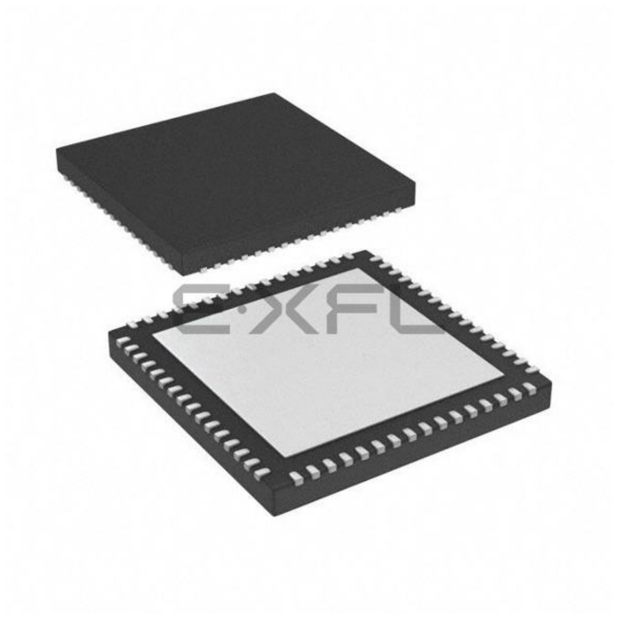
Microchip Technology - PIC16F1526-I/MR Datasheet



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	54
Program Memory Size	14KB (8K x 14)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 30x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-VFQFN Exposed Pad
Supplier Device Package	64-QFN (9x9)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16f1526-i-mr

TABLE 3-2: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

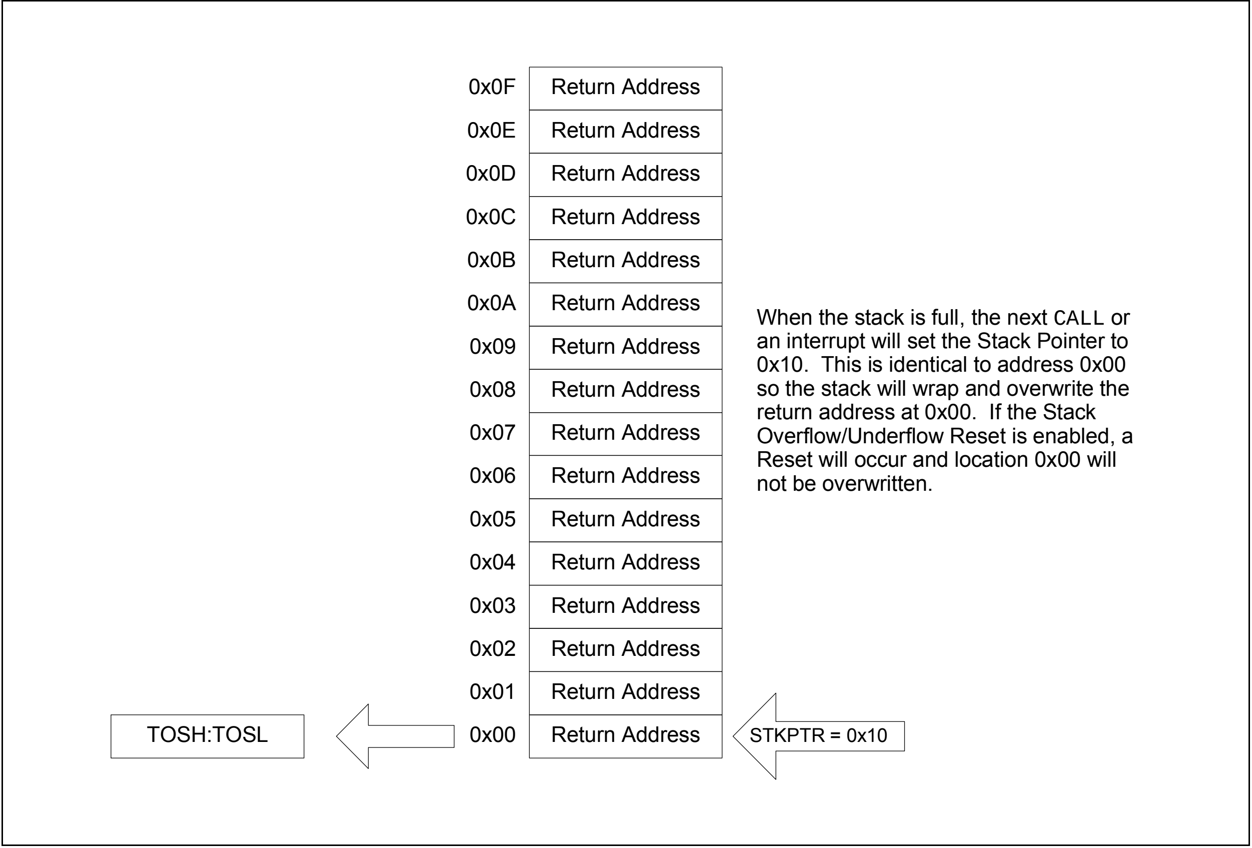
Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 2											
10Ch	LATA	PORTA Data Latch								`xxxx xxxx`	`uuuu uuuu`
10Dh	LATB	PORTB Data Latch								`xxxx xxxx`	`uuuu uuuu`
10Eh	LATC	PORTC Data Latch								`xxxx xxxx`	`uuuu uuuu`
10Fh	LATD	PORTD Data Latch								`xxxx xxxx`	`uuuu uuuu`
110h	LATE	PORTE Data Latch								`xxxx xxxx`	`uuuu uuuu`
111h to 115h	—	Unimplemented								—	—
116h	BORCON	SBOREN	BORFS	—	—	—	—	—	BORRDY	`10-- ---q`	`uu-- ---u`
117h	FVRCON	FVREN	FVRRDY	TSEN	TSRNG	—	—	ADFVR<1:0>		`0q00 0000`	`0q00 0000`
118h to 11Ch	—	Unimplemented								—	—
11Dh	APFCON	—	—	—	—	—	—	T3CKISEL	CCP2SEL	`---- --00`	`---- --00`
11Eh	—	Unimplemented								—	—
11Fh	—	Unimplemented								—	—
Bank 3											
18Ch	ANSELA	—	—	ANSA5	—	ANSA3	ANSA2	ANSA1	ANSA0	`--1- 1111`	`--1- 1111`
18Dh	ANSELB	—	—	ANSB5	ANSB4	ANSB3	ANSB2	ANSB1	ANSB0	`--11 1111`	`--11 1111`
18Eh	ANSELC	Unimplemented								—	—
18Fh	ANSELD	—	—	—	—	ANSD3	ANSD2	ANSD1	ANSD0	`---- 1111`	`---- 1111`
190h	ANSELE	—	—	—	—	—	ANSE2	ANSE1	ANSE0	`---- -111`	`---- -111`
191h	PMADRL	Program Memory Address Register Low Byte								`0000 0000`	`0000 0000`
192h	PMADRH	—(2)	Program Memory Address Register High Byte							`1000 0000`	`1000 0000`
193h	PMDATL	Program Memory Data Register Low Byte								`xxxx xxxx`	`uuuu uuuu`
194h	PMDATH	—	—	Program Memory Data Register High Byte						`--xx xxxx`	`--uu uuuu`
195h	PMCON1	—(2)	CFGS	LWLO	FREE	WRERR	WREN	WR	RD	`1000 x000`	`1000 q000`
196h	PMCON2	Program Memory control register 2								`0000 0000`	`0000 0000`
197h	VREGCON(1)	—	—	—	—	—	—	VREGPM	Reserved	`---- --01`	`---- --01`
198h	—	Unimplemented								—	—
199h	RC1REG	USART Receive Data Register								`0000 0000`	`0000 0000`
19Ah	TX1REG	USART Transmit Data Register								`0000 0000`	`0000 0000`
19Bh	SP1BRG	BRG<7:0>								`0000 0000`	`0000 0000`
19Ch	SP1BRGH	BRG<15:8>								`0000 0000`	`0000 0000`
19Dh	RC1STA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	`0000 000x`	`0000 000x`
19Eh	TX1STA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	`0000 0010`	`0000 0010`
19Fh	BAUD1CON	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	`01-0 0-00`	`01-0 0-00`

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

Note 1: PIC16F1526/7 only.

2: Unimplemented, read as ‘1’.

FIGURE 3-8: ACCESSING THE STACK EXAMPLE 4



3.7.2 OVERFLOW/UNDERFLOW RESET

If the STVREN bit in Configuration Words 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.

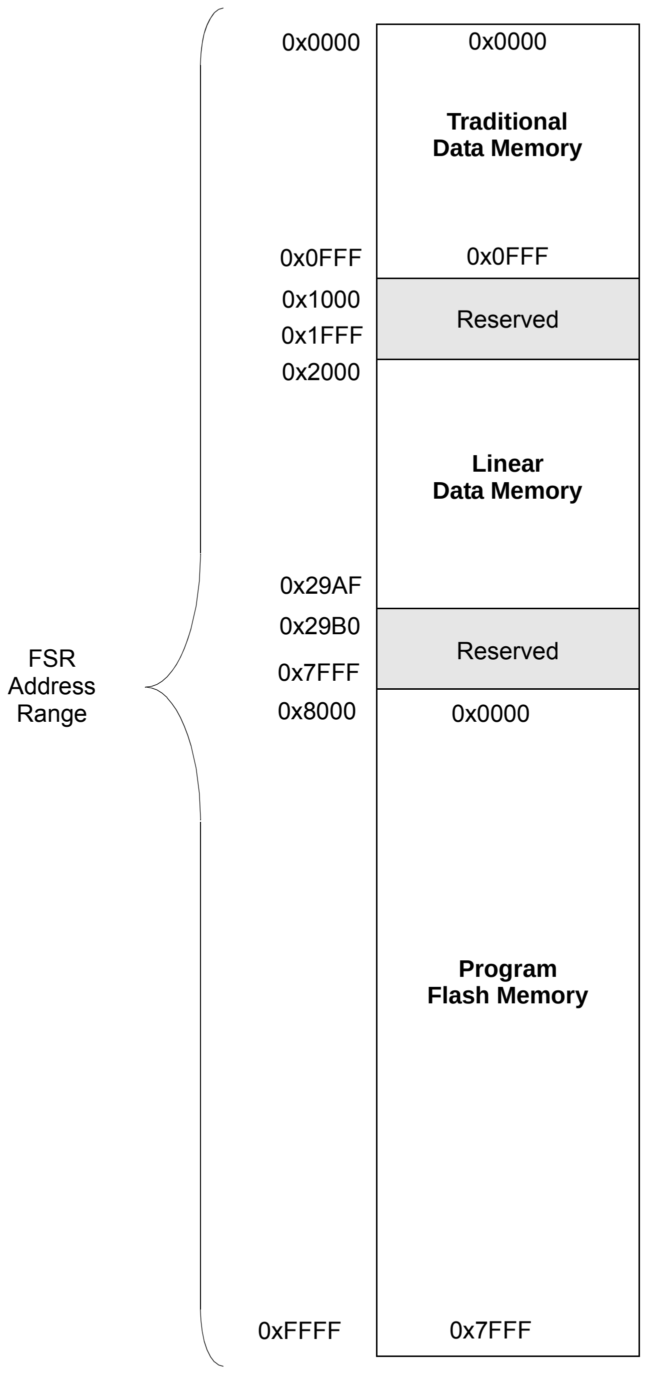
3.8 Indirect Addressing

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

FIGURE 3-9: INDIRECT ADDRESSING



Note: Not all memory regions are completely implemented. Consult device memory tables for memory limits.

REGISTER 4-2: CONFIG2: CONFIGURATION WORD 2

R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-1
LVP	$\overline{\text{DEBUG}}$	$\overline{\text{LPBOR}}$	BORV	STVREN	—
bit 13					bit 8

U-1	U-1	U-1	R/P-1	U-1	U-1	R/P-1	R/P-1
—	—	—	$\overline{\text{VCAPEN}}$(1)	—	—	WRT<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as ‘1’
‘0’ = Bit is cleared	‘1’ = Bit is set	-n = Value when blank or after Bulk Erase

bit 13 **LVP:** Low-Voltage Programming Enable bit
`1` = Low-voltage programming enabled
`0` = High-voltage on $\overline{\text{MCLR}}$ must be used for programming

bit 12 **$\overline{\text{DEBUG}}$:** In-Circuit Debugger Mode bit
`1` = In-Circuit Debugger disabled, ICSPCLK and ICSPDAT are general purpose I/O pins
`0` = In-Circuit Debugger enabled, ICSPCLK and ICSPDAT are dedicated to the debugger

bit 11 **$\overline{\text{LPBOR}}$:** Low-Power BOR bit
`1` = Low-Power BOR is disabled
`0` = Low-Power BOR is enabled

bit 10 **BORV:** Brown-out Reset Voltage Selection bit(2)
`1` = Brown-out Reset voltage (Vbor), low trip point selected.
`0` = Brown-out Reset voltage (Vbor), high trip point selected.

bit 9 **STVREN:** Stack Overflow/Underflow Reset Enable bit
`1` = Stack Overflow or Underflow will cause a Reset
`0` = Stack Overflow or Underflow will not cause a Reset

bit 8-5 **Unimplemented:** Read as ‘`1`’

bit 4 **$\overline{\text{VCAPEN}}$:** Voltage Regulator Capacitor Enable bits(1)
If PIC16LF1526/7 (regulator disabled):
These bits are ignored. All VCAP pin functions are disabled.
If PIC16F1526/7 (regulator enabled):
`0` = VCAP functionality is enabled on RF0.
`1` = All VCAP pin functions are disabled

bit 3-2 **Unimplemented:** Read as ‘`1`’

bit 1-0 **WRT<1:0>:** Flash Memory Self-Write Protection bits
8 kW Flash memory (PIC16(L)F1526 only):
`11` = Write protection off
`10` = 000h to 1FFh write-protected, 200h to 1FFFh may be modified by PMCON control
`01` = 000h to FFFh write-protected, 1000h to 1FFFh may be modified by PMCON control
`00` = 000h to 1FFFh write-protected, no addresses may be modified by PMCON control
16 kW Flash memory (PIC16(L)F1527 only):
`11` = Write protection off
`10` = 000h to 1FFh write-protected, 200h to 3FFFh may be modified by PMCON control
`01` = 000h to 1FFFh write-protected, 2000h to 3FFFh may be modified by PMCON control
`00` = 000h to 3FFFh write-protected, no addresses may be modified by PMCON control

Note 1: PIC16F1526/7 only.
2: See Vbor parameter for specific trip point voltages.

12.5 PORTB Registers

12.5.1 DATA REGISTER

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB (Register 12-7). Setting a TRISB bit (= `1`) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISB bit (= `0`) will make the corresponding PORTB 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 PORTB register (Register 12-6) 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 (LATB).

12.5.2 DIRECTION CONTROL

The TRISB register (Register 12-7) controls the PORTB pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISB register are maintained set when using them as analog inputs. I/O pins configured as analog input always read ‘`0`’.

12.5.3 ANALOG CONTROL

The ANSELB register (Register 12-9) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELB 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 ANSELB bits has no effect on digital output functions. A pin with TRIS clear and ANSELB 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 ANSELB 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.

12.5.4 PORTB FUNCTIONS AND OUTPUT PRIORITIES

Each PORTB pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 12-5.

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 12-5.

TABLE 12-5: PORTB OUTPUT PRIORITY

Pin Name	Function Priority[(1)]
RB0	RB0
RB1	RB1
RB2	RB2
RB3	CCP2 RB3
RB4	RB4
RB5	RB5
RB6	ICDCLK RB6
RB7	ICDDAT RB7

Note 1: Priority listed from highest to lowest.

12.15 PORTG Registers

12.15.1 DATA REGISTER

PORTG is a 6-bit wide, bidirectional port. The corresponding data direction register is TRISG (Register 12-29). Setting a TRISG bit (= `1`) will make the corresponding PORTG pin an input (i.e., disable the output driver). Clearing a TRISG bit (= `0`) will make the corresponding PORTG pin an output (i.e., enables output driver and puts the contents of the output latch on the selected pin). The exception is RG5, which is input only and its TRIS bit will always read as '`1`'. Example 12-1 shows how to initialize an I/O port.

Reading the PORTG register (Register 12-28) 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 (LATG).

12.15.2 DIRECTION CONTROL

The TRISG register (Register 12-29) controls the PORTG pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISG register are maintained set when using them as analog inputs. I/O pins configured as analog input always read '`0`'.

12.15.3 ANALOG CONTROL

The ANSELG register (Register 12-31) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELG 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 ANSELG bits has no effect on digital output functions. A pin with TRIS clear and ANSEL set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

Note: The ANSELG 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.

12.15.4 PORTG FUNCTIONS AND OUTPUT PRIORITIES

Each PORTG pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 12-16.

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

Analog input functions, such as ADC, are not shown in the priority lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority list.

TABLE 12-16: PORTG OUTPUT PRIORITY

Pin Name	Function Priority[1]
RG0	CCP3 RG0
RG1	CK2 TX2 RG1
RG2	DT2 RG2
RG3	CCP4 RG3
RG4	CCP5 RG4
RG5	Input only pin

Note 1: Priority listed from highest to lowest.

12.16 Register Definitions: PORTG

REGISTER 12-28: PORTG: PORTG REGISTER

U-0	U-0	R-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
—	—	RG5	RG4	RG3	RG2	RG1	RG0
bit 7							bit 0

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

bit 7-6 **Unimplemented**: Read as ‘0’

bit 5-0 **RG<5:0>**: PORTG I/O Pin bits(1)
1 = Port pin is > V_{IH}
0 = Port pin is < V_{IL}

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

REGISTER 12-29: TRISG: PORTG TRI-STATE REGISTER

U-0	U-0	U-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—(1)	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0
bit 7							bit 0

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

bit 7-6 **Unimplemented**: Read as ‘0’

bit 5 **Unimplemented**: Read as ‘1’

bit 4-0 **TRISG<4:0>:** RG<4:0> Tri-State Control bits(1)
1 = PORTG pin configured as an input (tri-stated)
0 = PORTG pin configured as an output

Note 1: Unimplemented, read as ‘1’.

17.1.3 SOFTWARE PROGRAMMABLE PRESCALER

A software programmable prescaler is available for exclusive use with Timer0. The prescaler is enabled by clearing the PSA bit of the OPTION_REG register.

Note:	The Watchdog Timer (WDT) uses its own independent prescaler.

There are 8 prescaler options for the Timer0 module ranging from 1:2 to 1:256. The prescale values are selectable via the PS<2:0> bits of the OPTION_REG register. In order to have a 1:1 prescaler value for the Timer0 module, the prescaler must be disabled by setting the PSA bit of the OPTION_REG register.

The prescaler is not readable or writable. All instructions writing to the TMR0 register will clear the prescaler.

17.1.4 TIMER0 INTERRUPT

Timer0 will generate an interrupt when the TMR0 register overflows from FFh to 00h. The TMR0IF interrupt flag bit of the INTCON register is set every time the TMR0 register overflows, regardless of whether or not the Timer0 interrupt is enabled. The TMR0IF bit can only be cleared in software. The Timer0 interrupt enable is the TMR0IE bit of the INTCON register.

Note:	The Timer0 interrupt cannot wake the processor from Sleep since the timer is frozen during Sleep.

17.1.5 8-BIT COUNTER MODE SYNCHRONIZATION

When in 8-bit Counter mode, the incrementing edge on the T0CKI pin must be synchronized to the instruction clock. Synchronization can be accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the instruction clock. The high and low periods of the external clocking source must meet the timing requirements as shown in **Section 25.0 “Electrical Specifications”**.

17.1.6 OPERATION DURING SLEEP

Timer0 cannot operate while the processor is in Sleep mode. The contents of the TMR0 register will remain unchanged while the processor is in Sleep mode.

TABLE 20-5: SUMMARY OF REGISTERS ASSOCIATED WITH COMPARE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
APFCON	—	—	—	—	—	—	T3CKISEL	CCP2SEL	112
CCP1CON	—	—	DC1B<1:0>		CCP1M<3:0>				189
CCP2CON	—	—	DC2B<1:0>		CCP2M<3:0>				189
CCP3CON	—	—	DC3B<1:0>		CCP3M<3:0>				189
CCP4CON	—	—	DC4B<1:0>		CCP4M<3:0>				189
CCP5CON	—	—	DC5B<1:0>		CCP5M<3:0>				189
CCP6CON	—	—	DC6B<1:0>		CCP6M<3:0>				189
CCP7CON	—	—	DC7B<1:0>		CCP7M<3:0>				189
CCP8CON	—	—	DC8B<1:0>		CCP8M<3:0>				189
CCP9CON	—	—	DC9B<1:0>		CCP9M<3:0>				189
CCP10CON	—	—	DC10B<1:0>		CCP10M<3:0>				189
CCPR1L	Capture/Compare/PWM Register 1 Low Byte (LSB)								176*
CCPR2L	Capture/Compare/PWM Register 2 Low Byte (LSB)								176*
CCPR3L	Capture/Compare/PWM Register 3 Low Byte (LSB)								176*
CCPR4L	Capture/Compare/PWM Register 4 Low Byte (LSB)								176*
CCPR5L	Capture/Compare/PWM Register 5 Low Byte (LSB)								176*
CCPR6L	Capture/Compare/PWM Register 6 Low Byte (LSB)								176*
CCPR7L	Capture/Compare/PWM Register 7 Low Byte (LSB)								176*
CCPR8L	Capture/Compare/PWM Register 8 Low Byte (LSB)								176*
CCPR9L	Capture/Compare/PWM Register 9 Low Byte (LSB)								176*
CCPR10L	Capture/Compare/PWM Register 10 Low Byte (LSB)								176*
CCPR1H	Capture/Compare/PWM Register 1 High Byte (MSB)								176*
CCPR2H	Capture/Compare/PWM Register 2 High Byte (MSB)								176*
CCPR3H	Capture/Compare/PWM Register 3 High Byte (MSB)								176*
CCPR4H	Capture/Compare/PWM Register 4 High Byte (MSB)								176*
CCPR5H	Capture/Compare/PWM Register 5 High Byte (MSB)								176*
CCPR6H	Capture/Compare/PWM Register 6 High Byte (MSB)								176*
CCPR7H	Capture/Compare/PWM Register 7 High Byte (MSB)								176*
CCPR8H	Capture/Compare/PWM Register 8 High Byte (MSB)								176*
CCPR9H	Capture/Compare/PWM Register 9 High Byte (MSB)								176*
CCPR10H	Capture/Compare/PWM Register 10 High Byte (MSB)								176*
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	76
PIE1	TMR1GIE	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	77
PIE2	OSFIE	TMR5GIE	TMR3GIE	—	BCL1IE	TMR10IE	TMR8IE	CCP2IE	78
PIE3	CCP6IE	CCP5IE	CCP4IE	CCP3IE	TMR6IE	TMR5IE	TMR4IE	TMR3IE	79
PIE4	CCP10IE	CCP9IE	RC2IE	TX2IE	CCP8IE	CCP7IE	BCL2IE	SSP2IE	80
PIR1	TMR1GIF	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	81
PIR2	OSFIF	TMR5GIF	TMR3GIF	—	BCL1IF	TMR10IF	TMR8IF	CCP2IF	82
PIR3	CCP6IF	CCP5IF	CCP4IF	CCP3IF	TMR6IF	TMR5IF	TMR4IF	TMR3IF	83
PIR4	CCP10IF	CCP9IF	RC2IF	TX2IF	CCP8IF	CCP7IF	BCL2IF	SSP2IF	84
T1CON	TMR1CS<1:0>		T1CKPS<1:0>		SOSCEN	$\overline{\text{T1SYNC}}$	—	TMR1ON	168
T3CON	TMR3CS<1:0>		T3CKPS<1:0>		SOSCEN	$\overline{\text{T3SYNC}}$	—	TMR3ON	168
T5CON	TMR5CS<1:0>		T5CKPS<1:0>		SOSCEN	$\overline{\text{T5SYNC}}$	—	TMR5ON	168
T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/DONE	T1GVAL	T1GSS<1:0>		169
T3GCON	TMR3GE	T3GPOL	T3GTM	T3GSPM	T3GGO/DONE	T3GVAL	T3GSS<1:0>		169

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by Compare mode.
* Page provides register information.

REGISTER 20-2: CCPTMRS0: CCP TIMER SELECTION CONTROL REGISTER 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	R/W-0/0
C4TSEL<1:0>		C3TSEL<1:0>		C2TSEL<1:0>		C1TSEL<1:0>	
bit 7							bit 0

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

bit 7-6 **C4TSEL<1:0>:** CCP4 Timer Selection bits

When in Capture/Compare mode:

`x1` = CCP4 is based off Timer3 in Capture/Compare mode
`x0` = CCP4 is based off Timer1 in Capture/Compare mode

When in PWM mode:

`11` = Reserved
`10` = CCP4 is based off Timer6 in PWM mode
`01` = CCP4 is based off Timer4 in PWM mode
`00` = CCP4 is based off Timer2 in PWM mode

bit 5-4 **C3TSEL<1:0>:** CCP3 Timer Selection bits

When in Capture/Compare mode:

`x1` = CCP3 is based off Timer3 in Capture/Compare mode
`x0` = CCP3 is based off Timer1 in Capture/Compare mode

When in PWM mode:

`11` = Reserved
`10` = CCP3 is based off Timer6 in PWM mode
`01` = CCP3 is based off Timer4 in PWM mode
`00` = CCP3 is based off Timer2 in PWM mode

bit 3-2 **C2TSEL<1:0>:** CCP2 Timer Selection bits

When in Capture/Compare mode:

`x1` = CCP2 is based off Timer3 in Capture/Compare mode
`x0` = CCP2 is based off Timer1 in Capture/Compare mode

When in PWM mode:

`11` = Reserved
`10` = CCP2 is based off Timer6 in PWM mode
`01` = CCP2 is based off Timer4 in PWM mode
`00` = CCP2 is based off Timer2 in PWM mode

bit 1-0 **C1TSEL<1:0>:** CCP1 Timer Selection bits

When in Capture/Compare mode:

`x1` = CCP1 is based off Timer3 in Capture/Compare mode
`x0` = CCP1 is based off Timer1 in Capture/Compare mode

When in PWM mode:

`11` = Reserved
`10` = CCP1 is based off Timer6 in PWM mode
`01` = CCP1 is based off Timer4 in PWM mode
`00` = CCP1 is based off Timer2 in PWM mode

The I^2C interface supports the following modes and features:

- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited Multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection
- General call address matching
- Address masking
- Address Hold and Data Hold modes
- Selectable SDAx hold times

Figure 21-2 is a block diagram of the I^2C Interface module in Master mode. Figure 21-3 is a diagram of the I^2C interface module in Slave mode.

The PIC16F1527 has two MSSP modules, MSSP1 and MSSP2, each module operating independently from the other.

Note 1: In devices with more than one MSSP module, it is very important to pay close attention to SSPxCONx register names. SSP1CON1 and SSP1CON2 registers control different operational aspects of the same module, while SSP1CON1 and SSP2CON1 control the same features for two different modules.

2: Throughout this section, generic references to an MSSP module in any of its operating modes may be interpreted as being equally applicable to MSSP1 or MSSP2. Register names, module I/O signals, and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module when required.

FIGURE 21-2: MSSPX BLOCK DIAGRAM (I^2C MASTER MODE)

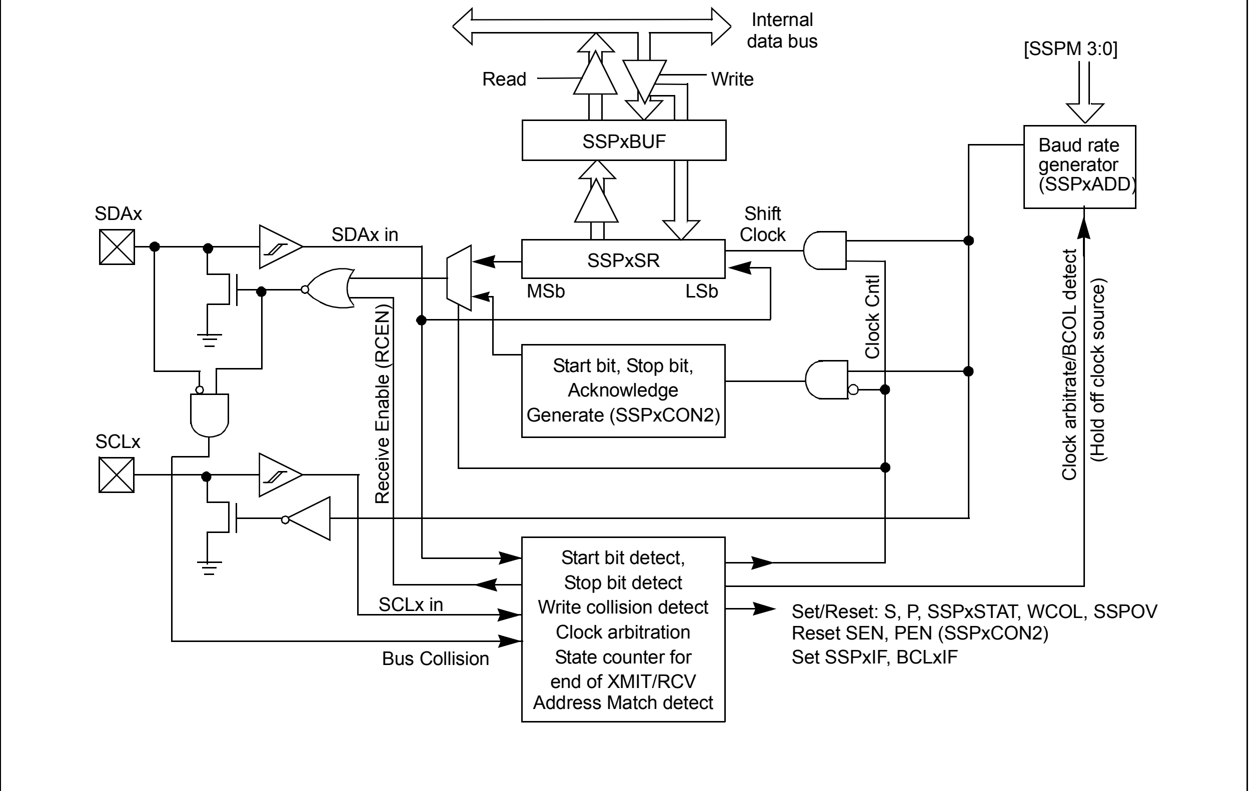

Internal data bus
Read
Write
[SSPM 3:0]
SSPxBUF
Baud rate generator (SSPxADD)
SDAx
SDAx in
Shift Clock
SSPxSR
MSb
LSb
Receive Enable (RCEN)
Clock Cntl
Clock arbitrate/BCOL detect (Hold off clock source)
Start bit, Stop bit, Acknowledge Generate (SSPxCON2)
SCLx
Start bit detect, Stop bit detect Write collision detect Clock arbitration State counter for end of XMIT/RCV Address Match detect
SCLx in
Bus Collision
Set/Reset: S, P, SSPxSTAT, WCOL, SSPOV
Reset SEN, PEN (SSPxCON2)
Set SSPxIF, BCLxIF

FIGURE 21-16: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)

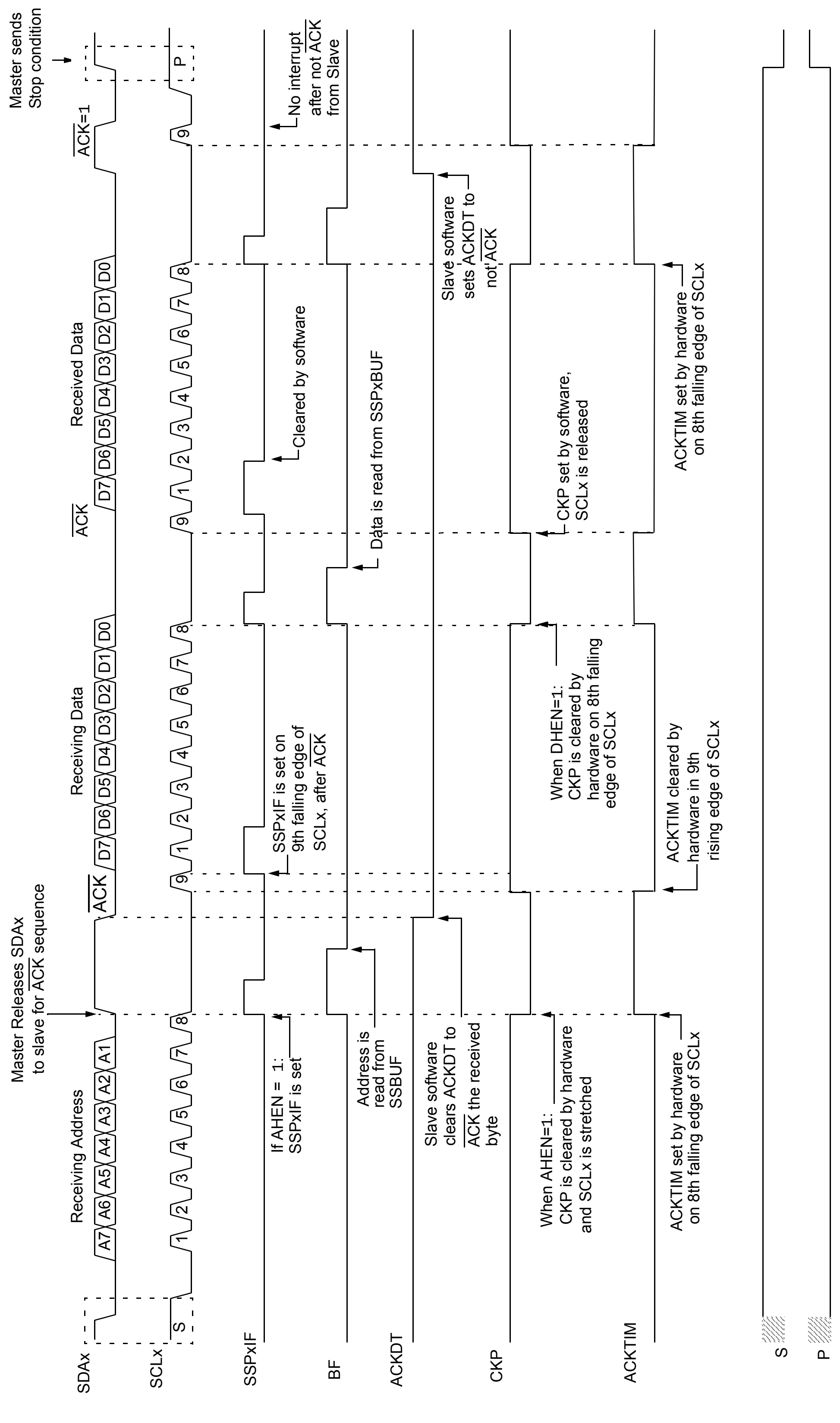
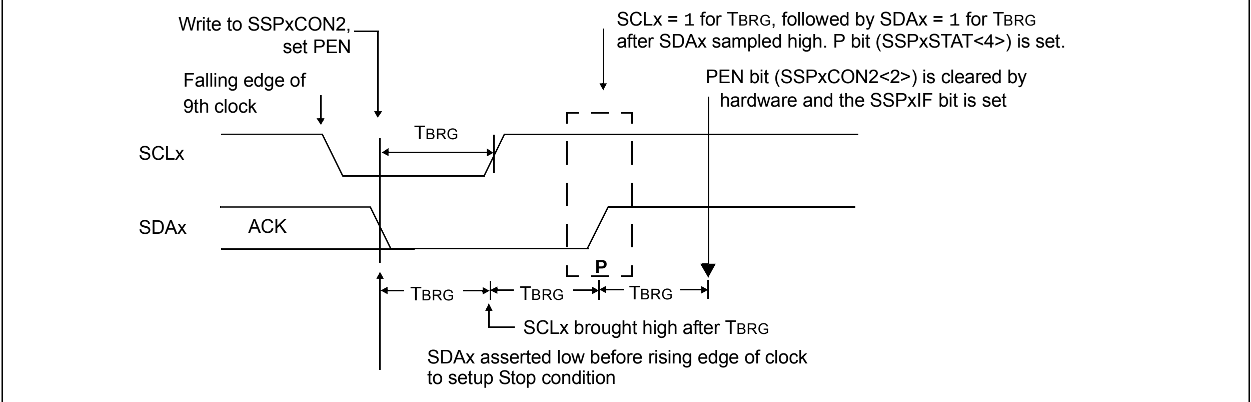

SDAx
SCLx
SSPxIF
BF
ACKDT
CKP
ACKTIM
S
P
Receiving Address
A7 A6 A5 A4 A3 A2 A1
Master Releases SDAx to slave for ACK sequence
ACK
Receiving Data
D7 D6 D5 D4 D3 D2 D1 D0
ACK
Received Data
D7 D6 D5 D4 D3 D2 D1 D0
ACK=1
Master sends Stop condition
S
1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9
P
If AHEN = 1: SSPxIF is set
SSPxIF is set on 9th falling edge of SCLx, after ACK
Cleared by software
No interrupt after not ACK from Slave
Address is read from SSBUF
Data is read from SSPxBUF
Slave software clears ACKDT to ACK the received byte
Slave software sets ACKDT to not ACK
When AHEN=1: CKP is cleared by hardware and SCLx is stretched
When DHEN=1: CKP is cleared by hardware on 8th falling edge of SCLx
CKP set by software, SCLx is released
ACKTIM set by hardware on 8th falling edge of SCLx
ACKTIM cleared by hardware in 9th rising edge of SCLx
ACKTIM set by hardware on 8th falling edge of SCLx

FIGURE 21-31: STOP CONDITION RECEIVE OR TRANSMIT MODE


Write to SSPxCON2, set PEN
Falling edge of 9th clock
SCLx = 1 for TBRG, followed by SDAx = 1 for TBRG after SDAx sampled high. P bit (SSPxSTAT<4>) is set.
PEN bit (SSPxCON2<2>) is cleared by hardware and the SSPxIF bit is set
SCLx
SDAx
ACK
TBRG
P
TBRG
TBRG
TBRG
SCLx brought high after TBRG
SDAx asserted low before rising edge of clock to setup Stop condition

Note: TBRG = one Baud Rate Generator period.

21.6.10 SLEEP OPERATION

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

21.6.11 EFFECTS OF A RESET

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

21.6.12 MULTI-MASTER MODE

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

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

The states where arbitration can be lost are:

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

21.6.13 MULTI -MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDAx pin, arbitration takes place when the master outputs a '1' on SDAx, by letting SDAx float high and another master asserts a '0'. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a '1' and the data sampled on the SDAx pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF and reset the I^{2}C port to its Idle state (Figure 21-32).

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

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

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

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

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

FIGURE 24-1: GENERAL FORMAT FOR INSTRUCTIONS

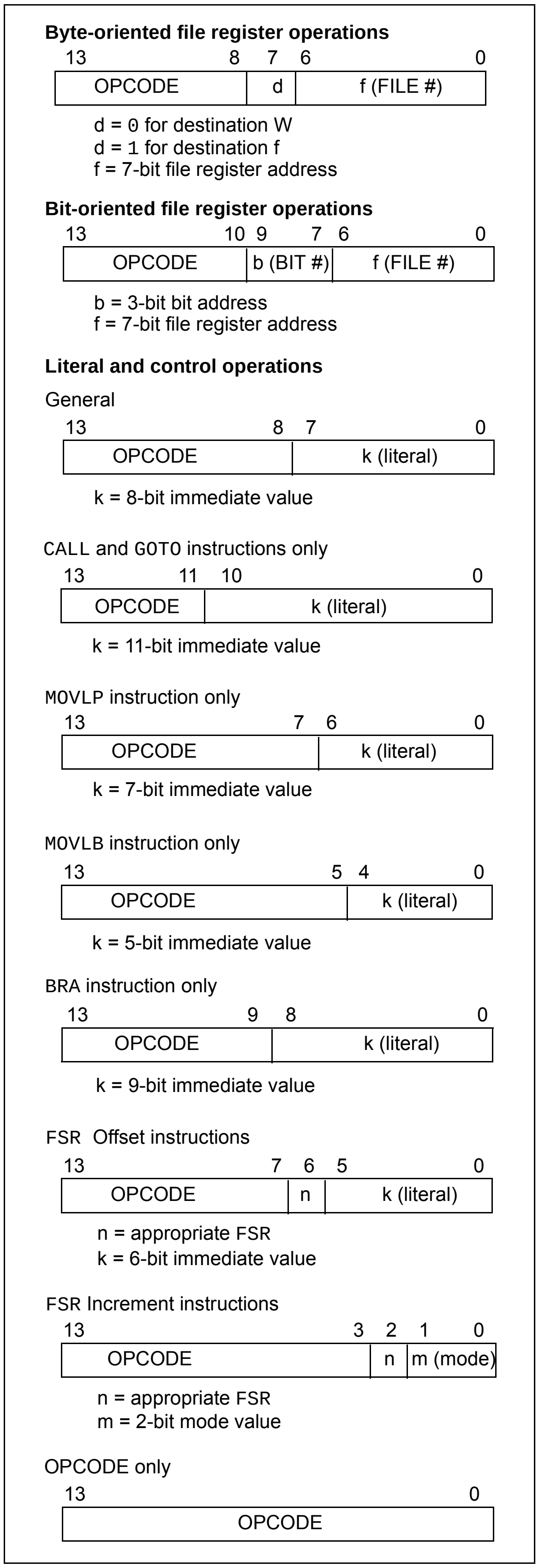


TABLE 24-3: INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
CONTROL OPERATIONS									
BRA	k	Relative Branch	2	`11`	`001k`	`kkkk`	`kkkk`		
BRW	–	Relative Branch with W	2	`00`	`0000`	`0000`	`1011`		
CALL	k	Call Subroutine	2	`10`	`0kkk`	`kkkk`	`kkkk`		
CALLW	–	Call Subroutine with W	2	`00`	`0000`	`0000`	`1010`		
GOTO	k	Go to address	2	`10`	`1kkk`	`kkkk`	`kkkk`		
RETFIE	k	Return from interrupt	2	`00`	`0000`	`0000`	`1001`		
RETLW	k	Return with literal in W	2	`11`	`0100`	`kkkk`	`kkkk`		
RETURN	–	Return from Subroutine	2	`00`	`0000`	`0000`	`1000`		
INHERENT OPERATIONS									
CLRWDT	–	Clear Watchdog Timer	1	`00`	`0000`	`0110`	`0100`	$\overline{TO}$, $\overline{PD}$	
NOP	–	No Operation	1	`00`	`0000`	`0000`	`0000`		
OPTION	–	Load OPTION_REG register with W	1	`00`	`0000`	`0110`	`0010`		
RESET	–	Software device Reset	1	`00`	`0000`	`0000`	`0001`		
SLEEP	–	Go into Standby mode	1	`00`	`0000`	`0110`	`0011`	$\overline{TO}$, $\overline{PD}$	
TRIS	f	Load TRIS register with W	1	`00`	`0000`	`0110`	`0fff`		
C-COMPILER OPTIMIZED									
ADDFSR	n, k	Add Literal k to FSRn	1	`11`	`0001`	`0nkk`	`kkkk`		
MOVIW	n mm	Move Indirect FSRn to W with pre/post inc/dec modifier, mm	1	`00`	`0000`	`0001`	`0nmm`	Z	**2, 3**
	k[n]	Move INDFn to W, Indexed Indirect.	1	`11`	`1111`	`0nkk`	`kkkk`	Z	**2**
MOVWI	n mm	Move W to Indirect FSRn with pre/post inc/dec modifier, mm	1	`00`	`0000`	`0001`	`1nmm`		**2, 3**
	k[n]	Move W to INDFn, Indexed Indirect.	1	`11`	`1111`	`1nkk`	`kkkk`		**2**

Note 1: If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

2: If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

3: See Table in the MOVIW and MOVWI instruction descriptions.

DECFSZ	Decrement f, Skip if 0
Syntax:	[*label*] DECFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Description:	The contents of register ‘f’ are decremented. If ‘d’ is ‘0’, the result is placed in the W register. If ‘d’ is ‘1’, the result is placed back in register ‘f’. If the result is ‘1’, the next instruction is executed. If the result is ‘0’, then a NOP is executed instead, making it a 2-cycle instruction.

GOTO	Unconditional Branch
Syntax:	[*label*] GOTO k
Operands:	$0 \le k \le 2047$
Operation:	k → PC<10:0> PCLATH<6:3> → PC<14:11>
Status Affected:	None
Description:	GOTO is an unconditional branch. The eleven-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a 2-cycle instruction.

INCF	Increment f
Syntax:	[*label*] INCF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 → (destination)
Status Affected:	Z
Description:	The contents of register ‘f’ are incremented. If ‘d’ is ‘0’, the result is placed in the W register. If ‘d’ is ‘1’, the result is placed back in register ‘f’.

INCFSZ	Increment f, Skip if 0
Syntax:	[*label*] INCFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 → (destination), skip if result = 0
Status Affected:	None
Description:	The contents of register ‘f’ are incremented. If ‘d’ is ‘0’, the result is placed in the W register. If ‘d’ is ‘1’, the result is placed back in register ‘f’. If the result is ‘1’, the next instruction is executed. If the result is ‘0’, a NOP is executed instead, making it a 2-cycle instruction.

IORLW	Inclusive OR literal with W
Syntax:	[*label*] IORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are OR’ed with the 8-bit literal ‘k’. The result is placed in the W register.

IORWF	Inclusive OR W with f
Syntax:	[*label*] IORWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .OR. (f) → (destination)
Status Affected:	Z
Description:	Inclusive OR the W register with register ‘f’. If ‘d’ is ‘0’, the result is placed in the W register. If ‘d’ is ‘1’, the result is placed back in register ‘f’.

RETFIE	**Return from Interrupt**
Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None
Description:	Return from Interrupt. Stack is POPed and Top-of-Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a 2-cycle instruction.
Words:	1
Cycles:	2
<u>Example:</u>	`RETFIE`

```
After Interrupt
        PC  =   TOS
        GIE =   1
```

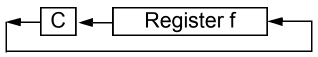
RETURN	**Return from Subroutine**
Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a 2-cycle instruction.

RETLW	**Return with literal in W**
Syntax:	[*label*] RETLW k
Operands:	$0 \leq k \leq 255$
Operation:	k → (W); TOS → PC
Status Affected:	None
Description:	The W register is loaded with the 8-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a 2-cycle instruction.
Words:	1
Cycles:	2
<u>Example:</u>	

```
         CALL TABLE;W contains table
             ;offset value
         •   ;W now has table value
TABLE    •
         •
         ADDWF PC  ;W = offset
         RETLW k1  ;Begin table
         RETLW k2  ;
         •
         •
         •
         RETLW kn  ; End of table
```

```
Before Instruction
        W   =   0x07
After Instruction
        W   =   value of k8
```

RLF	**Rotate Left f through Carry**
Syntax:	[*label*] RLF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is stored back in register 'f'.



Words:	1
Cycles:	1
<u>Example:</u>	`RLF     REG1,0`

```
Before Instruction
        REG1   =   1110 0110
        C      =   0
After Instruction
        REG1   =   1110 0110
        W      =   1100 1100
        C      =   1
```

SWAPF	Swap Nibbles in f
Syntax:	[*label*] SWAPF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(f<3:0>) → (destination<7:4>), (f<7:4>) → (destination<3:0>)
Status Affected:	None
Description:	The upper and lower nibbles of register ‘f’ are exchanged. If ‘d’ is ‘0’, the result is placed in the W register. If ‘d’ is ‘1’, the result is placed in register ‘f’.

TRIS	Load TRIS Register with W
Syntax:	[*label*] TRIS f
Operands:	$5 \leq f \leq 7$
Operation:	(W) → TRIS register ‘f’
Status Affected:	None
Description:	Move data from W register to TRIS register. When ‘f’ = 5, TRISA is loaded. When ‘f’ = 6, TRISB is loaded. When ‘f’ = 7, TRISC is loaded.

XORLW	Exclusive OR literal with W
Syntax:	[*label*] XORLW k
Operands:	$0 \leq k \leq 255$
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are XOR’ed with the 8-bit literal ‘k’. The result is placed in the W register.

XORWF	Exclusive OR W with f
Syntax:	[*label*] XORWF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(W) .XOR. (f) → (destination)
Status Affected:	Z
Description:	Exclusive OR the contents of the W register with register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

FIGURE 26-23: IDD TYPICAL, HFINTOSC, PIC16LF1526 ONLY

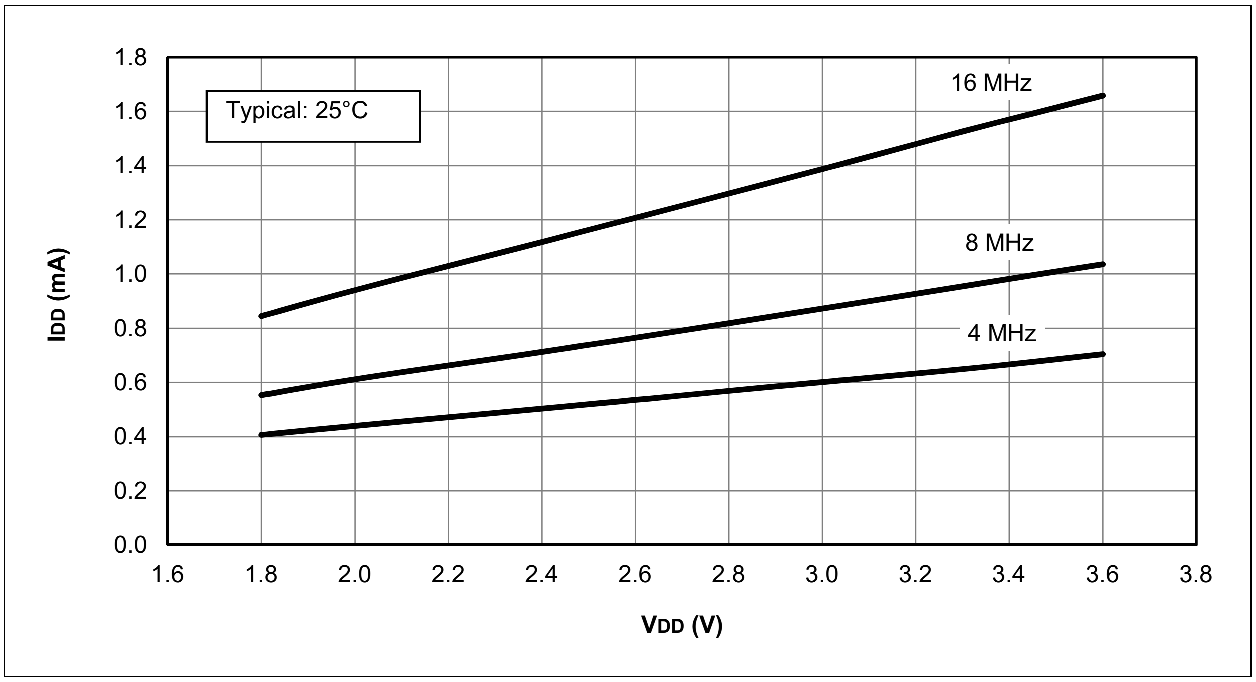


FIGURE 26-24: IDD MAXIMUM, HFINTOSC, PIC16LF1526 ONLY

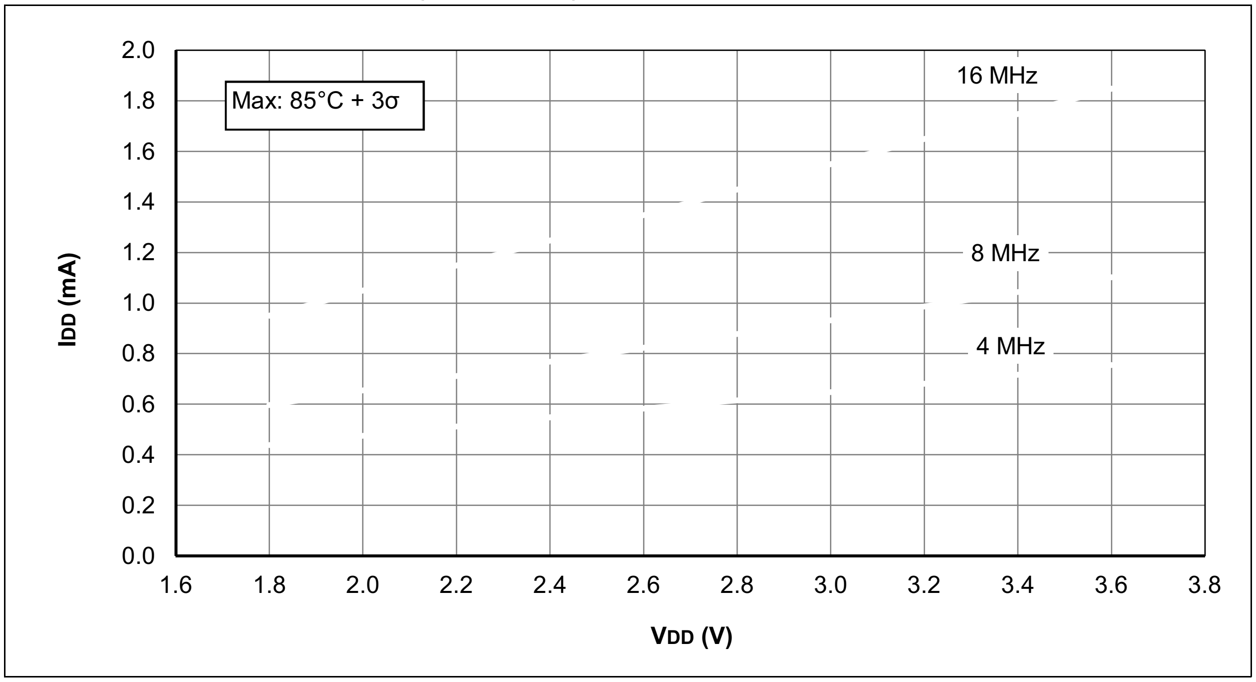


FIGURE 26-33: IPD, WATCHDOG TIMER (WDT), PIC16LF1526 ONLY

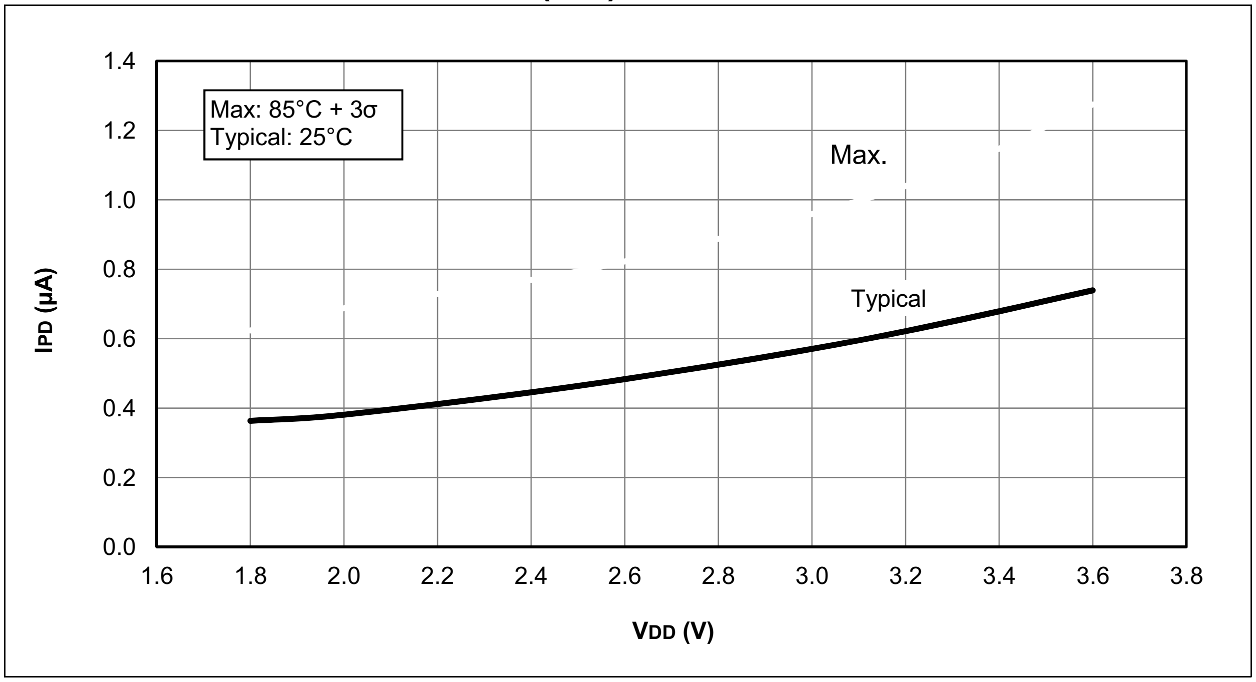
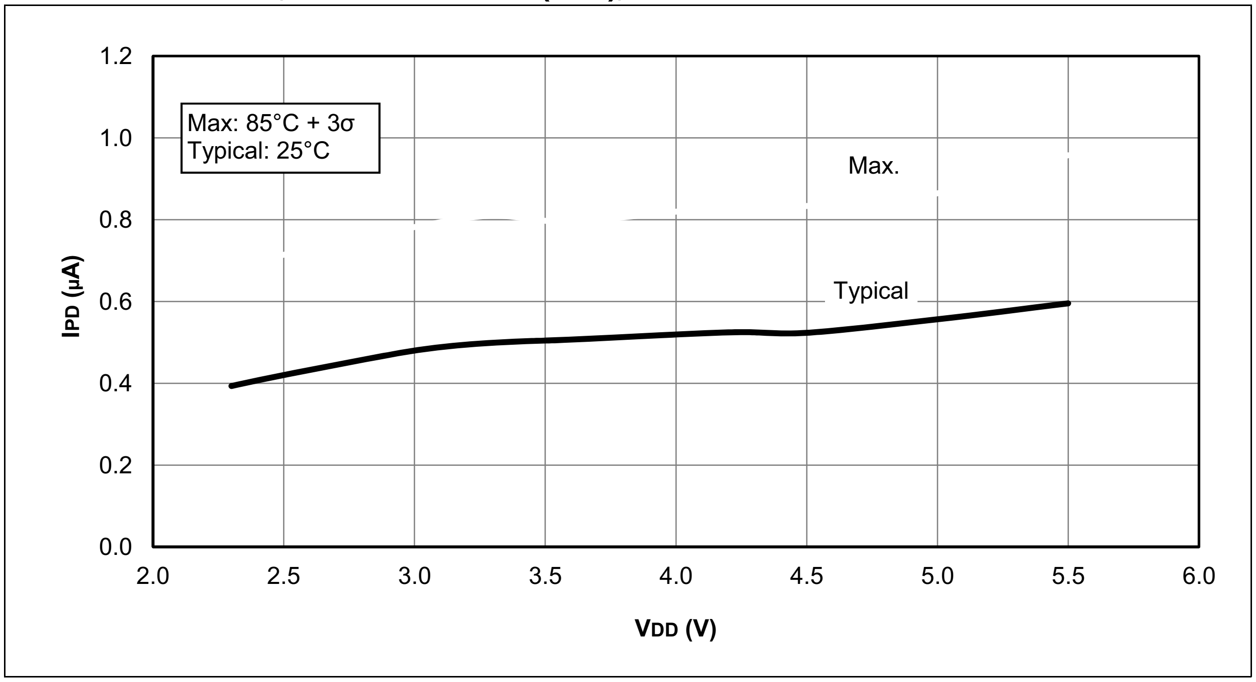


FIGURE 26-34: IPD, WATCHDOG TIMER (WDT), PIC16F1526/7 ONLY


Max: 85°C + 3σ
Typical: 25°C
Max.
Typical
IPD (µA)
VDD (V)