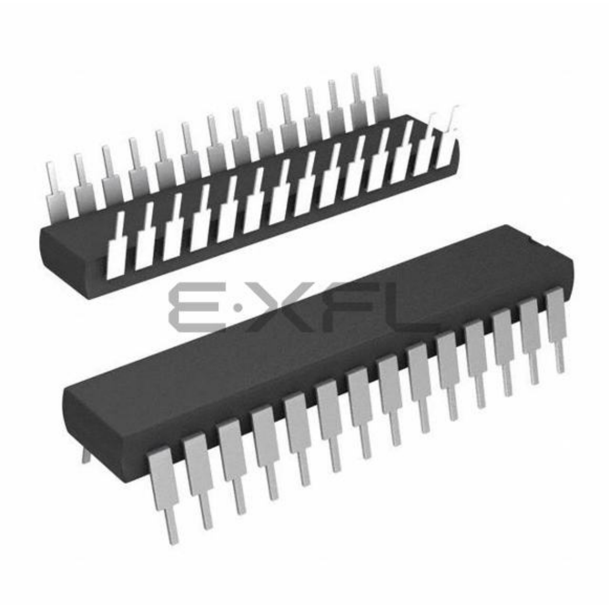


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	25
Program Memory Size	7KB (4K x 14)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 17x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1513-e-sp

TABLE 3-5: PIC16(L)F1512/3 MEMORY MAP (BANKS 8-30)

BANK 8	BANK 9	BANK 10	BANK 11	BANK 12	BANK 13	BANK 14	BANK 15
400h	480h	500h	580h	600h	680h	700h	780h
Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)
40Bh	48Bh	50Bh	58Bh	60Bh	68Bh	70Bh	78Bh
40Ch	48Ch	50Ch	58Ch	60Ch	68Ch	70Ch	78Ch
Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	See Table 3-6	Unimplemented Read as ‘0’
46Fh	4EFh	56Fh	5EFh	66Fh	6EFh	76Fh	7EFh
470h	4F0h	570h	5F0h	670h	6F0h	770h	7F0h
Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)
47Fh	4FFh	57Fh	5FFh	67Fh	6FFh	77Fh	7FFh

BANK 16	BANK 17	BANK 18	BANK 19	BANK 20	BANK 21	BANK 22	BANK 23
800h	880h	900h	980h	A00h	A80h	B00h	B80h
Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)
80Bh	88Bh	90Bh	98Bh	A0Bh	A8Bh	B0Bh	B8Bh
80Ch	88Ch	90Ch	98Ch	A0Ch	A8Ch	B0Ch	B8Ch
Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’
86Fh	8EFh	96Fh	9EFh	A6Fh	AEFh	B6Fh	BEFh
870h	8F0h	970h	9F0h	A70h	AF0h	B70h	BF0h
Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)
87Fh	8FFh	97Fh	9FFh	A7Fh	AFFh	B7Fh	BFFh

BANK 24	BANK 25	BANK 26	BANK 27	BANK 28	BANK 29	BANK 30	BANK 31
C00h	C80h	D00h	D80h	E00h	E80h	F00h	F80h
Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)	Core Registers (Table 3-2)
C0Bh	C8Bh	D0Bh	D8Bh	E0Bh	E8Bh	F0Bh	F8Bh
C0Ch	C8Ch	D0Ch	D8Ch	E0Ch	E8Ch	F0Ch	F8Ch
Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	Unimplemented Read as ‘0’	See (Table 3-7)
C6Fh	CEFh	D6Fh	DEFh	E6Fh	EEFh	F6Fh	FEFh
C70h	CF0h	D70h	DF0h	E70h	EF0h	F70h	FE0h
Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)	Common RAM (Accesses 70h – 7Fh)
C7Fh	CFFh	D7Fh	DFFh	E7Fh	EFFh	F7Fh	FEFh

Legend: = Unimplemented data memory locations, read as ‘0’.

3.4 Stack

All devices have a 16-level x 15-bit wide hardware stack (refer to Figures 3-5 through 3-8). The stack space is not part of either program or data space. The PC is PUSHed onto the stack when `CALL` or `CALLW` instructions are executed or an interrupt causes a branch. The stack is POPed in the event of a `RETURN`, `RETLW` or a `RETFIE` instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer if the STVREN bit is programmed to '0' (Configuration Word 2). This means that after the stack has been PUSHed sixteen times, the seventeenth PUSH overwrites the value that was stored from the first PUSH. The eighteenth PUSH overwrites the second PUSH (and so on). The STKOVF and STKUNF flag bits will be set on an Overflow/Underflow, regardless of whether the Reset is enabled.

Note 1: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the `CALL, CALLW, RETURN, RETLW` and `RETFIE` instructions or the vectoring to an interrupt address.

3.4.1 ACCESSING THE STACK

The stack is available through the TOSH, TOSL and STKPTR registers. STKPTR is the current value of the Stack Pointer. TOSH:TOSL register pair points to the TOP of the stack. Both registers are read/writable. TOS is split into TOSH and TOSL due to the 15-bit size of the PC. To access the stack, adjust the value of STKPTR, which will position TOSH:TOSL, then read/write to TOSH:TOSL. STKPTR is five bits to allow detection of overflow and underflow.

Note: Care should be taken when modifying the STKPTR while interrupts are enabled.

During normal program operation, `CALL, CALLW` and Interrupts will increment STKPTR while `RETLW`, `RETURN`, and `RETFIE` will decrement STKPTR. At any time STKPTR can be inspected to see how much stack is left. The STKPTR always points at the currently used place on the stack. Therefore, a `CALL` or `CALLW` will increment the STKPTR and then write the PC, and a return will unload the PC and then decrement STKPTR.

Reference Figure 3-5 through 3-8 for examples of accessing the stack.

FIGURE 3-5: ACCESSING THE STACK EXAMPLE 1

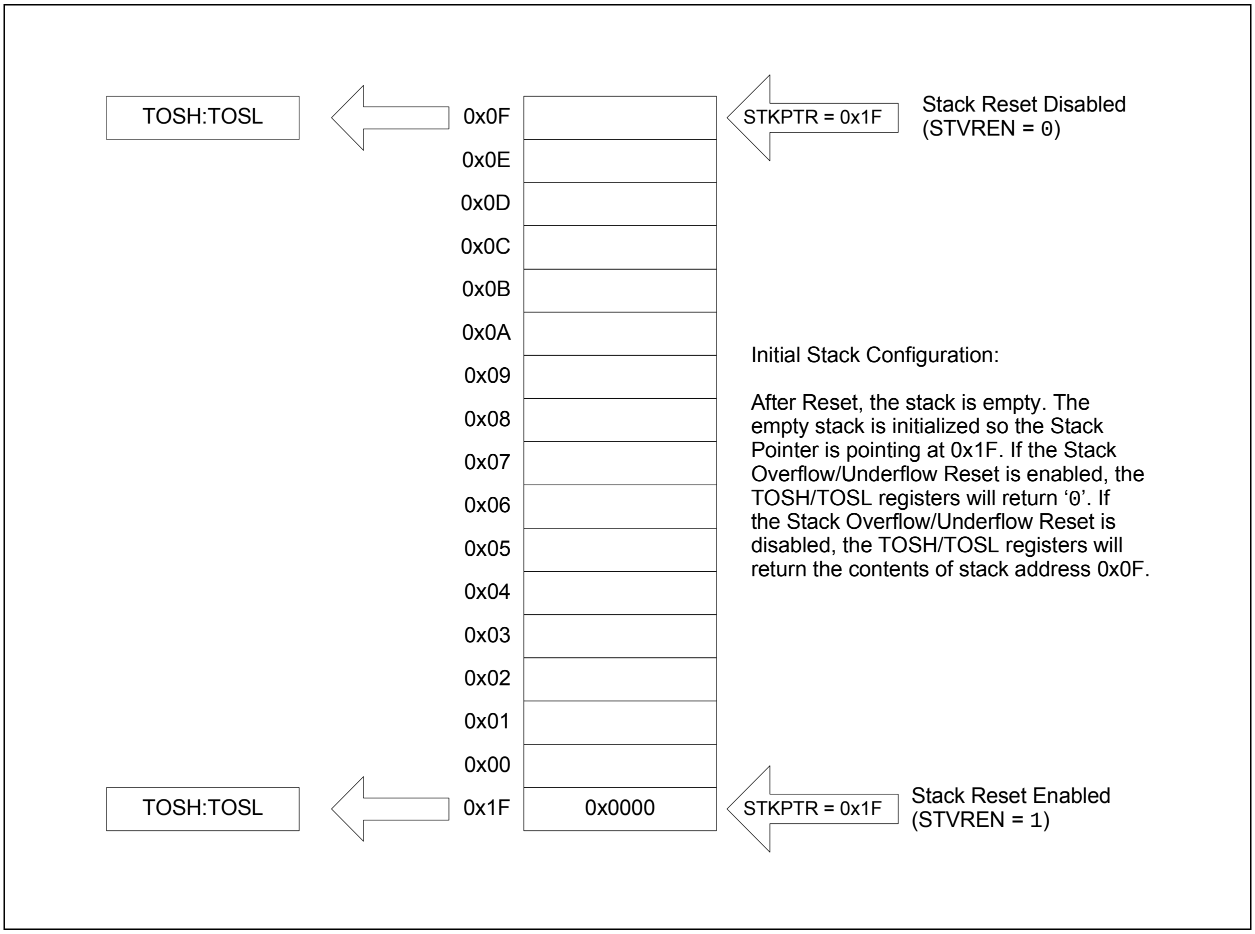
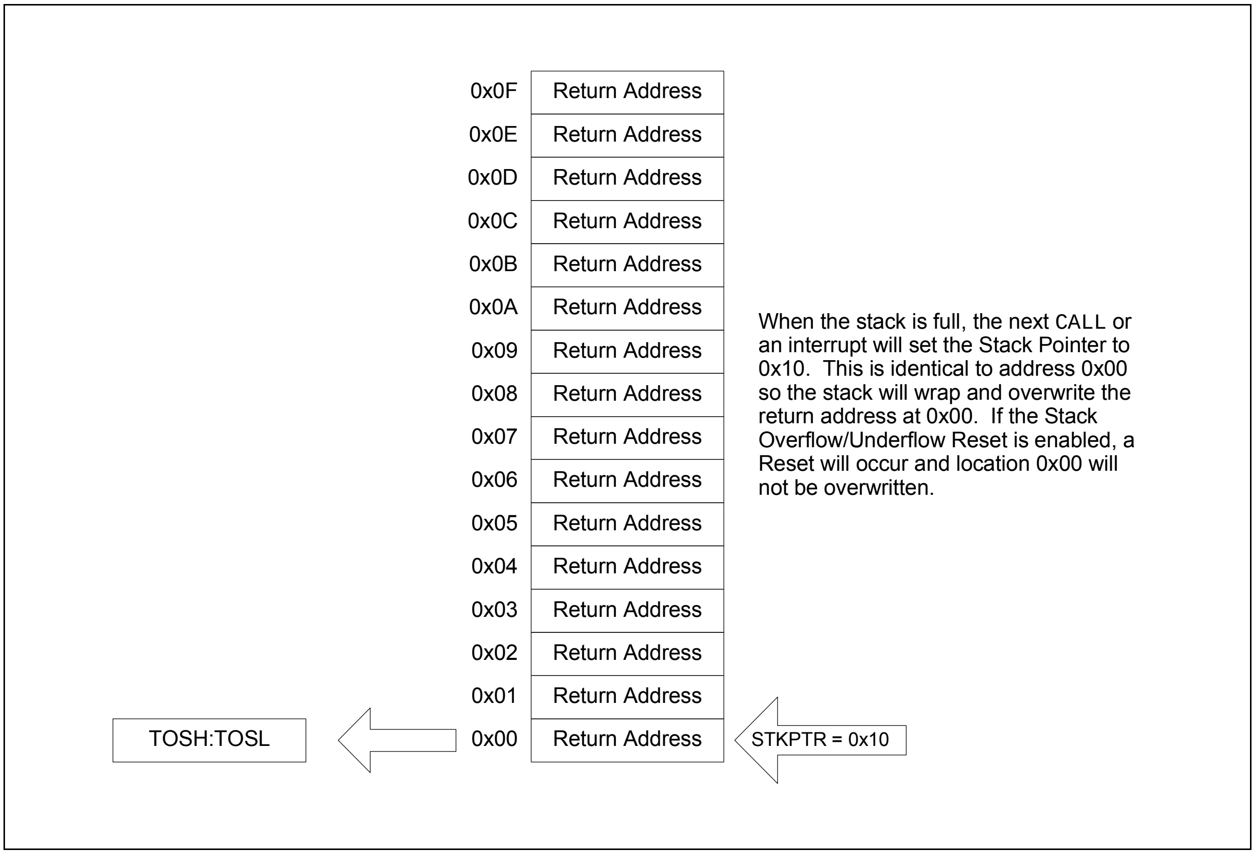


FIGURE 3-8: ACCESSING THE STACK EXAMPLE 4



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.

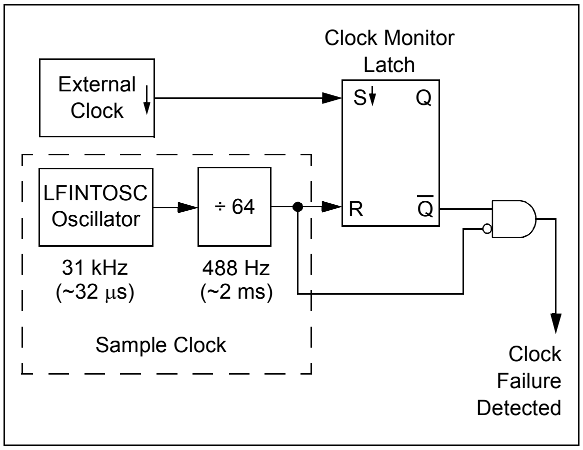
The FSR registers form a 16-bit address that allows an addressing space with 65536 locations. These locations are divided into three memory regions:

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

5.5 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM can detect oscillator failure any time after the Oscillator Start-up Timer (OST) has expired. The FSCM is enabled by setting the FCMEN bit in the Configuration Words. The FSCM is applicable to all external Oscillator modes (LP, XT, HS, EC, RC and secondary oscillator).

FIGURE 5-9: FSCM BLOCK DIAGRAM



5.5.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64. See Figure 5-9. Inside the fail detector block is a latch. The external clock sets the latch on each falling edge of the external clock. The sample clock clears the latch on each rising edge of the sample clock. A failure is detected when an entire half-cycle of the sample clock elapses before the external clock goes low.

5.5.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM switches the device clock to an internal clock source and sets the bit flag OSFIF of the PIR2 register. Setting this flag will generate an interrupt if the OSFIE bit of the PIE2 register is also set. The device firmware can then take steps to mitigate the problems that may arise from a failed clock. The system clock will continue to be sourced from the internal clock source until the device firmware successfully restarts the external oscillator and switches back to external operation.

The internal clock source chosen by the FSCM is determined by the IRCF<3:0> bits of the OSCCON register. This allows the internal oscillator to be configured before a failure occurs.

5.5.3 FAIL-SAFE CONDITION CLEARING

The Fail-Safe condition is cleared after a Reset or changing the SCS bits of the OSCCON register. When the SCS bits are changed, the OST is restarted. While the OST is running, the device continues to operate from the INTOSC selected in OSCCON. When the OST times out, the Fail-Safe condition is cleared and the device will be operating from the external clock source. The Fail-Safe condition must be cleared before the OSFIF flag can be cleared.

5.5.4 RESET OR WAKE-UP FROM SLEEP

The FSCM is designed to detect an oscillator failure after the Oscillator Start-up Timer (OST) has expired. The OST is used after waking up from Sleep and after any type of Reset. The OST is not used with the EC or RC Clock modes so that the FSCM will be active as soon as the Reset or wake-up has completed. When the FSCM is enabled, the Two-Speed Start-up is also enabled. Therefore, the device will always be executing code while the OST is operating.

Note:	Due to the wide range of oscillator start-up times, the Fail-Safe circuit is not active during oscillator start-up (i.e., after exiting Reset or Sleep). After an appropriate amount of time, the user should check the Status bits in the OSCSTAT register to verify the oscillator start-up and that the system clock switchover has successfully completed.

TABLE 10-3: SUMMARY OF REGISTERS ASSOCIATED WITH WATCHDOG TIMER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
OSCCON	—	IRCF<3:0>				—	SCS<1:0>		54
STATUS	—	—	—	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	18
WDTCON	—	—	WDTPS<4:0>					SWDTEN	82

Legend: x = unknown, u = unchanged, – = unimplemented locations read as '0'. Shaded cells are not used by Watchdog Timer.

TABLE 10-4: SUMMARY OF CONFIGURATION WORD WITH WATCHDOG TIMER

Name	Bits	Bit -/7	Bit -/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0	Register on Page
CONFIG1	13:8	—		FCMEN	IESO	$\overline{\text{CLKOUTEN}}$	BOREN<1:0>		—	37
	7:0	$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE<1:0>		FOSC<2:0>			

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by Watchdog Timer.

TABLE 11-1: FLASH MEMORY ORGANIZATION BY DEVICE

Device	Row Erase (words)	Write Latches (words)
PIC16(L)F1512/3	32	32

11.2.1 READING THE FLASH PROGRAM MEMORY

To read a program memory location, the user must:

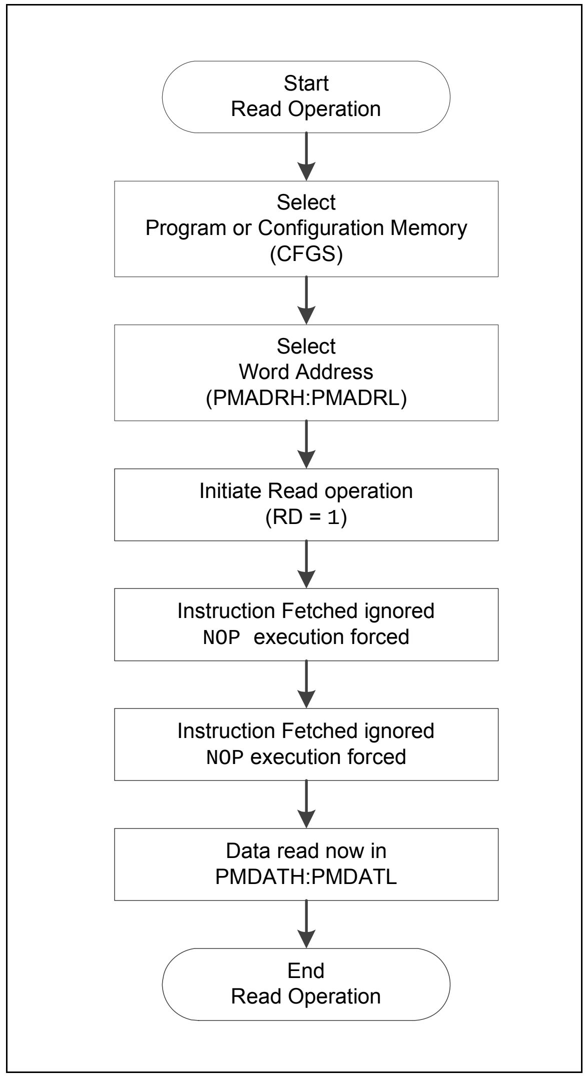
1. Write the desired address to the PMADRH:PMADRL register pair.
2. Clear the CFGS bit of the PMCON1 register.
3. Then, set control bit RD of the PMCON1 register.

Once the read control bit is set, the program memory Flash controller will use the second instruction cycle to read the data. This causes the second instruction immediately following the "`BSF PMCON1,RD`" instruction to be ignored. The data is available in the very next cycle, in the PMDATH:PMDATL register pair; therefore, it can be read as two bytes in the following instructions.

PMDATH:PMDATL register pair will hold this value until another read or until it is written to by the user.

Note: The two instructions following a program memory read are required to be `NOP`s. This prevents the user from executing a two-cycle instruction on the next instruction after the RD bit is set.

FIGURE 11-1: FLASH PROGRAM MEMORY READ FLOWCHART



11.2.2 FLASH MEMORY UNLOCK SEQUENCE

The unlock sequence is a mechanism that protects the Flash program memory from unintended self-write programming or erasing. The sequence must be executed and completed without interruption to successfully complete any of the following operations:

- Row Erase
- Load program memory write latches
- Write of program memory write latches to program memory
- Write of program memory write latches to User IDs

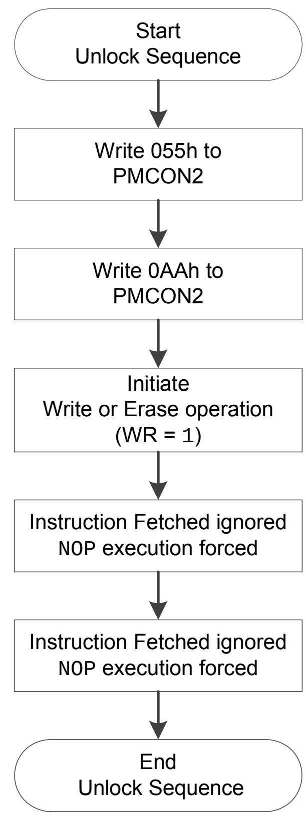
The unlock sequence consists of the following steps:

1. Write 55h to PMCON2
2. Write AAh to PMCON2
3. Set the WR bit in PMCON1
4. NOP instruction
5. NOP instruction

Once the WR bit is set, the processor will always force two NOP instructions. When an Erase Row or Program Row operation is being performed, the processor will stall internal operations (typical 2 ms), until the operation is complete and then resume with the next instruction. When the operation is loading the program memory write latches, the processor will always force the two NOP instructions and continue uninterrupted with the next instruction.

Since the unlock sequence must not be interrupted, global interrupts should be disabled prior to the unlock sequence and re-enabled after the unlock sequence is completed.

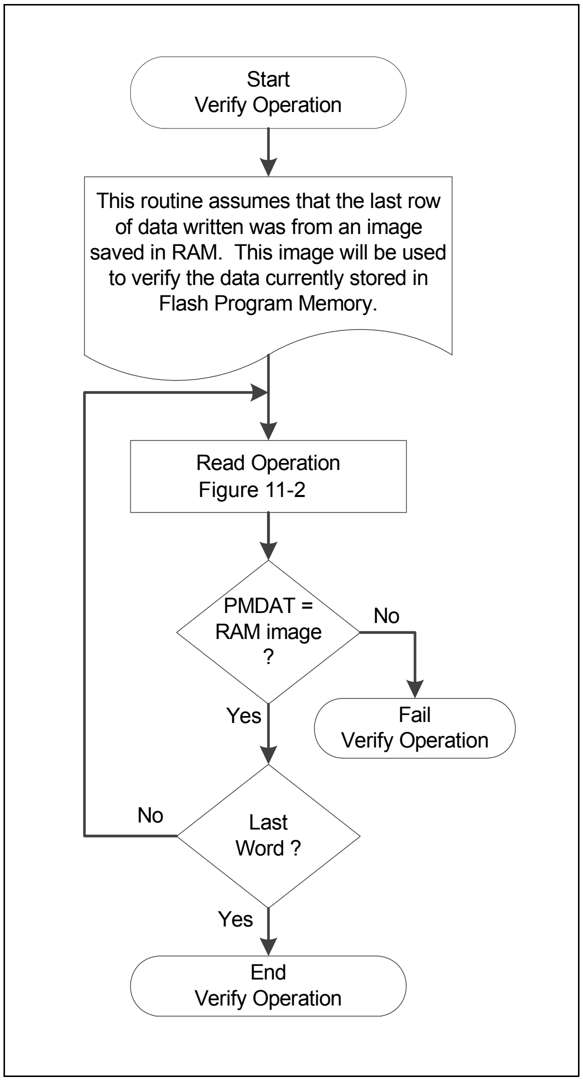
FIGURE 11-3: FLASH PROGRAM MEMORY UNLOCK SEQUENCE FLOWCHART



11.5 Write Verify

It is considered good programming practice to verify that program memory writes agree with the intended value. Since program memory is stored as a full page then the stored program memory contents are compared with the intended data stored in RAM after the last write is complete.

FIGURE 11-8: FLASH PROGRAM MEMORY VERIFY FLOWCHART



REGISTER 12-6: PORTB: PORTB REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
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 7-0 **RB<7:0>**: PORTB General Purpose I/O Pin bits(1)
1 = Port pin is $\geq V_{IH}$
0 = Port pin is $\leq V_{IL}$

Note 1: Writes to PORTB are actually written to corresponding LATB register. Reads from PORTB register is the return of actual I/O pin values.

REGISTER 12-7: TRISB: PORTB TRI-STATE REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
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 7-0 **TRISB<7:0>:** PORTB Tri-State Control bits
1 = PORTB pin configured as an input (tri-stated)
0 = PORTB pin configured as an output

REGISTER 12-8: LATB: PORTB DATA LATCH REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
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 7-0 **LATB<7:0>**: PORTB Output Latch Value bits(1)

Note 1: Writes to PORTB are actually written to corresponding LATB register. Reads from PORTB register is the return of actual I/O pin values.

12.4 PORTC Registers

PORTC is an 8-bit wide bidirectional port. The corresponding data direction register is TRISC (Register 12-12). 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 12-1 shows how to initialize an I/O port.

Reading the PORTC register (Register 12-11) 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 12-12) 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`'.

12.4.1 ANSELC REGISTER

The ANSELC register (Register 12-14) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELC 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 ANSELC bits has no effect on digital output functions. A pin with TRIS clear and ANSELC 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 ANSELC 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.

12.4.2 PORTC FUNCTIONS AND OUTPUT PRIORITIES

Each PORTC pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 12-7.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the highest priority.

Analog input and some digital input functions are not included in the list below. These input functions can remain active when the pin is configured as an output. Certain digital input functions override other port functions and are included in Table 12-7.

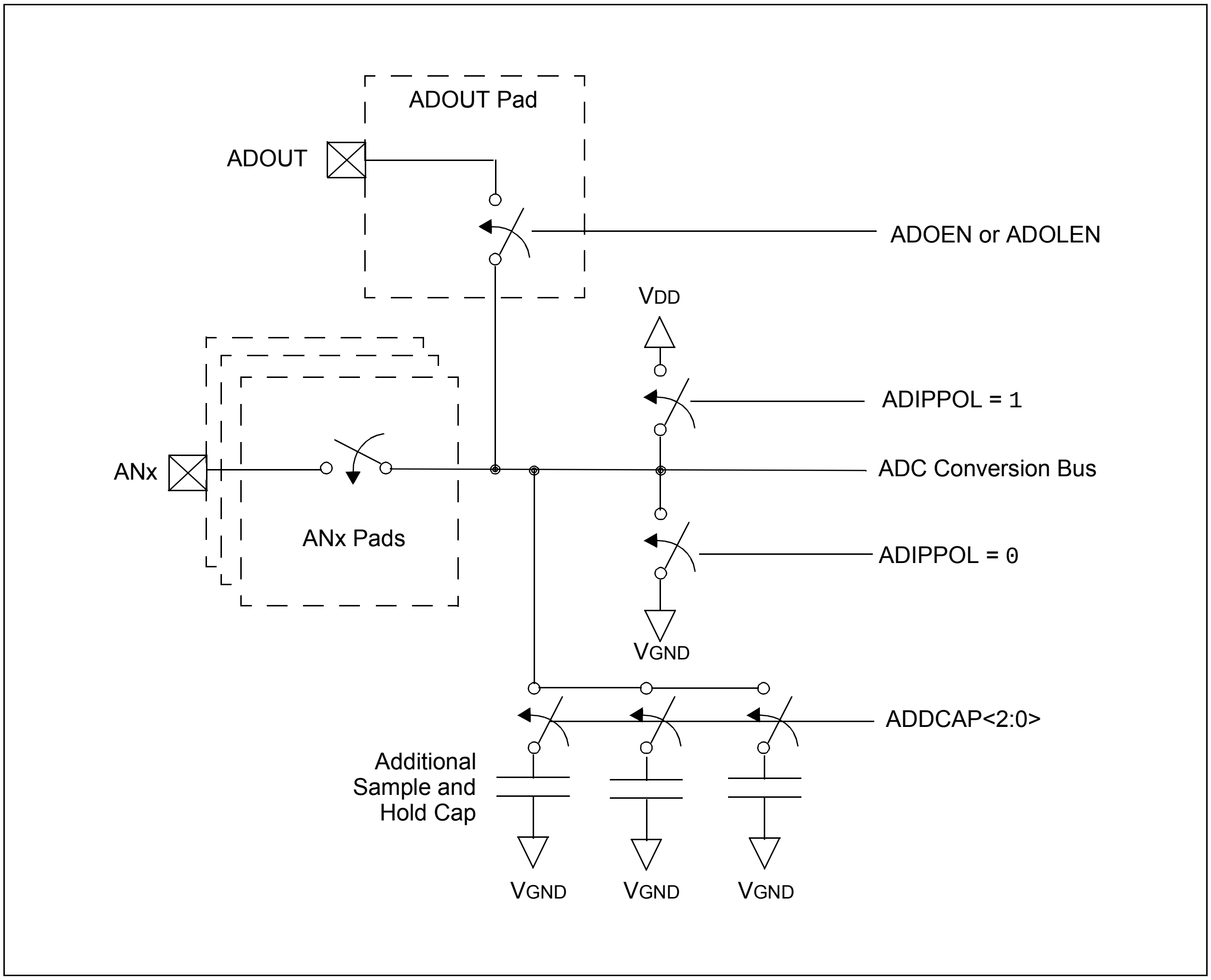
TABLE 12-7: PORTC OUTPUT PRIORITY

Pin Name	Function Priority[1]
RC0	SOSCO RC0
RC1	SOSCI CCP2 RC1
RC2	CCP1 RC2
RC3	SCL SCK RC3[2]
RC4	SDA RC4[2]
RC5	SDO RC5
RC6	CK TX RC6
RC7	DT RC7

Note 1: Priority listed from highest to lowest.

2: RC3 and RC4 read the I^2C ST input when I^2C mode is enabled.

FIGURE 16-7: HARDWARE CAPACITIVE VOLTAGE DIVIDER BLOCK DIAGRAM



REGISTER 16-9: AADCON2: HARDWARE CVD CONTROL REGISTER

U-0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0
—	TRIGSEL<2:0>(1,2)			—	—	—	—
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 **Unimplemented:** Read as '0'

bit 6-4 **TRIGSEL<2:0>:** ADC Special Event Trigger Source Selection bits(1,2)

111 = Reserved. Auto-conversion Trigger disabled.
110 = Reserved. Auto-conversion Trigger disabled.
101 = TMR2 Match to PR2
100 = TMR1 Overflow
011 = TMR0 Overflow
010 = CCP2
001 = CCP1
000 = No Auto Conversion Trigger Selection bits

bit 3-0 **Unimplemented:** Read as '0'

Note 1: This is a rising edge sensitive input for all sources.
2: Signal used to set the corresponding interrupt flag.

REGISTER 16-13: AADACQ: HARDWARE CVD ACQUISITION TIME CONTROL REGISTER

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	R/W-0/0
—	ADACQ<6:0>						
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 **Unimplemented:** Read as '0'

bit 6-0 **ADACQ<6:0>:** Acquisition/Charge Share Time Select bits(1)

`111 1111` = Acquisition/charge share for 127 instruction cycles
`111 1110` = Acquisition/charge share for 126 instruction cycles
•
•
•
`000 0001` = Acquisition/charge share for one instruction cycle (FOSC/4)
`000 0000` = ADC Acquisition/charge share time is disabled

Note 1: When the FRC clock is selected as the conversion clock source, it is also the clock used for the pre-charge and acquisition times.

REGISTER 16-14: AADGRD: HARDWARE CVD GUARD RING CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	U-0
GRDBOE(2)	GRDAOE(2)	GRDPOL(1,2)			—	—	—
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 **GRDBOE:** Guard Ring B Output Enable bit(2)
`1` = ADC guard ring output is enabled to ADGRDB pin. Its corresponding TRISx bit must be clear.
`0` = No ADC guard ring function to this pin is enabled

bit 6 **GRDAOE:** Guard Ring A Output Enable bit(2)
`1` = ADC Guard Ring Output is enabled to ADGRDA pin. Its corresponding TRISx, x bit must be clear.
`0` = No ADC Guard Ring function is enabled

bit 5 **GRDPOL:** Guard Ring Polarity selection bit(1,2)
`1` = ADC guard ring outputs start as digital high during pre-charge stage
`0` = ADC guard ring outputs start as digital low during pre-charge stage

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

Note 1: When the ADDSEN = `1` and ADIPEN = `1`; the polarity of this output is inverted for the second conversion time. The stored bit value does not change.

2: Guard Ring outputs are maintained while ADON = `1`. The ADGRDA output switches polarity at the start of the acquisition time.

20.2.6 SPI OPERATION IN SLEEP MODE

In SPI Master mode, module clocks may be operating at a different speed than when in Full-Power mode; in the case of the Sleep mode, all clocks are halted.

Special care must be taken by the user when the MSSP clock is much faster than the system clock.

In Slave mode, when MSSP interrupts are enabled, after the master completes sending data, an MSSP interrupt will wake the controller from Sleep.

If an exit from Sleep mode is not desired, MSSP interrupts should be disabled.

In SPI Master mode, when the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all eight bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

TABLE 20-1: SUMMARY OF REGISTERS ASSOCIATED WITH SPI OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
ANSELA	—	—	ANSA5	—	ANSA3	ANSA2	ANSA1	ANSA0	104
ANSELC	ANSC7	ANSC6	ANSC5	ANSC4	ANSC3	ANSC2	—	—	111
APFCON	—	—	—	—	—	—	SSSEL	CCP2SEL	101
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	69
PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	70
PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	72
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								179*
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				224
SSPCON3	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN	226
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	224
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	103
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	110

Legend: — = Unimplemented location, read as ‘0’. Shaded cells are not used by the MSSP in SPI mode.
* Page provides register information.

20.4.5 START CONDITION

The I^2C specification defines a Start condition as a transition of SDA from a high to a low state while SCL line is high. A Start condition is always generated by the master and signifies the transition of the bus from an Idle to an Active state. Figure 20-12 shows wave forms for Start and Stop conditions.

A bus collision can occur on a Start condition if the module samples the SDA line low before asserting it low. This does not conform to the I^2C Specification that states no bus collision can occur on a Start.

20.4.6 STOP CONDITION

A Stop condition is a transition of the SDA line from low-to-high state while the SCL line is high.

Note:	At least one SCL low time must appear before a Stop is valid, therefore, if the SDA line goes low then high again while the SCL line stays high, only the Start condition is detected.

20.4.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave.

In 10-bit Addressing Slave mode a Restart is required for the master to clock data out of the addressed slave. Once a slave has been fully addressed, matching both high and low address bytes, the master can issue a Restart and the high address byte with the R/$\overline{W}$ bit set. The slave logic will then hold the clock and prepare to clock out data.

After a full match with R/$\overline{W}$ clear in 10-bit mode, a prior match flag is set and maintained. Until a Stop condition, a high address with R/$\overline{W}$ clear, or high address match fails.

20.4.8 START/STOP CONDITION INTERRUPT MASKING

The SCIE and PCIE bits of the SSPCON3 register can enable the generation of an interrupt in Slave modes that do not typically support this function. Slave modes where interrupt on Start and Stop detect are already enabled, these bits will have no effect.

FIGURE 20-12: I^2C START AND STOP CONDITIONS

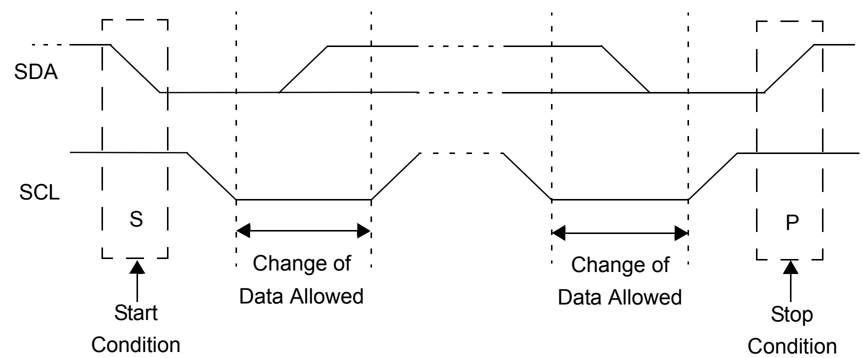


FIGURE 20-13: I^2C RESTART CONDITION

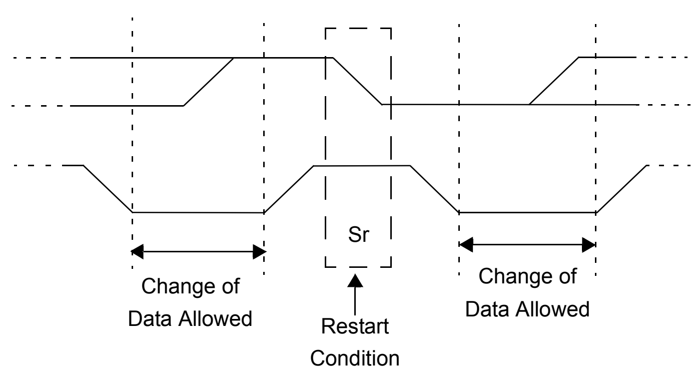
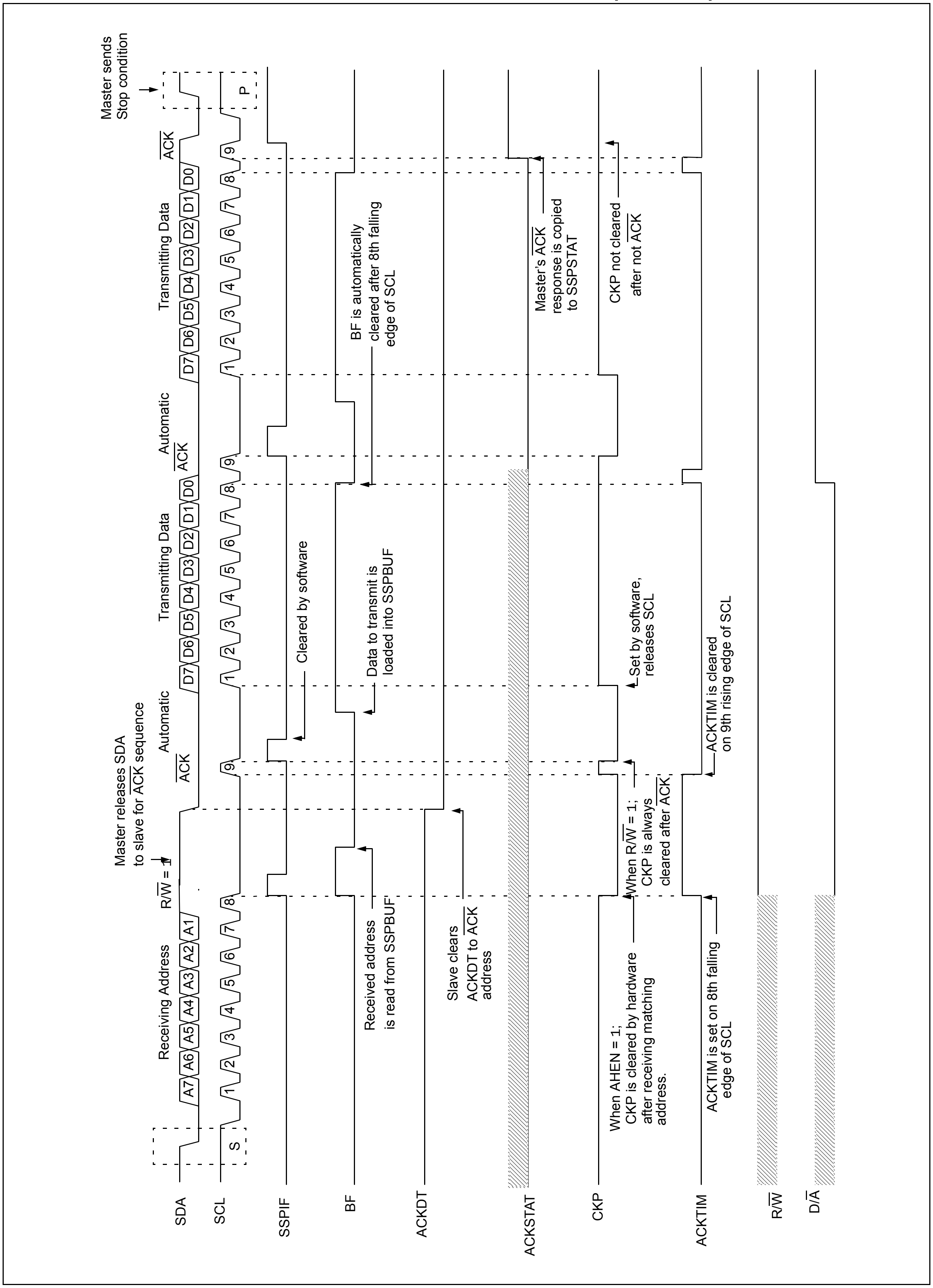


FIGURE 20-19: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)



21.3.6 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is 10 bits when PR2 is 255. The resolution is a function of the PR2 register value as shown by Equation 21-4.

EQUATION 21-4: PWM RESOLUTION

$$Resolution = \frac{\log[4(PR2 + 1)]}{\log(2)} \text{ bits}$$

> **Note:** If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

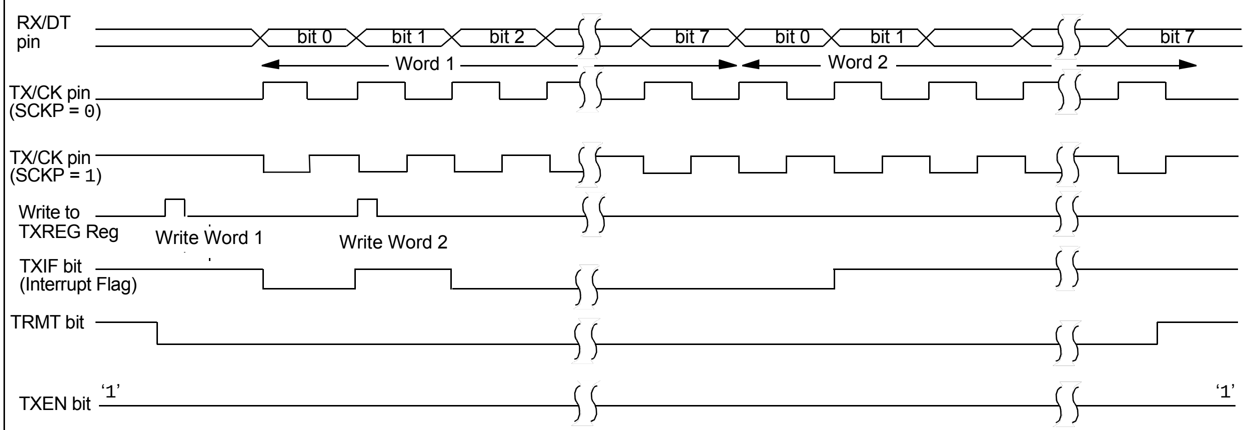
TABLE 21-1: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 20 MHz)

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescale (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	6.6

TABLE 21-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 8 MHz)

PWM Frequency	1.22 kHz	4.90 kHz	19.61 kHz	76.92 kHz	153.85 kHz	200.0 kHz
Timer Prescale (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0x65	0x65	0x65	0x19	0x0C	0x09
Maximum Resolution (bits)	8	8	8	6	5	5

FIGURE 22-10: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode, SPBRGL = 0, continuous transmission of two 8-bit words.

FIGURE 22-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

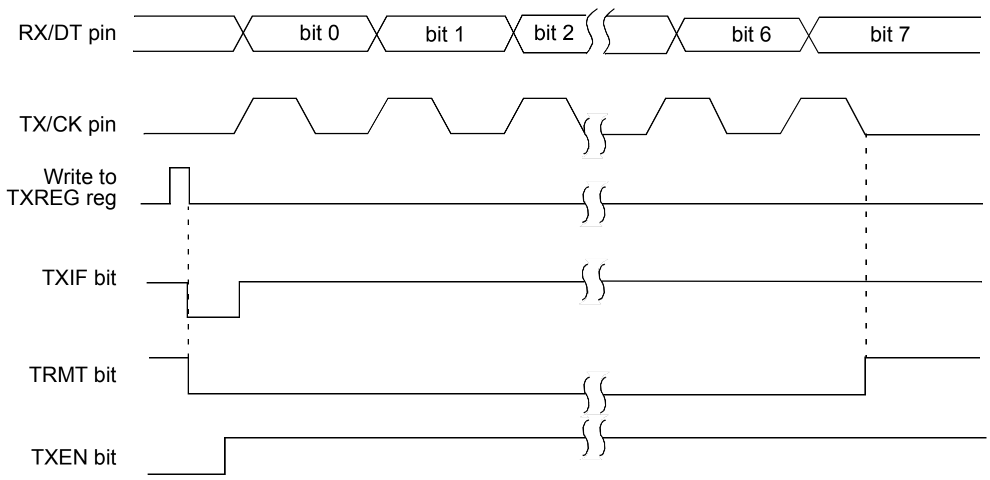


TABLE 22-6: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BAUDCON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	249
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	69
PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	70
PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	72
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	248
SPBRGL	BRG<7:0>								250*
SPBRGH	BRG<15:8>								250*
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	110
TXREG	EUSART Transmit Data Register								239*
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	247

Legend: — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.
* Page provides register information.

TABLE 25-3: POWER-DOWN CURRENTS (IPD)(1,2,4)

Param No.	Device Characteristics	Min.	Typ†	Max. +85°C	Max. +125°C	Units	Conditions VDD	Conditions Note
PIC16LF1512/3		Standard Operating Conditions (unless otherwise stated)						
PIC16F1512/3								
D022		—	0.02	1.0	8.0	μA	1.8	WDT, BOR, FVR, and SOSC disabled, all Peripherals Inactive
		—	0.03	2.0	9.0	μA	3.0	
D022		—	0.20	3.0	11	μA	2.3	WDT, BOR, FVR, and SOSC disabled, all Peripherals Inactive
		—	0.30	4.0	12	μA	3.0	
		—	0.40	6	15	μA	5.0	
D023		—	0.30	6	14	μA	1.8	LPWDT Current
		—	0.60	7	17	μA	3.0	
D023		—	0.50	6	15	μA	2.3	LPWDT Current
		—	0.77	7	20	μA	3.0	
		—	0.85	8	22	μA	5.0	
D023A		—	10	28	30	μA	1.8	FVR current
		—	12	30	33	μA	3.0	
D023A		—	18	33	35	μA	2.3	FVR current
		—	19	36	37	μA	3.0	
		—	20	37	45	μA	5.0	
D024		—	8.0	17	20	μA	3.0	BOR Current
D024		—	8	17	30	μA	3.0	BOR Current
		—	9	20	40	μA	5.0	
D024A		—	0.80	4	8	μA	3.0	LPBOR Current
D024A		—	0.30	4	14	μA	3.0	LPBOR Current
		—	0.45	8	17	μA	5.0	
D025		—	0.6	5	9	μA	1.8	SOSC Current
		—	2.5	8.5	12	μA	3.0	
D025		—	1	6	10	μA	2.3	SOSC Current
		—	2.2	8.5	20	μA	3.0	
		—	5.5	15	25	μA	5.0	
D026		—	0.1	1.5	9	μA	1.8	A/D Current **(Note 3)**, no conversion in progress
		—	0.2	2.7	10	μA	3.0	
D026		—	0.3	4	11	μA	2.3	A/D Current **(Note 3)**, no conversion in progress
		—	0.35	5	13	μA	3.0	
		—	0.45	8	16	μA	5.0	

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.

2: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD.

3: A/D oscillator source is FRC.

4: Specification for PIC16F1512/3 devices assumes that Low-Power Sleep mode is selected, when available, via the VREGCON register (see **Section 8.2.2 "Peripheral Usage in Sleep"** and Register 8-1).

FIGURE 25-18: SPI SLAVE MODE TIMING (CKE = 0)

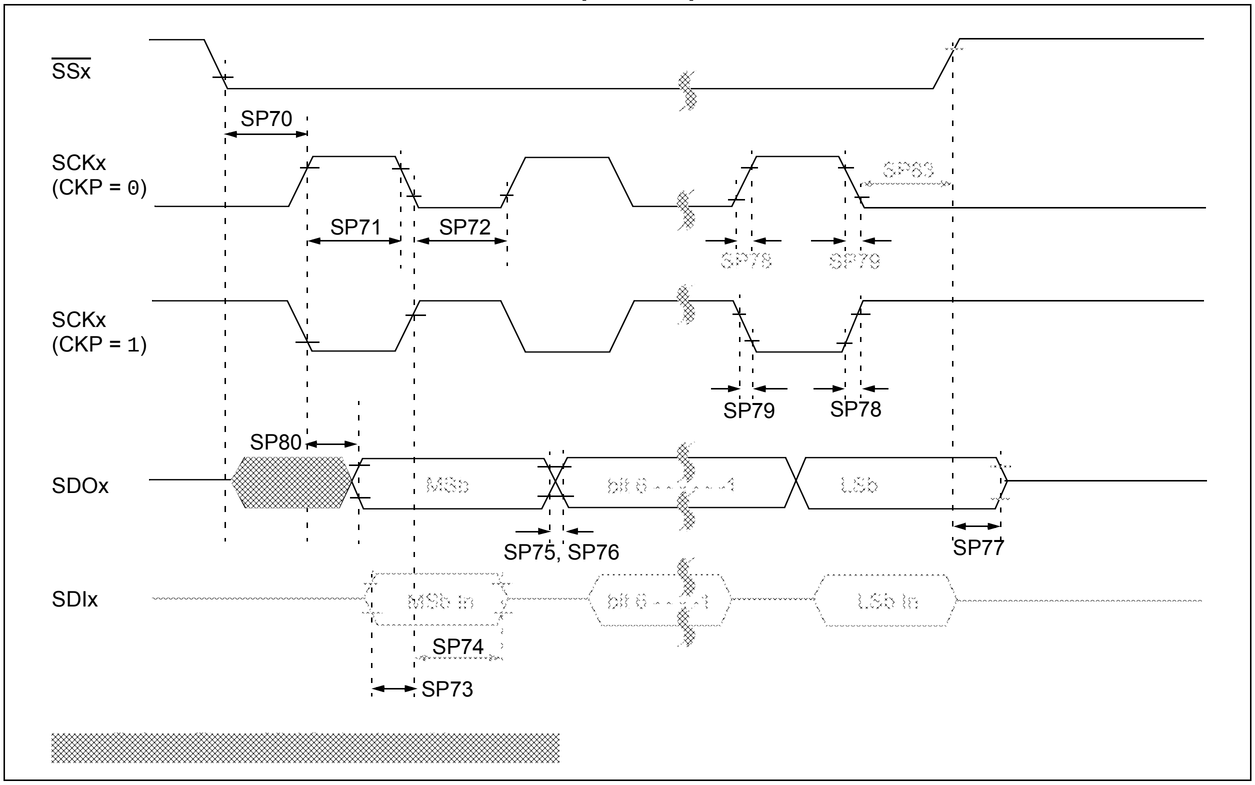


FIGURE 25-19: SPI SLAVE MODE TIMING (CKE = 1)