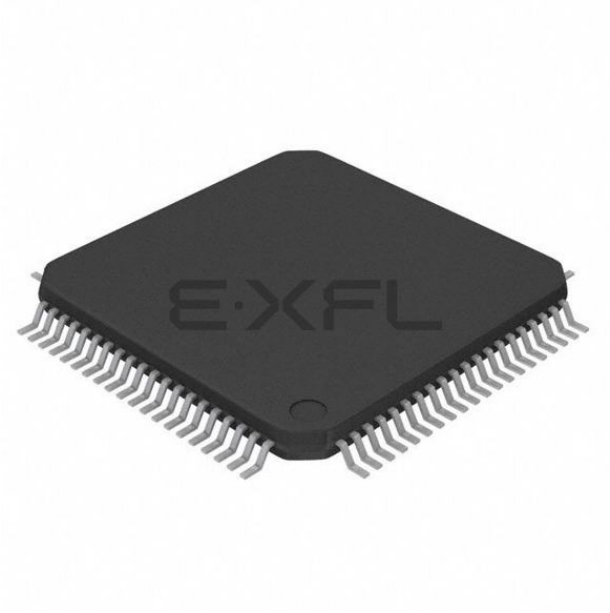


Welcome to E-XFL.COM

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	EBI/EMI, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	70
Program Memory Size	96KB (48K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f8627t-e-pt

TABLE 1-4: PIC18F8527/8622/8627/8722 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number TQFP	Pin Type	Buffer Type	Description
				PORTH is a bidirectional I/O port.
RH0/A16	79			
RH0		I/O	ST	Digital I/O.
A16		I/O	TTL	External memory address/data 16.
RH1/A17	80			
RH1		I/O	ST	Digital I/O.
A17		I/O	TTL	External memory address/data 17.
RH2/A18	1			
RH2		I/O	ST	Digital I/O.
A18		I/O	TTL	External memory address/data 18.
RH3/A19	2			
RH3		I/O	ST	Digital I/O.
A19		I/O	TTL	External memory address/data 19.
RH4/AN12/P3C	22			
RH4		I/O	ST	Digital I/O.
AN12		I	Analog	Analog input 12.
P3C(5)		O	—	ECCP3 PWM output C.
RH5/AN13/P3B	21			
RH5		I/O	ST	Digital I/O.
AN13		I	Analog	Analog input 13.
P3B(5)		O	—	ECCP3 PWM output B.
RH6/AN14/P1C	20			
RH6		I/O	ST	Digital I/O.
AN14		I	Analog	Analog input 14.
P1C(5)		O	—	ECCP1 PWM output C.
RH7/AN15/P1B	19			
RH7		I/O	ST	Digital I/O.
AN15		I	Analog	Analog input 15.
P1B(5)		O	—	ECCP1 PWM output B.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; I^2C™/SMB = I^2C/SMBus input buffer

Note 1: Alternate assignment for ECCP2 when Configuration bit, CCP2MX, is cleared (all operating modes except Microcontroller mode).

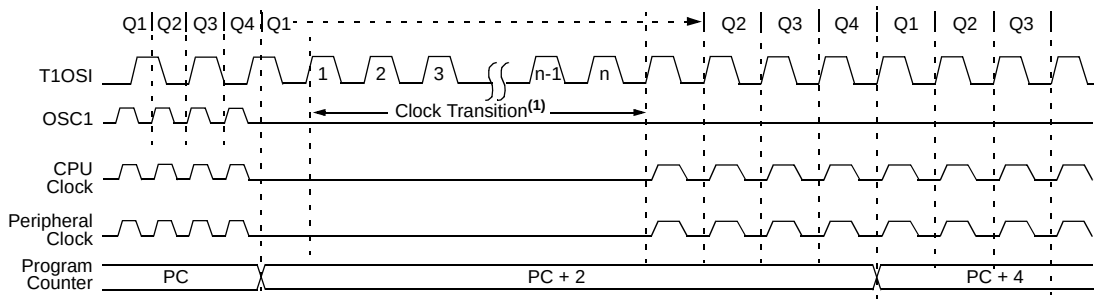
2: Default assignment for ECCP2 in all operating modes (CCP2MX is set).

3: Alternate assignment for ECCP2 when CCP2MX is cleared (Microcontroller mode only).

4: Default assignment for P1B/P1C/P3B/P3C (ECCPMX is set).

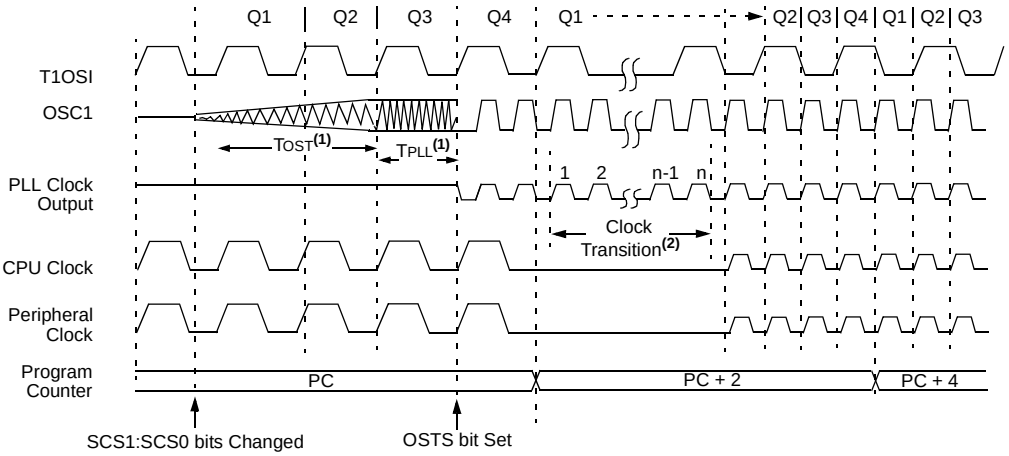
5: Alternate assignment for P1B/P1C/P3B/P3C (ECCPMX is clear).

FIGURE 3-1: TRANSITION TIMING FOR ENTRY TO SEC_RUN MODE



Note 1: Clock transition typically occurs within 2-4 TOSC.

FIGURE 3-2: TRANSITION TIMING FROM SEC_RUN MODE TO PRI_RUN MODE (HSPLL)



Note 1: TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

2: Clock transition typically occurs within 2-4 TOSC.

3.2.3 RC_RUN MODE

In RC_RUN mode, the CPU and peripherals are clocked from the internal oscillator block using the INTOSC multiplexer. In this mode, the primary clock is shut down. When using the INTRC source, this mode provides the best power conservation of all the Run modes, while still executing code. It works well for user applications which are not highly timing-sensitive or do not require high-speed clocks at all times.

If the primary clock source is the internal oscillator block (either INTRC or INTOSC), there are no distinguishable differences between PRI_RUN and RC_RUN modes during execution. However, a clock switch delay will occur during entry to and exit from RC_RUN mode. Therefore, if the primary clock source is the internal oscillator block, the use of RC_RUN mode is not recommended.

This mode is entered by setting the SCS1 bit to ‘1’. Although it is ignored, it is recommended that the SCS0 bit also be cleared; this is to maintain software compatibility with future devices. When the clock source is switched to the INTOSC multiplexer (see Figure 3-3), the primary oscillator is shut down and the OSTS bit is cleared. The IRCF bits may be modified at any time to immediately change the clock speed.

Note:	Caution should be used when modifying a single IRCF bit. If VDD is less than 3V, it is possible to select a higher clock speed than is supported by the low VDD. Improper device operation may result if the VDD/FOSC specifications are violated.

7.2 Address and Data Width

PIC18F8527/8622/8627/8722 devices can be independently configured for different address and data widths on the same memory bus. Both address and data width are set by Configuration bits in the CONFIG3L register. As Configuration bits, this means that these options can only be configured by programming the device and are not controllable in software.

The BW bit selects an 8-bit or 16-bit data bus width. Setting this bit (default) selects a data width of 16 bits.

The ADW<1:0> bits determine the address bus width. The available options are 20-bit (default), 16-bit, 12-bit and 8-bit. Selecting any of the options other than 20-bit width makes a corresponding number of high-order lines available for I/O functions; these pins are no longer affected by the setting of the EBDIS bit. For example, selecting a 16-bit Address mode (ADW<1:0> = `10`) disables A<19:16> and allows PORTH<3:0> to function without interruptions from the bus. Using smaller address widths allows users to tailor the memory bus to the size of the external memory space for a particular design while freeing up pins for dedicated I/O operation.

Because the ADW bits have the effect of disabling pins for memory bus operations, it is important to always select an address width at least equal to the data width. If 8-bit or 12-bit address widths are used with a 16-bit data width, the upper bits of data will not be available on the bus.

All combinations of address and data widths require multiplexing of address and data information on the same lines. The address and data multiplexing, as well as I/O ports made available by the use of smaller address widths, are summarized in Table 7-2.

7.2.1 21-BIT ADDRESSING

As an extension of 20-bit address width operation, the External Memory Bus can also fully address a 2 Mbyte memory space. This is done by using the Bus Address bit 0 (BA0) control line as the Least Significant bit of the address. The $\overline{UB}$ and $\overline{LB}$ control signals may also be used with certain memory devices to select the upper and lower bytes within a 16-bit wide data word.

This addressing mode is available in both 8-bit and certain 16-bit Data Width modes. Additional details are provided in **Section 7.5.3 “16-bit Byte Select Mode”** and **Section 7.6 “8-Bit Data Width Modes”**.

7.3 Wait States

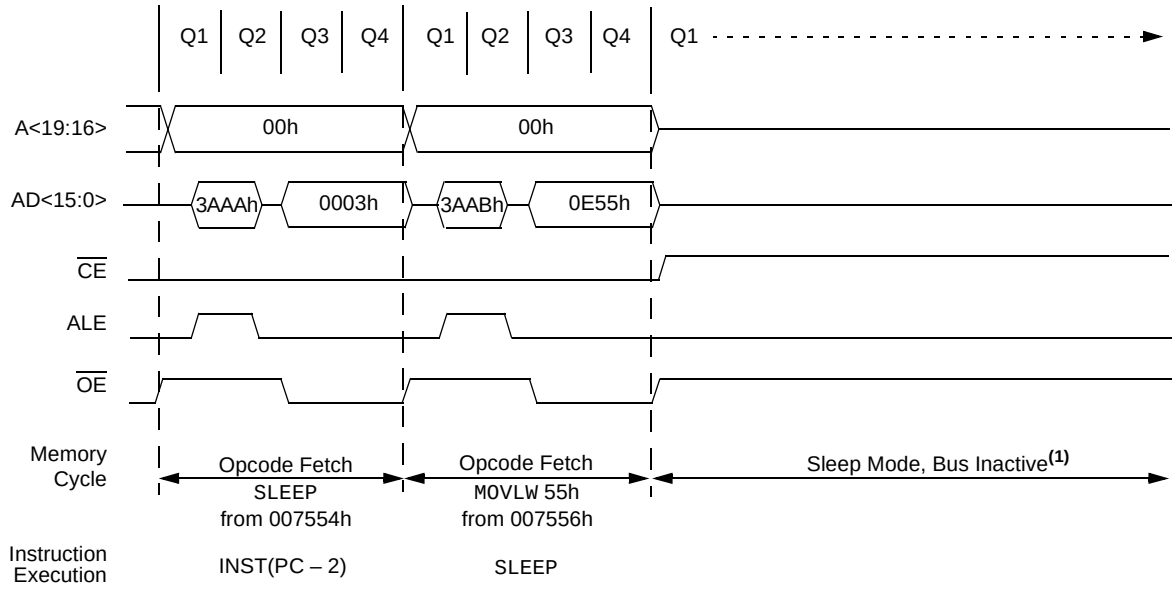
While it may be assumed that external memory devices will operate at the microcontroller clock rate, this is often not the case. In fact, many devices require longer times to write or retrieve data than the time allowed by the execution of table read or table write operations.

To compensate for this, the External Memory Bus can be configured to add a fixed delay to each table operation using the bus. Wait states are enabled by setting the WAITx bit. When enabled, the amount of delay is set by the WAIT<1:0> bits (MEMCON<5:4>). The delay is based on multiples of microcontroller instruction cycle time and are added following the instruction cycle when the table operation is executed. The range is from no delay to 3 TCY (default value).

TABLE 7-2: ADDRESS AND DATA LINES FOR DIFFERENT ADDRESS AND DATA WIDTHS

Data Width	Address Width	Multiplexed Data and Address Lines (and Corresponding Ports)	Address-Only Lines (and Corresponding Ports)	Ports Available for I/O
8-bit	8-bit	AD<7:0> (PORTD<7:0>)	—	All of PORTE and PORTH
	12-bit		AD<11:8> (PORTE<3:0>)	PORTE<7:4>, All of PORTH
	16-bit		AD<15:8> (PORTE<7:0>)	All of PORTH
	20-bit		A<19:16>, AD<15:8> (PORTH<3:0>, PORTE<7:0>)	—
16-bit	16-bit	AD<15:0> (PORTD<7:0>, PORTE<7:0>)	—	All of PORTH
	20-bit		A<19:16> (PORTH<3:0>)	—

FIGURE 7-6: EXTERNAL MEMORY BUS TIMING FOR SLEEP (MICROPROCESSOR MODE)



Note 1: Bus becomes inactive regardless of power-managed mode entered when SLEEP is executed.

10.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Priority registers (IPR1, IPR2, IPR3). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

REGISTER 10-10: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
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 **PSPIP:** Parallel Slave Port Read/Write Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **ADIP:** A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RC1IP:** EUSART1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TX1IP:** EUSART1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSP1IP:** MSSP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP:** ECCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 10-11: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

R/W-1	R/W-1	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
OSCFIP	CMIP	—	EEIP	BCL1IP	HLVDIP	TMR3IP	CCP2IP
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 **OSCFIP:** Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **CMIP:** Comparator Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **EEIP:** Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **BCL1IP:** MSSP1 Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **HLVDIP:** High/Low-Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP:** TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP2IP:** ECCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

18.1 ECCP Outputs and Configuration

Each of the Enhanced CCP modules may have up to four PWM outputs, depending on the selected operating mode. These outputs, designated PxA through PxD, are multiplexed with various I/O pins. Some ECCPx pin assignments are constant, while others change based on device configuration. For those pins that do change, the controlling bits are:

- CCP2MX Configuration bit (CONFIG3H<0>)
- ECCPMX Configuration bit (CONFIG3H<1>)
- Program Memory mode (set by Configuration bits, CONFIG3L<1:0>)

The pin assignments for the Enhanced CCP modules are summarized in Table 18-1, Table 18-2 and Table 18-3. To configure the I/O pins as PWM outputs, the proper PWM mode must be selected by setting the PxMx and CCPxMx bits (CCPxCON<7:6> and <3:0>, respectively). The appropriate TRIS direction bits for the corresponding port pins must also be set as outputs.

18.1.1 USE OF CCP4 AND CCP5 WITH ECCP1 AND ECCP3

Only the ECCP2 module has four dedicated output pins available for use. Assuming that the I/O ports or other multiplexed functions on those pins are not needed, they may be used whenever needed without interfering with any other CCP module.

ECCP1 and ECCP3, on the other hand, only have three dedicated output pins: ECCPx/P3A, PxB and PxC. Whenever these modules are configured for Quad PWM mode, the pin used for CCP4 or CCP5 takes priority over the D output pins for ECCP3 and ECCP1, respectively.

18.1.2 ECCP MODULE OUTPUTS, PROGRAM MEMORY MODES AND EMB ADDRESS BUS WIDTH

For PIC18F8527/8622/8627/8722 devices, the program memory mode of the device (**Section 7.2 “Address and Data Width”** and **Section 7.4 “Program Memory Modes and the External Memory Bus”**) impacts both pin multiplexing and the operation of the module.

The ECCP2 input/output (ECCP2/P2A) can be multiplexed to one of three pins. By default, this is RC1 for all devices; in this case, the default is in effect when CCP2MX is set and the device is operating in Microcontroller mode. With PIC18F8527/8622/8627/8722 devices, three other options exist. When CCP2MX is not set (= `0`) and the device is in Microcontroller mode, ECCP2/P2A is multiplexed to RE7; in all other program memory modes, it is multiplexed to RB3.

Another option is for ECCPMX to be set while the device is operating in one of the three other program memory modes. In this case, ECCP1 and ECCP3 operate as compatible (i.e., single output) CCP modules. The pins used by their other outputs (PxB through PxD) are available for other multiplexed functions. ECCP2 continues to operate as an Enhanced CCP module regardless of the program memory mode.

The final option is that the ABW<1:0> Configuration bits can be used to select 8, 12, 16 or 20-bit EMB addressing. Pins not assigned to EMB address pins are available for peripheral or port functions.

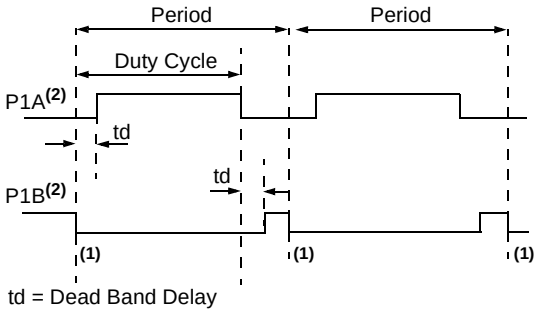
18.4.4 HALF-BRIDGE MODE

In the Half-Bridge Output mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the P1A pin, while the complementary PWM output signal is output on the P1B pin (Figure 18-4). This mode can be used for half-bridge applications, as shown in Figure 18-5, or for full-bridge applications, where four power switches are being modulated with two PWM signals.

In Half-Bridge Output mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of bits, P1DC<6:0> sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See **Section 18.4.6 "Programmable Dead-Band Delay"** for more details on dead-band delay operations.

The P1A and P1B outputs are multiplexed with the PORTC<2> and PORTE<6> data latches. Alternatively, P1B can be assigned to PORTH<7> by programming the ECCPMX Configuration bit to '0'. See Table 18-1, Table 18-2 and Table 18-3 for more information. The associated TRIS bit must be cleared to configure P1A and P1B as outputs.

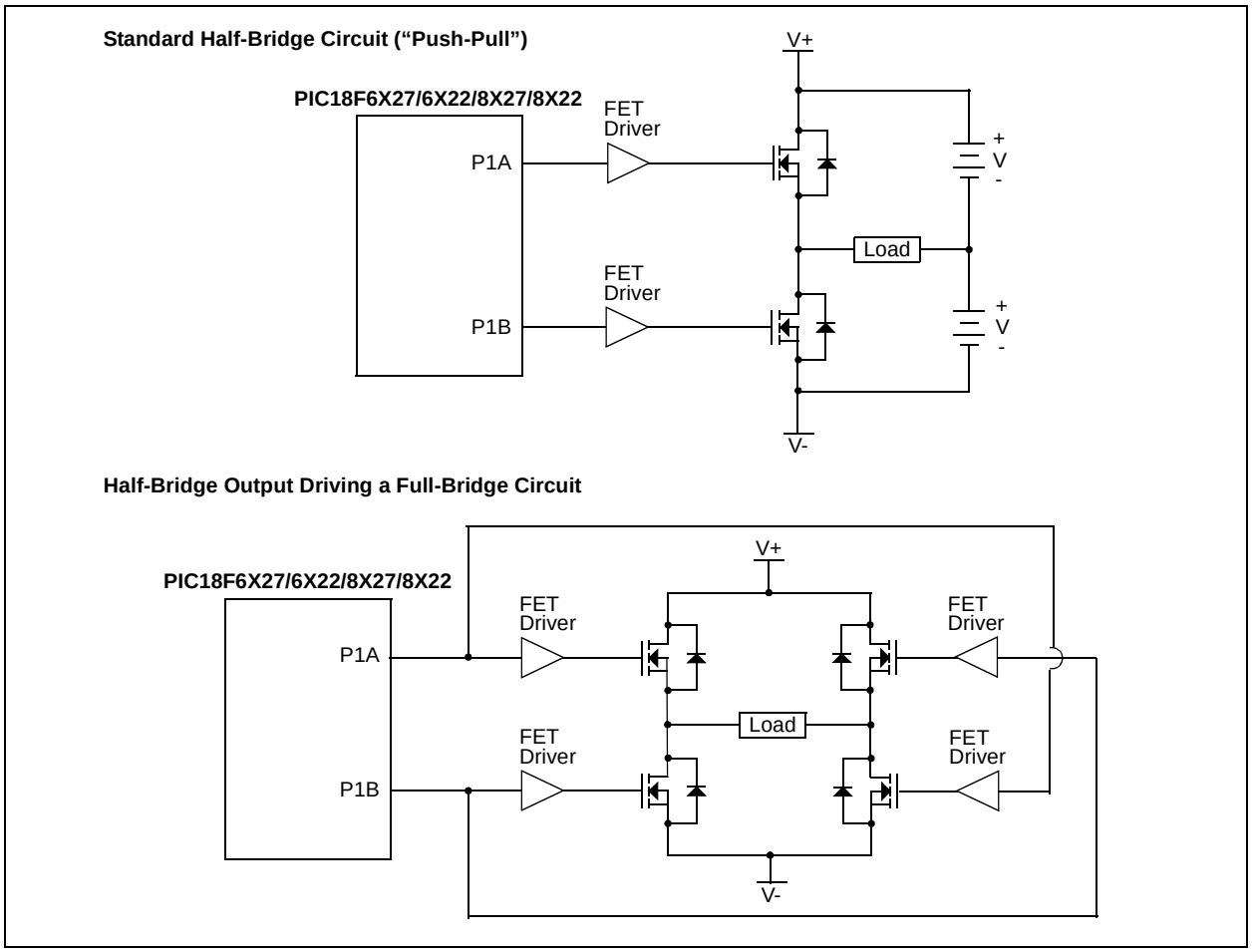
FIGURE 18-4: HALF-BRIDGE PWM OUTPUT



Note 1: At this time, the TMR2 register is equal to the PR2 register.

2: Output signals are shown as active-high.

FIGURE 18-5: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS



19.3.1 REGISTERS

Each MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register 1 (SSPxCON1)
- MSSP Status Register (SSPxSTAT)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSP Shift Register (SSPxSR) – Not directly accessible

SSPxCON1 and SSPxSTAT are the control and status registers in SPI mode operation. The SSPxCON1 register is readable and writable. The lower 6 bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

SSPxSR is the shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPxSR and SSPxBUF together create a double-buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

REGISTER 19-1: SSPxSTAT: MSSPx STATUS REGISTER (SPI MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
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 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Select bit
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state

Note: Polarity of clock state is set by the CKP bit (SSPxCON1<4>).

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit
Used in I^2C mode only.

bit 4 **P:** Stop bit
Used in I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

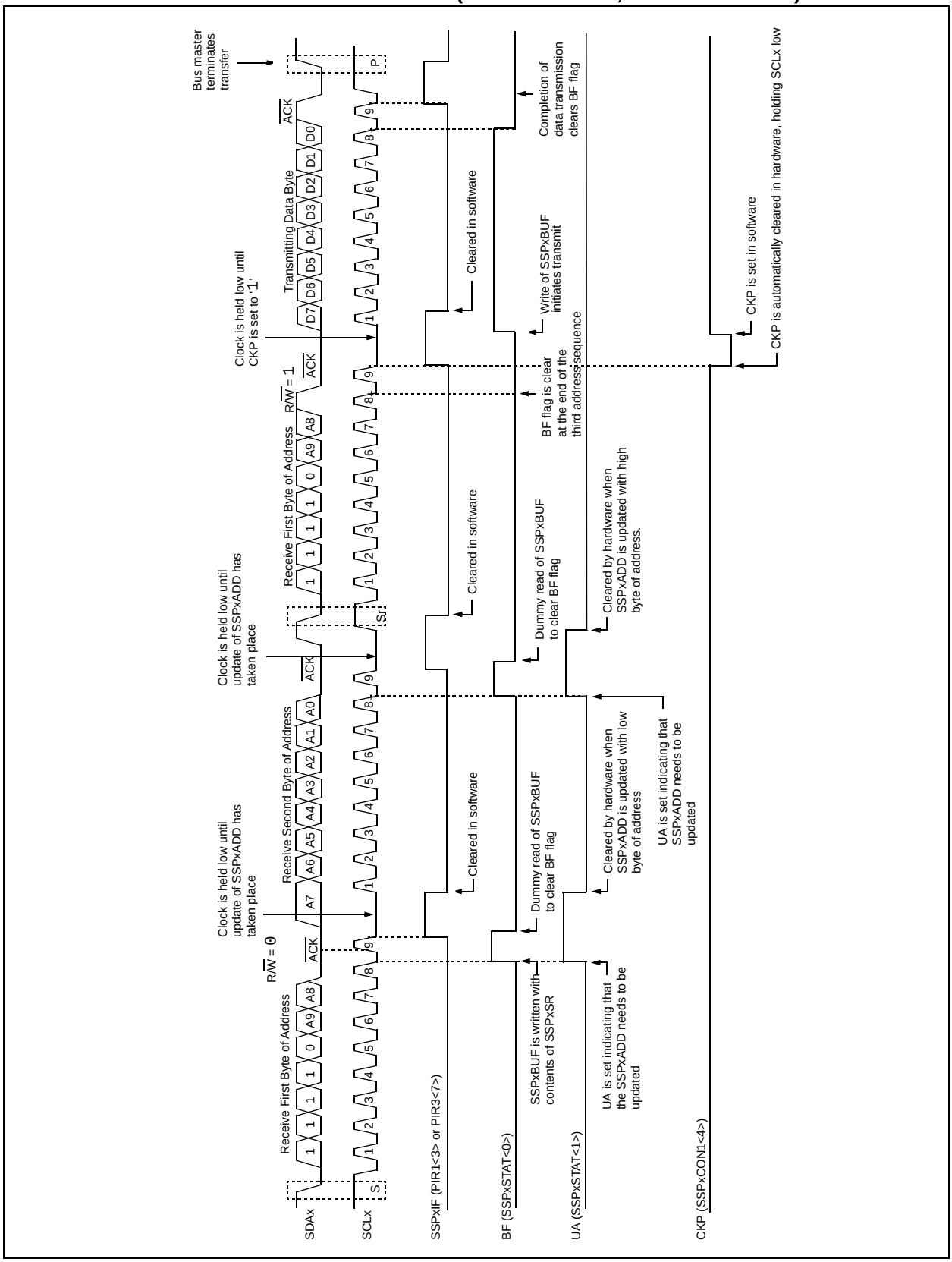
bit 3 **S:** Start bit
Used in I^2C mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{\text{Write}}$ Information bit
Used in I^2C mode only.

bit 1 **UA:** Update Address bit
Used in I^2C mode only.

bit 0 **BF:** Buffer Full Status bit (Receive mode only)
1 = Receive complete, SSPxBUF is full
0 = Receive not complete, SSPxBUF is empty

FIGURE 19-11: I²C™ SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)



19.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPxCON1 and by setting the SSPEN bit. In Master mode, the SCLx and SDAx lines are manipulated by the MSSP hardware if the TRIS bits are set.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit is set, or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all I^2C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

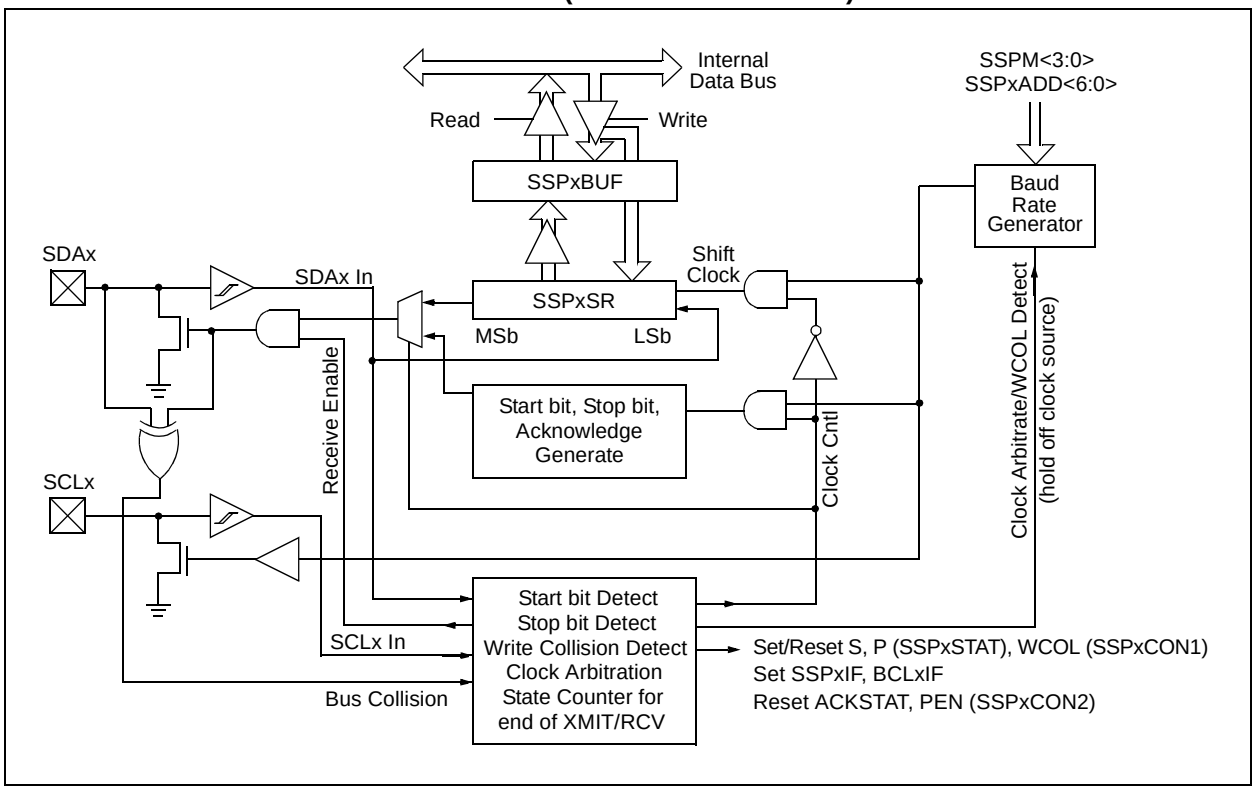
1. Assert a Start condition on SDAx and SCLx.
2. Assert a Repeated Start condition on SDAx and SCLx.
3. Write to the SSPxBUF register initiating transmission of data/address.
4. Configure the I^2C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDAx and SCLx.

> **Note:** The MSSP module, when configured in I^2C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur.

The following events will cause the SSP Interrupt Flag bit, SSPxIF, to be set (and SSP interrupt, if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmit
- Repeated Start

FIGURE 19-16: MSSP BLOCK DIAGRAM (I^2C™ MASTER MODE)



19.4.7.2 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts the SCLx pin (SCLx allowed to float high). When the SCLx pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCLx pin is actually sampled high. When the SCLx pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<6:0> and begins counting. This ensures that the SCLx high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 19-18).

FIGURE 19-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION

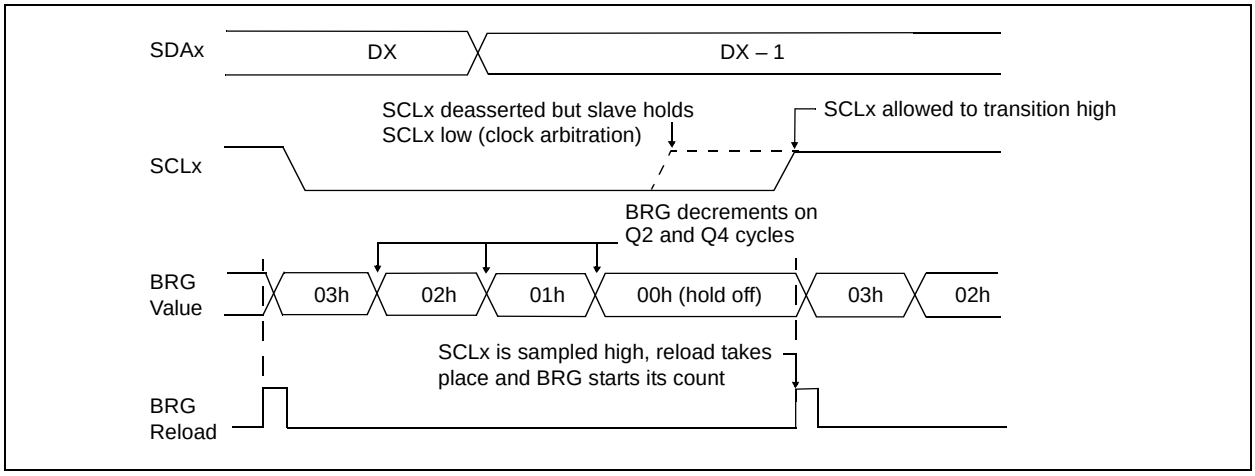


FIGURE 19-27: BUS COLLISION DURING START CONDITION (SCLx = 0)

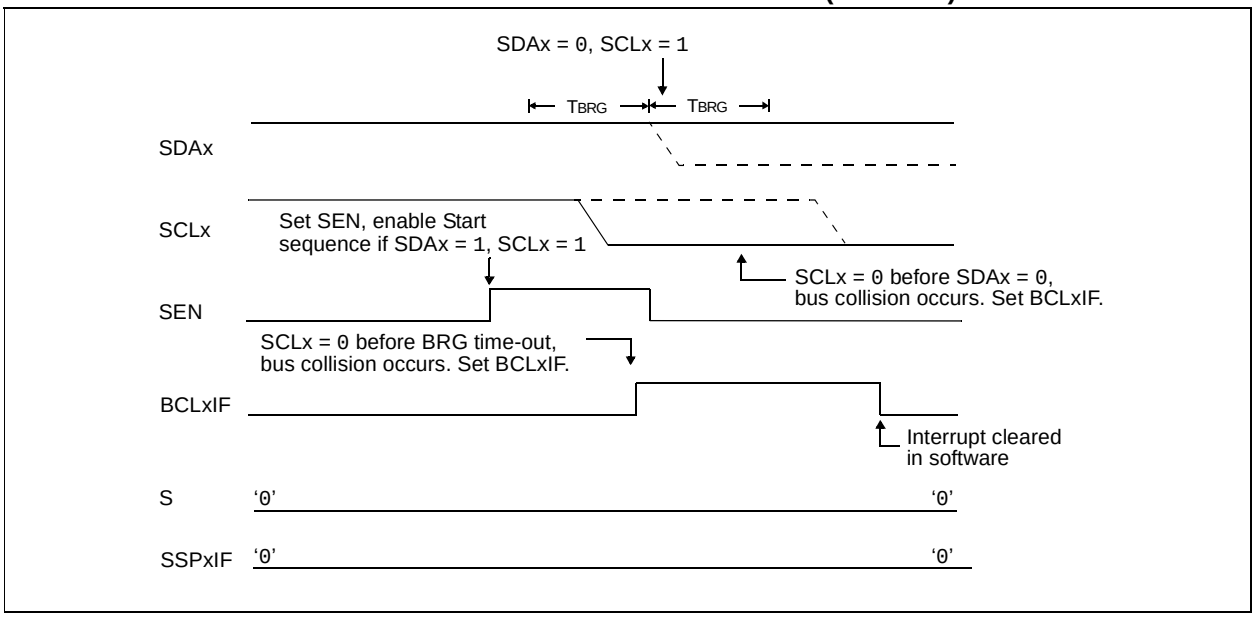


FIGURE 19-28: BRG RESET DUE TO SDAx ARBITRATION DURING START CONDITION

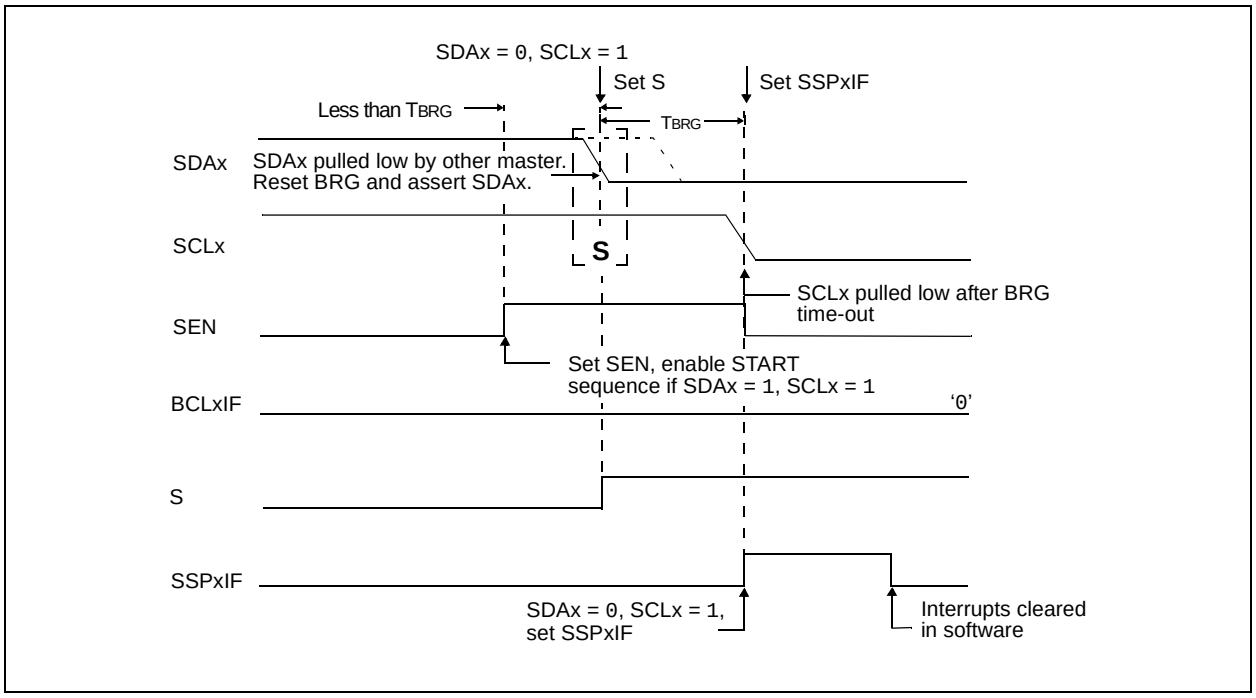
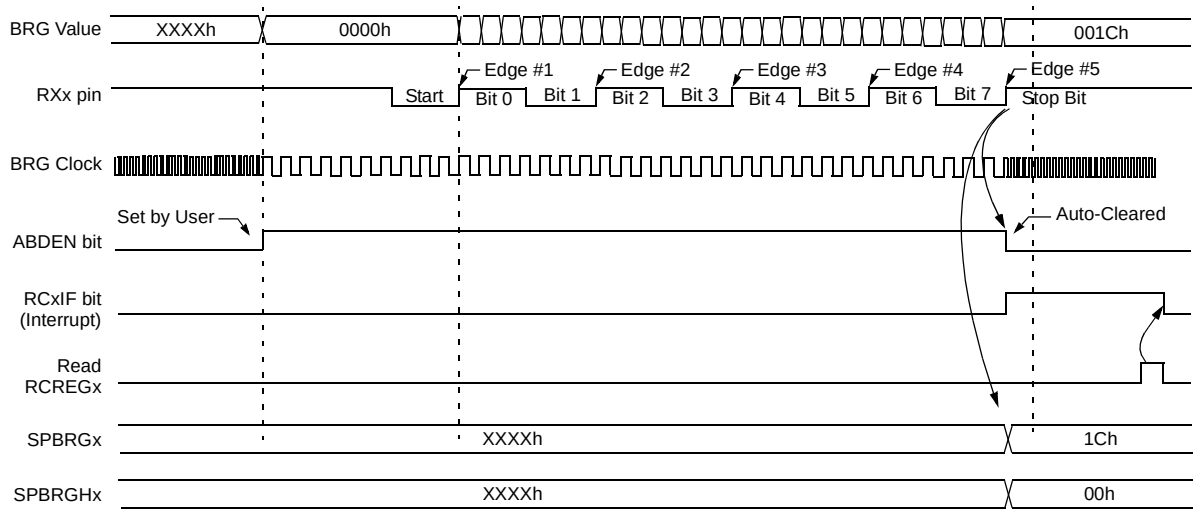
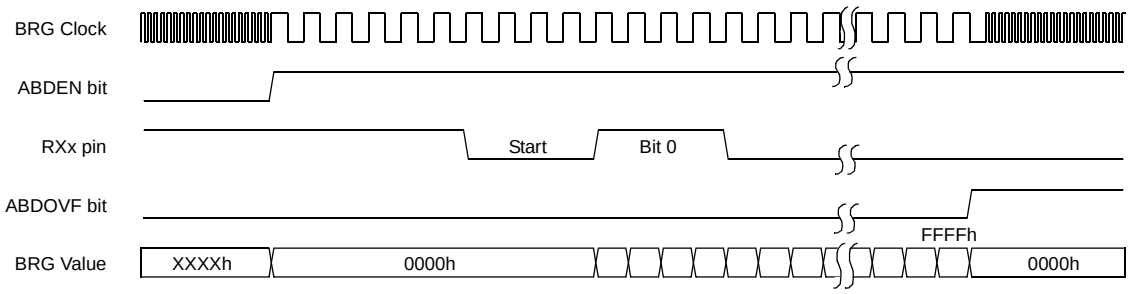


FIGURE 20-1: AUTOMATIC BAUD RATE CALCULATION



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

FIGURE 20-2: BRG OVERFLOW SEQUENCE



20.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 20-6. The data is received on the RXx pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

To set up an Asynchronous Reception:

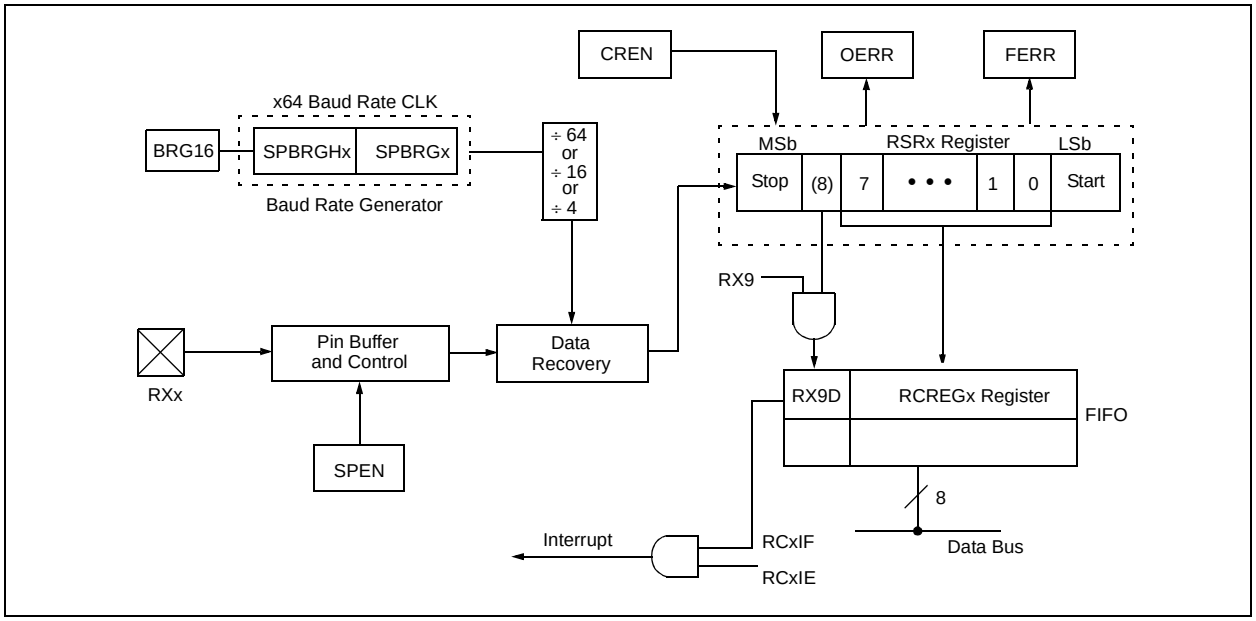
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RCxIE.
4. If 9-bit reception is desired, set bit, RX9.
5. Enable the reception by setting bit, CREN.
6. Flag bit, RCxIF, will be set when reception is complete and an interrupt will be generated if enable bit, RCxIE, was set.
7. Read the RCSTAx register to get the 9th bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREGx register.
9. If any error occurred, clear the error by clearing enable bit, CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

20.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCxIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCxIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCxIE and GIE bits are set.
8. Read the RCSTAx register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREGx to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 20-6: EUSART RECEIVE BLOCK DIAGRAM



REGISTER 25-6: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/P-1	R/P-0	R/P-0	R/P-0	U-0	R/P-1	U-0	R/P-1
$\overline{\text{DEBUG}}$	XINST	BBSIZ1	BBSIZ0	—	LVP	—	STVREN
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 **$\overline{\text{DEBUG}}$:** 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 5-4 **BBSIZ<1:0>:** Boot Block Size Select bits
11 = 4K words (8 Kbytes) boot block size
10 = 4K words (8 Kbytes) boot block size
01 = 2K words (4 Kbytes) boot block size
00 = 1K word (2 Kbytes) boot block size

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

DAW — Decimal Adjust W Register

Syntax:	DAW
Operands:	None

Operation:	If [W<3:0> > 9] or [DC = 1] then
	(W<3:0>) + 6 → W<3:0>;
	else
	(W<3:0>) → W<3:0>
	
	If [W<7:4> > 9] or [C = 1] then
	(W<7:4>) + 6 → W<7:4>;
	C = 1;
	else
	(W<7:4>) → W<7:4>

Status Affected:	C
Encoding:	0000 | 0000 | 0000 | 0111
Description:	DAW adjusts the eight-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example 1: DAW

Before Instruction
W = A5h
C = 0
DC = 0
After Instruction
W = 05h
C = 1
DC = 0

Example 2:

Before Instruction
W = CEh
C = 0
DC = 0
After Instruction
W = 34h
C = 1
DC = 0

DECF — Decrement f

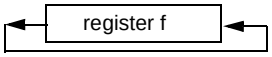
Syntax:	DECF f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – 1 → dest
Status Affected:	C, DC, N, OV, Z
Encoding:	0000 | 01da | ffff | ffff
Description:	Decrement register ‘f’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default). 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 to destination

Example: DECF CNT, 1, 0

Before Instruction
CNT = 01h
Z = 0
After Instruction
CNT = 00h
Z = 1

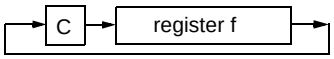
RLNCF	**Rotate Left f (no carry)**
Syntax:	RLNCF f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f<n>) → dest<n + 1>, (f<7>) → dest<0>
Status Affected:	N, Z
Encoding:	0100 | 01da | ffff | ffff
Description:	The contents of register ‘f’ are rotated one bit to the left. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default). 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 to destination

Example: `RLNCF REG, 1, 0`

Before Instruction
REG = 1010 1011
After Instruction
REG = 0101 0111

RRCF	**Rotate Right f through Carry**
Syntax:	RRCF f {,d {,a}}
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f<n>) → dest<n – 1>, (f<0>) → C, (C) → dest<7>
Status Affected:	C, N, Z
Encoding:	0011 | 00da | ffff | ffff
Description:	The contents of register ‘f’ are rotated one bit to the right through the Carry flag. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default). 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 to destination

Example: `RRCF REG, 0, 0`

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 0111 0011
C = 0

TSTFSZ	Test f, Skip if 0
Syntax:	TSTFSZ f {,a}
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	skip if f = 0
Status Affected:	None
Encoding:	0110 | 011a | ffff | ffff
Description:	If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a NOP is executed, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). 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(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    TSTFSZ  CNT, 1
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)
After Instruction
If CNT = 00h,
PC = Address (ZERO)
If CNT ≠ 00h,
PC = Address (NZERO)

XORLW	Exclusive OR Literal with W
Syntax:	XORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .XOR. k → W
Status Affected:	N, Z
Encoding:	0000 | 1010 | kkkk | kkkk
Description:	The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: XORLW 0AFh

Before Instruction
W = B5h
After Instruction
W = 1Ah

ADDWF	ADD W to Indexed (Indexed Literal Offset mode)
Syntax:	ADDWF [k] {,d}
Operands:	$0 \le k \le 95$ $d \in [0,1]$
Operation:	(W) + ((FSR2) + k) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	0010 | 01d0 | kkkk | kkkk
Description:	The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `ADDWF [OFST],0`

Before Instruction
- W = 17h
- OFST = 2Ch
- FSR2 = 0A00h
- Contents of 0A2Ch = 20h

After Instruction
- W = 37h
- Contents of 0A2Ch = 20h

BSF	Bit Set Indexed (Indexed Literal Offset mode)
Syntax:	BSF [k], b
Operands:	$0 \le f \le 95$ $0 \le b \le 7$
Operation:	1 → ((FSR2) + k)<b>
Status Affected:	None
Encoding:	1000 | bbb0 | kkkk | kkkk
Description:	Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `BSF [FLAG_OFST], 7`

Before Instruction
- FLAG_OFST = 0Ah
- FSR2 = 0A00h
- Contents of 0A0Ah = 55h

After Instruction
- Contents of 0A0Ah = D5h

SETF	Set Indexed (Indexed Literal Offset mode)
Syntax:	SETF [k]
Operands:	$0 \le k \le 95$
Operation:	FFh → ((FSR2) + k)
Status Affected:	None
Encoding:	0110 | 1000 | kkkk | kkkk
Description:	The contents of the register indicated by FSR2, offset by 'k', are set to FFh.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `SETF [OFST]`

Before Instruction
- OFST = 2Ch
- FSR2 = 0A00h
- Contents of 0A2Ch = 00h

After Instruction
- Contents of 0A2Ch = FFh