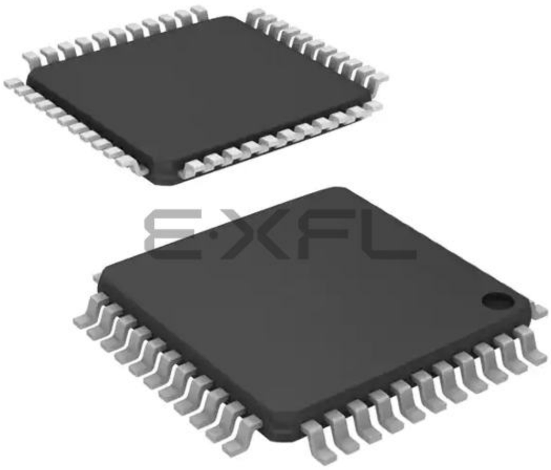


Welcome to E-XFL.COM

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	32MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LCD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	7KB (4K x 14)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 14x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°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/pic16f1934-e-pt

TABLE 1-2: PIC16(L)F1934/6/7 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
RA0/AN0/C12IN0-/C2OUT(1)/ SRNQ(1)/$\overline{SS}$(1)/VCAP(2)/SEG12	RA0	TTL	CMOS	General purpose I/O.
	AN0	AN	—	A/D Channel 0 input.
	C12IN0-	AN	—	Comparator C1 or C2 negative input.
	C2OUT	—	CMOS	Comparator C2 output.
	SRNQ	—	CMOS	SR Latch inverting output.
	$\overline{SS}$	ST	—	Slave Select input.
	VCAP	Power	Power	Filter capacitor for Voltage Regulator (PIC16F1934/6/7 only).
	SEG12	—	AN	LCD Analog output.
RA1/AN1/C12IN1-/SEG7	RA1	TTL	CMOS	General purpose I/O.
	AN1	AN	—	A/D Channel 1 input.
	C12IN1-	AN	—	Comparator C1 or C2 negative input.
	SEG7	—	AN	LCD Analog output.
RA2/AN2/C2IN+/VREF-/ DACOUT/COM2	RA2	TTL	CMOS	General purpose I/O.
	AN2	AN	—	A/D Channel 2 input.
	C2IN+	AN	—	Comparator C2 positive input.
	VREF-	AN	—	A/D Negative Voltage Reference input.
	DACOUT	—	AN	Voltage Reference output.
	COM2	—	AN	LCD Analog output.
RA3/AN3/C1IN+/VREF+/ COM3(3)/SEG15	RA3	TTL	CMOS	General purpose I/O.
	AN3	AN	—	A/D Channel 3 input.
	C1IN+	AN	—	Comparator C1 positive input.
	VREF+	AN	—	A/D Voltage Reference input.
	COM3(3)	—	AN	LCD Analog output.
	SEG15	—	AN	LCD Analog output.
RA4/C1OUT/CPS6/T0CKI/SRQ/ CCP5/SEG4	RA4	TTL	CMOS	General purpose I/O.
	C1OUT	—	CMOS	Comparator C1 output.
	CPS6	AN	—	Capacitive sensing input 6.
	T0CKI	ST	—	Timer0 clock input.
	SRQ	—	CMOS	SR Latch non-inverting output.
	CCP5	ST	CMOS	Capture/Compare/PWM5.
	SEG4	—	AN	LCD Analog output.
RA5/AN4/C2OUT(1)/CPS7/ SRNQ(1)/$\overline{SS}$(1)/VCAP(2)/SEG5	RA5	TTL	CMOS	General purpose I/O.
	AN4	AN	—	A/D Channel 4 input.
	C2OUT	—	CMOS	Comparator C2 output.
	CPS7	AN	—	Capacitive sensing input 7.
	SRNQ	—	CMOS	SR Latch inverting output.
	$\overline{SS}$	ST	—	Slave Select input.
	VCAP	Power	Power	Filter capacitor for Voltage Regulator (PIC16F1934/6/7 only).
	SEG5	—	AN	LCD Analog output.

Legend: AN = Analog input or output; CMOS = CMOS compatible input or output; OD = Open Drain; TTL = TTL compatible input; ST = Schmitt Trigger input with CMOS levels; I^{2}C™ = Schmitt Trigger input with I^{2}C levels; HV = High Voltage; XTAL = Crystal

Note 1: Pin function is selectable via the APFCON register.
2: PIC16F1934/6/7 devices only.
3: PIC16(L)F1936 devices only.
4: PORTD is available on PIC16(L)F1934/7 devices only.
5: RE<2:0> are available on PIC16(L)F1934/7 devices only.

TABLE 3-7: PIC16(L)F1934/6/7 MEMORY MAP, BANKS 16-23

	BANK 16		BANK 17		BANK 18		BANK 19		BANK 20		BANK 21		BANK 22		BANK 23
800h	INDF0	880h	INDF0	900h	INDF0	980h	INDF0	A00h	INDF0	A80h	INDF0	B00h	INDF0	B80h	INDF0
801h	INDF1	881h	INDF1	901h	INDF1	981h	INDF1	A01h	INDF1	A81h	INDF1	B01h	INDF1	B81h	INDF1
802h	PCL	882h	PCL	902h	PCL	982h	PCL	A02h	PCL	A82h	PCL	B02h	PCL	B82h	PCL
803h	STATUS	883h	STATUS	903h	STATUS	983h	STATUS	A03h	STATUS	A83h	STATUS	B03h	STATUS	B83h	STATUS
804h	FSR0L	884h	FSR0L	904h	FSR0L	984h	FSR0L	A04h	FSR0L	A84h	FSR0L	B04h	FSR0L	B84h	FSR0L
805h	FSR0H	885h	FSR0H	905h	FSR0H	985h	FSR0H	A05h	FSR0H	A85h	FSR0H	B05h	FSR0H	B85h	FSR0H
806h	FSR1L	886h	FSR1L	906h	FSR1L	986h	FSR1L	A06h	FSR1L	A86h	FSR1L	B06h	FSR1L	B86h	FSR1L
807h	FSR1H	887h	FSR1H	907h	FSR1H	987h	FSR1H	A07h	FSR1H	A87h	FSR1H	B07h	FSR1H	B87h	FSR1H
808h	BSR	888h	BSR	908h	BSR	988h	BSR	A08h	BSR	A88h	BSR	B08h	BSR	B88h	BSR
809h	WREG	889h	WREG	909h	WREG	989h	WREG	A09h	WREG	A89h	WREG	B09h	WREG	B89h	WREG
80Ah	PCLATH	88Ah	PCLATH	90Ah	PCLATH	98Ah	PCLATH	A0Ah	PCLATH	A8Ah	PCLATH	B0Ah	PCLATH	B8Ah	PCLATH
80Bh	INTCON	88Bh	INTCON	90Bh	INTCON	98Bh	INTCON	A0Bh	INTCON	A8Bh	INTCON	B0Bh	INTCON	B8Bh	INTCON
80Ch	—	88Ch	—	90Ch	—	98Ch	—	A0Ch	—	A8Ch	—	B0Ch	—	B8Ch	—
80Dh	—	88Dh	—	90Dh	—	98Dh	—	A0Dh	—	A8Dh	—	B0Dh	—	B8Dh	—
80Eh	—	88Eh	—	90Eh	—	98Eh	—	A0Eh	—	A8Eh	—	B0Eh	—	B8Eh	—
80Fh	—	88Fh	—	90Fh	—	98Fh	—	A0Fh	—	A8Fh	—	B0Fh	—	B8Fh	—
810h	—	890h	—	910h	—	990h	—	A10h	—	A90h	—	B10h	—	B90h	—
811h	—	891h	—	911h	—	991h	—	A11h	—	A91h	—	B11h	—	B91h	—
812h	—	892h	—	912h	—	992h	—	A12h	—	A92h	—	B12h	—	B92h	—
813h	—	893h	—	913h	—	993h	—	A13h	—	A93h	—	B13h	—	B93h	—
814h	—	894h	—	914h	—	994h	—	A14h	—	A94h	—	B14h	—	B94h	—
815h	—	895h	—	915h	—	995h	—	A15h	—	A95h	—	B15h	—	B95h	—
816h	—	896h	—	916h	—	996h	—	A16h	—	A96h	—	B16h	—	B96h	—
817h	—	897h	—	917h	—	997h	—	A17h	—	A97h	—	B17h	—	B97h	—
818h	—	898h	—	918h	—	998h	—	A18h	—	A98h	—	B18h	—	B98h	—
819h	—	899h	—	919h	—	999h	—	A19h	—	A99h	—	B19h	—	B99h	—
81Ah	—	89Ah	—	91Ah	—	99Ah	—	A1Ah	—	A9Ah	—	B1Ah	—	B9Ah	—
81Bh	—	89Bh	—	91Bh	—	99Bh	—	A1Bh	—	A9Bh	—	B1Bh	—	B9Bh	—
81Ch	—	89Ch	—	91Ch	—	99Ch	—	A1Ch	—	A9Ch	—	B1Ch	—	B9Ch	—
81Dh	—	89Dh	—	91Dh	—	99Dh	—	A1Dh	—	A9Dh	—	B1Dh	—	B9Dh	—
81Eh	—	89Eh	—	91Eh	—	99Eh	—	A1Eh	—	A9Eh	—	B1Eh	—	B9Eh	—
81Fh	—	89Fh	—	91Fh	—	99Fh	—	A1Fh	—	A9Fh	—	B1Fh	—	B9Fh	—
820h 86Fh	Unimplemented Read as '0'	8A0h 8EFh	Unimplemented Read as '0'	920h 96Fh	Unimplemented Read as '0'	9A0h 9EFh	Unimplemented Read as '0'	A20h A6Fh	Unimplemented Read as '0'	AA0h AEFh	Unimplemented Read as '0'	B20h B6Fh	Unimplemented Read as '0'	BA0h BEFh	Unimplemented Read as '0'
870h 87Fh	Accesses 70h – 7Fh	8F0h 8FFh	Accesses 70h – 7Fh	970h 97Fh	Accesses 70h – 7Fh	9F0h 9FFh	Accesses 70h – 7Fh	A70h A7Fh	Accesses 70h – 7Fh	AF0h AFFh	Accesses 70h – 7Fh	B70h B7Fh	Accesses 70h – 7Fh	BF0h BFFh	Accesses 70h – 7Fh

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

TABLE 3-12: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 4											
200h(2)	INDF0	Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register)								xxxx xxxx	xxxx xxxx
201h(2)	INDF1	Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register)								xxxx xxxx	xxxx xxxx
202h(2)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	0000 0000
203h(2)	STATUS	—	—	—	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	---1 1000	---q quuu
204h(2)	FSR0L	Indirect Data Memory Address 0 Low Pointer								0000 0000	uuuu uuuu
205h(2)	FSR0H	Indirect Data Memory Address 0 High Pointer								0000 0000	0000 0000
206h(2)	FSR1L	Indirect Data Memory Address 1 Low Pointer								0000 0000	uuuu uuuu
207h(2)	FSR1H	Indirect Data Memory Address 1 High Pointer								0000 0000	0000 0000
208h(2)	BSR	—	—	—	BSR<4:0>					---0 0000	---0 0000
209h(2)	WREG	Working Register								0000 0000	uuuu uuuu
20Ah(1, 2)	PCLATH	—	Write Buffer for the upper 7 bits of the Program Counter							-000 0000	-000 0000
20Bh(2)	INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	0000 0000	0000 0000
20Ch	—	Unimplemented								—	—
20Dh	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	1111 1111	1111 1111
20Eh	—	Unimplemented								—	—
20Fh	—	Unimplemented								—	—
210h	WPUE	—	—	—	—	WPUE3	—	—	—	---- 1---	---- 1---
211h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
212h	SSPADD	ADD<7:0>								0000 0000	0000 0000
213h	SSPMSK	MSK<7:0>								1111 1111	1111 1111
214h	SSPSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000	0000 0000
215h	SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM<3:0>				0000 0000	0000 0000
216h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	0000 0000
217h	SSPCON3	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN	0000 0000	0000 0000
218h	—	Unimplemented								—	—
219h	—	Unimplemented								—	—
21Ah	—	Unimplemented								—	—
21Bh	—	Unimplemented								—	—
21Ch	—	Unimplemented								—	—
21Dh	—	Unimplemented								—	—
21Eh	—	Unimplemented								—	—
21Fh	—	Unimplemented								—	—

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: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<14:8>, whose contents are transferred to the upper byte of the program counter.

2: These registers can be addressed from any bank.

3: These registers/bits are not implemented on PIC16(L)F1936 devices, read as '0'.

4: Unimplemented, read as '1'.

7.6.4 PIE3 REGISTER

The PIE3 register contains the interrupt enable bits, as shown in Register 7-4.

Note:	Bit PEIE of the INTCON register must be set to enable any peripheral interrupt.

REGISTER 7-4: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	U-0
—	CCP5IE	CCP4IE	CCP3IE	TMR6IE	—	TMR4IE	—
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 **CCP5IE:** CCP5 Interrupt Enable bit
1 = Enables the CCP5 interrupt
0 = Disables the CCP5 interrupt

bit 5 **CCP4IE:** CCP4 Interrupt Enable bit
1 = Enables the CCP4 interrupt
0 = Disables the CCP4 interrupt

bit 4 **CCP3IE:** CCP3 Interrupt Enable bit
1 = Enables the CCP3 interrupt
0 = Disables the CCP3 interrupt

bit 3 **TMR6IE:** TMR6 to PR6 Match Interrupt Enable bit
1 = Enables the TMR6 to PR6 Match interrupt
0 = Disables the TMR6 to PR6 Match interrupt

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **TMR4IE:** TMR4 to PR4 Match Interrupt Enable bit
1 = Enables the TMR4 to PR4 Match interrupt
0 = Disables the TMR4 to PR4 Match interrupt

bit 0 **Unimplemented:** Read as ‘0’

12.2 PORTA Registers

PORTA is a 8-bit wide, bidirectional port. The corresponding data direction register is TRISA (Register 12-3). Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., disable the output driver). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., enables output driver and puts the contents of the output latch on the selected pin). Example 12-1 shows how to initialize an I/O port.

Reading the PORTA register (Register 12-2) 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 (LATA).

The TRISA register (Register 12-3) controls the PORTA pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISA register are maintained set when using them as analog inputs. I/O pins configured as analog input always read '0'.

12.2.1 ANSELA REGISTER

The ANSELA register (Register 12-5) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELA 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 ANSELA 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 ANSELA 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.

12.2.2 PORTA FUNCTIONS AND OUTPUT PRIORITIES

Each PORTA pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 12-2.

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

Analog input functions, such as ADC, comparator and CapSense inputs, 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 shown in Table 12-2.

TABLE 12-2: PORTA OUTPUT PRIORITY

Pin Name	Function Priority(1)
RA0	VCAP SEG12 (LCD) SRNQ (SR Latch) C2OUT (Comparator) RA0
RA1	SEG7 (LCD) RA1
RA2	COM2 (LCD) AN2 (DAC) RA2
RA3	COM3 (LCD) 28-pin only SEG15 RA3
RA4	SEG4 (LCD) SRQ (SR Latch) C1OUT (Comparator) CCP5, 28-pin only RA4
RA5	VCAP (enabled by Config. Word) SEG5 (LCD) SRNQ (SR Latch) C2OUT (Comparator) RA5
RA6	VCAP (enabled by Config. Word) OSC2 (enabled by Config. Word) CLKOUT (enabled by Config. Word) SEG1 (LCD) RA6
RA7	OSC1/CLKIN (enabled by Config. Word) SEG2 (LCD) RA7

Note 1: Priority listed from highest to lowest.

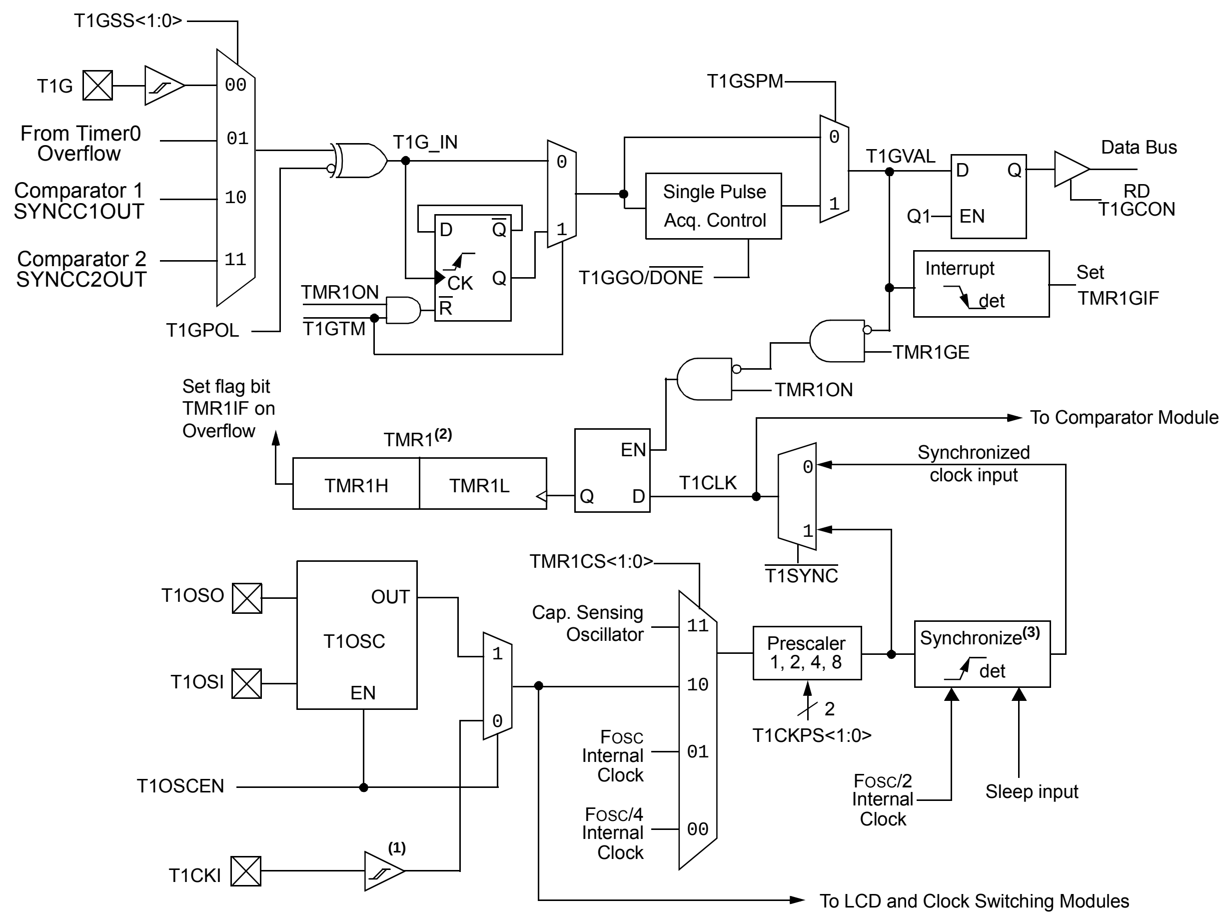
21.0 TIMER1 MODULE WITH GATE CONTROL

The Timer1 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMR1H:TMR1L)
- Programmable internal or external clock source
- 2-bit prescaler
- Dedicated 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- Time base for the Capture/Compare function
- Special Event Trigger (with CCP/ECCP)
- Selectable Gate Source Polarity
- Gate Toggle Mode
- Gate Single-pulse Mode
- Gate Value Status
- Gate Event Interrupt

Figure 21-1 is a block diagram of the Timer1 module.

FIGURE 21-1: TIMER1 BLOCK DIAGRAM



Note 1: ST Buffer is high speed type when using T1CKI.
2: Timer1 register increments on rising edge.
3: Synchronize does not operate while in Sleep.

NOTES:

23.1.6 ALTERNATE PIN LOCATIONS

This module incorporates I/O pins that can be moved to other locations with the use of the alternate pin function register, APFCON. To determine which pins can be moved and what their default locations are upon a Reset, see **Section 12.1 "Alternate Pin Function"** for more information.

TABLE 23-2: SUMMARY OF REGISTERS ASSOCIATED WITH CAPTURE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
APFCON	—	CCP3SEL	T1GSEL	P2BSEL	SRNQSEL	C2OUTSEL	SSSEL	CCP2SEL	131
CCPxCON	PxM<1:0>(1)		DCxB<1:0>		CCPxM<3:0>				234
CCPRxL	Capture/Compare/PWM Register x Low Byte (LSB)								212
CCPRxH	Capture/Compare/PWM Register x High Byte (MSB)								212
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	98
PIE1	TMR1GIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	99
PIE2	OSFIE	C2IE	C1IE	EEIE	BCLIE	LCDIE	—	CCP2IE	100
PIE3	—	CCP5IE	CCP4IE	CCP3IE	TMR6IE	—	TMR4IE	—	101
PIR1	TMR1GIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	102
PIR2	OSFIF	C2IF	C1IF	EEIF	BCLIF	LCDIF	—	CCP2IF	103
PIR3	—	CCP5IF	CCP4IF	CCP3IF	TMR6IF	—	TMR4IF	—	104
T1CON	TMR1CS<1:0>		T1CKPS<1:0>		T1OSCEN	$\overline{\text{T1SYNC}}$	—	TMR1ON	203
T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/DONE	T1GVAL	T1GSS<1:0>		204
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								199
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								199
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	133
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	138
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	142
TRISD(2)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	145
TRISE	—	—	—	—	—(3)	TRISE2(2)	TRISE1(2)	TRISE0(2)	148

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by Capture mode.

Note 1: Applies to ECCP modules only.

2: These registers/bits are not implemented on PIC16(L)F1936 devices, read as '0'.

3: Unimplemented, read as '1'.

FIGURE 24-12: I²C START AND STOP CONDITIONS

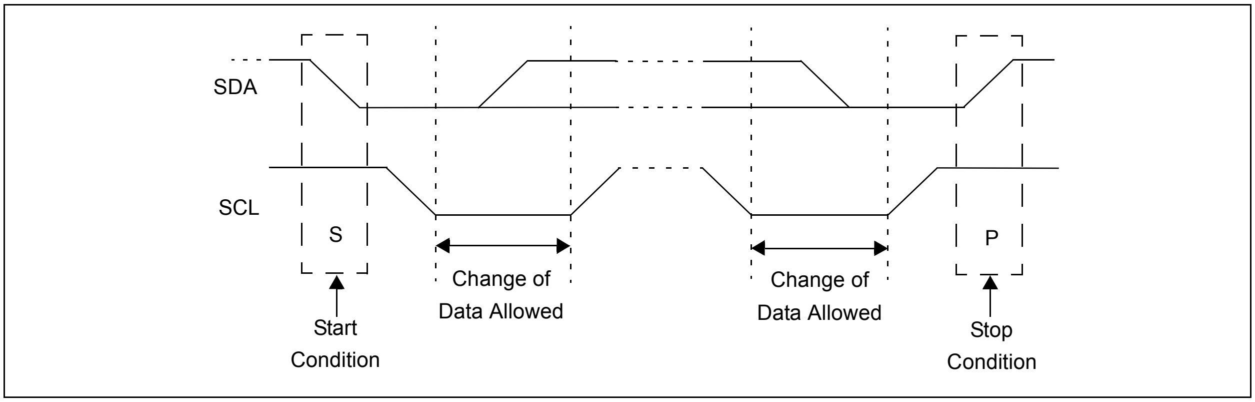
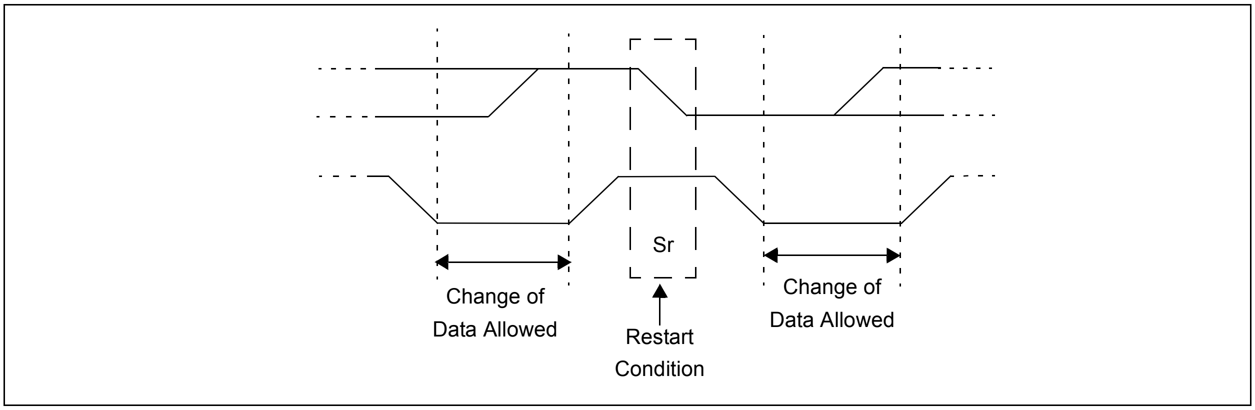


FIGURE 24-13: I²C RESTART CONDITION



24.6.10 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 MSSP interrupt is enabled).

24.6.11 EFFECTS OF A RESET

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

24.6.12 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 MSSP module is disabled. Control of the I^{2}C bus may be taken when the P bit of the SSPSTAT register is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the SSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDA 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 BCLIF bit.

The states where arbitration can be lost are:

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

24.6.13 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 SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF, and reset the I^{2}C port to its Idle state (Figure 24-31).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are deasserted and the SSPBUF 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.

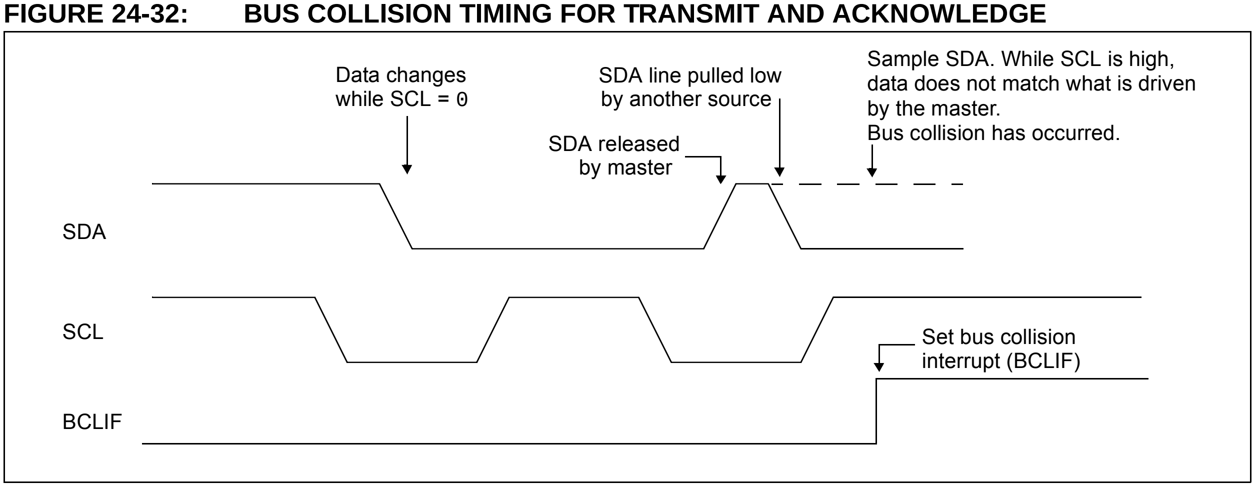
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine 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 SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF 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 SSPSTAT register, or the bus is Idle and the S and P bits are cleared.

FIGURE 24-32: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



25.1.2.8 Asynchronous Reception Set-up:

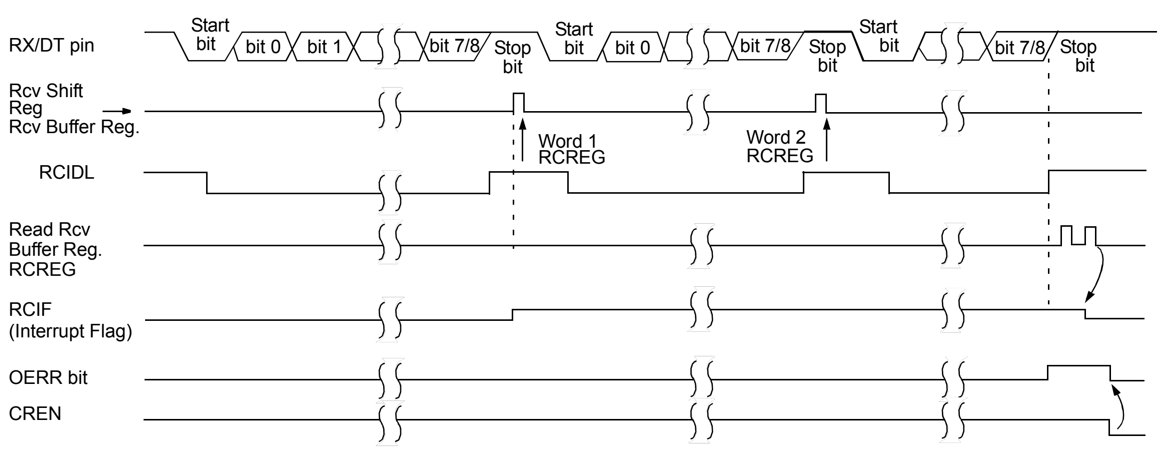
1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 25.3 “EUSART Baud Rate Generator (BRG)”**).
2. Clear the ANSEL bit for the RX pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. If 9-bit reception is desired, set the RX9 bit.
6. Enable reception by setting the CREN bit.
7. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
8. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
9. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register.
10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

25.1.2.9 9-bit Address Detection Mode Set-up

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 25.3 “EUSART Baud Rate Generator (BRG)”**).
2. Clear the ANSEL bit for the RX pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. Enable 9-bit reception by setting the RX9 bit.
6. Enable address detection by setting the ADDEN bit.
7. Enable reception by setting the CREN bit.
8. The RCIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
9. Read the RCSTA register to get the error flags. The ninth data bit will always be set.
10. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register. Software determines if this is the device’s address.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

FIGURE 25-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

REGISTER 26-1: CPSCON0: CAPACITIVE SENSING CONTROL REGISTER 0

R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0	R-0/0	R/W-0/0
CPSON	—	—	—	CPSRNG<1:0>		CPSOUT	T0XCS
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 **CPSON:** CPS Module Enable bit
1 = CPS module is enabled
0 = CPS module is disabled

bit 6-4 **Unimplemented:** Read as ‘0’

bit 3-2 **CPSRNG<1:0>:** Capacitive Sensing Current Range
00 = Oscillator is off
01 = Oscillator is in Low Range. Charge/Discharge Current is nominally 0.1 µA
10 = Oscillator is in Medium Range. Charge/Discharge Current is nominally 1.2 µA
11 = Oscillator is in High Range. Charge/Discharge Current is nominally 18 µA

bit 1 **CPSOUT:** Capacitive Sensing Oscillator Status bit
1 = Oscillator is sourcing current (Current flowing out of the pin)
0 = Oscillator is sinking current (Current flowing into the pin)

bit 0 **T0XCS:** Timer0 External Clock Source Select bit
If TMR0CS = 1:
The T0XCS bit controls which clock external to the core/Timer0 module supplies Timer0:
1 = Timer0 clock source is the capacitive sensing oscillator
0 = Timer0 clock source is the T0CKI pin
If TMR0CS = 0:
Timer0 clock source is controlled by the core/Timer0 module and is FOSC/4

29.0 INSTRUCTION SET SUMMARY

Each PIC16 instruction is a 14-bit word containing the operation code (opcode) and all required operands. The opcodes are broken into three broad categories.

- Byte Oriented
- Bit Oriented
- Literal and Control

The literal and control category contains the most varied instruction word format.

Table 29-3 lists the instructions recognized by the MPASM™ assembler.

All instructions are executed within a single instruction cycle, with the following exceptions, which may take two or three cycles:

- Subroutine takes two cycles (CALL, CALLW)
- Returns from interrupts or subroutines take two cycles (RETURN, RETLW, RETFIE)
- Program branching takes two cycles (GOTO, BRA, BRW, BTFSS, BTFSC, DECFSZ, INCSFZ)
- One additional instruction cycle will be used when any instruction references an indirect file register and the file select register is pointing to program memory.

One instruction cycle consists of 4 oscillator cycles; for an oscillator frequency of 4 MHz, this gives a nominal instruction execution rate of 1 MHz.

All instruction examples use the format '0xhh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

29.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the instruction, or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

TABLE 29-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
n	FSR or INDF number. (0-1)
mm	Pre-post increment-decrement mode selection

TABLE 29-2: ABBREVIATION DESCRIPTIONS

Field	Description
PC	Program Counter
$\overline{\text{TO}}$	Time-out bit
C	Carry bit
DC	Digit carry bit
Z	Zero bit
$\overline{\text{PD}}$	Power-down bit

29.2 Instruction Descriptions

ADDFSR — Add Literal to FSRn

Syntax:	[*label*] ADDFSR FSRn, k
Operands:	-32 ≤ k ≤ 31 n ∈ [0, 1]
Operation:	FSR(n) + k → FSR(n)
Status Affected:	None
Description:	The signed 6-bit literal ‘k’ is added to the contents of the FSRnH:FSRnL register pair. FSRn is limited to the range 0000h - FFFFh. Moving beyond these bounds will cause the FSR to wrap around.

ADDLW — Add literal and W

Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight-bit literal ‘k’ and the result is placed in the W register.

ADDWF — Add W and f

Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

ADDWFC — ADD W and CARRY bit to f

Syntax:	[*label*] ADDWFC f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) + (C) → dest
Status Affected:	C, DC, Z
Description:	Add W, the Carry flag and data memory location ‘f’. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed in data memory location ‘f’.

ANDLW — AND literal with W

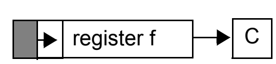
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND’ed with the eight-bit literal ‘k’. The result is placed in the W register.

ANDWF — AND W with f

Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

ASRF — Arithmetic Right Shift

Syntax:	[*label*] ASRF f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<7>)→ dest<7> (f<7:1>) → dest<6:0>, (f<0>) → C,
Status Affected:	C, Z
Description:	The contents of register ‘f’ are shifted one bit to the right through the Carry flag. The MSb remains unchanged. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’.



MOVIW — Move INDFn to W

Syntax: [*label*] MOVIW ++FSRn
[*label*] MOVIW --FSRn
[*label*] MOVIW FSRn++
[*label*] MOVIW FSRn--
[*label*] MOVIW k[FSRn]

Operands: n ∈ [0,1]
mm ∈ [00,01, 10, 11]
$-32 \leq k \leq 31$

Operation: INDFn → W
Effective address is determined by
- FSR + 1 (preincrement)
- FSR - 1 (predecrement)
- FSR + k (relative offset)

After the Move, the FSR value will be either:
- FSR + 1 (all increments)
- FSR - 1 (all decrements)
- Unchanged

Status Affected: Z

Mode	Syntax	mm
Preincrement	++FSRn	00
Predecrement	--FSRn	01
Postincrement	FSRn++	10
Postdecrement	FSRn--	11

Description: This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it.

Note: The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn.

FSRn is limited to the range 0000h - FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wrap around.

MOVLB — Move literal to BSR

Syntax: [*label*] MOVLB k

Operands: $0 \leq k \leq 15$

Operation: k → BSR

Status Affected: None

Description: The five-bit literal ‘k’ is loaded into the Bank Select Register (BSR).

MOVLP — Move literal to PCLATH

Syntax: [*label*] MOVLP k

Operands: $0 \leq k \leq 127$

Operation: k → PCLATH

Status Affected: None

Description: The seven-bit literal ‘k’ is loaded into the PCLATH register.

MOVLW — Move literal to W

Syntax: [*label*] MOVLW k

Operands: $0 \leq k \leq 255$

Operation: k → (W)

Status Affected: None

Description: The eight-bit literal ‘k’ is loaded into W register. The “don’t cares” will assemble as ‘0’s.

Words: 1

Cycles: 1

Example: `MOVLW 0x5A`

After Instruction
W = 0x5A

MOVWF — Move W to f

Syntax: [*label*] MOVWF f

Operands: $0 \leq f \leq 127$

Operation: (W) → (f)

Status Affected: None

Description: Move data from W register to register ‘f’.

Words: 1

Cycles: 1

Example: `MOVWF OPTION_REG`

Before Instruction
OPTION_REG = 0xFF
W = 0x4F
After Instruction
OPTION_REG = 0x4F
W = 0x4F

30.1 DC Characteristics: PIC16(L)F1934/6/7-I/E (Industrial, Extended)

PIC16LF1934/36/37			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
PIC16F1934/36/37			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
D001	VDD	**Supply Voltage**					
		PIC16LF1934/36/37	1.8 2.3	— —	3.6 3.6	V V	FOSC ≤ 16 MHz: FOSC ≤ 32 MHz (**Note 2**)
D001		PIC16F1934/36/37	1.8 2.3	— —	5.5 5.5	V V	FOSC ≤ 16 MHz: FOSC ≤ 32 MHz (**Note 2**)
D002*	VDR	**RAM Data Retention Voltage(1)**					
		PIC16LF1934/36/37	1.5	—	—	V	Device in Sleep mode
D002*		PIC16F1934/36/37	1.7	—	—	V	Device in Sleep mode
	VPOR*	**Power-on Reset Release Voltage**	—	1.6	—	V	
	VPORR*	**Power-on Reset Rearm Voltage**					
		PIC16LF1934/36/37	—	0.8	—	V	Device in Sleep mode
		PIC16F1934/36/37	—	1.7	—	V	Device in Sleep mode
D003	VADFVR	**Fixed Voltage Reference Voltage for ADC**	-8		6	%	1.024V, VDD ≥ 2.5V 2.048V, VDD ≥ 2.5V 4.096V, VDD ≥ 4.75V
D003A	VCDAFVR	**Fixed Voltage Reference Voltage for Comparator and DAC**	-11		7	%	1.024V, VDD ≥ 2.5V 2.048V, VDD ≥ 2.5V 4.096V, VDD ≥ 4.75V
D003B	VLCDFVR	**Fixed Voltage Reference Voltage for LCD Bias**	-11	—	10	%	3.072V, VDD ≥ 3.6V
D004*	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05	—	—	V/ms	See **Section 6.1 “Power-on Reset (POR)”** for details.

* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

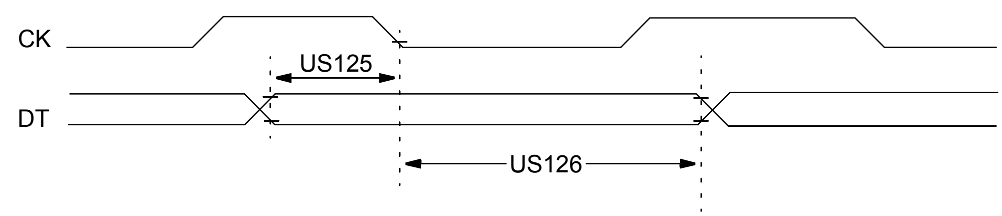
Note 1: This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.

2: PLL required for 32 MHz operation.

TABLE 30-12: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Standard Operating Conditions (unless otherwise stated) Operating Temperature -40°C ≤ TA ≤ +125°C							
Param. No.	**Symbol**	**Characteristic**		**Min.**	**Max.**	**Units**	**Conditions**
US120	TCKH2DTV	SYNC XMIT (Master and Slave) Clock high to data-out valid	3.0-5.5V	—	80	ns	
			1.8-5.5V	—	100	ns	
US121	TCKRF	Clock out rise time and fall time (Master mode)	3.0-5.5V	—	45	ns	
			1.8-5.5V	—	50	ns	
US122	TDTRF	Data-out rise time and fall time	3.0-5.5V	—	45	ns	
			1.8-5.5V	—	50	ns	

FIGURE 30-15: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 30-5 for load conditions.

TABLE 30-13: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Standard Operating Conditions (unless otherwise stated) Operating Temperature -40°C ≤ TA ≤ +125°C						
Param. No.	**Symbol**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
US125	TDTV2CKL	SYNC RCV (Master and Slave) Data-hold before CK ↓ (DT hold time)	10	—	ns	
US126	TCKL2DTL	Data-hold after CK ↓ (DT hold time)	15	—	ns	

FIGURE 31-72: PIC16LF1937 BASE IPD

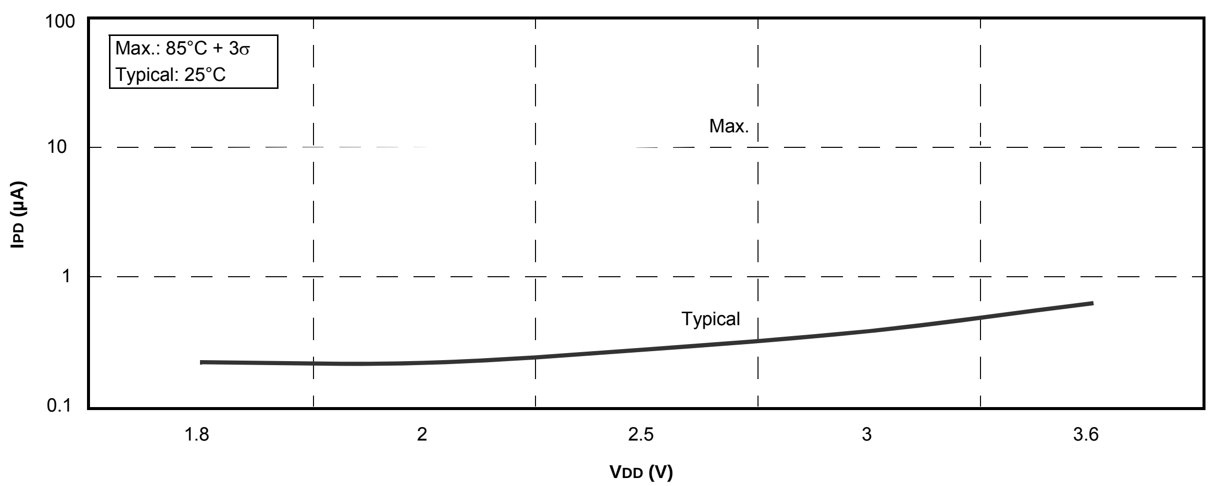
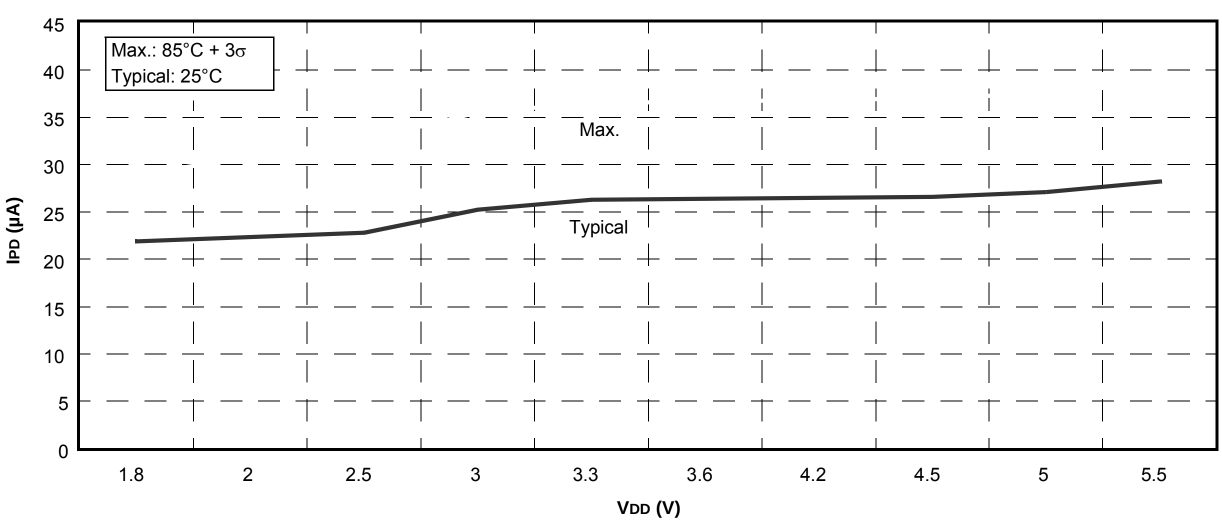


FIGURE 31-73: PIC16F1937 BASE IPD



32.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers and dsPIC® digital signal controllers are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB C Compiler for Various Device Families
  - HI-TECH C for Various Device Families
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- Device Programmers
  - PICkit™ 2 Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits, and Starter Kits

32.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - In-Circuit Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either C or assembly)
- One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (C or assembly)
  - Mixed C and assembly
  - Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

NOTES: