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	40MHz
Connectivity	EBI/EMI, 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	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°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/pic18lf8520t-i-pt

### 2.6.2 OSCILLATOR TRANSITIONS

PIC18FXX20 devices contain circuitry to prevent "glitches" when switching between oscillator sources. Essentially, the circuitry waits for eight rising edges of the clock source that the processor is switching to. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

A timing diagram indicating the transition from the main oscillator to the Timer1 oscillator is shown in Figure 2-8. The Timer1 oscillator is assumed to be running all the time. After the SCS bit is set, the processor is frozen at the next occurring Q1 cycle. After eight synchronization cycles are counted from the Timer1 oscillator, operation resumes. No additional delays are required after the synchronization cycles.

FIGURE 2-8: TIMING DIAGRAM FOR TRANSITION FROM OSC1 TO TIMER1 OSCILLATOR

Note 1: Delay on internal system clock is eight oscillator cycles for synchronization.

The sequence of events that takes place when switching from the Timer1 oscillator to the main oscillator will depend on the mode of the main oscillator. In addition to eight clock cycles of the main oscillator, additional delays may take place.

If the main oscillator is configured for an external crystal (HS, XT, LP), then the transition will take place after an oscillator start-up time (T_{OST}) has occurred. A timing diagram, indicating the transition from the Timer1 oscillator to the main oscillator for HS, XT and LP modes, is shown in Figure 2-9.

FIGURE 2-9: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS, XT, LP)

Note 1: T_{OST} = 1024 T_{OSC} (drawing not to scale).

4.9 Data Memory Organization

The data memory is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The data memory map is in turn divided into 16 banks of 256 bytes each. The lower 4 bits of the Bank Select Register (BSR<3:0>) select which bank will be accessed. The upper 4 bits of the BSR are not implemented.

The data memory space contains both Special Function Registers (SFR) and General Purpose Registers (GPR). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratch pad operations in the user's application. The SFRs start at the last location of Bank 15 (0FFFh) and extend downwards. Any remaining space beyond the SFRs in the Bank may be implemented as GPRs. GPRs start at the first location of Bank 0 and grow upwards. Any read of an unimplemented location will read as '0's.

PIC18FX520 devices have 2048 bytes of data RAM, extending from Bank 0 to Bank 7 (000h through 7FFh). PIC18FX620 and PIC18FX720 devices have 3840 bytes of data RAM, extending from Bank 0 to Bank 14 (000h through EFFh). The organization of the data memory space for these devices is shown in Figure 4-6 and Figure 4-7.

The entire data memory may be accessed directly or indirectly. Direct addressing may require the use of the BSR register. Indirect addressing requires the use of a File Select Register (FSRn) and a corresponding Indirect File Operand (INDFn). Each FSR holds a 12-bit address value that can be used to access any location in the data memory map without banking.

The instruction set and architecture allow operations across all banks. This may be accomplished by indirect addressing, or by the use of the MOVFF instruction. The MOVFF instruction is a two-word/two-cycle instruction that moves a value from one register to another.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM. **Section 4.10 "Access Bank"** provides a detailed description of the Access RAM.

4.9.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly or indirectly. Indirect addressing operates using a File Select Register and corresponding Indirect File Operand. The operation of indirect addressing is shown in **Section 4.12 "Indirect Addressing, INDF and FSR Registers"**.

Enhanced MCU devices may have banked memory in the GPR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

Data RAM is available for use as General Purpose Registers by all instructions. The top section of Bank 15 (F60h to FFFh) contains SFRs. All other banks of data memory contain GPR registers, starting with Bank 0.

4.9.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 4-2 and Table 4-3.

The SFRs can be classified into two sets: those associated with the "core" function and those related to the peripheral functions. Those registers related to the "core" are described in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature. The SFRs are typically distributed among the peripherals whose functions they control.

The unused SFR locations are unimplemented and read as '0's. The addresses for the SFRs are listed in Table 4-2.

REGISTER 5-1:   EECON1 REGISTER (ADDRESS FA6h)

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD

bit 7   bit 0

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit

1 = Access Flash program memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit

1 = Access configuration registers
0 = Access Flash program or data EEPROM memory

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

bit 4 **FREE:** Flash Row Erase Enable bit

1 = Erase the program memory row addressed by TBLPTR on the next WR command
(cleared by completion of erase operation)
0 = Perform write only

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit

1 = A write operation is prematurely terminated
(any Reset during self-timed programming in normal operation)
0 = The write operation completed

 Note: When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.

bit 2 **WREN:** Flash Program/Data EEPROM Write Enable bit

1 = Allows write cycles to Flash program/data EEPROM
0 = Inhibits write cycles to Flash program/data EEPROM

bit 1 **WR:** Write Control bit

1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit

1 = Initiates an EEPROM read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)
0 = Does not initiate an EEPROM read

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

EXAMPLE 5-3: WRITING TO FLASH PROGRAM MEMORY

```
                MOVLW   D'64                    ; number of bytes in erase block
                MOVWF   COUNTER
                MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
                MOVWF   FSR0H
                MOVLW   BUFFER_ADDR_LOW
                MOVWF   FSR0L
                MOVLW   CODE_ADDR_UPPER         ; Load TBLPTR with the base
                MOVWF   TBLPTRU                 ; address of the memory block
                MOVLW   CODE_ADDR_HIGH
                MOVWF   TBLPTRH
                MOVLW   CODE_ADDR_LOW
                MOVWF   TBLPTRL
READ_BLOCK
                TBLRD*+                         ; read into TABLAT, and inc
                MOVF    TABLAT, W               ; get data
                MOVWF   POSTINC0                ; store data
                DECFSZ  COUNTER                 ; done?
                BRA     READ_BLOCK              ; repeat
MODIFY_WORD
                MOVLW   DATA_ADDR_HIGH          ; point to buffer
                MOVWF   FSR0H
                MOVLW   DATA_ADDR_LOW
                MOVWF   FSR0L
                MOVLW   NEW_DATA_LOW            ; update buffer word
                MOVWF   POSTINC0
                MOVLW   NEW_DATA_HIGH
                MOVWF   INDF0
ERASE_BLOCK
                MOVLW   CODE_ADDR_UPPER         ; load TBLPTR with the base
                MOVWF   TBLPTRU                 ; address of the memory block
                MOVLW   CODE_ADDR_HIGH
                MOVWF   TBLPTRH
                MOVLW   CODE_ADDR_LOW
                MOVWF   TBLPTRL
                BSF     EECON1, EEPGD           ; point to Flash program memory
                BCF     EECON1, CFGS            ; access Flash program memory
                BSF     EECON1, WREN            ; enable write to memory
                BSF     EECON1, FREE            ; enable Row Erase operation
                BCF     INTCON, GIE             ; disable interrupts
                MOVLW   55h
                MOVWF   EECON2                  ; write 55H
    Required    MOVLW   AAh
    Sequence    MOVWF   EECON2                  ; write AAH
                BSF     EECON1, WR              ; start erase (CPU stall)
                NOP
                BSF     INTCON, GIE             ; re-enable interrupts
                TBLRD*-                         ; dummy read decrement
WRITE_BUFFER_BACK
                MOVLW   8                       ; number of write buffer groups of 8 bytes
                MOVWF   COUNTER_HI
                MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
                MOVWF   FSR0H
                MOVLW   BUFFER_ADDR_LOW
                MOVWF   FSR0L
PROGRAM_LOOP
                MOVLW   8                       ; number of bytes in holding register
                MOVWF   COUNTER
WRITE_WORD_TO_HREGS
                MOVFF   POSTINC0, WREG          ; get low byte of buffer data
                                                ; present data to table latch
                TBLWT+*                         ; write data, perform a short write
                                                ; to internal TBLWT holding register.
                DECFSZ  COUNTER                 ; loop until buffers are full
                BRA     WRITE_WORD_TO_HREGS
```

FIGURE 10-6: **BLOCK DIAGRAM OF RB2:RB0 PINS**

Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (OPTION_REG<7>).
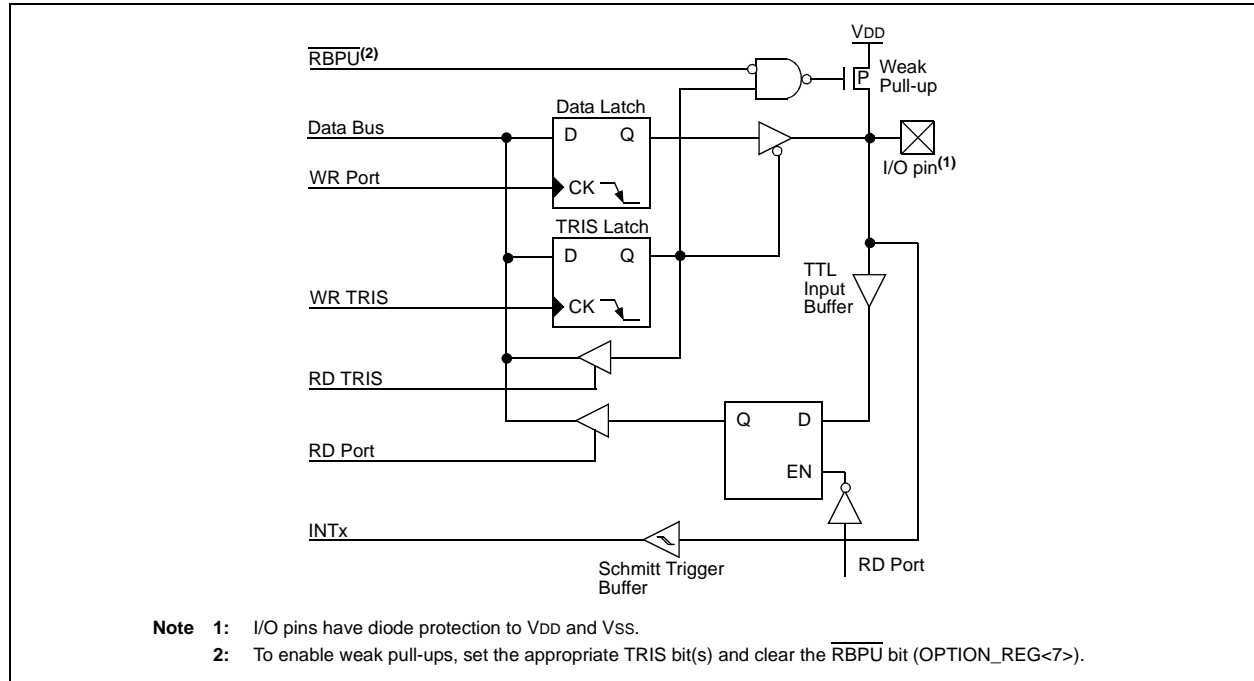
FIGURE 10-7: **BLOCK DIAGRAM OF RB3 PIN**

Note 1: I/O pin has diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2<7>).
3: The CCP2 input/output is multiplexed with RB3 if the CCP2MX bit is enabled (= 0) in the Configuration register and the device is operating in Microprocessor, Microprocessor with Boot Block or Extended Microcontroller mode.
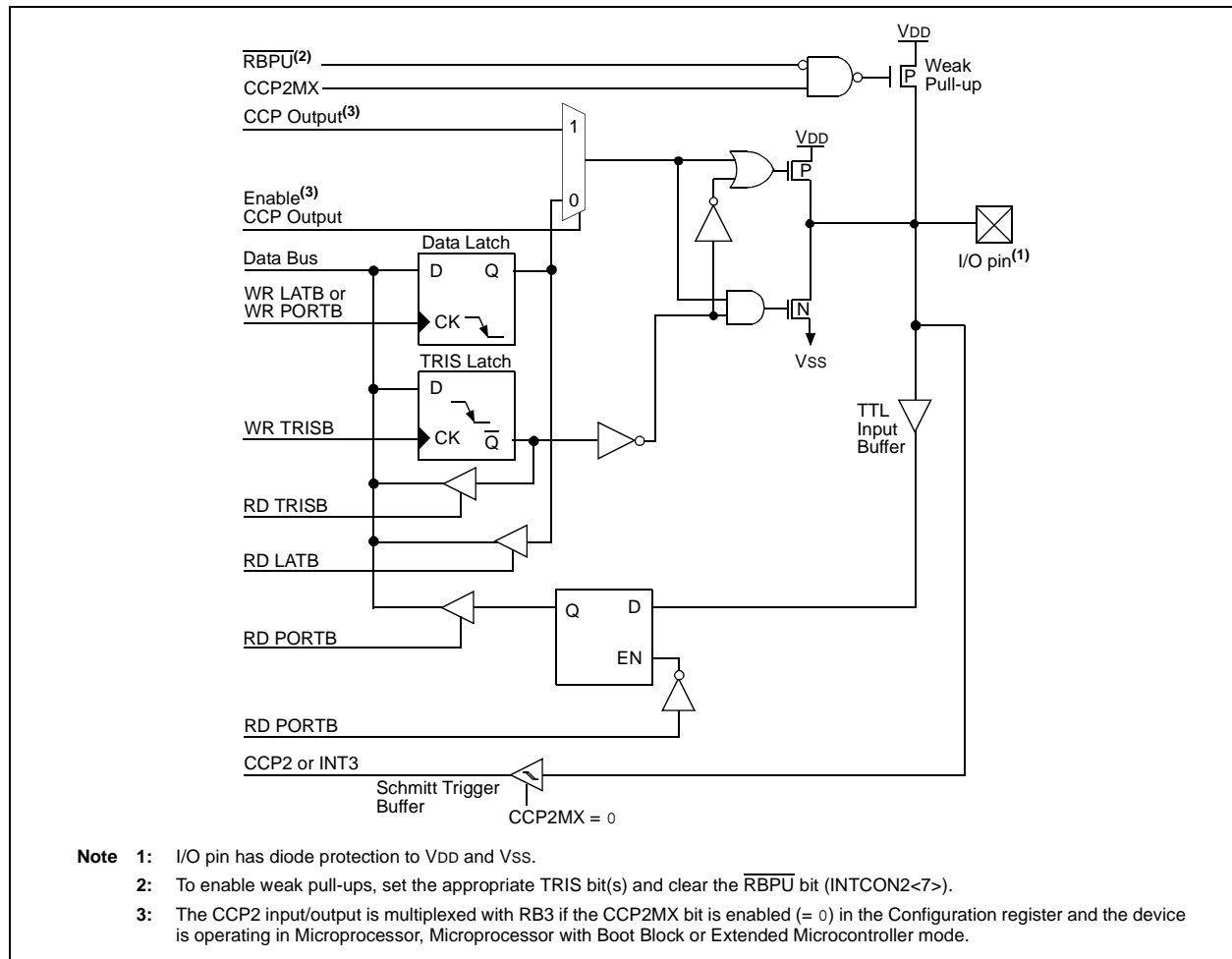
PIC18F6520/8520/6620/8620/6720/8720

10.10 Parallel Slave Port

PORTD also operates as an 8-bit wide Parallel Slave Port, or microprocessor port, when control bit PSPMODE (PSPCON<4>) is set. It is asynchronously readable and writable by the external world through the $\overline{RD}$ control input pin, RE0/$\overline{RD}$/AD8 and the $\overline{WR}$ control input pin, RE1/$\overline{WR}$/AD9.

> **Note:** For PIC18F8X20 devices, the Parallel Slave Port is available only in Microcontroller mode.

The PSP can directly interface to an 8-bit micro-processor data bus. The external microprocessor can read or write the PORTD latch as an 8-bit latch. Setting bit PSPMODE enables port pin RE0/$\overline{RD}$/AD8 to be the $\overline{RD}$ input, RE1/$\overline{WR}$/AD9 to be the $\overline{WR}$ input and RE2/$\overline{CS}$/AD10 to be the $\overline{CS}$ (Chip Select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (set). The A/D port configuration bits, PCFG2:PCFG0 (ADCON1<2:0>), must be set which will configure pins RE2:RE0 as digital I/O.

A write to the PSP occurs when both the $\overline{CS}$ and $\overline{WR}$ lines are first detected low. A read from the PSP occurs when both the $\overline{CS}$ and $\overline{RD}$ lines are first detected low.

The PORTE I/O pins become control inputs for the microprocessor port when bit PSPMODE (PSPCON<4>) is set. In this mode, the user must make sure that the TRISE<2:0> bits are set (pins are config-ured as digital inputs) and the ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

FIGURE 10-23: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)

Note: I/O pin has protection diodes to VDD and VSS.
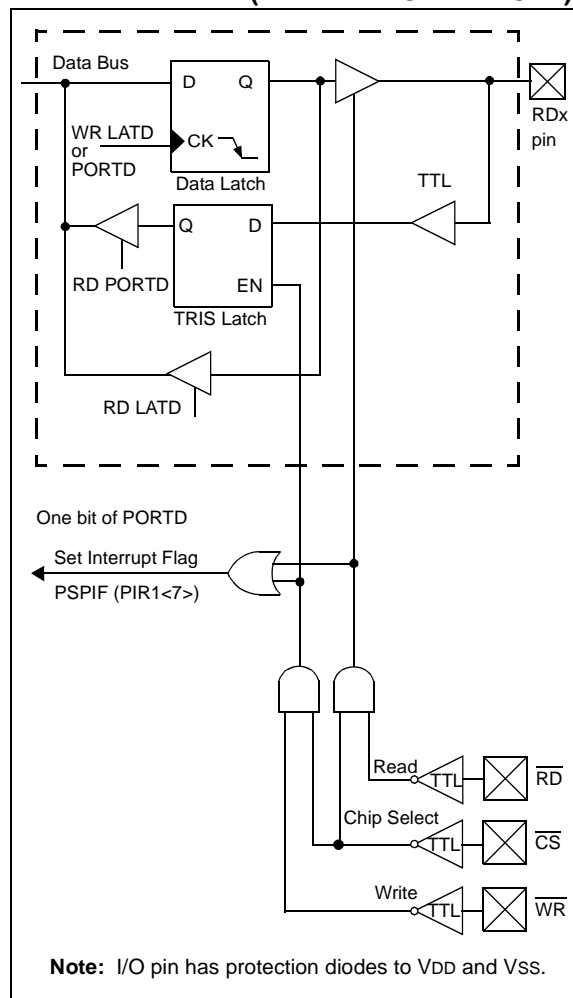
REGISTER 17-5: SSPCON2: MSSP CONTROL REGISTER 2 (I²C MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN

bit 7   bit 0

bit 7   **GCEN:** General Call Enable bit (Slave mode only)
            1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
            0 = General call address disabled

bit 6   **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
            1 = Acknowledge was not received from slave
            0 = Acknowledge was received from slave

bit 5   **ACKDT:** Acknowledge Data bit (Master Receive mode only)
            1 = Not Acknowledge
            0 = Acknowledge

         Note: Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

bit 4   **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)
            1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware.
            0 = Acknowledge sequence Idle

bit 3   **RCEN:** Receive Enable bit (Master mode only)
            1 = Enables Receive mode for I²C
            0 = Receive Idle

bit 2   **PEN:** Stop Condition Enable bit (Master mode only)
            1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
            0 = Stop condition Idle

bit 1   **RSEN:** Repeated Start Condition Enabled bit (Master mode only)
            1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
            0 = Repeated Start condition Idle

bit 0   **SEN:** Start Condition Enabled/Stretch Enabled bit
         <u>In Master mode:</u>
            1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
            0 = Start condition Idle
         <u>In Slave mode:</u>
            1 = Clock stretching is enabled for both Slave Transmit and Slave Receive (stretch enabled)
            0 = Clock stretching is disabled

> **Note:**   For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I²C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

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

17.4.3.2   Reception

When the R/$\overline{\text{W}}$ bit of the address byte is clear and an address match occurs, the R/$\overline{\text{W}}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and the SDA line is held low ($\overline{\text{ACK}}$).

When the address byte overflow condition exists, then the no Acknowledge ($\overline{\text{ACK}}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT<0>) is set, or bit SSPOV (SSPCON1<6>) is set.

An MSSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR1<3>) must be cleared in software. The SSPSTAT register is used to determine the status of the byte.

If SEN is enabled (SSPCON1<0> = 1), RC3/SCK/SCL will be held low (clock stretch) following each data transfer. The clock must be released by setting bit CKP (SSPCON<4>).   See **Section 17.4.4 "Clock Stretching"** for more detail.

17.4.3.3   Transmission

When the R/$\overline{\text{W}}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{\text{W}}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{\text{ACK}}$ pulse will be sent on the ninth bit and pin RC3/SCK/SCL is held low, regardless of SEN (see **Section 17.4.4 "Clock Stretching"**, for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then pin RC3/SCK/SCL should be enabled by setting bit CKP (SSPCON1<4>). The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time (Figure 17-9).

The $\overline{\text{ACK}}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not $\overline{\text{ACK}}$), then the data transfer is complete. In this case, when the $\overline{\text{ACK}}$ is latched by the slave, the slave logic is reset (resets SSPSTAT register) and the slave monitors for another occurrence of the Start bit. If the SDA line was low ($\overline{\text{ACK}}$), the next transmit data must be loaded into the SSPBUF register. Again, pin RC3/SCK/SCL must be enabled by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

FIGURE 17-9: **I²C SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)**
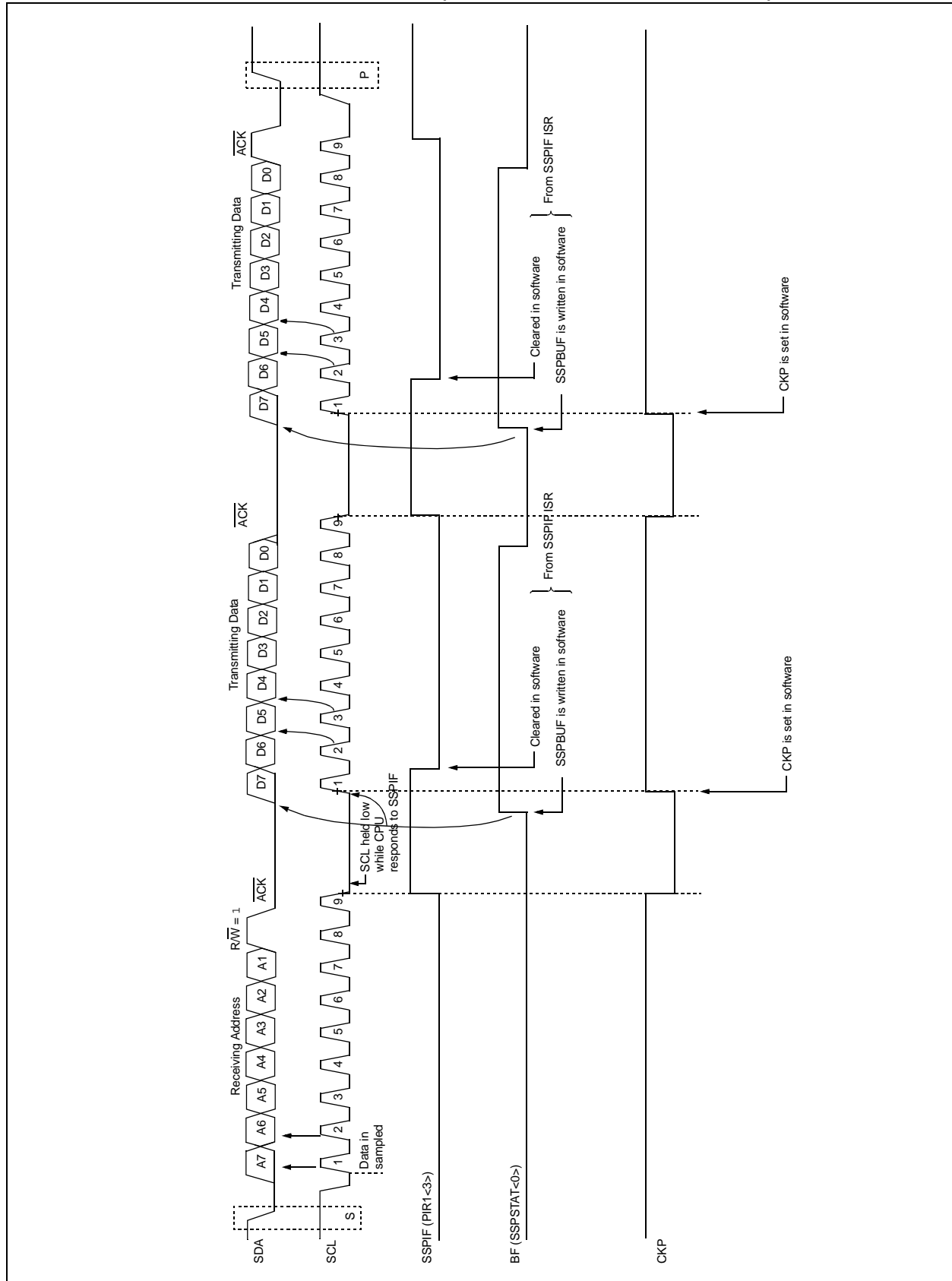
17.4.4 CLOCK STRETCHING

Both 7- and 10-bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

17.4.4.1 Clock Stretching for 7-bit Slave Receive Mode (SEN = 1)

In 7-bit Slave Receive mode, on the falling edge of the ninth clock at the end of the $\overline{\text{ACK}}$ sequence, if the BF bit is set, the CKP bit in the SSPCON1 register is automatically cleared, forcing the SCL output to be held low. The CKP being cleared to '0' will assert the SCL line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the SSPBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 17-13).

> **Note 1:** If the user reads the contents of the SSPBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.
>
> **2:** The CKP bit can be set in software, regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence, in order to prevent an overflow condition.

17.4.4.2 Clock Stretching for 10-bit Slave Receive Mode (SEN = 1)

In 10-bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{\text{W}}$ bit cleared to '0'. The release of the clock line occurs upon updating SSPADD. Clock stretching will occur on each data receive sequence, as described in 7-bit mode.

> **Note:** If the user polls the UA bit and clears it by updating the SSPADD register before the falling edge of the ninth clock occurs and if the user hasn't cleared the BF bit by reading the SSPBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

17.4.4.3 Clock Stretching for 7-bit Slave Transmit Mode

7-bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock, if the BF bit is clear. This occurs, regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the SSPBUF before the master device can initiate another transmit sequence (see Figure 17-9).

> **Note 1:** If the user loads the contents of SSPBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.
>
> **2:** The CKP bit can be set in software, regardless of the state of the BF bit.

17.4.4.4 Clock Stretching for 10-bit Slave Transmit Mode

In 10-bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-bit Slave Receive mode. The first two addresses are followed by a third address sequence, which contains the high-order bits of the 10-bit address and the R/$\overline{\text{W}}$ bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled as in 7-bit Slave Transmit mode (see Figure 17-11).

18.4   USART Synchronous Slave Mode

Synchronous Slave mode differs from the Master mode in the fact that the shift clock is supplied externally at the TXx pin (RC6/TX1/CK1 or RG1/TX2/CK2), instead of being supplied internally in Master mode. TRISC<6> must be set for this mode. This allows the device to transfer or receive data while in Sleep mode. Slave mode is entered by clearing bit CSRC (TXSTAx<7>).

18.4.1   USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes are identical, except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.

b) The second word will remain in TXREG register.

c) Flag bit TXxIF will not be set.

d) When the first word has been shifted out of TSR, the TXREGx register will transfer the second word to the TSR and flag bit TXxIF will now be set.

e) If enable bit TXxIE is set, the interrupt will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting bits SYNC and SPEN and clearing bit CSRC.

2. Clear bits CREN and SREN.

3. If interrupts are desired, set enable bit TXxIE.

4. If 9-bit transmission is desired, set bit TX9.

5. Enable the transmission by setting enable bit TXEN.

6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.

7. Start transmission by loading data to the TXREGx register.

8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 18-10:   REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE 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 0000	0000 0000
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0111 1111	0111 1111
PIR3	—	—	RC2IF	TX2IF	TMR4IF	CCP5IF	CCP4IF	CCP3IF	--00 0000	--00 0000
PIE3	—	—	RC2IE	TX2IE	TMR4IE	CCP5IE	CCP4IE	CCP3IE	--00 0000	--00 0000
IPR3	—	—	RC2IP	TX2IP	TMR4IP	CCP5IP	CCP4IP	CCP3IP	--11 1111	--11 1111
RCSTAx[1]	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
TXREGx[1]	USART Transmit Register								0000 0000	0000 0000
TXSTAx[1]	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRGx[1]	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, − = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

Note   1:   Register names generically refer to both of the identically named registers for the two USART modules, where 'x' indicates the particular module. Bit names and Reset values are identical between modules.

BTG	Bit Toggle f
Syntax:	[*label*] BTG f,b[,a]
Operands:	$0 \leq f \leq 255$ $0 \leq b < 7$ $a \in [0,1]$
Operation:	$(\overline{f<b>}) \rightarrow f<b>$
Status Affected:	None

Encoding:

0111	bbba	ffff	ffff

Description: Bit 'b' in data memory location 'f' is inverted. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: BTG PORTC, 4, 0

Before Instruction:
PORTC = 0111 0101 [0x75]
After Instruction:
PORTC = 0110 0101 [0x65]

BOV	Branch if Overflow
Syntax:	[*label*] BOV n
Operands:	$-128 \leq n \leq 127$
Operation:	if Overflow bit is '1' $(PC) + 2 + 2n \rightarrow PC$
Status Affected:	None

Encoding:

1110	0100	nnnn	nnnn

Description: If the Overflow bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: HERE BOV Jump

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

COMF	Complement f

Syntax:	[*label*] COMF f [,d [,a]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	$(\overline{f}) \rightarrow$ dest
Status Affected:	N, Z
Encoding:	0001 | 11da | ffff | ffff
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' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: COMF   REG, 0, 0

Before Instruction
  REG   =   0x13
After Instruction
  REG   =   0x13
  W   =   0xEC

CPFSEQ	Compare f with W, skip if f = W

Syntax:	[*label*] CPFSEQ f [,a]
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(f) – (W), skip if (f) = (W) (unsigned comparison)
Status Affected:	None
Encoding:	0110 | 001a | ffff | ffff
Description:	Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
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   CPFSEQ REG, 0
                  NEQUAL   :
                  EQUAL   :

Before Instruction
  PC Address   =   HERE
  W   =   ?
  REG   =   ?
After Instruction
  If REG   =   W;
      PC   =   Address (EQUAL)
  If REG   ≠   W;
      PC   =   Address (NEQUAL)

MOVLW	**Move literal to W**
Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → W
Status Affected:	None

Encoding:

0000	1110	kkkk	kkkk

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

        W   =   0x5A

MOVWF	**Move W to f**
Syntax:	[*label*] MOVWF f [,a]
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(W) → f
Status Affected:	None

Encoding:

0110	111a	ffff	ffff

Description:	Move data from W 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' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: MOVWF   REG, 0

    Before Instruction

        W   =   0x4F
        REG   =   0xFF

    After Instruction

        W   =   0x4F
        REG   =   0x4F

RCALL	Relative Call
Syntax:	[*label*] RCALL n
Operands:	$-1024 \leq n \leq 1023$
Operation:	(PC) + 2 $\rightarrow$ TOS, (PC) + 2 + 2n $\rightarrow$ PC
Status Affected:	None

Encoding:

1101	1nnn	nnnn	nnnn

Description: Subroutine call with a jump up to 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: HERE   RCALL Jump

Before Instruction
PC = Address (HERE)

After Instruction
PC = Address (Jump)
TOS = Address (HERE+2)

RESET	Reset
Syntax:	[*label*] RESET
Operands:	None
Operation:	Reset all registers and flags that are affected by a $\overline{MCLR}$ Reset.
Status Affected:	All

Encoding:

0000	0000	1111	1111

Description: This instruction provides a way to execute a $\overline{MCLR}$ Reset in software.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start Reset	No operation	No operation

Example: RESET

After Instruction
Registers =   Reset Value
Flags*   =   Reset Value

RLNCF	Rotate Left f (no carry)
Syntax:	[*label*] RLNCF f [,d [,a]
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$(f<n>) \rightarrow dest<n+1>$, $(f<7>) \rightarrow dest<0>$
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 will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).
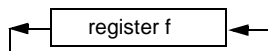
register f

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:	[*label*] RRCF f [,d [,a]
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$(f<n>) \rightarrow dest<n-1>$, $(f<0>) \rightarrow C$, $(C) \rightarrow dest<7>$
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 will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).
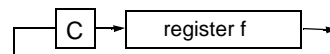
C → register f

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

SLEEP	Enter SLEEP mode
Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h $\rightarrow$ WDT, 0 $\rightarrow$ WDT postscaler, 1 $\rightarrow$ $\overline{\text{TO}}$, 0 $\rightarrow$ $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$

Encoding:

0000	0000	0000	0011

Description: The Power-down status bit ($\overline{\text{PD}}$) is cleared. The Time-out status bit ($\overline{\text{TO}}$) is set. Watchdog Timer and its postscaler are cleared.
The processor is put into Sleep mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Go to Sleep

Example: SLEEP

Before Instruction
$\overline{\text{TO}}$ = ?
$\overline{\text{PD}}$ = ?

After Instruction
$\overline{\text{TO}}$ = 1 †
$\overline{\text{PD}}$ = 0

† If WDT causes wake-up, this bit is cleared.

SUBFWB	Subtract f from W with borrow
Syntax:	[*label*] SUBFWB f [,d [,a]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) $-$ (f) $-$ ($\overline{\text{C}}$) $\rightarrow$ dest
Status Affected:	N, OV, C, DC, Z

Encoding:

0101	01da	ffff	ffff

Description: Subtract register 'f' and Carry flag (borrow) from W (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1:   SUBFWB   REG, 1, 0

Before Instruction
REG = 3
W = 2
C = 1
After Instruction
REG = FF
W = 2
C = 0
Z = 0
N = 1   ; result is negative

Example 2:   SUBFWB   REG, 0, 0

Before Instruction
REG = 2
W = 5
C = 1
After Instruction
REG = 2
W = 3
C = 1
Z = 0
N = 0   ; result is positive

Example 3:   SUBFWB   REG, 1, 0

Before Instruction
REG = 1
W = 2
C = 0
After Instruction
REG = 0
W = 2
C = 1
Z = 1   ; result is zero
N = 0

26.1   DC Characteristics: Supply Voltage
PIC18F6520/8520/6620/8620/6720/8720 (Industrial, Extended)
PIC18LF6520/8520/6620/8620/6720/8720 (Industrial)

PIC18LF6520/8520/6620/8620/6720/8720 (Industrial)			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
PIC18F6520/8520/6620/8620/6720/8720 (Industrial, Extended)			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
D001	VDD	**Supply Voltage**					
		PIC18LFXX20	2.0	—	5.5	V	HS, XT, RC and LP Oscillator mode
		PIC18FXX20	4.2	—	5.5	V	
D001A	AVDD	**Analog Supply Voltage**	VDD – 0.3	—	VDD + 0.3	V	
D002	VDR	**RAM Data Retention Voltage**[1]	1.5	—	—	V	
D003	VPOR	**VDD Start Voltage** to ensure internal Power-on Reset signal	—	—	0.7	V	See section on Power-on Reset for details
D004	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05	—	—	V/ms	See section on Power-on Reset for details
D005	VBOR	**Brown-out Reset Voltage**					
		BORV1:BORV0 = 11	N/A	—	N/A	V	Reserved
		BORV1:BORV0 = 10	2.64	—	2.92	V	
		BORV1:BORV0 = 01	4.11	—	4.55	V	
		BORV1:BORV0 = 00	4.41	—	4.87	V	

Legend: Shading of rows is to assist in readability of the table.

Note   1:   This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

PIC18F6520/8520/6620/8620/6720/8720

QUALITY MANAGEMENT SYSTEM

CERTIFIED BY DNV

═ ISO/TS 16949 ═