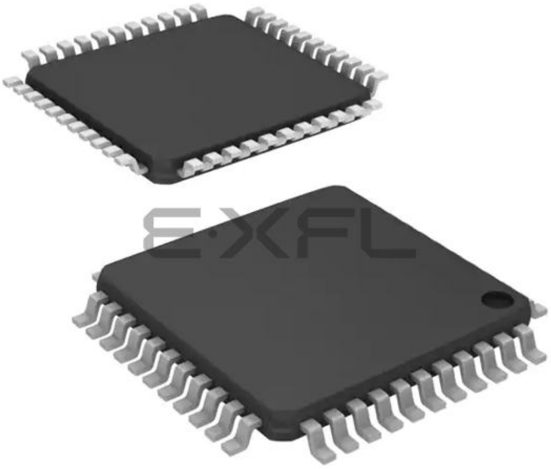
Microchip Technology - PIC18F4620T-I/PT Datasheet



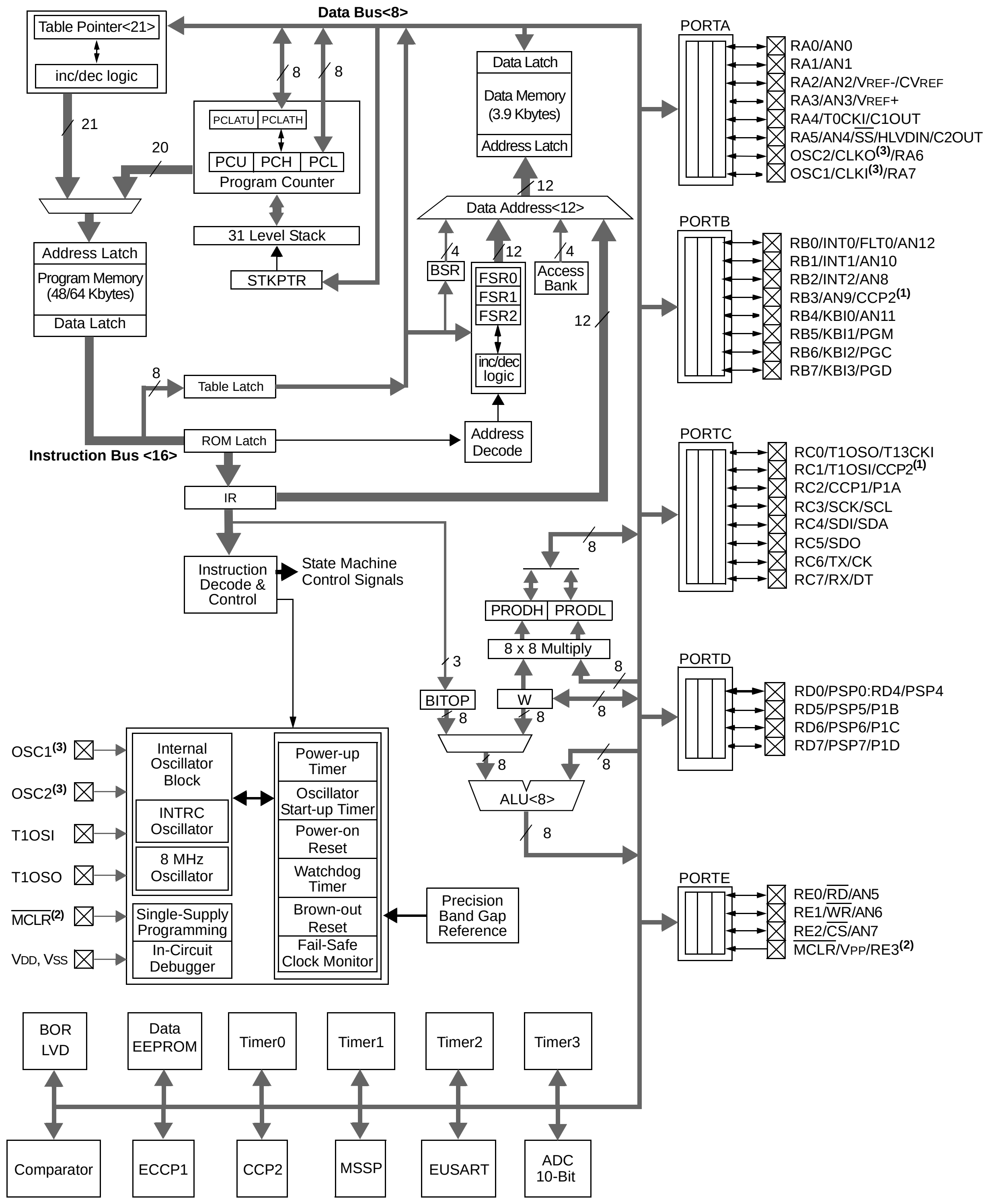
What is "Embedded - Microcontrollers"?

"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	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	64KB (32K 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 13x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f4620t-i-pt

FIGURE 1-2: PIC18F4525/4620 (40/44-PIN) BLOCK DIAGRAM



Note 1: CCP2 is multiplexed with RC1 when Configuration bit, CCP2MX, is set, or RB3 when CCP2MX is not set.

2: RE3 is only available when MCLR functionality is disabled.

3: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 “Oscillator Configurations”** for additional information.

2.7 Clock Sources and Oscillator Switching

Like previous PIC18 devices, the PIC18F2525/2620/4525/4620 family includes a feature that allows the device clock source to be switched from the main oscillator to an alternate, low-frequency clock source. PIC18F2525/2620/4525/4620 devices offer two alternate clock sources. When an alternate clock source is enabled, the various power-managed operating modes are available.

Essentially, there are three clock sources for these devices:

- Primary oscillators
- Secondary oscillators
- Internal oscillator block

The **primary oscillators** include the External Crystal and Resonator modes, the External RC modes, the External Clock modes and the internal oscillator block. The particular mode is defined by the FOSC3:FOSC0 Configuration bits. The details of these modes are covered earlier in this chapter.

The **secondary oscillators** are those external sources not connected to the OSC1 or OSC2 pins. These sources may continue to operate even after the controller is placed in a power-managed mode.

PIC18F2525/2620/4525/4620 devices offer the Timer1 oscillator as a secondary oscillator. This oscillator, in all power-managed modes, is often the time base for functions such as a Real-Time Clock (RTC).

Most often, a 32.768 kHz watch crystal is connected between the RC0/T1OSO/T13CKI and RC1/T1OSI pins. Like the LP Oscillator mode circuit, loading capacitors are also connected from each pin to ground.

The Timer1 oscillator is discussed in greater detail in **Section 12.3 "Timer1 Oscillator"**.

In addition to being a primary clock source, the **internal oscillator block** is available as a power-managed mode clock source. The INTRC source is also used as the clock source for several special features, such as the WDT and Fail-Safe Clock Monitor.

The clock sources for the PIC18F2525/2620/4525/4620 devices are shown in Figure 2-8. See **Section 23.0 "Special Features of the CPU"** for Configuration register details.

FIGURE 2-8: PIC18F2525/2620/4525/4620 CLOCK DIAGRAM

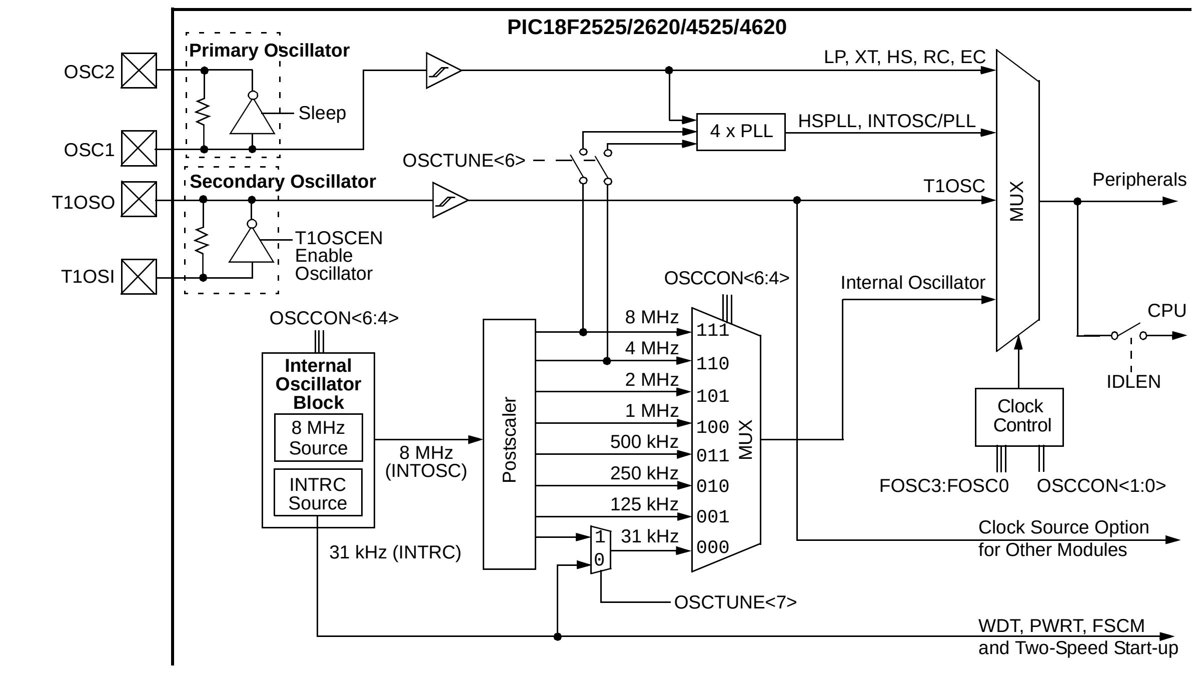


TABLE 4-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Applicable Devices				Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
TOSU	2525	2620	4525	4620	`---0 0000`	`---0 0000`	`---0 uuuu`(3)
TOSH	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`(3)
TOSL	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`(3)
STKPTR	2525	2620	4525	4620	`00-0 0000`	`uu-0 0000`	`uu-u uuuu`(3)
PCLATU	2525	2620	4525	4620	`---0 0000`	`---0 0000`	`---u uuuu`
PCLATH	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
PCL	2525	2620	4525	4620	`0000 0000`	`0000 0000`	PC + 2(2)
TBLPTRU	2525	2620	4525	4620	`--00 0000`	`--00 0000`	`--uu uuuu`
TBLPTRH	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
TBLPTRL	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
TABLAT	2525	2620	4525	4620	`0000 0000`	`0000 0000`	`uuuu uuuu`
PRODH	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PRODL	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
INTCON	2525	2620	4525	4620	`0000 000x`	`0000 000u`	`uuuu uuuu`(1)
INTCON2	2525	2620	4525	4620	`1111 -1-1`	`1111 -1-1`	`uuuu -u-u`(1)
INTCON3	2525	2620	4525	4620	`11-0 0-00`	`11-0 0-00`	`uu-u u-uu`(1)
INDF0	2525	2620	4525	4620	N/A	N/A	N/A
POSTINC0	2525	2620	4525	4620	N/A	N/A	N/A
POSTDEC0	2525	2620	4525	4620	N/A	N/A	N/A
PREINC0	2525	2620	4525	4620	N/A	N/A	N/A
PLUSW0	2525	2620	4525	4620	N/A	N/A	N/A
FSR0H	2525	2620	4525	4620	`---- 0000`	`---- 0000`	`---- uuuu`
FSR0L	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
WREG	2525	2620	4525	4620	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
INDF1	2525	2620	4525	4620	N/A	N/A	N/A
POSTINC1	2525	2620	4525	4620	N/A	N/A	N/A
POSTDEC1	2525	2620	4525	4620	N/A	N/A	N/A
PREINC1	2525	2620	4525	4620	N/A	N/A	N/A
PLUSW1	2525	2620	4525	4620	N/A	N/A	N/A

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 4-3 for Reset value for specific condition.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

5.1.2.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 5-1) contains the Stack Pointer value, the STKFUL (Stack Full) status bit and the STKUNF (Stack Underflow) status bits. The value of the Stack Pointer can be 0 through 31. The Stack Pointer increments before values are pushed onto the stack and decrements after values are popped off the stack. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit is cleared by software or by a POR.

The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 23.1 “Configuration Bits”** for a description of the device Configuration bits.) If STVREN is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit and reset the device. The STKFUL bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the Stack Pointer will increment to 31. Any additional pushes will not overwrite the 31st push and STKPTR will remain at 31.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and sets the STKUNF bit, while the Stack Pointer remains at zero. The STKUNF bit will remain set until cleared by software or until a POR occurs.

Note:	Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken. This is not the same as a Reset, as the contents of the SFRs are not affected.

5.1.2.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable feature. The PIC18 instruction set includes two instructions, PUSH and POP, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The PUSH instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The POP instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

REGISTER 5-1: STKPTR: STACK POINTER REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL[(1)]	STKUNF[(1)]	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

Legend:	C = Clearable only bit		
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 **STKFUL:** Stack Full Flag bit[(1)]
1 = Stack became full or overflowed
0 = Stack has not become full or overflowed

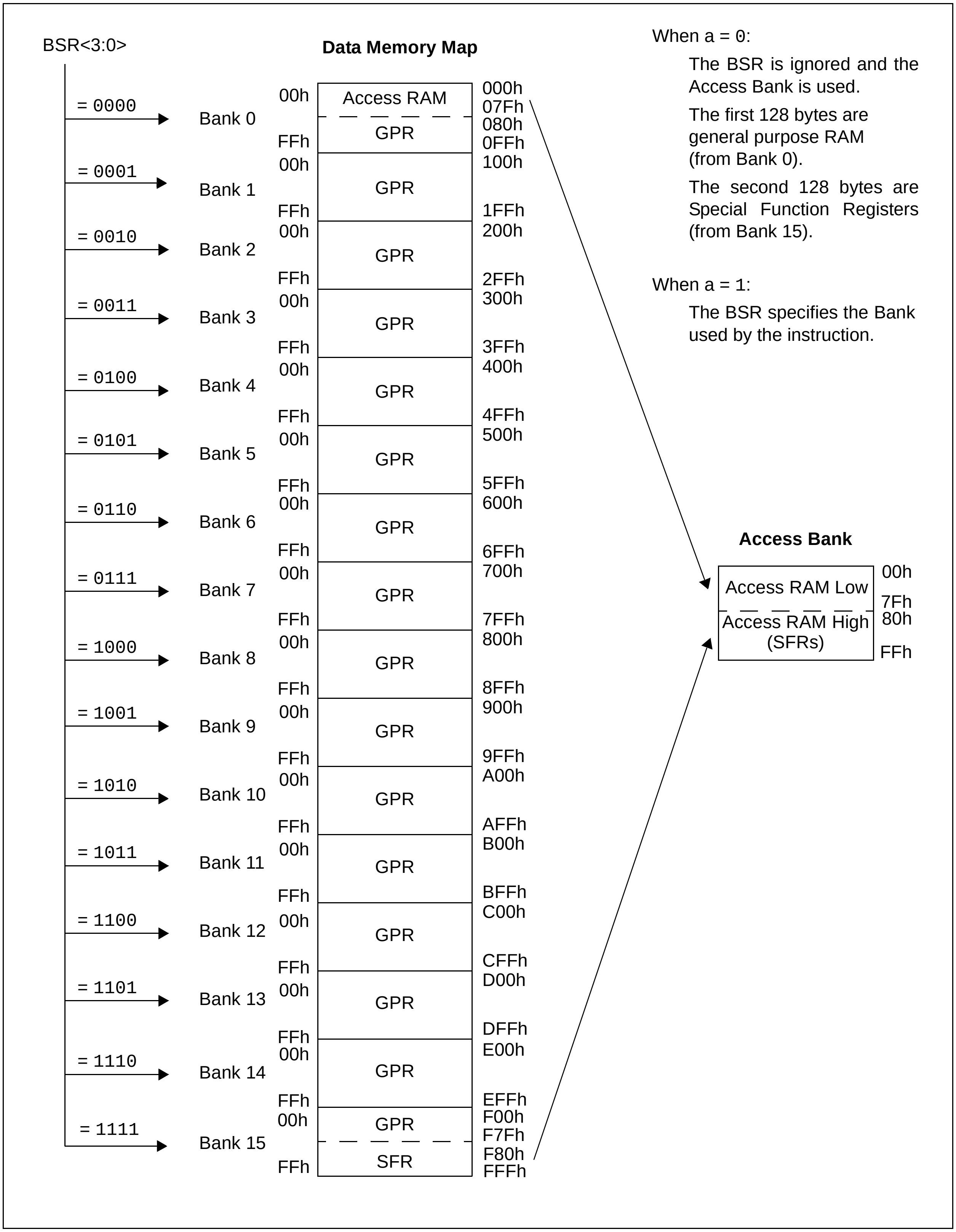
bit 6 **STKUNF:** Stack Underflow Flag bit[(1)]
1 = Stack underflow occurred
0 = Stack underflow did not occur

bit 5 **Unimplemented:** Read as ‘0’

bit 4-0 **SP4:SP0:** Stack Pointer Location bits

Note 1: Bit 7 and bit 6 are cleared by user software or by a POR.

FIGURE 5-5: DATA MEMORY MAP FOR PIC18F2525/2620/4525/4620 DEVICES



When a = 0:

The BSR is ignored and the Access Bank is used.

The first 128 bytes are general purpose RAM (from Bank 0).

The second 128 bytes are Special Function Registers (from Bank 15).

When a = 1:

The BSR specifies the Bank used by the instruction.

TABLE 5-2: REGISTER FILE SUMMARY (PIC18F2525/2620/4525/4620)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
TOSU	—	—	—	Top-of-Stack Upper Byte (TOS<20:16>)					---0 0000	49, 54
TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	49, 54
TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	49, 54
STKPTR	STKFUL(6)	STKUNF(6)	—	SP4	SP3	SP2	SP1	SP0	00-0 0000	49, 55
PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000	49, 54
PCLATH	Holding Register for PC<15:8>								0000 0000	49, 54
PCL	PC Low Byte (PC<7:0>)								0000 0000	49, 54
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--00 0000	49, 82
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	49, 82
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	49, 82
TABLAT	Program Memory Table Latch								0000 0000	49, 82
PRODH	Product Register High Byte								xxxx xxxx	49, 89
PRODL	Product Register Low Byte								xxxx xxxx	49, 89
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	49, 111
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1	49, 112
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	49, 113
INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								N/A	49, 68
POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								N/A	49, 68
POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								N/A	49, 68
PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								N/A	49, 68
PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W								N/A	49, 68
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte				---- 0000	49, 68
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	49, 68
WREG	Working Register								xxxx xxxx	49
INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								N/A	49, 68
POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								N/A	49, 68
POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								N/A	49, 68
PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								N/A	49, 68
PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W								N/A	49, 68
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte				---- 0000	50, 68
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	50, 68
BSR	—	—	—	—	Bank Select Register				---- 0000	50, 59
INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								N/A	50, 68
POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								N/A	50, 68
POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								N/A	50, 68
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								N/A	50, 68
PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W								N/A	50, 68
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte				---- 0000	50, 68
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	50, 68
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	50, 66

Legend: x = unknown, u = unchanged, – = unimplemented, q = value depends on condition

Note 1: The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as ‘0’. See **Section 4.4 “Brown-out Reset (BOR)”**.

2: These registers and/or bits are not implemented on 28-pin devices and are read as ‘0’. Reset values are shown for 40/44-pin devices; individual unimplemented bits should be interpreted as ‘-’.

3: The PLLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as ‘0’. See **Section 2.6.4 “PLL in INTOSC Modes”**.

4: The RE3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0); otherwise, RE3 reads as ‘0’. This bit is read-only.

5: RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as ‘0’.

6: Bit 7 and bit 6 are cleared by user software or by a POR.

REGISTER 7-1: EECON1: DATA EEPROM CONTROL REGISTER 1

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[1]	WREN	WR	RD
bit 7							bit 0

Legend:	S = Set only bit (cannot be cleared in software)		
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 **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]

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

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 or CFGS = 1.)
0 = Does not initiate an EEPROM read

Note 1: When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.

EXAMPLE 7-3: 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
Required            MOVWF     EECON2                   ; write 55h
Sequence            MOVLW     0AAh
                    MOVWF     EECON2                   ; write 0AAh
                    BSF       EECON1, WR               ; start erase (CPU stall)
                    BSF       INTCON, GIE              ; re-enable interrupts
                    TBLRD*-                            ; dummy read decrement
                    MOVLW     BUFFER_ADDR_HIGH         ; point to buffer
                    MOVWF     FSR0H
                    MOVLW     BUFFER_ADDR_LOW
                    MOVWF     FSR0L
WRITE_BUFFER_BACK
                    MOVLW     D’64                     ; number of bytes in holding register
                    MOVWF     COUNTER
WRITE_BYTE_TO_HREGS
                    MOVFF     POSTINC0, WREG           ; get low byte of buffer data
                    MOVWF     TABLAT                   ; 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
```

REGISTER 10-2: INTCON2: INTERRUPT CONTROL REGISTER 2

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
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 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0:** External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **RBIP:** RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global interrupt enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

10.6 INTx Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1 and RB2/INT2 pins are edge-triggered. If the corresponding INTEDGx bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. When a valid edge appears on the RBx/INTx pin, the corresponding flag bit, INTxIF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxIE. Flag bit, INTxIF, must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1 and INT2) can wake-up the processor from Idle or Sleep modes if bit INTxIE was set prior to going into those modes. If the Global Interrupt Enable bit, GIE, is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority for INT1 and INT2 is determined by the value contained in the interrupt priority bits, INT1IP (INTCON3<6>) and INT2IP (INTCON3<7>). There is no priority bit associated with INT0. It is always a high-priority interrupt source.

10.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit, TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP (INTCON2<2>). See **Section 11.0 “Timer0 Module”** for further details on the Timer0 module.

10.8 PORTB Interrupt-on-Change

An input change on PORTB<7:4> sets flag bit, RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE (INTCON<3>). Interrupt priority for PORTB interrupt-on-change is determined by the value contained in the interrupt priority bit, RBIP (INTCON2<0>).

10.9 Context Saving During Interrupts

During interrupts, the return PC address is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the Fast Return Stack. If a fast return from interrupt is not used (see **Section 5.3 “Data Memory Organization”**), the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user’s application, other registers may also need to be saved. Example 10-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

EXAMPLE 10-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

```
MOVWF    W_TEMP                    ; W_TEMP is in virtual bank
MOVFF    STATUS, STATUS_TEMP       ; STATUS_TEMP located anywhere
MOVFF    BSR, BSR_TEMP             ; BSR_TMEP located anywhere
;
; USER ISR CODE
;
MOVFF    BSR_TEMP, BSR             ; Restore BSR
MOVF     W_TEMP, W                 ; Restore WREG
MOVFF    STATUS_TEMP, STATUS       ; Restore STATUS
```

17.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the Start condition (Figure 17-26).

b) SCL is sampled low before SDA is asserted low (Figure 17-27).

During a Start condition, both the SDA and the SCL pins are monitored.

If the SDA pin is already low, or the SCL pin is already low, then all of the following occur:

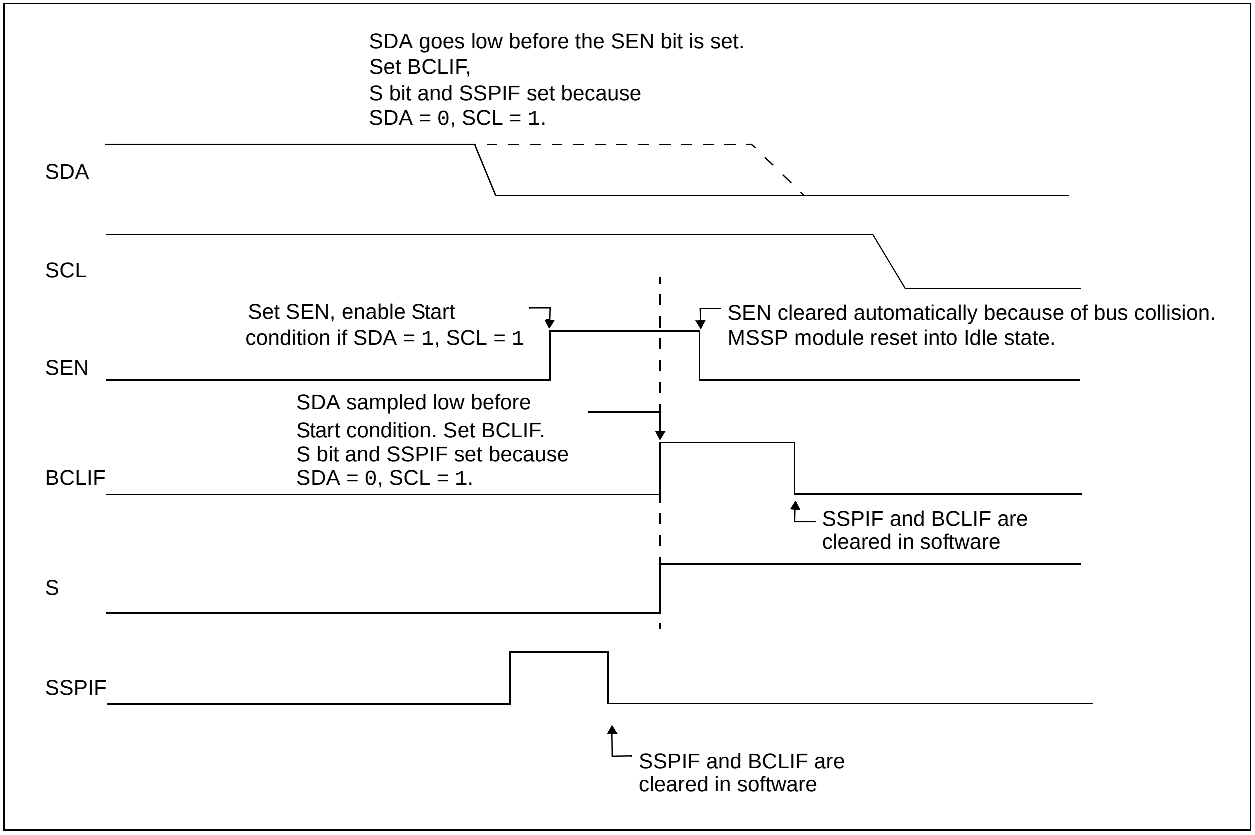
- the Start condition is aborted,
- the BCLIF flag is set and
- the MSSP module is reset to its Idle state (Figure 17-26).

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded from SSPADD<6:0> and counts down to 0. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 17-28). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to 0; if the SCL pin is sampled as '0' during this time, a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

> **Note:** The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

FIGURE 17-26: BUS COLLISION DURING START CONDITION (SDA ONLY)


SDA goes low before the SEN bit is set.
Set BCLIF,
S bit and SSPIF set because
SDA = 0, SCL = 1.
SDA
SCL
Set SEN, enable Start
condition if SDA = 1, SCL = 1
SEN cleared automatically because of bus collision.
MSSP module reset into Idle state.
SEN
SDA sampled low before
Start condition. Set BCLIF.
S bit and SSPIF set because
SDA = 0, SCL = 1.
BCLIF
SSPIF and BCLIF are
cleared in software
S
SSPIF
SSPIF and BCLIF are
cleared in software

REGISTER 18-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN[(1)]	SYNC	SENDB	BRGH	TRMT	TX9D
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 **CSRC:** Clock Source Select bit

Asynchronous mode:
Don’t care.

Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-Bit Transmit Enable bit

1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit[(1)]

1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit

1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit

Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed

Synchronous mode:
Don’t care.

bit 2 **BRGH:** High Baud Rate Select bit

Asynchronous mode:
1 = High speed
0 = Low speed

Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit

1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th Bit of Transmit Data

Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode.

ADDWFC ADD W and Carry bit to f

Syntax: ADDWFC f {,d {,a}}

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

Operation: (W) + (f) + (C) → dest

Status Affected: N,OV, C, DC, Z

Encoding:

`0010`	`00da`	`ffff`	`ffff`

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’.
If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.
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 24.2.3 “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: `ADDWFC REG, 0, 1`

```
Before Instruction
    Carry bit  =  1
    REG        =  02h
    W          =  4Dh
After Instruction
    Carry bit  =  0
    REG        =  02h
    W          =  50h
```

ANDLW AND Literal with W

Syntax: ANDLW k

Operands: $0 \le k \le 255$

Operation: (W) .AND. k → W

Status Affected: N, Z

Encoding:

`0000`	`1011`	`kkkk`	`kkkk`

Description: The contents of W are ANDed 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: `ANDLW 05Fh`

```
Before Instruction
    W  =  A3h
After Instruction
    W  =  03h
```

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> + DC > 9] or [C = 1] then,
(W<7:4>) + 6 + DC → W<7:4> ;
else,
(W<7:4>) + DC → 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:
$0 \leq f \leq 255$
$d \in [0,1]$
$a \in [0,1]$

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.
If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 24.2.3 “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

RETURN Return from Subroutine

Syntax: RETURN {s}

Operands: $s \in [0,1]$

Operation: (TOS) → PC;
if s = 1,
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

`0000`	`0000`	`0001`	`001s`

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's'= `1`, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If 's' = `0`, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	POP PC from stack
No operation	No operation	No operation	No operation

Example: `RETURN`

After Instruction:
PC = TOS

RLCF Rotate Left f through Carry

Syntax: RLCF f {,d {,a}}

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

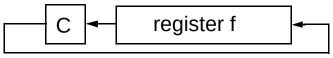
Operation: (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C, N, Z

Encoding:

`0011`	`01da`	`ffff`	`ffff`

Description: The contents of register 'f' are rotated one bit to the left through the Carry flag. 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.
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 24.2.3 "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: `RLCF REG, 0, 0`

Before Instruction
REG = `1110 0110`
C = `0`
After Instruction
REG = `1110 0110`
`W` = `1100 1100`
C = `1`

RLNCF Rotate Left f (No Carry)

Syntax: RLNCF f {,d {,a}}

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

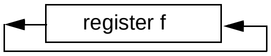
Operation: (f<n>) → dest<n + 1>,
(f<7>) → dest<0>

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.
If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 “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

<u>Example:</u> `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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

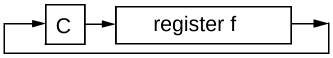
Operation: (f<n>) → dest<n – 1>,
(f<0>) → C,
(C) → dest<7>

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.
If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 “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

<u>Example:</u> `RRCF REG, 0, 0`

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

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

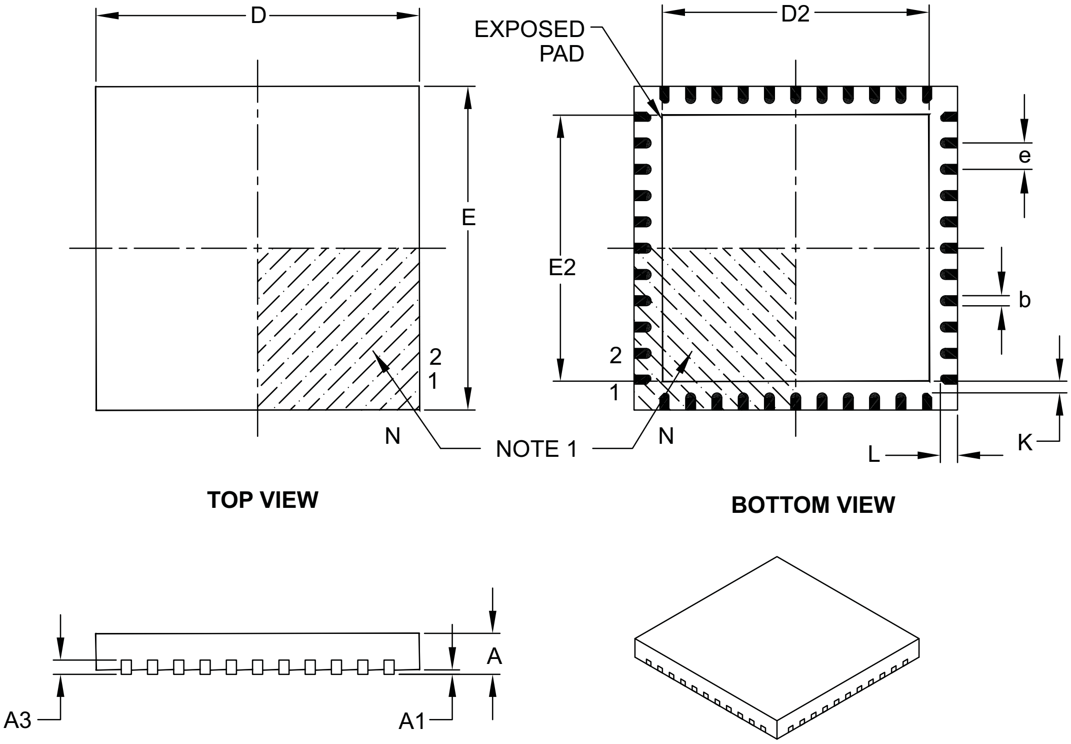
OFST	=	2Ch
FSR2	=	0A00h
Contents of 0A2Ch	=	00h

After Instruction

Contents of 0A2Ch	=	FFh

44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW

BOTTOM VIEW

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	44		
Pitch	e	0.65 BSC		
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Contact Thickness	A3	0.20 REF		
Overall Width	E	8.00 BSC		
Exposed Pad Width	E2	6.30	6.45	6.80
Overall Length	D	8.00 BSC		
Exposed Pad Length	D2	6.30	6.45	6.80
Contact Width	b	0.25	0.30	0.38
Contact Length	L	0.30	0.40	0.50
Contact-to-Exposed Pad	K	0.20	–	–

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-103B

INDEX