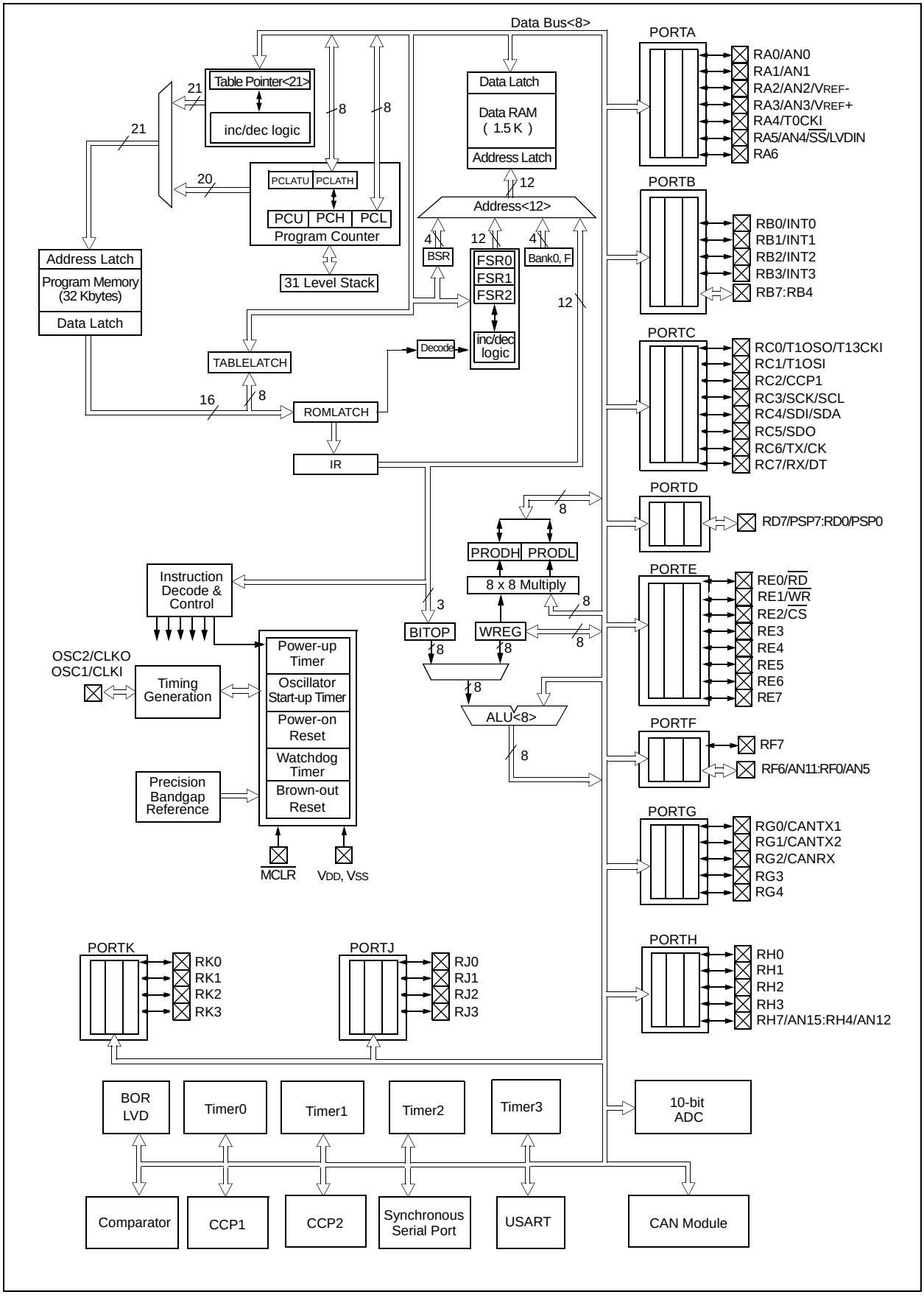
What is "Embedded - Microcontrollers"?

"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	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	68
Program Memory Size	32KB (16K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	84-LCC (J-Lead)
Supplier Device Package	84-PLCC (29.31x29.31)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lc858-i-l

FIGURE 1-2: PIC18C858 BLOCK DIAGRAM



2.0 OSCILLATOR CONFIGURATIONS

2.1 Oscillator Types

The PIC18CXX8 can be operated in one of eight oscillator modes, programmable by three configuration bits (FOSC2, FOSC1, and FOSC0).

1. LP Low Power Crystal
2. XT Crystal/Resonator
3. HS High Speed Crystal/Resonator
4. HS4 High Speed Crystal/Resonator with PLL enabled
5. RC External Resistor/Capacitor
6. RCIO External Resistor/Capacitor with I/O pin enabled
7. EC External Clock
8. ECIO External Clock with I/O pin enabled

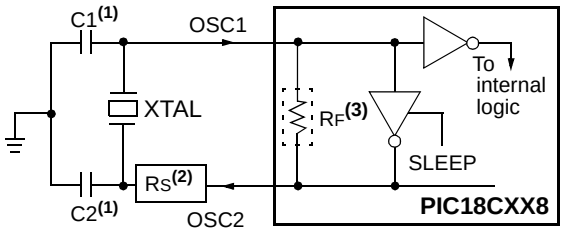
2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HS4 (PLL) oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-1 shows the pin connections. An external clock source may also be connected to the OSC1 pin, as shown in Figure 2-3 and Figure 2-4.

The PIC18CXX8 oscillator design requires the use of a parallel cut crystal.

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

FIGURE 2-1: CRYSTAL/CERAMIC RESONATOR OPERATION (HS, XT OR LP OSC CONFIGURATION)



Note 1: See Table 2-1 and Table 2-2 for recommended values of C1 and C2.

2: A series resistor (RS) may be required for AT strip cut crystals.

3: RF varies with the crystal chosen.

REGISTER 7-5: PIR REGISTERS (CONT'D)

PIR2

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

bit 6 **CMIF:** Comparator Interrupt Flag bit
1 = Comparator input has changed
0 = Comparator input has not changed

bit 5-4 **Unimplemented:** Read as'0'

bit 3 **BCLIF:** Bus Collision Interrupt Flag bit
1 = A Bus Collision occurred
(must be cleared in software)
0 = No Bus Collision occurred

bit 2 **LVDIF:** Low Voltage Detect Interrupt Flag bit
1 = A low voltage condition occurred
(must be cleared in software)
0 = The device voltage is above the Low Voltage Detect trip point

bit 1 **TMR3IF:** TMR3 Overflow Interrupt Flag bit
1 = TMR3 register overflowed
(must be cleared in software)
0 = TMR3 register did not overflow

bit 0 **CCP2IF:** CCPx Interrupt Flag bit
Capture Mode
1 = A TMR1 register capture occurred
(must be cleared in software)
0 = No TMR1 register capture occurred
Compare Mode
1 = A TMR1 register compare match occurred
(must be cleared in software)
0 = No TMR1 register compare match occurred
PWM Mode
Unused in this mode

TABLE 8-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$	bit0	ST/TTL(1)	Input/output port pin or Read control input in Parallel Slave Port mode.
RE1/$\overline{WR}$	bit1	ST/TTL(1)	Input/output port pin or Write control input in Parallel Slave Port mode.
RE2/$\overline{CS}$	bit2	ST/TTL(1)	Input/output port pin or Chip Select control input in Parallel Slave Port mode.
RE3	bit3	ST	Input/output port pin.
RE4	bit4	ST	Input/output port pin.
RE5	bit5	ST	Input/output port pin.
RE6	bit6	ST	Input/output port pin.
RE7/CCP2	bit7	ST	Input/output port pin or Capture 2 input/Compare 2 output.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffer when in Parallel Slave Port mode.

TABLE 8-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
TRISE	PORTE Data Direction Control Register								1111 1111	1111 1111
PORTE	Read PORTE pin/Write PORTE Data Latch								xxxx xxxx	uuuu uuuu
LATE	Read PORTE Data Latch/Write PORTE Data Latch								xxxx xxxx	uuuu uuuu
PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—	0000 ----	0000 ----

Legend: x = unknown, u = unchanged

8.8 PORTH, LATH, and TRISH Registers

Note: This port is available on PIC18C858.

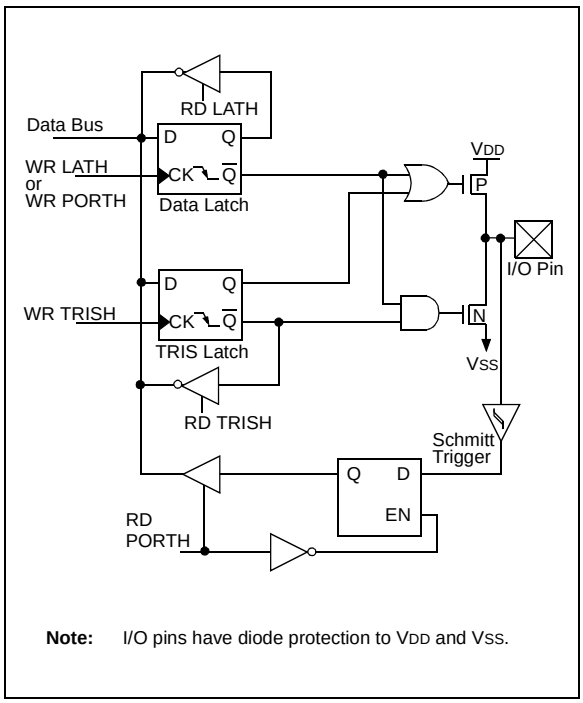
PORTH is a 5-bit wide, bi-directional port available only on the PIC18C858 devices. The corresponding Data Direction register is TRISH. Setting a TRISH bit (=1) will make the corresponding PORTH pin an input (i.e., put the corresponding output driver in a hi-impedance mode). Clearing a TRISH bit (=0) will make the corresponding PORTH pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATH register read and write the latched output value for PORTH.

Pins RH0-RH3 on the PIC18C858 are bi-directional I/O pins with ST input buffers. Pins RH4-RH7 on all devices are multiplexed with A/D converter inputs.

Note: On a Power-on Reset, the RH7:RH4 pins are configured as inputs and read as '0'.

FIGURE 8-16: RH3:RH0 PINS BLOCK DIAGRAM

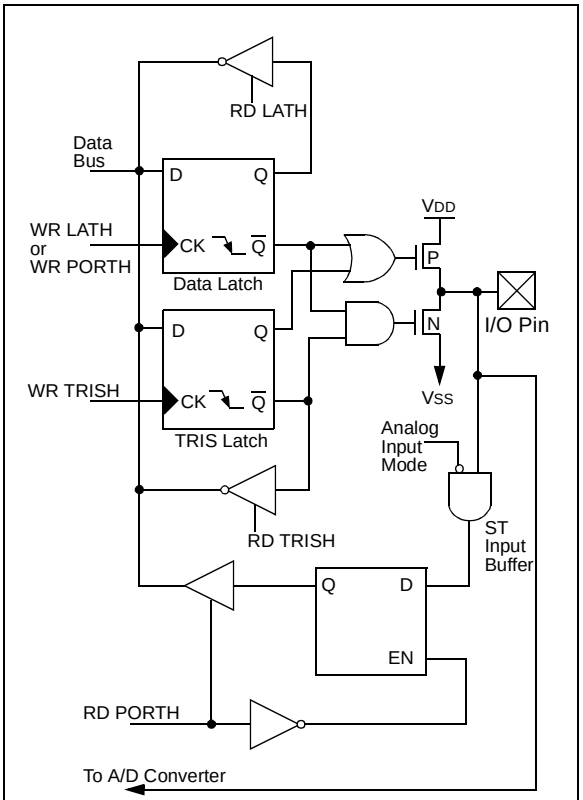


Note: I/O pins have diode protection to VDD and VSS.

EXAMPLE 8-8: INITIALIZING PORTH

```
CLRF    PORTH       ; Initialize PORTH by
                    ; clearing output
                    ; data latches
CLRF    LATH        ; Alternate method
                    ; to clear output
                    ; data latches
MOVLW   0x0F        ;
MOVWF   ADCON1      ;
MOVLW   0xCF        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISH       ; Set RH3:RH0 as inputs
                    ; RH5:RH4 as outputs
                    ; RH7:RH6 as inputs
```

FIGURE 8-17: RH7:RH4 PINS BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

14.0 CAPTURE/COMPARE/PWM (CCP) MODULES

Each CCP (Capture/Compare/PWM) module contains a 16-bit register that can operate as a 16-bit capture register, as a 16-bit compare register, or as a PWM Duty Cycle register. Table 14-1 shows the timer resources of the CCP module modes.

The operation of CCP1 is identical to that of CCP2, with the exception of the special event trigger and the CAN message timestamp received. (Refer to “CAN Module”, Section 17.0 for CAN operation.) Therefore, operation of a CCP module in the following sections is described with respect to CCP1.

Table 14-2 shows the interaction of the CCP modules.

Register 14-1 shows the CCPx Control registers (CCPxCON). For the CCP1 module, the register is called CCP1CON and for the CCP2 module, the register is called CCP2CON.

REGISTER 14-1: CCP1CON REGISTER
CCP2CON REGISTER

	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
	bit 7							bit 0

	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
	bit 7							bit 0

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

bit 5-4 **DCxB1:DCxB0**: PWM Duty Cycle bit1 and bit0

Capture Mode:
Unused

Compare Mode:
Unused

PWM Mode:
These bits are the two LSbs (bit1 and bit0) of the 10-bit PWM duty cycle. The upper eight bits (DCx9:DCx2) of the duty cycle are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets CCPx module)
0001 = Reserved
0010 = Compare mode, toggle output on match (CCPxIF bit is set)
0011 = Capture mode, CAN message received (CCP1 only)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode,
Initialize CCP pin Low, on compare match force CCP pin High (CCPIF bit is set)
1001 = Compare mode,
Initialize CCP pin High, on compare match force CCP pin Low (CCPIF bit is set)
1010 = Compare mode,
Generate software interrupt on compare match
(CCPIF bit is set, CCP pin is unaffected)
1011 = Compare mode,
Trigger special event (CCPIF bit is set, reset TMR1 or TMR3)
11xx = PWM mode

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

14.3.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit CCP1IE (PIE registers) clear to avoid false interrupts and should clear the flag bit CCP1IF, following any such change in operating mode.

14.3.4 CCP PRESCALER

There are four prescaler settings, specified by bits CCP1M3:CCP1M0. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. This means that any RESET will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared; therefore, the first capture may be from a non-zero prescaler. Example 14-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

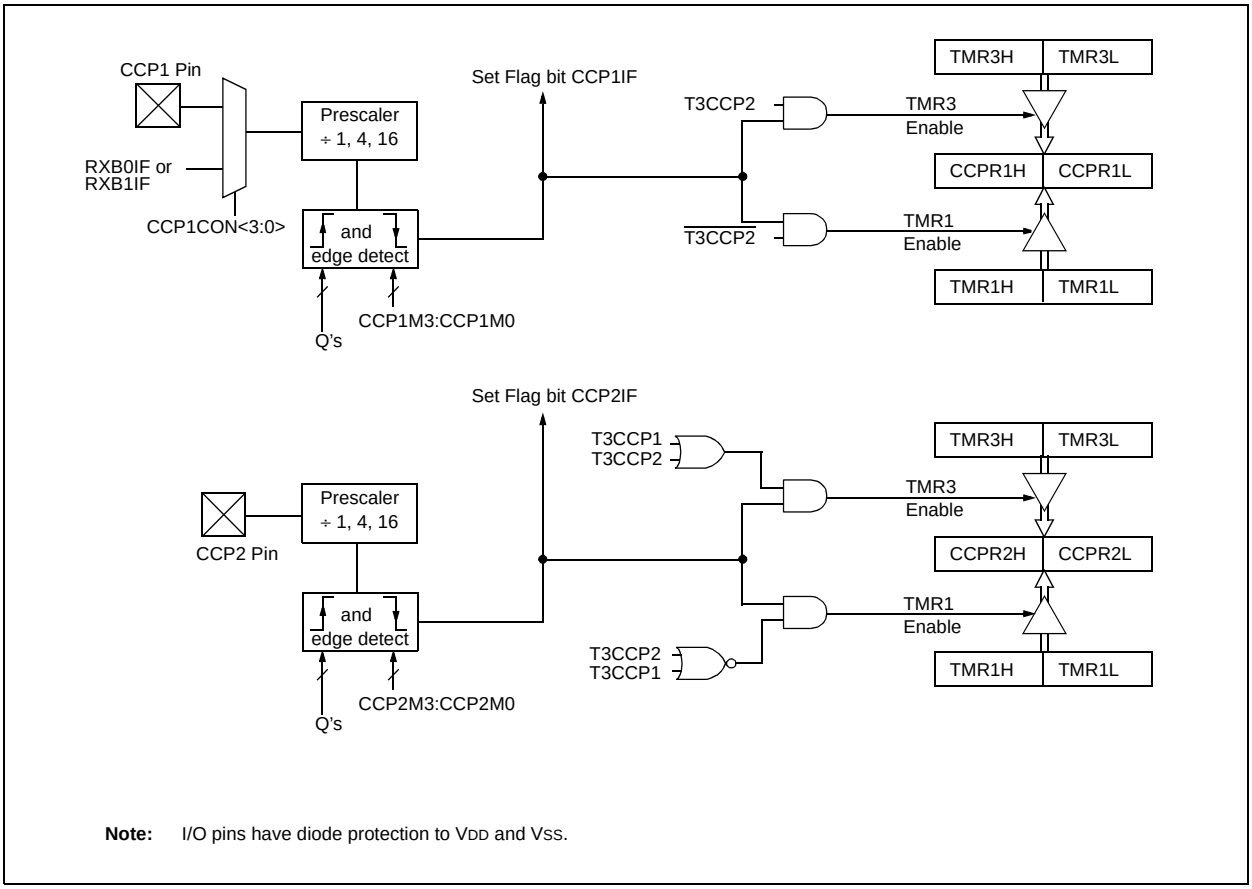
14.3.5 CAN MESSAGE RECEIVED

The CAN capture event occurs when a message is received in either receive buffer. The CAN module provides a rising edge to the CCP module to cause a capture event. This feature is provided to time-stamp the received CAN messages.

EXAMPLE 14-1: CHANGING BETWEEN CAPTURE PRESCALERS

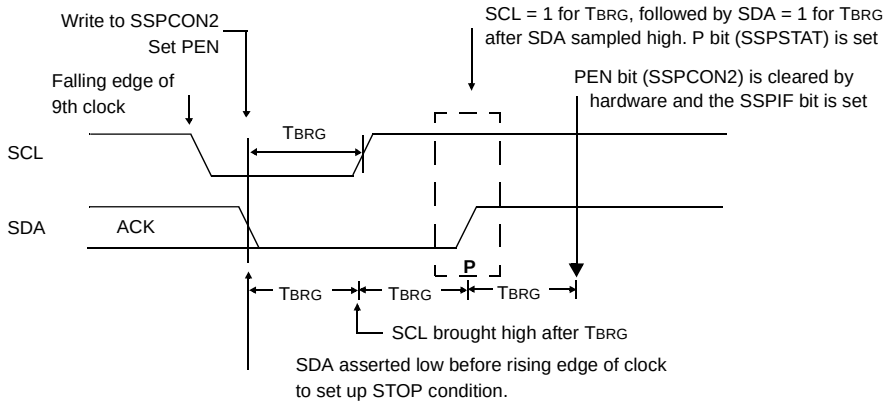
```
CLRF   CCP1CON, F  ; Turn CCP module off
MOVLW  NEW_CAPT_PS ; Load WREG with the
                   ; new prescaler mode
                   ; value and CCP ON
MOVWF  CCP1CON     ; Load CCP1CON with
                   ; this value
```

FIGURE 14-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

FIGURE 15-18: STOP CONDITION RECEIVE OR TRANSMIT MODE



Note: TBRG = one baud rate generator period.

16.2 USART Asynchronous Mode

In this mode, the USART uses standard non-return-to-zero (NRZ) format (one START bit, eight or nine data bits and one STOP bit). The most common data format is 8-bits. An on-chip dedicated 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The USART's transmitter and receiver are functionally independent, but use the same data format and baud rate. The baud rate generator produces a clock, either x16 or x64 of the bit shift rate, depending on the BRGH bit (TXSTA register). Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing the SYNC bit (TXSTA register).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

16.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 16-1. The heart of the transmitter is the Transmit (serial) Shift Register (TSR). The TSR register obtains its data from the Read/Write Transmit Buffer register (TXREG). The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit TXIF (PIR registers) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE registers). Flag bit TXIF will be set, regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicated the status of the TXREG register, another bit TRMT (TXSTA register) shows the status of the TSR register. Status bit TRMT is a read only bit, which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

Note 1:	The TSR register is not mapped in data memory, so it is not available to the user.
2:	Flag bit TXIF is set when enable bit TXEN is set.

Steps to follow when setting up an Asynchronous Transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 16.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit TXEN, which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

FIGURE 16-1: USART TRANSMIT BLOCK DIAGRAM

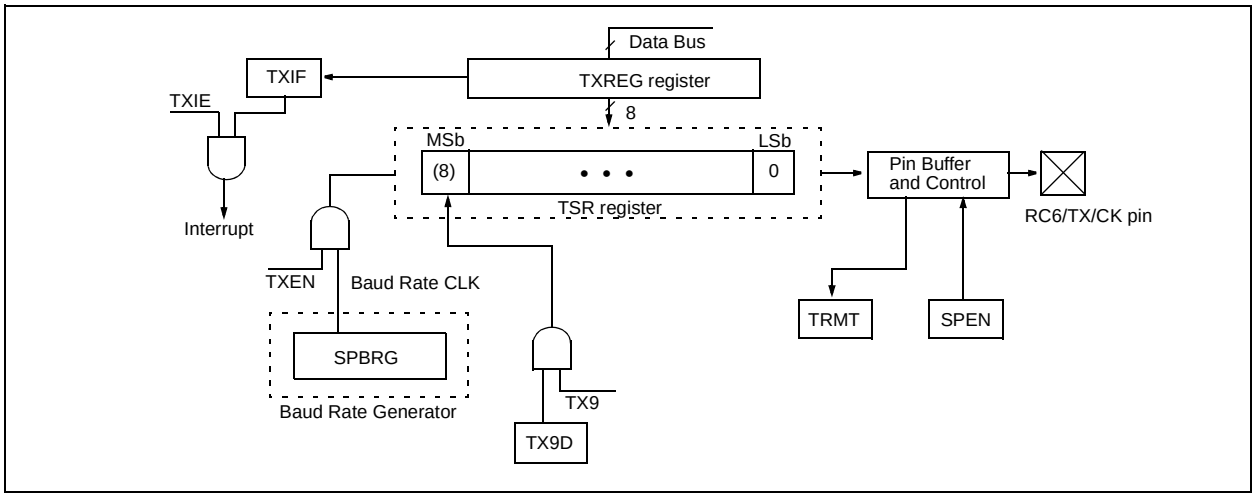


FIGURE 16-2: ASYNCHRONOUS TRANSMISSION

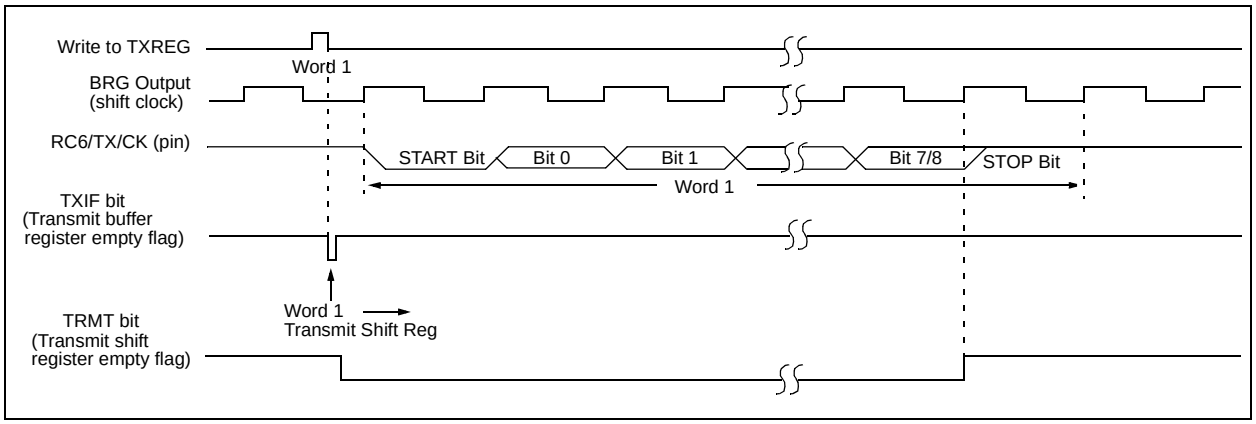
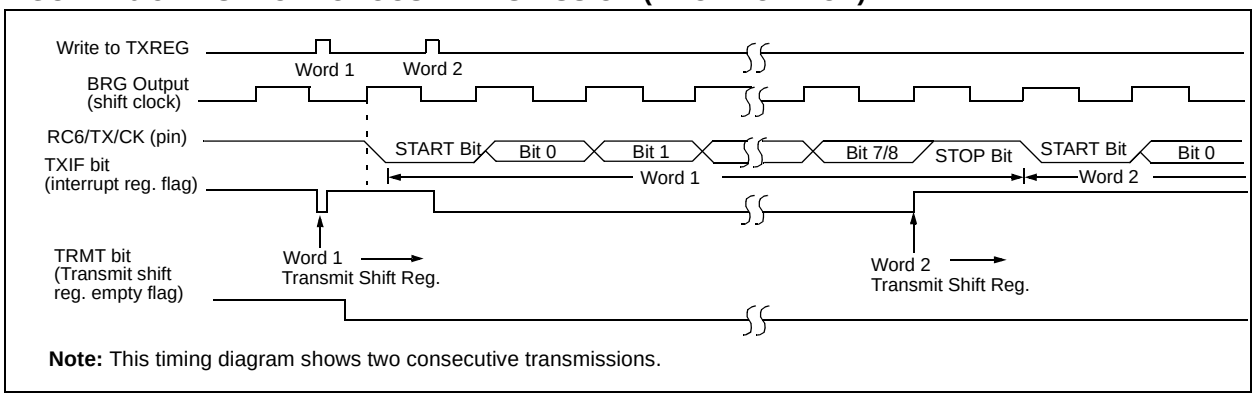


FIGURE 16-3: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 16-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	ADDEN	BRGH	TRMT	TX9D	0000 0010	0000 0010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'.
Shaded cells are not used for Asynchronous Transmission.

REGISTER 17-2: CANSTAT – CAN STATUS REGISTER

R-1	R-0	R-0	U-0	R-0	R-0	R-0	U-0
OPMODE2	OPMODE1	OPMODE0	—	ICODE2	ICODE1	ICODE0	—
bit 7							bit 0

bit 7-5 **OPMODE2:OPMODE0:** Operation Mode Status bits
111 = Reserved
110 = Reserved
101 = Reserved
100 = Configuration mode
011 = Listen Only mode
010 = Loopback mode
001 = Disable mode
000 = Normal mode

Note: Before the device goes into SLEEP mode, select Disable mode.

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

bit 3-1 **ICODE2:ICODE0:** Interrupt Code bits
When an interrupt occurs, a prioritized coded interrupt value will be present in the ICODE2:ICODE0 bits. These codes indicate the source of the interrupt. The ICODE2:ICODE0 bits can be copied to the WIN2:WIN0 bits to select the correct buffer to map into the Access Bank area. See Example 17-1 for code example.
111 = Wake-up on Interrupt
110 = RXB0 Interrupt
101 = RXB1 Interrupt
100 = TXB0 Interrupt
011 = TXB1 Interrupt
010 = TXB2 Interrupt
001 = Error Interrupt
000 = No Interrupt

bit 0 **Unimplemented:** Read as '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

REGISTER 17-6: TXBnSIDL – TRANSMIT BUFFER n STANDARD IDENTIFIER LOW BYTE REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDE	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier bits, if EXIDE = 0.
Extended Identifier bits EID20:EID18, if EXIDE = 1.

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

bit 3 **EXIDE:** Extended Identifier Enable bit
1 = Message will transmit Extended ID, SID10:SID0 becomes EID28:EID18
0 = Message will transmit Standard ID, EID17:EID0 are ignored

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

bit 1-0 **EID17:EID16:** Extended Identifier bits

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

REGISTER 17-7: TXBnEIDH – TRANSMIT BUFFER n EXTENDED IDENTIFIER HIGH BYTE REGISTER

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

bit 7-0 **EID15:EID8:** Extended Identifier bits

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

REGISTER 17-8: TXBnEIDL – TRANSMIT BUFFER n EXTENDED IDENTIFIER LOW BYTE REGISTER

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

bit 7-0 **EID7:EID0:** Extended Identifier bits

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

REGISTER 17-15: RXBnSIDL – RECEIVE BUFFER n STANDARD IDENTIFIER LOW BYTE REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	SRR	EXID	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier bits, if EXID = 0.
Extended Identifier bits EID20:EID18, if EXID = 1.

bit 4 **SRR:** Substitute Remove Request bit (only when EXID = '1')
1 = Remote transfer request occurred
0 = No remote transfer request occurred

bit 3 **EXID:** Extended Identifier bit
1 = Received message is an Extended Data Frame, SID10:SID0 are EID28:EID18
0 = Received message is a Standard Data Frame

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

bit 1-0 **EID17:EID16:** Extended Identifier bits

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

REGISTER 17-16: RXBnEIDH – RECEIVE BUFFER n EXTENDED IDENTIFIER HIGH BYTE REGISTER

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

bit 7-0 **EID15:EID8:** Extended Identifier bits

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

REGISTER 17-17: RXBnEIDL – RECEIVE BUFFER n EXTENDED IDENTIFIER LOW BYTE REGISTER

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

bit 7-0 **EID7:EID0:** Extended Identifier bits

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

17.3 CAN Modes of Operation

The PIC18CXX8 has the following modes of operation. These modes are:

- Configuration mode
- Disable mode
- Normal Operation mode
- Listen Only mode
- Loopback mode
- Error Recognition mode (selected through CANRXM bits)

Modes are requested by setting the REQOP bits, except the Error Recognition mode, which is requested through the CANRXM bits. Entry into a mode is acknowledged by monitoring the OPMODE bits.

When changing modes, the mode will not actually change until all pending message transmissions are complete. Because of this, the user must verify that the device has actually changed into the requested mode before further operations are executed.

17.3.1 CONFIGURATION MODE

The CAN module has to be initialized before the activation. This is only possible if the module is in the Configuration mode. The Configuration mode is requested by setting REQOP2 bit. Only when the status bit OPMODE2 has a high level, the initialization can be performed. Afterwards, the configuration registers and the acceptance mask registers and the acceptance filter registers can be written. The module is activated by setting the control bits CFGREQ to zero.

The module will protect the user from accidentally violating the CAN protocol through programming errors. All registers which control the configuration of the module can not be modified while the module is on-line. The CAN module will not be allowed to enter the Configuration mode while a transmission is taking place. The CONFIG bit serves as a lock to protect the following registers.

- Configuration registers
- Bus Timing registers
- Identifier Acceptance Filter registers
- Identifier Acceptance Mask registers

In the Configuration mode, the module will not transmit or receive. The error counters are cleared and the interrupt flags remain unchanged. The programmer will have access to configuration registers that are access restricted in other modes.

17.3.2 DISABLE MODE

In Disable mode, the module will not transmit or receive. The module has the ability to set the WAKIF bit due to bus activity, however, any pending interrupts will remain and the error counters will retain their value.

If REQOP<2:0> is set to 001, the module will enter the module Disable mode. This mode is similar to disabling other peripheral modules by turning off the module enables. This causes the module internal clock to stop unless the module is active (i.e., receiving or transmitting a message). If the module is active, the module will wait for 11 recessive bits on the CAN bus, detect that condition as an idle bus, then accept the module disable command. OPMODE<2:0>=001 indicates whether the module successfully went into module Disable mode

The WAKIF interrupt is the only module interrupt that is still active in the module Disable mode. If the WAKIE is set, the processor will receive an interrupt whenever the CAN bus detects a dominant state, as occurs with a SOF.

The I/O pins will revert to normal I/O function when the module is in the module Disable mode.

17.3.3 NORMAL MODE

This is the standard operating mode of the PIC18CXX8. In this mode, the device actively monitors all bus messages and generates acknowledge bits, error frames, etc. This is also the only mode in which the PIC18CXX8 will transmit messages over the CAN bus.

17.3.4 LISTEN ONLY MODE

Listen Only mode provides a means for the PIC18CXX8 to receive all messages, including messages with errors. This mode can be used for bus monitor applications, or for detecting the baud rate in ‘hot plugging’ situations. For auto-baud detection, it is necessary that there are at least two other nodes which are communicating with each other. The baud rate can be detected empirically by testing different values until valid messages are received. The Listen Only mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or acknowledge signals. The filters and masks can be used to allow only particular messages to be loaded into the receive registers, or the filter masks can be set to all zeros to allow a message with any identifier to pass. The error counters are reset and deactivated in this state. The Listen Only mode is activated by setting the mode request bits in the CANCON register.

17.6 Message Acceptance Filters and Masks

The Message Acceptance Filters and Masks are used to determine if a message in the message assembly buffer should be loaded into either of the receive buffers. Once a valid message has been received into the MAB, the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer. The filter masks are used to determine which bits in the identifier are examined with the filters. A truth table is shown below in Table 17-2 that indicates how each bit in the identifier is compared to the masks and filters to determine if a the message should be loaded into a receive buffer. The mask essentially determines which bits to apply the acceptance filters to. If any mask bit is set to a zero, then that bit will automatically be accepted, regardless of the filter bit.

TABLE 17-2: FILTER/MASK TRUTH TABLE

Mask bit n	Filter bit n	Message Identifier bit n001	Accept or Reject bit n
0	X	X	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Legend: X = don't care

As shown in the Receive Buffers Block Diagram (Figure 17-3), acceptance filters RXF0 and RXF1, and filter mask RXM0 are associated with RXB0. Filters RXF2, RXF3, RXF4, and RXF5 and mask RXM1 are associated with RXB1. When a filter matches and a message is loaded into the receive buffer, the filter number that enabled the message reception is loaded into the FILHIT bit(s). For RXB1, the RXB1CON register contains the FILHIT<2:0> bits. They are coded as follows:

- 101 = Acceptance Filter 5 (RXF5)
- 100 = Acceptance Filter 4 (RXF4)
- 011 = Acceptance Filter 3 (RXF3)
- 010 = Acceptance Filter 2 (RXF2)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0 (RXF0)

Note:	000 and 001 can only occur if the RXB0DBEN bit is set in the RXB0CON register, allowing RXB0 messages to roll over into RXB1.

The coding of the RXB0DBEN bit enables these three bits to be used similarly to the FILHIT bits and to distinguish a hit on filter RXF0 and RXF1, in either RXB0, or after a roll over into RXB1.

- 111 = Acceptance Filter 1 (RXF1)
- 110 = Acceptance Filter 0 (RXF0)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0

If the RXB0DBEN bit is clear, there are six codes corresponding to the six filters. If the RXB0DBEN bit is set, there are six codes corresponding to the six filters, plus two additional codes corresponding to RXF0 and RXF1 filters that roll over into RXB1.

If more than one acceptance filter matches, the FILHIT bits will encode the binary value of the lowest numbered filter that matched. In other words, if filter RXF2 and filter RXF4 match, FILHIT will be loaded with the value for RXF2. This essentially prioritizes the acceptance filters with a lower number filter having higher priority. Messages are compared to filters in ascending order of filter number.

The mask and filter registers can only be modified when the PIC18CXX8 is in Configuration mode. The mask and filter registers cannot be read outside of Configuration mode. When outside of Configuration mode, all mask and filter registers will be read as '0'.

17.8 Synchronization

To compensate for phase shifts between the oscillator frequencies of each of the nodes on the bus, each CAN controller must be able to synchronize to the relevant signal edge of the incoming signal. When an edge in the transmitted data is detected, the logic will compare the location of the edge to the expected time (Sync Seg). The circuit will then adjust the values of phase segment 1 and phase segment 2, as necessary. There are two mechanisms used for synchronization.

17.8.1 HARD SYNCHRONIZATION

Hard Synchronization is only done when there is a recessive to dominant edge during a BUS IDLE condition, indicating the start of a message. After hard synchronization, the bit time counters are restarted with Sync Seg. Hard synchronization forces the edge, which has occurred to lie within the synchronization segment of the restarted bit time. Due to the rules of synchronization, if a hard synchronization occurs, there will not be a resynchronization within that bit time.

17.8.2 RESYNCHRONIZATION

As a result of Resynchronization, phase segment 1 may be lengthened, or phase segment 2 may be shortened. The amount of lengthening or shortening of the phase buffer segments has an upper bound given by the Synchronization Jump Width (SJW). The value of the SJW will be added to phase segment 1 (see Figure 17-7), or subtracted from phase segment 2 (see Figure 17-8). The SJW is programmable between 1 T_Q and 4 T_Q.

Clocking information will only be derived from recessive to dominant transitions. The property that only a fixed maximum number of successive bits have the same value, ensures resynchronization to the bit stream during a frame.

The phase error of an edge is given by the position of the edge relative to Sync Seg, measured in T_Q. The phase error is defined in magnitude of T_Q as follows:

- e = 0 if the edge lies within SYNCESEG.
- e > 0 if the edge lies before the SAMPLE POINT.
- e < 0 if the edge lies after the SAMPLE POINT of the previous bit.

If the magnitude of the phase error is less than, or equal to, the programmed value of the synchronization jump width, the effect of a resynchronization is the same as that of a hard synchronization.

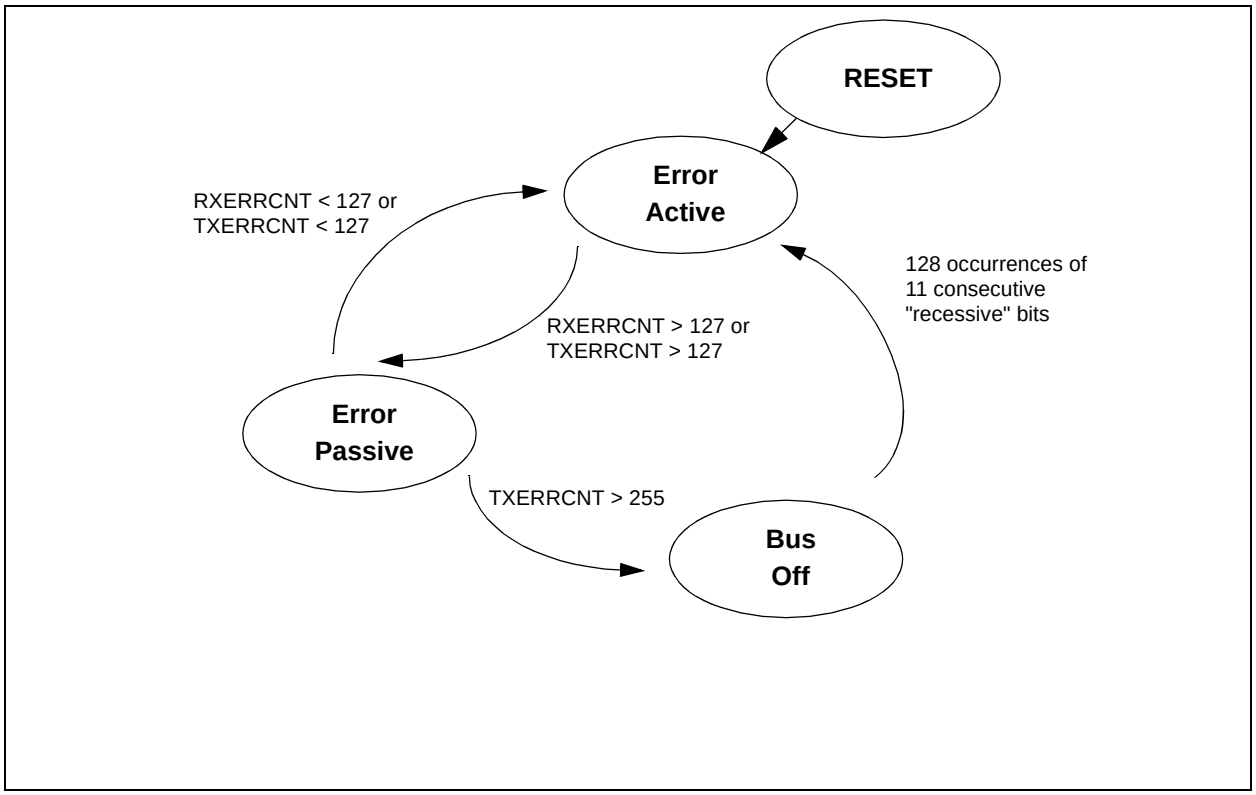
If the magnitude of the phase error is larger than the synchronization jump width, and if the phase error is positive, then phase segment 1 is lengthened by an amount equal to the synchronization jump width.

If the magnitude of the phase error is larger than the resynchronization jump width, and if the phase error is negative, then phase segment 2 is shortened by an amount equal to the synchronization jump width.

17.8.3 SYNCHRONIZATION RULES

- Only one synchronization within one bit time is allowed.
- An edge will be used for synchronization only if the value detected at the previous sample point (previously read bus value) differs from the bus value immediately after the edge.
- All other recessive to dominant edges, fulfilling rules 1 and 2, will be used for resynchronization with the exception that a node transmitting a dominant bit will not perform a resynchronization, as a result of a recessive to dominant edge with a positive phase error.

FIGURE 17-9: ERROR MODES STATE DIAGRAM



MOVLW **Move literal to WREG**

Syntax: [*label*] MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow$ WREG

Status Affected: None

Encoding:

0000	1110	kkkk	kkkk

Description: The eight bit literal 'k' is loaded into WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `MOVLW 0x5A`

After Instruction
WREG = 0x5A

MOVWF **Move WREG to f**

Syntax: [*label*] MOVWF f [,a]

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

Operation: (WREG) $\rightarrow$ f

Status Affected: None

Encoding:

0110	111a	ffff	ffff

Description: Move data from WREG to register 'f'. Location 'f' can be anywhere in the 256 byte Bank. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `MOVWF REG`

Before Instruction
WREG = 0x4F
REG = 0xFF

After Instruction
WREG = 0x4F
REG = 0x4F

MULLW **Multiply Literal with WREG**

Syntax: [*label*] MULLW k

Operands: $0 \le k \le 255$

Operation: (WREG) x k → PRODH:PRODL

Status Affected: None

Encoding:

0000	1101	kkkk	kkkk

Description: An unsigned multiplication is carried out between the contents of WREG and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte.

WREG 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 0xC4`

Before Instruction
- WREG = 0xE2
- PRODH = ?
- PRODL = ?

After Instruction
- WREG = 0xE2
- PRODH = 0xAD
- PRODL = 0x08

MULWF **Multiply WREG with f**

Syntax: [*label*] MULWF f [,a]

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

Operation: (WREG) x (f) → PRODH:PRODL

Status Affected: None

Encoding:

0000	001a	ffff	ffff

Description: An unsigned multiplication is carried out between the contents of WREG and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte.

Both WREG 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 will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

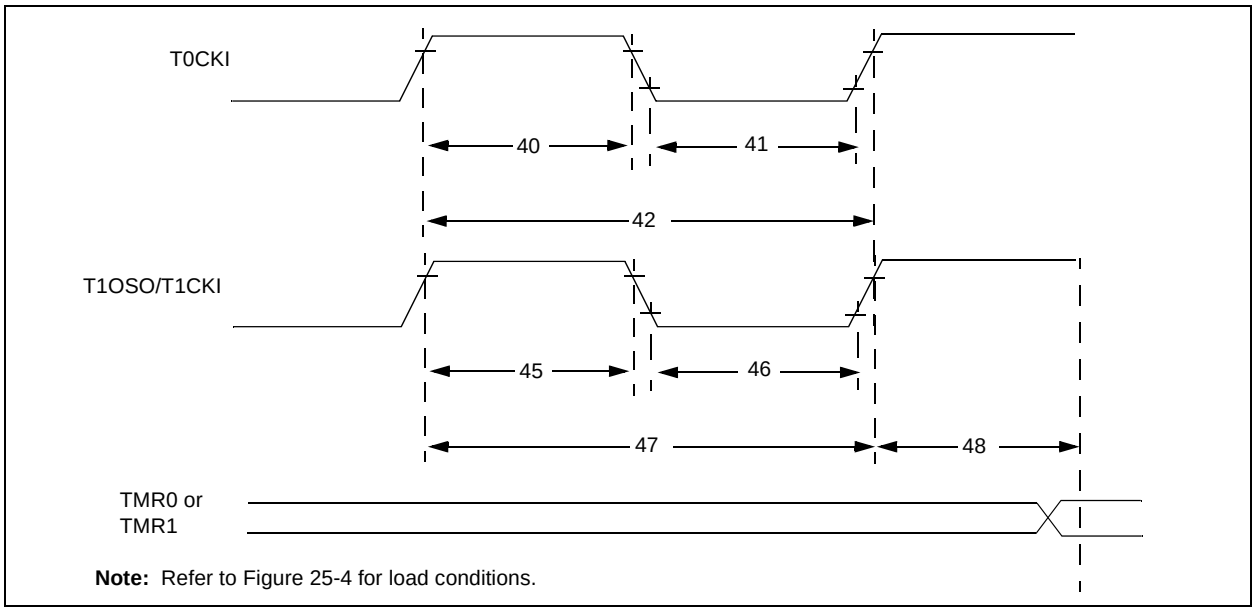
Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

Example: `MULWF REG`

Before Instruction
- WREG = 0xC4
- REG = 0xB5
- PRODH = ?
- PRODL = ?

After Instruction
- WREG = 0xC4
- REG = 0xB5
- PRODH = 0x8A
- PRODL = 0x94

FIGURE 25-9: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS



Note: Refer to Figure 25-4 for load conditions.

TABLE 25-8: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param No.	Symbol	Characteristic			Min	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width		No Prescaler	0.5TCY + 20	—	ns	
				With Prescaler	10	—	ns	
41	Tt0L	T0CKI Low Pulse Width		No Prescaler	0.5TCY + 20	—	ns	
				With Prescaler	10	—	ns	
42	Tt0P	T0CKI Period		No Prescaler	TCY + 10	—	ns	
				With Prescaler	Greater of: 20 ns or TCY + 40 / N	—	ns	N = prescale value (1, 2, 4,..., 256)
45	Tt1H	T1CKI High Time	Synchronous, no prescaler		0.5TCY + 20	—	ns	
			Synchronous, with prescaler	PIC18**C**XX8	10	—	ns	
				PIC18**LC**XX8	25	—	ns	
			Asynchronous	PIC18**C**XX8	30	—	ns	
				PIC18**LC**XX8	50	—	ns	
46	Tt1L	T1CKI Low Time	Synchronous, no prescaler		0.5TCY + 5	—	ns	
			Synchronous, with prescaler	PIC18**C**XX8	10	—	ns	
				PIC18**LC**XX8	25	—	ns	
			Asynchronous	PIC18**C**XX8	30	—	ns	
				PIC18**LC**XX8	TBD	TBD	ns	
47	Tt1P	T1CKI Input Period	Synchronous		Greater of: 20 ns or TCY + 40 / N	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous		60	—	ns	
	Ft1	T1CKI oscillator input frequency range			DC	50	kHz	
48	Tcke2tmrl	Delay from external T1CKI clock edge to timer increment			2Tosc	7Tosc	—	

