

Welcome to [E-XFL.COM](https://www.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.

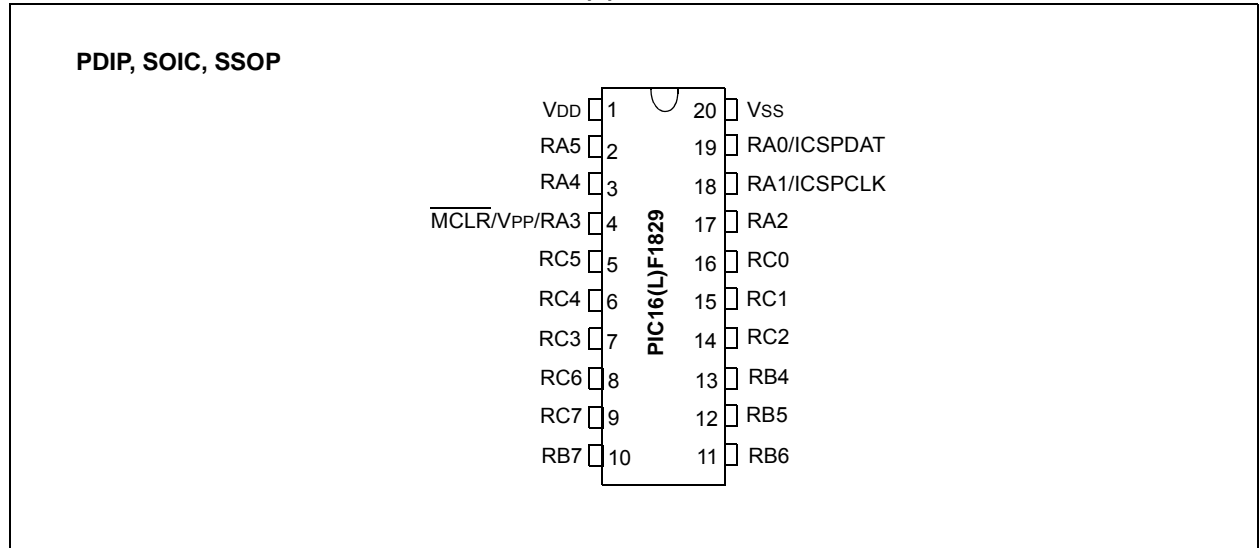
### Applications of "[Embedded - Microcontrollers](#)"

#### Details

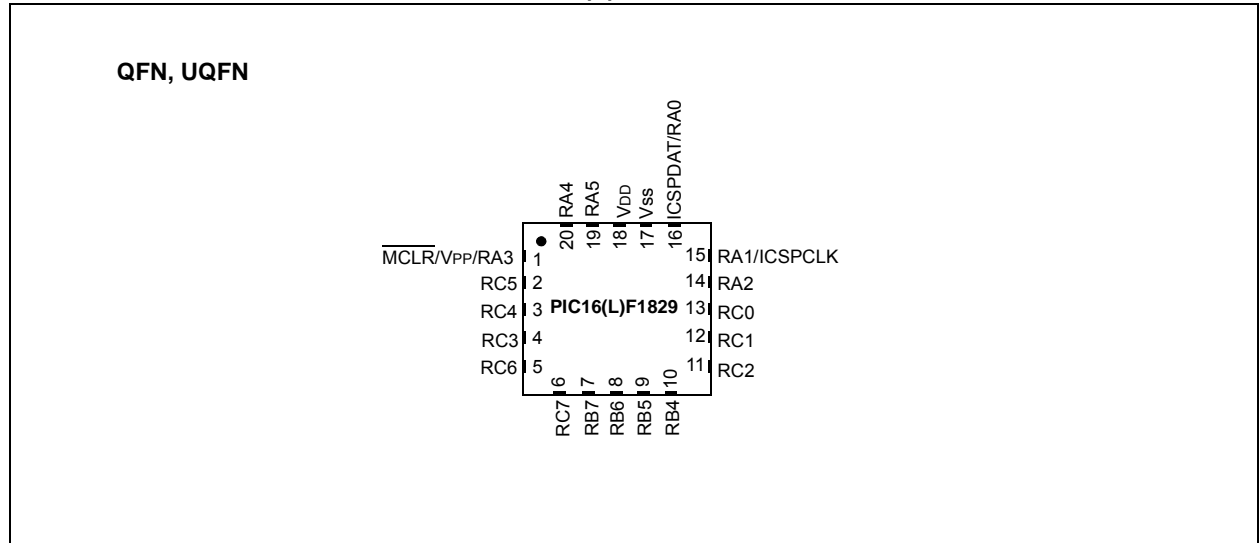
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	32MHz
Connectivity	I <sup>2</sup> C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	17
Program Memory Size	14KB (8K x 14)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/pic16f1829-i-so">https://www.e-xfl.com/product-detail/microchip-technology/pic16f1829-i-so</a>



**FIGURE 3: 20-PIN DIAGRAM FOR PIC16(L)F1829**



**FIGURE 4: 20-PIN DIAGRAM FOR PIC16(L)F1829**





**TABLE 1-2: PIC16(L)F1825 PINOUT DESCRIPTION (CONTINUED)**

Name	Function	Input Type	Output Type	Description
RC3/AN7/CPS7/C12IN3-/P2A <sup>(1,2)</sup> /CCP2 <sup>(1,2)</sup> /P1C <sup>(1,2)</sup> /SS1 <sup>(1,2)</sup> /MDMIN	RC3	TTL	CMOS	General purpose I/O.
	AN7	AN	—	A/D Channel 7 input.
	CPS7	AN	—	Capacitive sensing input 7.
	C12IN3-	AN	—	Comparator C1 or C2 negative input.
	P2A	—	CMOS	PWM output.
	CCP2	AN	—	Capture/Compare/PWM2.
	P1C	—	CMOS	PWM output.
	SS1	ST	—	Slave Select input.
	MDMIN	ST	—	Modulator source input.
RC4/C2OUT/SRNQ/P1B/TX <sup>(1,2)</sup> /CK <sup>(1,2)</sup> /MDOUT	RC4	TTL	CMOS	General purpose I/O.
	C2OUT	—	CMOS	Comparator C2 output.
	SRNQ	—	CMOS	SR Latch inverting output.
	P1B	—	CMOS	PWM output.
	TX	—	CMOS	USART asynchronous transmit.
	CK	ST	CMOS	USART synchronous clock.
	MDOUT	—	CMOS	Modulator output.
RC5/P1A/CCP1/DT <sup>(1,2)</sup> /RX <sup>(1,2)</sup> /MDCIN2	RC5	TTL	CMOS	General purpose I/O.
	P1A	—	CMOS	PWM output.
	CCP1	ST	CMOS	Capture/Compare/PWM1.
	RX	ST	—	USART asynchronous input.
	DT	ST	CMOS	USART synchronous data.
	MDCIN2	ST	—	Modulator Carrier Input 2.
VDD	VDD	Power	—	Positive supply.
VSS	VSS	Power	—	Ground reference.

**Legend:** AN = Analog input or output    CMOS = CMOS compatible input or output    OD = Open Drain  
TTL = TTL compatible input    ST = Schmitt Trigger input with CMOS levels    I<sup>2</sup>C™ = Schmitt Trigger input with I<sup>2</sup>C levels  
HV = High Voltage    XTAL = Crystal

**Note 1:** Pin functions can be moved using the APFCON0 or APFCON1 register.

**2:** Default function location.



# PIC16(L)F1825/9

**TABLE 3-8: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)**

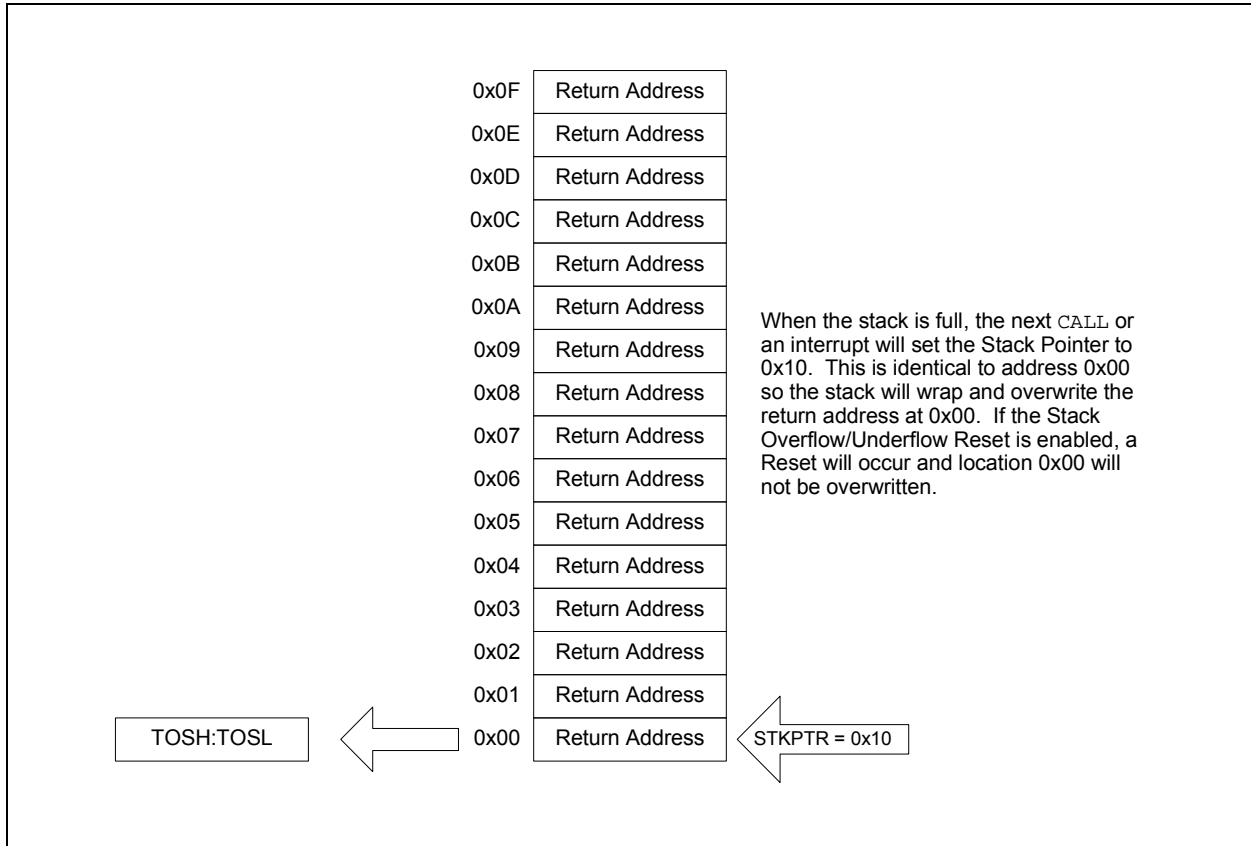
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets	
Bank 31												
F80h <sup>(1)</sup>	INDF0	Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register)								xxxx  xxxx	xxxx  xxxx	
F81h <sup>(1)</sup>	INDF1	Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register)								xxxx  xxxx	xxxx  xxxx	
F82h <sup>(1)</sup>	PCL	Program Counter (PC) Least Significant Byte								0000  0000	0000  0000	
F83h <sup>(1)</sup>	STATUS	—	—	—	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	---1  1000	---q  quuu	
F84h <sup>(1)</sup>	FSR0L	Indirect Data Memory Address 0 Low Pointer								0000  0000	uuuu  uuuu	
F85h <sup>(1)</sup>	FSR0H	Indirect Data Memory Address 0 High Pointer								0000  0000	0000  0000	
F86h <sup>(1)</sup>	FSR1L	Indirect Data Memory Address 1 Low Pointer								0000  0000	uuuu  uuuu	
F87h <sup>(1)</sup>	FSR1H	Indirect Data Memory Address 1 High Pointer								0000  0000	0000  0000	
F88h <sup>(1)</sup>	BSR	—	—	—	BSR<4:0>					---0  0000	---0  0000	
F89h <sup>(1)</sup>	WREG	Working Register								0000  0000	uuuu  uuuu	
F8Ah <sup>(1)</sup>	PCLATH	—	Write Buffer for the upper 7 bits of the Program Counter								-000  0000	-000  0000
F8Bh <sup>(1)</sup>	INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	0000  0000	0000  0000	
F8Ch — FE3h	—	Unimplemented								—	—	
FE4h	STATUS_ SHAD	—	—	—	—	—	Z_SHAD	DC_SHAD	C_SHAD	----  -xxx	----  -uuu	
FE5h	WREG_ SHAD	Working Register Shadow								0000  0000	uuuu  uuuu	
FE6h	BSR_ SHAD	—	—	—	Bank Select Register Shadow					---x  xxxx	---u  uuuu	
FE7h	PCLATH_ SHAD	—	Program Counter Latch High Register Shadow								-xxx  xxxx	uuuu  uuuu
FE8h	FSR0L_ SHAD	Indirect Data Memory Address 0 Low Pointer Shadow								xxxx  xxxx	uuuu  uuuu	
FE9h	FSR0H_ SHAD	Indirect Data Memory Address 0 High Pointer Shadow								xxxx  xxxx	uuuu  uuuu	
FEAh	FSR1L_ SHAD	Indirect Data Memory Address 1 Low Pointer Shadow								xxxx  xxxx	uuuu  uuuu	
FEBh	FSR1H_ SHAD	Indirect Data Memory Address 1 High Pointer Shadow								xxxx  xxxx	uuuu  uuuu	
FECh	—	Unimplemented								—	—	
FEDh	STKPTR	—	—	—	Current Stack pointer					---1  1111	---1  1111	
FEEh	TOSL	Top-of-Stack Low byte								xxxx  xxxx	uuuu  uuuu	
FEFh	TOSH	—	Top-of-Stack High byte								-xxx  xxxx	-uuu  uuuu

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, r = reserved.  
Shaded locations are unimplemented, read as '0'.

- Note** 1: These registers can be addressed from any bank.  
2: PIC16(L)F1829 only.  
3: PIC16(L)F1825 only.  
4: Unimplemented, read as '1'.



**FIGURE 3-7: ACCESSING THE STACK EXAMPLE 4**



### 3.4.2 OVERFLOW/UNDERFLOW RESET

If the `STVREN` bit in Configuration Word 2 is programmed to '1', the device will be reset if the stack is `PUSHed` beyond the sixteenth level or `POPed` beyond the first level, setting the appropriate bits (`STKOVF` or `STKUNF`, respectively) in the `PCON` register.

## 3.5 Indirect Addressing

The `INDFn` registers are not physical registers. Any instruction that accesses an `INDFn` register actually accesses the register at the address specified by the File Select Registers (`FSR`). If the `FSRn` address specifies one of the two `INDFn` registers, the read will return '0' and the write will not occur (though Status bits may be affected). The `FSRn` register value is created by the pair `FSRnH` and `FSRnL`.

The `FSR` registers form a 16-bit address that allows an addressing space with 65536 locations. These locations are divided into three memory regions:

- Traditional Data Memory
- Linear Data Memory
- Program Flash Memory



**TABLE 7-5: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BORCON	SBOREN	—	—	—	—	—	—	BORRDY	76
PCON	STKOVF	STKUNF	—	—	RMCLR	RI	POR	BOR	80
STATUS	—	—	—	TO	PD	Z	DC	C	22
WDTCON	—	—	WDTPS<4:0>					SWDTEN	100

**Legend:** — Unimplemented bit, reads as '0'. Shaded cells are not used by Resets.



## 12.4 PORTC Registers

PORTC is a 6-bit wide (8-bit wide for PIC16(L)F1829), bidirectional port. The corresponding data direction register is TRISC (Register 12-16). Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin). Example 12-2 shows how to initialize a port.

Reading the PORTC register (Register 12-15) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATC).

The TRISC register (Register 12-16) controls the PORTC pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISC register are maintained set when using them as analog inputs. I/O pins configured as analog input always read '0'.

The INLVLC register (Register 12-20) controls the input voltage threshold for each of the available PORTC input pins. A selection between the Schmitt Trigger CMOS or the TTL Compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTC register and also the level at which an Interrupt-on-Change occurs, if that feature is enabled. See **Section 30.4 “DC Characteristics: PIC16(L)F1825/9-I/E”** for more information on threshold levels.

**Note:** Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin.

### 12.4.1 ANSEL register

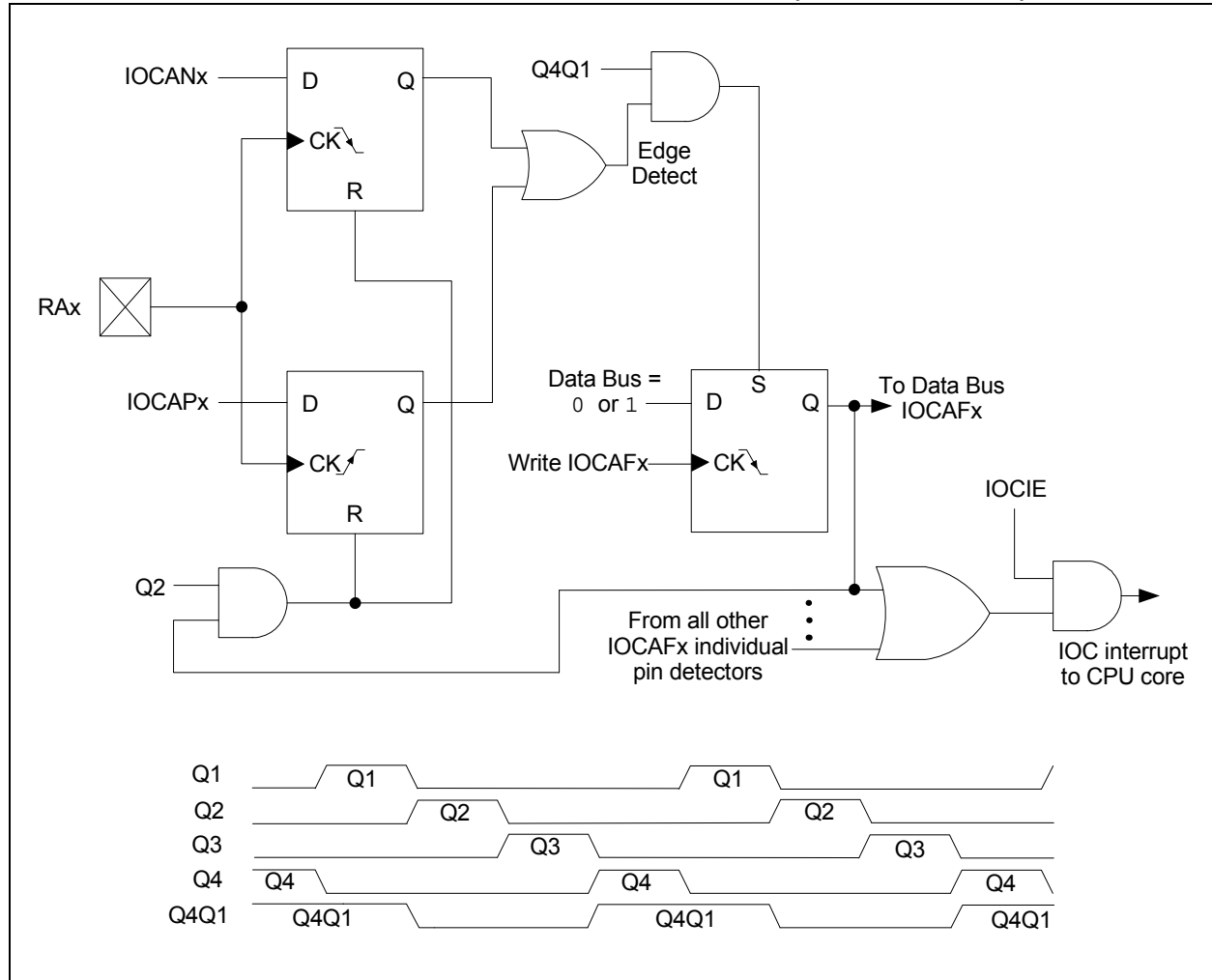
The ANSEL register (Register 12-18) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSEL bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSEL bits has no effect on digital output functions. A pin with TRIS clear and ANSEL set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

**Note:** The ANSEL bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to '0' by user software.



**FIGURE 13-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTA EXAMPLE)**



## 13.6 Interrupt-on-Change Registers

**REGISTER 13-1: IOCAP: INTERRUPT-ON-CHANGE PORTA POSITIVE EDGE REGISTER**

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
bit 7							bit 0

### Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

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

bit 5-0 **IOCAP<5:0>:** Interrupt-on-Change PORTA Positive Edge Enable bits

- 1 = Interrupt-on-change enabled on the pin for a positive going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.
- 0 = Interrupt-on-change disabled for the associated pin.

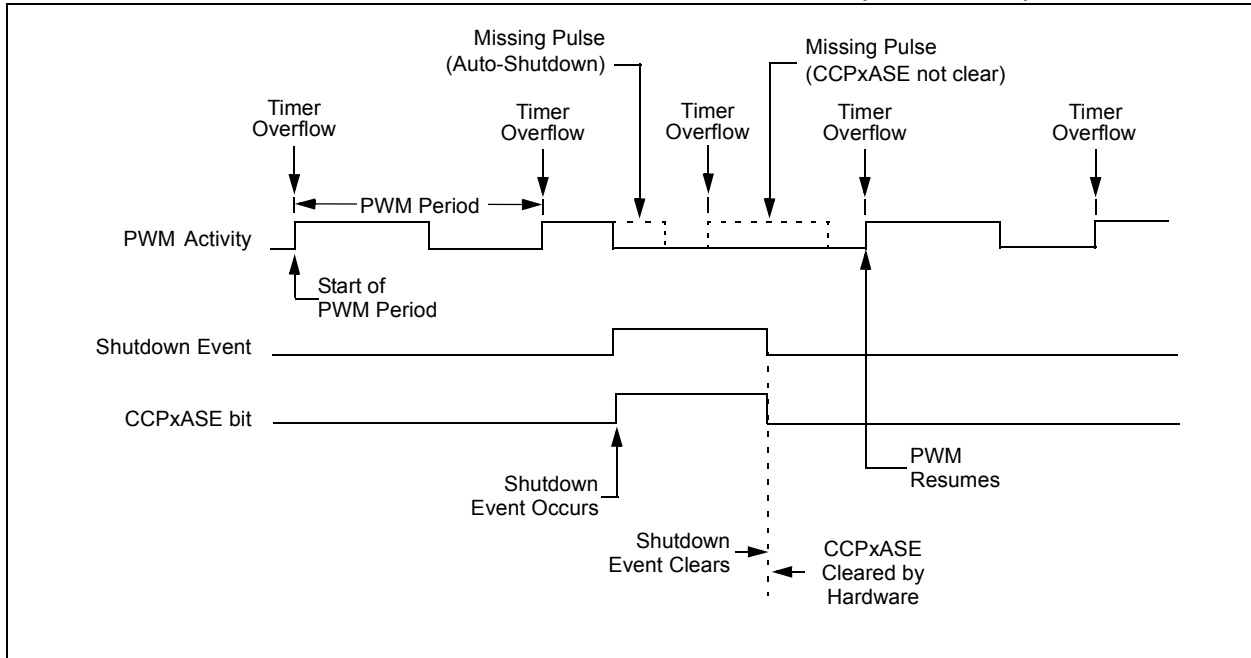


## 24.4.4 AUTO-RESTART MODE

The Enhanced PWM can be configured to automatically restart the PWM signal once the auto-shutdown condition has been removed. Auto-restart is enabled by setting the PxRSEN bit in the PWMxCON register.

If auto-restart is enabled, the CCPxASE bit will remain set as long as the auto-shutdown condition is active. When the auto-shutdown condition is removed, the CCPxASE bit will be cleared via hardware and normal operation will resume.

**FIGURE 24-15: PWM AUTO-SHUTDOWN WITH AUTO-RESTART (PxRSEN = 1)**





# PIC16(L)F1825/9

## REGISTER 24-3: CCPxAS: CCPx AUTO-SHUTDOWN CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
CCPxASE	CCPxAS<2:0>			PSSxAC<1:0>		PSSxBD<1:0>	
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7      **CCPxASE:** CCPx Auto-Shutdown Event Status bit  
1 = A shutdown event has occurred; CCPx outputs are in shutdown state  
0 = CCPx outputs are operating
- bit 6-4      **CCPxAS<2:0>:** CCPx Auto-Shutdown Source Select bits  
000 = Auto-shutdown is disabled  
001 = Comparator C1 output high<sup>(1)</sup>  
010 = Comparator C2 output high<sup>(1)</sup>  
011 = Either Comparator C1 or C2 high<sup>(1)</sup>  
100 = VIL on FLT0 pin  
101 = VIL on FLT0 pin or Comparator C1 high<sup>(1)</sup>  
110 = VIL on FLT0 pin or Comparator C2 high<sup>(1)</sup>  
111 = VIL on FLT0 pin or Comparator C1 or Comparator C2 high<sup>(1)</sup>
- bit 3-2      **PSSxAC<1:0>:** Pins PxA and PxC Shutdown State Control bits  
00 = Drive pins PxA and PxC to '0'  
01 = Drive pins PxA and PxC to '1'  
1x = Pins PxA and PxC tri-state
- bit 1-0      **PSSxBD<1:0>:** Pins PxB and PxD Shutdown State Control bits  
00 = Drive pins PxB and PxD to '0'  
01 = Drive pins PxB and PxD to '1'  
1x = Pins PxB and PxD tri-state

**Note 1:** If CxSYNC is enabled, the shutdown will be delayed by Timer1.



## 25.4.9 ACKNOWLEDGE SEQUENCE

The 9th SCLx pulse for any transferred byte in I<sup>2</sup>C is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDAx line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge ( $\overline{\text{ACK}}$ ) is an active-low signal, pulling the SDAx line low indicated to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an  $\overline{\text{ACK}}$  is placed in the ACKSTAT bit of the SSPxCON2 register.

Slave software, when the AHEN and DHEN bits are set, allow the user to set the  $\overline{\text{ACK}}$  value sent back to the transmitter. The ACKDT bit of the SSPxCON2 register is set/cleared to determine the response.

Slave hardware will generate an  $\overline{\text{ACK}}$  response if the AHEN and DHEN bits of the SSPxCON3 register are clear.

There are certain conditions where an  $\overline{\text{ACK}}$  will not be sent by the slave. If the BF bit of the SSPxSTAT register or the SSPOV bit of the SSPxCON1 register are set when a byte is received.

When the module is addressed, after the eighth falling edge of SCLx on the bus, the ACKTIM bit of the SSPxCON3 register is set. The ACKTIM bit indicates the acknowledge time of the active bus. The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is enabled.

## 25.5 I<sup>2</sup>C SLAVE MODE OPERATION

The MSSPx Slave mode operates in one of four modes selected in the SSPM bits of SSPxCON1 register. The modes can be divided into 7-bit and 10-bit Addressing mode. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

Modes with Start and Stop bit interrupts operated the same as the other modes with SSPxIF additionally getting set upon detection of a Start, Restart or Stop condition.

### 25.5.1 SLAVE MODE ADDRESSES

The SSPxADD register (Register 25-6) contains the Slave mode address. The first byte received after a Start or Restart condition is compared against the value stored in this register. If the byte matches, the value is loaded into the SSPxBUF register and an interrupt is generated. If the value does not match, the module goes Idle and no indication is given to the software that anything happened.

The SSPx Mask register (Register 25-5) affects the address matching process. See **Section 25.5.8 “SSPx Mask Register”** for more information.

#### 25.5.1.1 I<sup>2</sup>C Slave 7-bit Addressing Mode

In 7-bit Addressing mode, the LSb of the received data byte is ignored when determining if there is an address match.

#### 25.5.1.2 I<sup>2</sup>C Slave 10-bit Addressing Mode

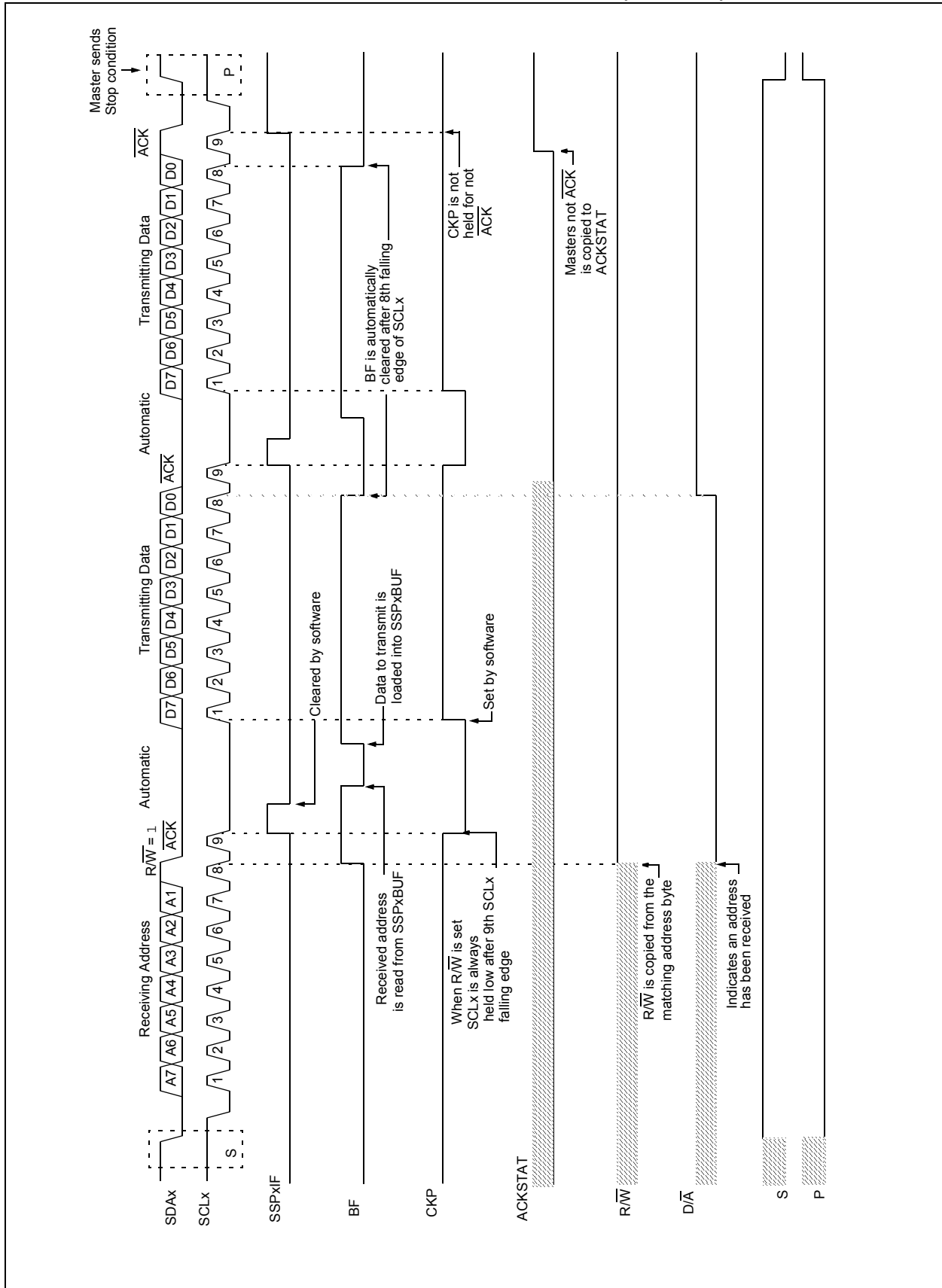
In 10-bit Addressing mode, the first received byte is compared to the binary value of ‘1 1 1 1 0 A9 A8 0’. A9 and A8 are the two MSb of the 10-bit address and stored in bits 2 and 1 of the SSPxADD register.

After the acknowledge of the high byte the UA bit is set and SCLx is held low until the user updates SSPxADD with the low address. The low address byte is clocked in and all eight bits are compared to the low address value in SSPxADD. Even if there is not an address match; SSPxIF and UA are set, and SCLx is held low until SSPxADD is updated to receive a high byte again. When SSPxADD is updated the UA bit is cleared. This ensures the module is ready to receive the high address byte on the next communication.

A high and low address match as a write request is required at the start of all 10-bit addressing communication. A transmission can be initiated by issuing a Restart once the slave is addressed, and clocking in the high address with the R/W bit set. The slave hardware will then acknowledge the read request and prepare to clock out data. This is only valid for a slave after it has received a complete high and low address byte match.



**FIGURE 25-18: I<sup>2</sup>C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 0)**





## 25.6.5 I<sup>2</sup>C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition (Figure 25-27) occurs when the RSEN bit of the SSPxCON2 register is programmed high and the Master state machine is no longer active. When the RSEN bit is set, the SCLx pin is asserted low. When the SCLx pin is sampled low, the Baud Rate Generator is loaded and begins counting. The SDAx pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDAx is sampled high, the SCLx pin will be deasserted (brought high). When SCLx is sampled high, the Baud Rate Generator is reloaded and begins counting. SDAx and SCLx must be sampled high for one TBRG. This action is then followed by assertion of the SDAx pin (SDAx = 0) for one TBRG while SCLx is high. SCLx is asserted low. Following this, the RSEN bit of the SSPxCON2 register will be

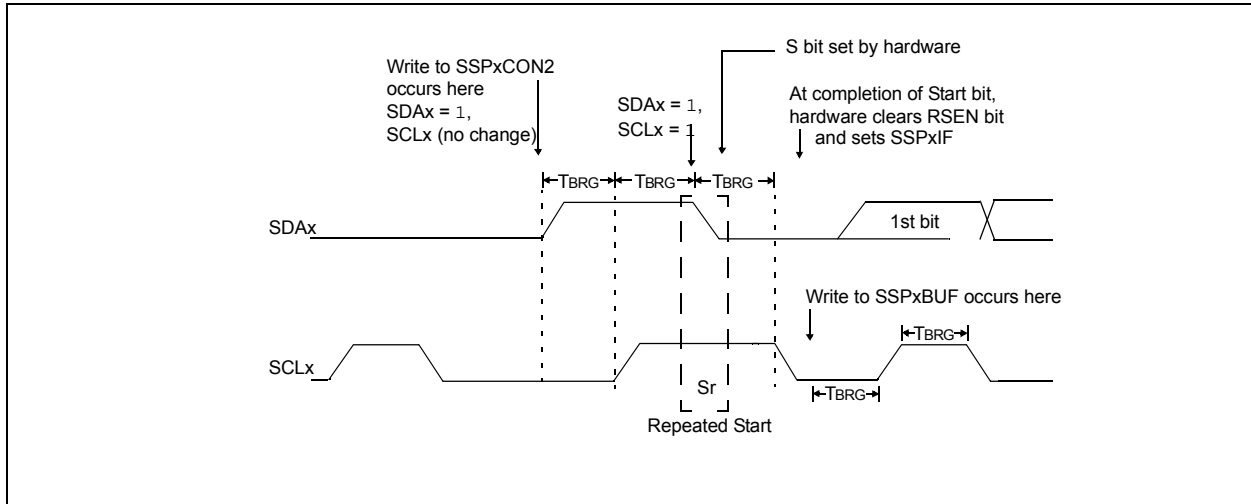
automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDAx pin held low. As soon as a Start condition is detected on the SDAx and SCLx pins, the S bit of the SSPxSTAT register will be set. The SSPxIF bit will not be set until the Baud Rate Generator has timed out.

**Note 1:** If RSEN is programmed while any other event is in progress, it will not take effect.

**2:** A bus collision during the Repeated Start condition occurs if:

- SDAx is sampled low when SCLx goes from low-to-high.
- SCLx goes low before SDAx is asserted low. This may indicate that another master is attempting to transmit a data '1'.

**FIGURE 25-27: REPEAT START CONDITION WAVEFORM**





## 25.6.10 SLEEP OPERATION

While in Sleep mode, the I<sup>2</sup>C slave module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSPx interrupt is enabled).

## 25.6.11 EFFECTS OF A RESET

A Reset disables the MSSPx module and terminates the current transfer.

## 25.6.12 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSPx module is disabled. Control of the I<sup>2</sup>C bus may be taken when the P bit of the SSPxSTAT register is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the SSPx interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDAx line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLxIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

## 25.6.13 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDAx pin, arbitration takes place when the master outputs a '1' on SDAx, by letting SDAx float high and another master asserts a '0'. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a '1' and the data sampled on the SDAx pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF and reset the I<sup>2</sup>C port to its Idle state (Figure 25-32).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDAx and SCLx lines are deasserted and the SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I<sup>2</sup>C bus is free, the user can resume communication by asserting a Start condition.

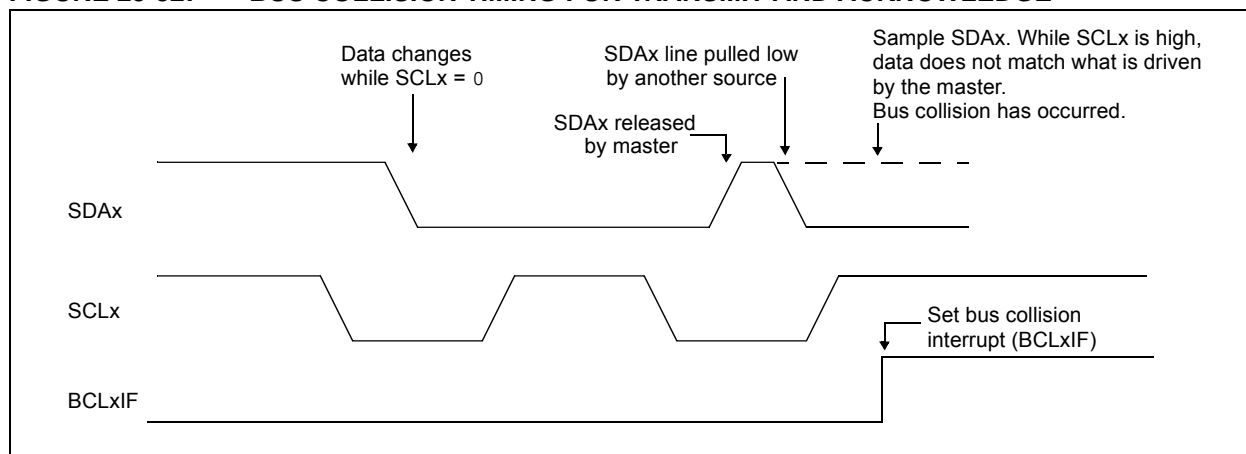
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDAx and SCLx lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I<sup>2</sup>C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDAx and SCLx pins. If a Stop condition occurs, the SSPxIF bit will be set.

A write to the SSPxBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I<sup>2</sup>C bus can be taken when the P bit is set in the SSPxSTAT register, or the bus is Idle and the S and P bits are cleared.

**FIGURE 25-32: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**





# PIC16(L)F1825/9

## REGISTER 25-3: SSPxCON2: SSPx CONTROL REGISTER 2

R/W-0/0	R-0/0	R/W-0/0	R/S/HS-0/0	R/S/HS-0/0	R/S/HS-0/0	R/S/HS-0/0	R/W/HS-0/0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

### Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HC = Cleared by hardware S = User set

- bit 7 **GCEN:** General Call Enable bit (in I<sup>2</sup>C Slave mode only)  
1 = Enable interrupt when a general call address (0x00 or 00h) is received in the SSPxSR  
0 = General call address disabled
- bit 6 **ACKSTAT:** Acknowledge Status bit (in I<sup>2</sup>C mode only)  
1 = Acknowledge was not received  
0 = Acknowledge was received
- bit 5 **ACKDT:** Acknowledge Data bit (in I<sup>2</sup>C mode only)  
In Receive mode:  
Value transmitted when the user initiates an Acknowledge sequence at the end of a receive  
1 = Not Acknowledge  
0 = Acknowledge
- bit 4 **ACKEN:** Acknowledge Sequence Enable bit (in I<sup>2</sup>C Master mode only)  
In Master Receive mode:  
1 = Initiate Acknowledge sequence on SDAx and SCLx pins, and transmit ACKDT data bit.  
Automatically cleared by hardware.  
0 = Acknowledge sequence Idle
- bit 3 **RCEN:** Receive Enable bit (in I<sup>2</sup>C Master mode only)  
1 = Enables Receive mode for I<sup>2</sup>C  
0 = Receive Idle
- bit 2 **PEN:** Stop Condition Enable bit (in I<sup>2</sup>C Master mode only)  
SCKx Release Control:  
1 = Initiate Stop condition on SDAx and SCLx pins. Automatically cleared by hardware.  
0 = Stop condition Idle
- bit 1 **RSEN:** Repeated Start Condition Enabled bit (in I<sup>2</sup>C Master mode only)  
1 = Initiate Repeated Start condition on SDAx and SCLx pins. Automatically cleared by hardware.  
0 = Repeated Start condition Idle
- bit 0 **SEN:** Start Condition Enable/Stretch Enable bit  
In Master mode:  
1 = Initiate Start condition on SDAx and SCLx pins. Automatically cleared by hardware.  
0 = Start condition Idle  
In Slave mode:  
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)  
0 = Clock stretching is disabled

**Note 1:** For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I<sup>2</sup>C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).



## 26.1.2 EUSART ASYNCHRONOUS RECEIVER

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in Figure 26-2. The data is received on the RX/DT pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all eight or nine bits of the character have been shifted in, they are immediately transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must start servicing the EUSART receiver. The FIFO and RSR registers are not directly accessible by software. Access to the received data is via the RCREG register.

### 26.1.2.1 Enabling the Receiver

The EUSART receiver is enabled for asynchronous operation by configuring the following three control bits:

- CREN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the CREN bit of the RCSTA register enables the receiver circuitry of the EUSART. Clearing the SYNC bit of the TXSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTA register enables the EUSART. The programmer must set the corresponding TRIS bit to configure the RX/DT I/O pin as an input.

**Note 1:** If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

### 26.1.2.2 Receiving Data

The receiver data recovery circuit initiates character reception on the falling edge of the first bit. The first bit, also known as the Start bit, is always a zero. The data recovery circuit counts one-half bit time to the center of the Start bit and verifies that the bit is still a zero. If it is not a zero then the data recovery circuit aborts character reception, without generating an error, and resumes looking for the falling edge of the Start bit. If the Start bit zero verification succeeds then the data recovery circuit counts a full bit time to the center of the next bit. The bit is then sampled by a majority detect circuit and the resulting '0' or '1' is shifted into the RSR. This repeats until all data bits have been sampled and shifted into the RSR. One final bit time is measured and the level sampled. This is the Stop bit, which is always a '1'. If the data recovery circuit samples a '0' in the Stop bit position then a framing error is set for this character, otherwise the framing error is cleared for this character. See **Section 26.1.2.4 "Receive Framing Error"** for more information on framing errors.

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the EUSART receive FIFO and the RCIF interrupt flag bit of the PIR1 register is set. The top character in the FIFO is transferred out of the FIFO by reading the RCREG register.

**Note:** If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See **Section 26.1.2.5 "Receive Overrun Error"** for more information on overrun errors.

### 26.1.2.3 Receive Interrupts

The RCIF interrupt flag bit of the PIR1 register is set whenever the EUSART receiver is enabled and there is an unread character in the receive FIFO. The RCIF interrupt flag bit is read-only, it cannot be set or cleared by software.

RCIF interrupts are enabled by setting all of the following bits:

- RCIE interrupt enable bit of the PIE1 register
- PEIE Peripheral Interrupt Enable bit of the INTCON register
- GIE Global Interrupt Enable bit of the INTCON register

The RCIF interrupt flag bit will be set when there is an unread character in the FIFO, regardless of the state of interrupt enable bits.



# PIC16(L)F1825/9

## 29.2 Instruction Descriptions

### ADDFSR Add Literal to FSRn

Syntax:	[ <i>label</i> ] ADDFSR FSRn, k
Operands:	$-32 \leq k \leq 31$ $n \in [0, 1]$
Operation:	$FSR(n) + k \rightarrow FSR(n)$
Status Affected:	None
Description:	The signed 6-bit literal 'k' is added to the contents of the FSRnH:FSRnL register pair.  FSRn is limited to the range 0000h - FFFFh. Moving beyond these bounds will cause the FSR to wrap-around.

### ANDLW AND literal with W

Syntax:	[ <i>label</i> ] ANDLW k
Operands:	$0 \leq k \leq 255$
Operation:	$(W) .AND. (k) \rightarrow (W)$
Status Affected:	Z
Description:	The contents of W register are AND'ed with the 8-bit literal 'k'. The result is placed in the W register.

### ADDLW Add literal and W

Syntax:	[ <i>label</i> ] ADDLW k
Operands:	$0 \leq k \leq 255$
Operation:	$(W) + k \rightarrow (W)$
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the 8-bit literal 'k' and the result is placed in the W register.

### ANDWF AND W with f

Syntax:	[ <i>label</i> ] ANDWF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(W) .AND. (f) \rightarrow (\text{destination})$
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

### ADDWF Add W and f

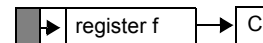
Syntax:	[ <i>label</i> ] ADDWF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(W) + (f) \rightarrow (\text{destination})$
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

### ASRF Arithmetic Right Shift

Syntax:	[ <i>label</i> ] ASRF f,{d}
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(f < 7) \rightarrow \text{dest} < 7>$ $(f < 7:1) \rightarrow \text{dest} < 6:0>$ , $(f < 0) \rightarrow C$ ,
Status Affected:	C, Z
Description:	The contents of register 'f' are shifted one bit to the right through the Carry flag. The MSb remains unchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'.

### ADDWFC ADD W and CARRY bit to f

Syntax:	[ <i>label</i> ] ADDWFC f,{d}
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(W) + (f) + (C) \rightarrow \text{dest}$
Status Affected:	C, DC, Z
Description:	Add W, the Carry flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'.





# PIC16(L)F1825/9

## CALL Call Subroutine

Syntax: [ *label* ] CALL k  
Operands:  $0 \leq k \leq 2047$   
Operation: (PC)+1 → TOS,  
k → PC<10:0>,  
(PCLATH<6:3>) → PC<14:11>  
Status Affected: None  
Description: Call Subroutine. First, return address (PC + 1) is pushed onto the stack. The 11-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a 2-cycle instruction.

## CALLW Subroutine Call With W

Syntax: [ *label* ] CALLW  
Operands: None  
Operation: (PC) + 1 → TOS,  
(W) → PC<7:0>,  
(PCLATH<6:0>) → PC<14:8>  
Status Affected: None  
Description: Subroutine call with W. First, the return address (PC + 1) is pushed onto the return stack. Then, the contents of W is loaded into PC<7:0>, and the contents of PCLATH into PC<14:8>. CALLW is a two-cycle instruction.

## CLRF Clear f

Syntax: [ *label* ] CLRF f  
Operands:  $0 \leq f \leq 127$   
Operation: 00h → (f)  
1 → Z  
Status Affected: Z  
Description: The contents of register 'f' are cleared and the Z bit is set.

## CLRW Clear W

Syntax: [ *label* ] CLRW  
Operands: None  
Operation: 00h → (W)  
1 → Z  
Status Affected: Z  
Description: W register is cleared. Zero bit (Z) is set.

## CLRWDTClear Watchdog Timer

Syntax: [ *label* ] CLRWDTClear Watchdog Timer  
Operands: None  
Operation: 00h → WDT  
0 → WDT prescaler,  
1 →  $\overline{TO}$   
1 →  $\overline{PD}$   
Status Affected:  $\overline{TO}$ ,  $\overline{PD}$   
Description: CLRWDTClear Watchdog Timer instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits  $\overline{TO}$  and  $\overline{PD}$  are set.

## COMF Complement f

Syntax: [ *label* ] COMF f,d  
Operands:  $0 \leq f \leq 127$   
d ∈ [0,1]  
Operation: (f) → (destination)  
Status Affected: Z  
Description: The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

## DECF Decrement f

Syntax: [ *label* ] DECF f,d  
Operands:  $0 \leq f \leq 127$   
d ∈ [0,1]  
Operation: (f) - 1 → (destination)  
Status Affected: Z  
Description: Decrement register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.



# PIC16(L)F1825/9

## RRF Rotate Right f through Carry

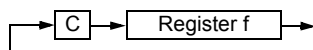
Syntax: [ *label* ] RRF *f*,*d*

Operands:  $0 \leq f \leq 127$   
 $d \in [0,1]$

Operation: See description below

Status Affected: C

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 the W register. If 'd' is '1', the result is placed back in register 'f'.



## SLEEP Enter Sleep mode

Syntax: [ *label* ] SLEEP

Operands: None

Operation: 00h → WDT,  
 0 → WDT prescaler,  
 1 →  $\overline{TO}$ ,  
 0 →  $\overline{PD}$

Status Affected:  $\overline{TO}$ ,  $\overline{PD}$

Description: The power-down Status bit,  $\overline{PD}$  is cleared. Time-out Status bit,  $\overline{TO}$  is set. Watchdog Timer and its prescaler are cleared. The processor is put into Sleep mode with the oscillator stopped.

## SUBLW Subtract W from literal

Syntax: [ *label* ] SUBLW *k*

Operands:  $0 \leq k \leq 255$

Operation:  $k - (W) \rightarrow (W)$

Status Affected: C, DC, Z

Description: The W register is subtracted (2's complement method) from the 8-bit literal 'k'. The result is placed in the W register.

C = 0	$W > k$
C = 1	$W \leq k$
DC = 0	$W<3:0> > k<3:0>$
DC = 1	$W<3:0> \leq k<3:0>$

## SUBWF Subtract W from f

Syntax: [ *label* ] SUBWF *f*,*d*

Operands:  $0 \leq f \leq 127$   
 $d \in [0,1]$

Operation:  $(f) - (W) \rightarrow (\text{destination})$

Status Affected: C, DC, Z

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

C = 0	$W > f$
C = 1	$W \leq f$
DC = 0	$W<3:0> > f<3:0>$
DC = 1	$W<3:0> \leq f<3:0>$

## SUBWFB Subtract W from f with Borrow

Syntax: SUBWFB *f*{,*d*}

Operands:  $0 \leq f \leq 127$   
 $d \in [0,1]$

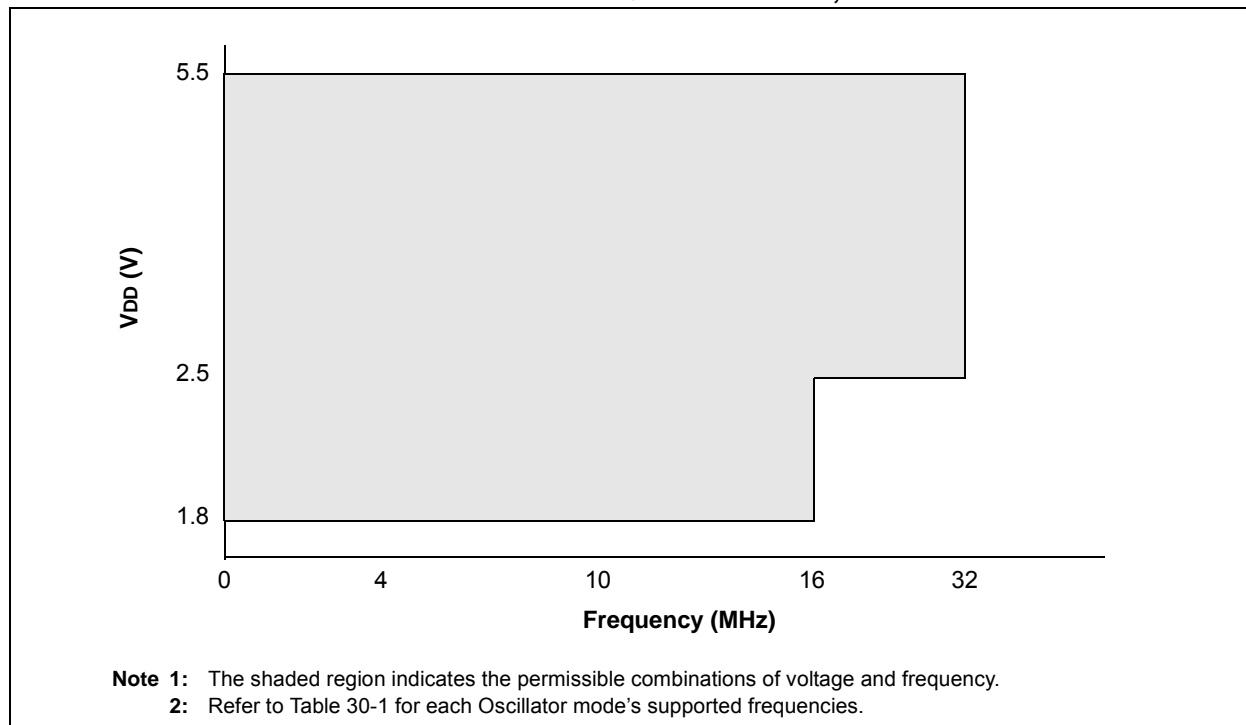
Operation:  $(f) - (W) - (\overline{B}) \rightarrow \text{dest}$

Status Affected: C, DC, Z

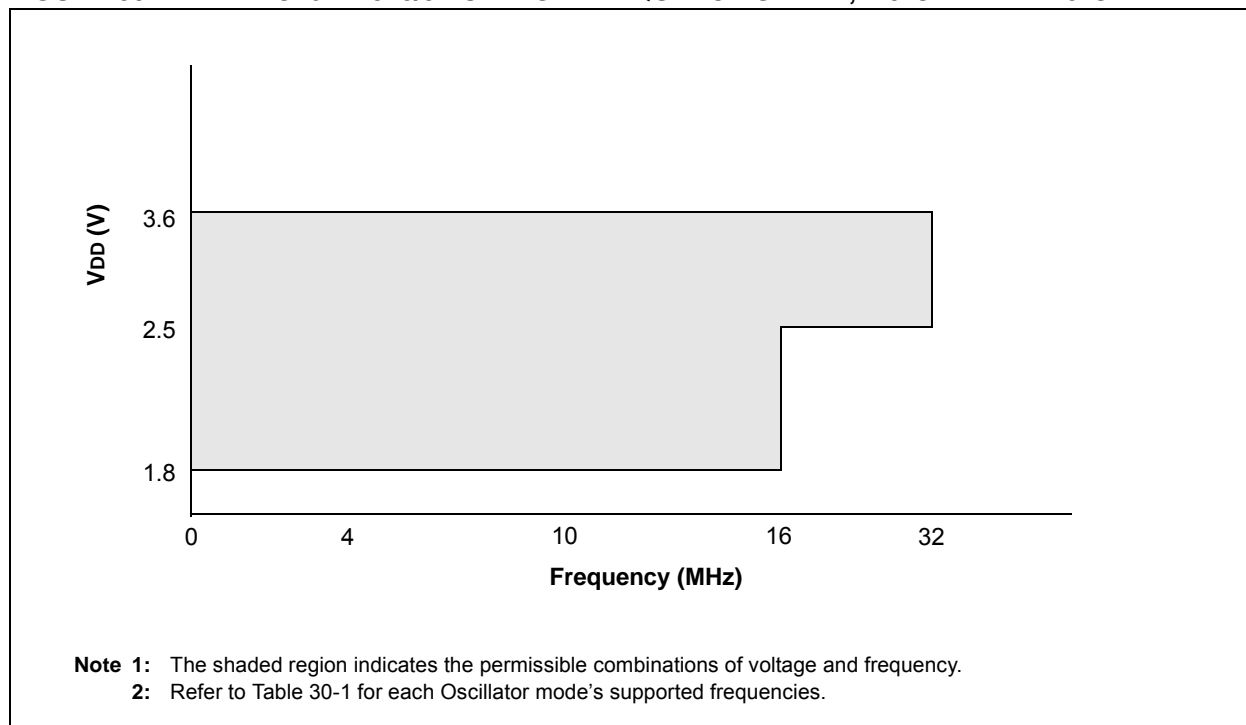
Description: Subtract W and the BORROW flag (CARRY) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.



**FIGURE 30-1: PIC16F1825/9 VOLTAGE FREQUENCY GRAPH,  $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$**



**FIGURE 30-2: PIC16LF1825/9 VOLTAGE FREQUENCY GRAPH,  $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$**





# PIC16(L)F1825/9

FIGURE 30-4: POR AND POR REARM WITH SLOW RISING  $V_{DD}$

