



Welcome to [E-XFL.COM](https://www.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	-
Peripherals	Brown-out Detect/Reset, LCD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1902t-i-ml

FIGURE 3-1: PROGRAM MEMORY MAP AND STACK FOR PIC16LF1902

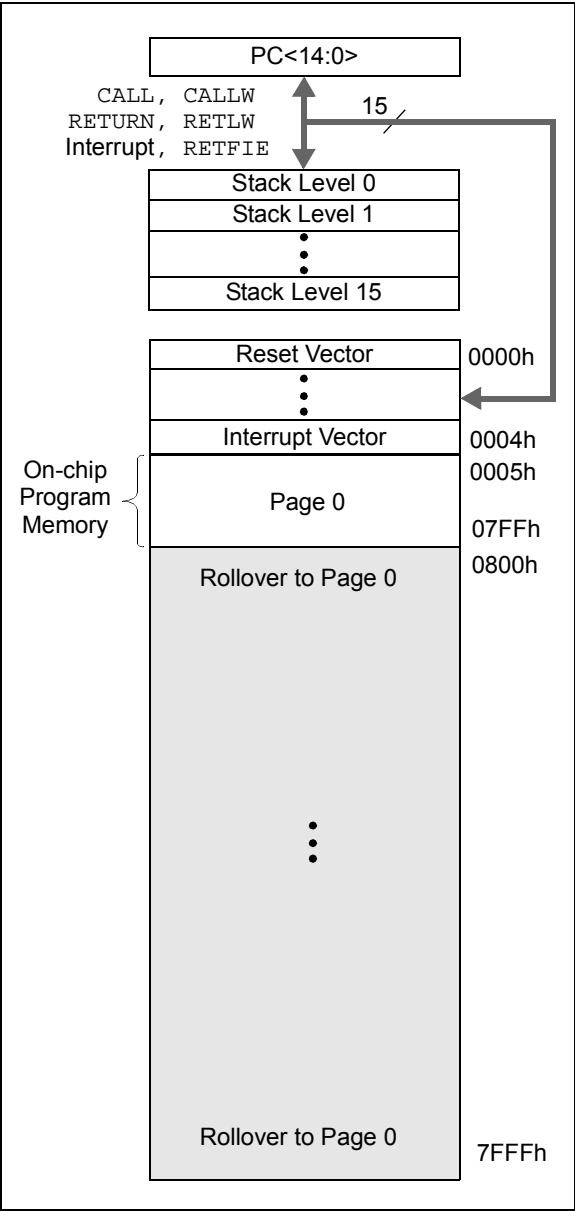


FIGURE 3-2: PROGRAM MEMORY MAP AND STACK FOR PIC16LF1903

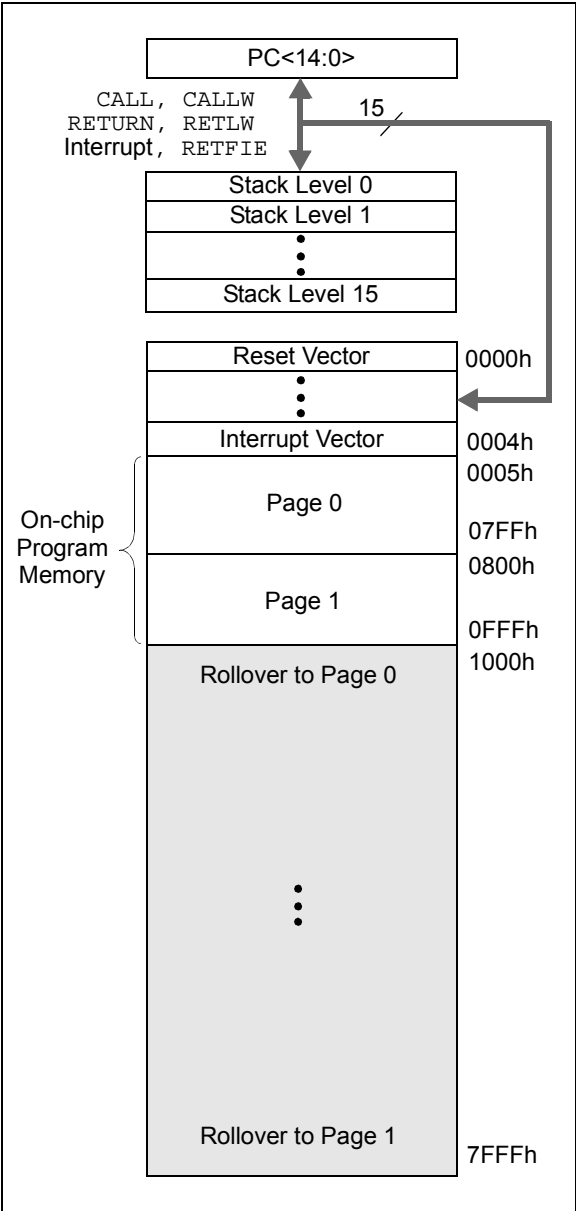


TABLE 3-3: PIC16LF1902/3 MEMORY MAP (CONTINUED)

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

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

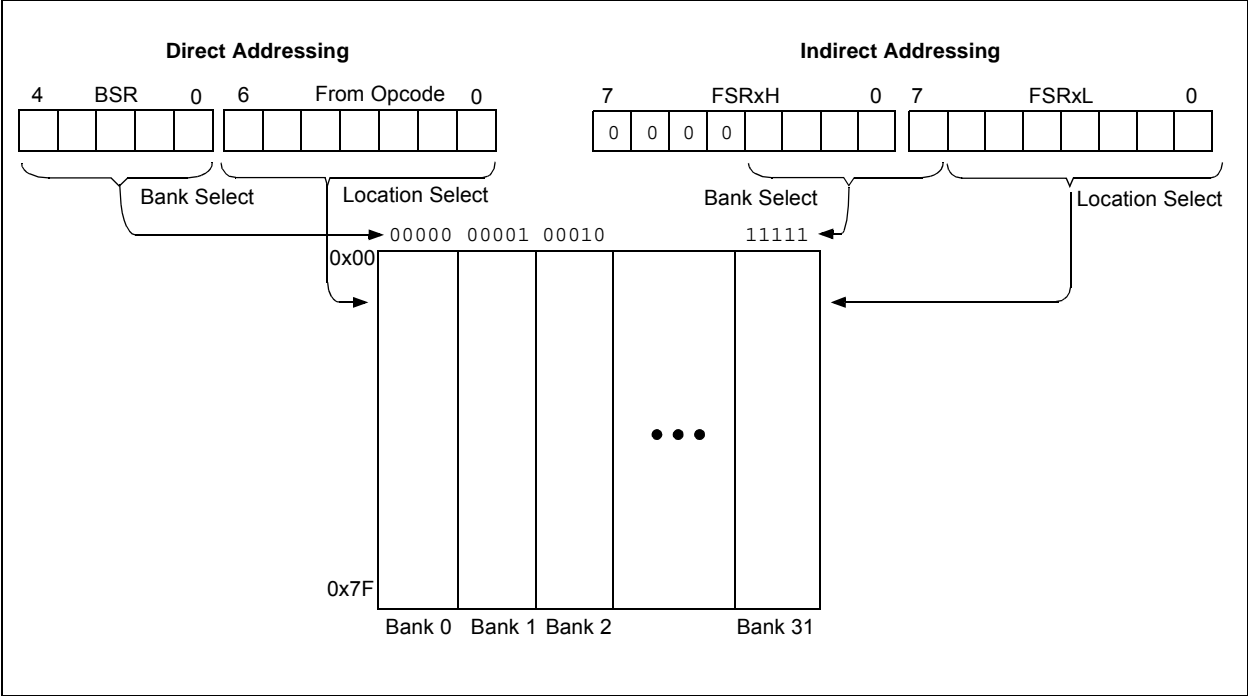
BANK 24		BANK 25		BANK 26		BANK 27		BANK 28		BANK 29		BANK 30	
C00h	Core Registers (Table 3-2)	C80h	Core Registers (Table 3-2)	D00h	Core Registers (Table 3-2)	D80h	Core Registers (Table 3-2)	E00h	Core Registers (Table 3-2)	E80h	Core Registers (Table 3-2)	F00h	Core Registers (Table 3-2)
C0Bh C0Ch	Unimplemented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

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

3.5.1 TRADITIONAL DATA MEMORY

The traditional data memory is a region from FSR address 0x000 to FSR address 0xFFF. The addresses correspond to the absolute addresses of all SFR, GPR and common registers.

FIGURE 3-10: TRADITIONAL DATA MEMORY MAP



4.2 Code Protection

Code protection allows the device to be protected from unauthorized access. Program memory protection is controlled independently. Internal access to the program memory is unaffected by any code protection setting.

4.2.1 PROGRAM MEMORY PROTECTION

The entire program memory space is protected from external reads and writes by the CP bit in Configuration Word 1. When $\overline{CP} = 0$, external reads and writes of program memory are inhibited and a read will return all '0's. The CPU can continue to read program memory, regardless of the protection bit settings. Writing the program memory is dependent upon the write protection setting. See **Section 4.3 “Write Protection”** for more information.

4.3 Write Protection

Write protection allows the device to be protected from unintended self-writes. Applications, such as boot loader software, can be protected while allowing other regions of the program memory to be modified.

The WRT<1:0> bits in Configuration Word 2 define the size of the program memory block that is protected.

4.4 User ID

Four memory locations (8000h-8003h) are designated as ID locations where the user can store checksum or other code identification numbers. These locations are readable and writable during normal execution. See **Section 10.4 “User ID, Device ID and Configuration Word Access”** for more information on accessing these memory locations. For more information on checksum calculation, see the *“PIC16F193X/LF193X/PIC16F194X/LF194X/PIC16LF190X Memory Programming Specification”* (DS41397).

10.2.2 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

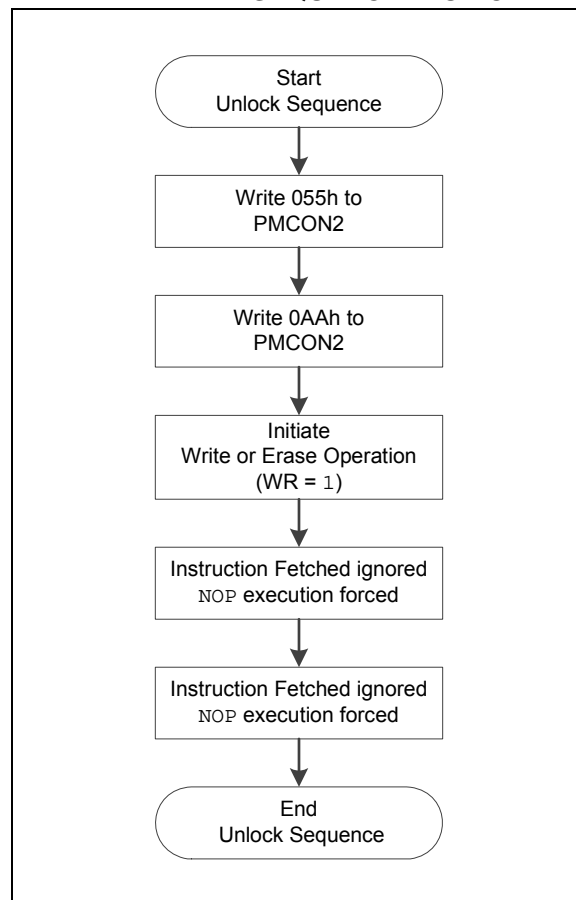
The unlock sequence consists of the following steps:

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 10-3: FLASH PROGRAM MEMORY UNLOCK SEQUENCE FLOWCHART



10.2.4 WRITING TO FLASH PROGRAM MEMORY

Program memory is programmed using the following steps:

1. Load the address in PMADRH:PMADRL of the row to be programmed.
2. Load each write latch with data.
3. Initiate a programming operation.
4. Repeat steps 1 through 3 until all data is written.

Before writing to program memory, the word(s) to be written must be erased or previously unwritten. Program memory can only be erased one row at a time. No automatic erase occurs upon the initiation of the write.

Program memory can be written one or more words at a time. The maximum number of words written at one time is equal to the number of write latches. See Figure 10-5 (row writes to program memory with 32 write latches) for more details.

The write latches are aligned to the Flash row address boundary defined by the upper ten bits of PMADRH:PMADRL, (PMADRH<6:0>:PMADRL<7:5>) with the lower five bits of PMADRL, (PMADRL<4:0>) determining the write latch being loaded. Write operations do not cross these boundaries. At the completion of a program memory write operation, the data in the write latches is reset to contain 0x3FFF.

The following steps should be completed to load the write latches and program a row of program memory. These steps are divided into two parts. First, each write latch is loaded with data from the PMDATH:PMDATL using the unlock sequence with LWLO = 1. When the last word to be loaded into the write latch is ready, the LWLO bit is cleared and the unlock sequence executed. This initiates the programming operation, writing all the latches into Flash Program Memory.

Note: The special unlock sequence is required to load a write latch with data or initiate a Flash programming operation. If the unlock sequence is interrupted, writing to the latches or program memory will not be initiated.

1. Set the WREN bit of the PMCON1 register.
2. Clear the CFGS bit of the PMCON1 register.
3. Set the LWLO bit of the PMCON1 register. When the LWLO bit of the PMCON1 register is '1', the write sequence will only load the write latches and will not initiate the write to Flash Program Memory.
4. Load the PMADRH:PMADRL register pair with the address of the location to be written.
5. Load the PMDATH:PMDATL register pair with the program memory data to be written.
6. Execute the unlock sequence (**Section 10.2.2 "Flash Memory Unlock Sequence"**). The write latch is now loaded.
7. Increment the PMADRH:PMADRL register pair to point to the next location.
8. Repeat steps 5 through 7 until all but the last write latch has been loaded.
9. Clear the LWLO bit of the PMCON1 register. When the LWLO bit of the PMCON1 register is '0', the write sequence will initiate the write to Flash Program Memory.
10. Load the PMDATH:PMDATL register pair with the program memory data to be written.
11. Execute the unlock sequence (**Section 10.2.2 "Flash Memory Unlock Sequence"**). The entire program memory latch content is now written to Flash Program Memory.

Note: The program memory write latches are reset to the blank state (0x3FFF) at the completion of every write or erase operation. As a result, it is not necessary to load all the program memory write latches. Unloaded latches will remain in the blank state.

An example of the complete write sequence is shown in Example 10-3. The initial address is loaded into the PMADRH:PMADRL register pair; the data is loaded using indirect addressing.

REGISTER 10-6: PMCON2: PROGRAM MEMORY CONTROL 2 REGISTER

W-0/0	W-0/0	W-0/0	W-0/0	W-0/0	W-0/0	W-0/0	W-0/0
Program Memory Control Register 2							
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

S = Bit can only be set

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

Flash Memory Unlock Pattern bits

To unlock writes, a 55h must be written first, followed by an AAh, before setting the WR bit of the PMCON1 register. The value written to this register is used to unlock the writes. There are specific timing requirements on these writes.

TABLE 10-3: SUMMARY OF REGISTERS ASSOCIATED WITH FLASH PROGRAM MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
PMCON1	— ⁽¹⁾	CFG5	LWLO	FREE	WRERR	WREN	WR	RD	86
PMCON2	Program Memory Control Register 2								87
PMADRL	PMADRL<7:0>								85
PMADRH	— ⁽¹⁾	PMADRH<6:0>							85
PMDATL	PMDATL<7:0>								85
PMDATH	—	—	PMDATH<5:0>						85
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	60

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by Flash Program Memory module.

Note 1: Unimplemented, read as '1'.

TABLE 10-4: SUMMARY OF CONFIGURATION WORD WITH FLASH PROGRAM MEMORY

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	—	—	—	—	CLKOUTEN	BOREN<1:0>		—	34
	7:0	CP	MCLRE	PWRTE	WDTE<1:0>		—	FOSC<1:0>		
CONFIG2	13:8	—	—	LVP	DEBUG	LPBOR	BORV	STVREN	—	35
	7:0	—	—	—	—	—	—	WRT<1:0>		

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

11.1 PORTA Registers

PORTA is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA (Register 11-2). 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). The exception is RA3, which is input only and its TRIS bit will always read as '1'. Example 11-1 shows how to initialize PORTA.

Reading the PORTA register (Register 11-1) 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 11-2) 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'.

11.1.1 ANSELA REGISTER

The ANSELA register (Register 11-4) 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.

11.1.2 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 11-2.

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 11-2.

TABLE 11-2: PORTA OUTPUT PRIORITY

Pin Name	Function Priority ⁽¹⁾
RA0	SEG12 (LCD) AN0 RA0
RA1	SEG7 AN1 RA1
RA2	COM2 AN2 RA2
RA3	VREF+ COM3 SEG15 AN3 RA3
RA4	SEG4 T0CKI RA4
RA5	SEG6 AN5 RA5
RA6	CLKOUT SEG1 RA6
RA7	CLKIN SEG2 RA7

Note 1: Priority listed from highest to lowest.

PIC16LF1902/3

11.4 PORTE Registers

RE3 is input only, and also functions as $\overline{\text{MCLR}}$. The $\overline{\text{MCLR}}$ feature can be disabled via a configuration fuse. RE3 also supplies the programming voltage. The TRIS bit for RE3 (TRISE3) always reads '1'.

11.4.1 PORTE FUNCTIONS AND OUTPUT PRIORITIES

No output priorities, RE3 is an input only pin.

REGISTER 11-13: PORTE: PORTE REGISTER

U-0	U-0	U-0	U-0	R-x/u	U-0	U-0	U-0
—	—	—	—	RE3	—	—	—
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 7-4 **Unimplemented:** Read as '0'

bit 3 **RE3:** PORTE Input Pin bit

1 = Port pin is > V_{IH}

0 = Port pin is < V_{IL}

bit 2-0 **Unimplemented:** Read as '0'

REGISTER 11-14: TRISE: PORTE TRI-STATE REGISTER

U-0	U-0	U-0	U-0	U-1 ⁽¹⁾	U-0	U-0	U-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 7-4 **Unimplemented:** Read as '0'

bit 3 **Unimplemented:** Read as '1'

bit 2-0 **Unimplemented:** Read as '0'

Note 1: Unimplemented, read as '1'.

12.0 INTERRUPT-ON-CHANGE

The PORTB pins can be configured to operate as Interrupt-on-Change (IOC) pins. An interrupt can be generated by detecting a signal that has either a rising edge or a falling edge. Any individual PORTB pin, or combination of PORTB pins, can be configured to generate an interrupt. The interrupt-on-change module has the following features:

- Interrupt-on-Change enable (Master Switch)
- Individual pin configuration
- Rising and falling edge detection
- Individual pin interrupt flags

Figure 12-1 is a block diagram of the IOC module.

12.1 Enabling the Module

To allow individual PORTB pins to generate an interrupt, the IOCIE bit of the INTCON register must be set. If the IOCIE bit is disabled, the edge detection on the pin will still occur, but an interrupt will not be generated.

12.2 Individual Pin Configuration

For each PORTB pin, a rising edge detector and a falling edge detector are present. To enable a pin to detect a rising edge, the associated IOCBP_x bit of the IOCBP register is set. To enable a pin to detect a falling edge, the associated IOCBN_x bit of the IOCBN register is set.

A pin can be configured to detect rising and falling edges simultaneously by setting both the IOCBP_x bit and the IOCBN_x bit of the IOCBP and IOCBN registers, respectively.

12.3 Interrupt Flags

The IOCBF_x bits located in the IOCBF register are status flags that correspond to the interrupt-on-change pins of PORTB. If an expected edge is detected on an appropriately enabled pin, then the status flag for that pin will be set, and an interrupt will be generated if the IOCIE bit is set. The IOCIF bit of the INTCON register reflects the status of all IOCBF_x bits.

12.4 Clearing Interrupt Flags

The individual status flags, (IOCBF_x bits), can be cleared by resetting them to zero. If another edge is detected during this clearing operation, the associated status flag will be set at the end of the sequence, regardless of the value actually being written.

In order to ensure that no detected edge is lost while clearing flags, only AND operations masking out known changed bits should be performed. The following sequence is an example of what should be performed.

EXAMPLE 12-1:

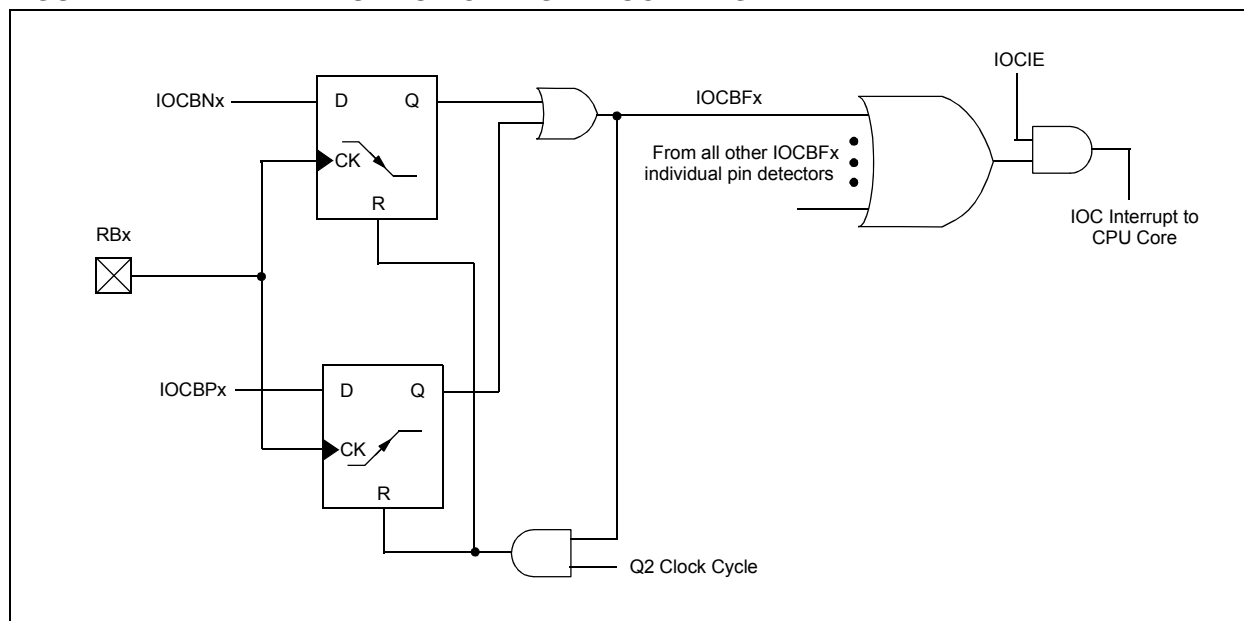
```
MOVLW 0xff
XORWF IOCBF, W
ANDWF IOCBF, F
```

12.5 Operation in Sleep

The interrupt-on-change interrupt sequence will wake the device from Sleep mode, if the IOCIE bit is set.

If an edge is detected while in Sleep mode, the IOCBF register will be updated prior to the first instruction executed out of Sleep.

FIGURE 12-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM



PIC16LF1902/3

TABLE 17-5: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	60
PIE1	TMR1GIE	ADIE	—	—	—	—	—	TMR1IE	61
PIR1	TMR1GIF	ADIF	—	—	—	—	—	TMR1IF	63
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								126*
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								126*
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	96
T1CON	TMR1CS1	TMR1CS0	T1CKPS<1:0>		T1OSCEN	T1SYNC	—	TMR1ON	130
T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/ DONE	T1GVAL	T1GSS1	T1GSS0	131

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by the Timer1 module.

* Page provides register information.

18.4 LCD Bias Internal Reference Ladder

The internal reference ladder can be used to divide the LCD bias voltage two or three equally spaced voltages that will be supplied to the LCD segment pins. To create this, the reference ladder consists of three matched resistors. Refer to Figure 18-3.

18.4.1 BIAS MODE INTERACTION

When in 1/2 Bias mode (BIASMD = 1), then the middle resistor of the ladder is shorted out so that only two voltages are generated. The current consumption of the ladder is higher in this mode, with the one resistor removed.

TABLE 18-3: LCD INTERNAL LADDER POWER MODES (1/3 BIAS)

Power Mode	Nominal Resistance of Entire Ladder	Nominal I _{DD}
Low	3 Mohm	1 μ A
Medium	300 kohm	10 μ A
High	30 kohm	100 μ A

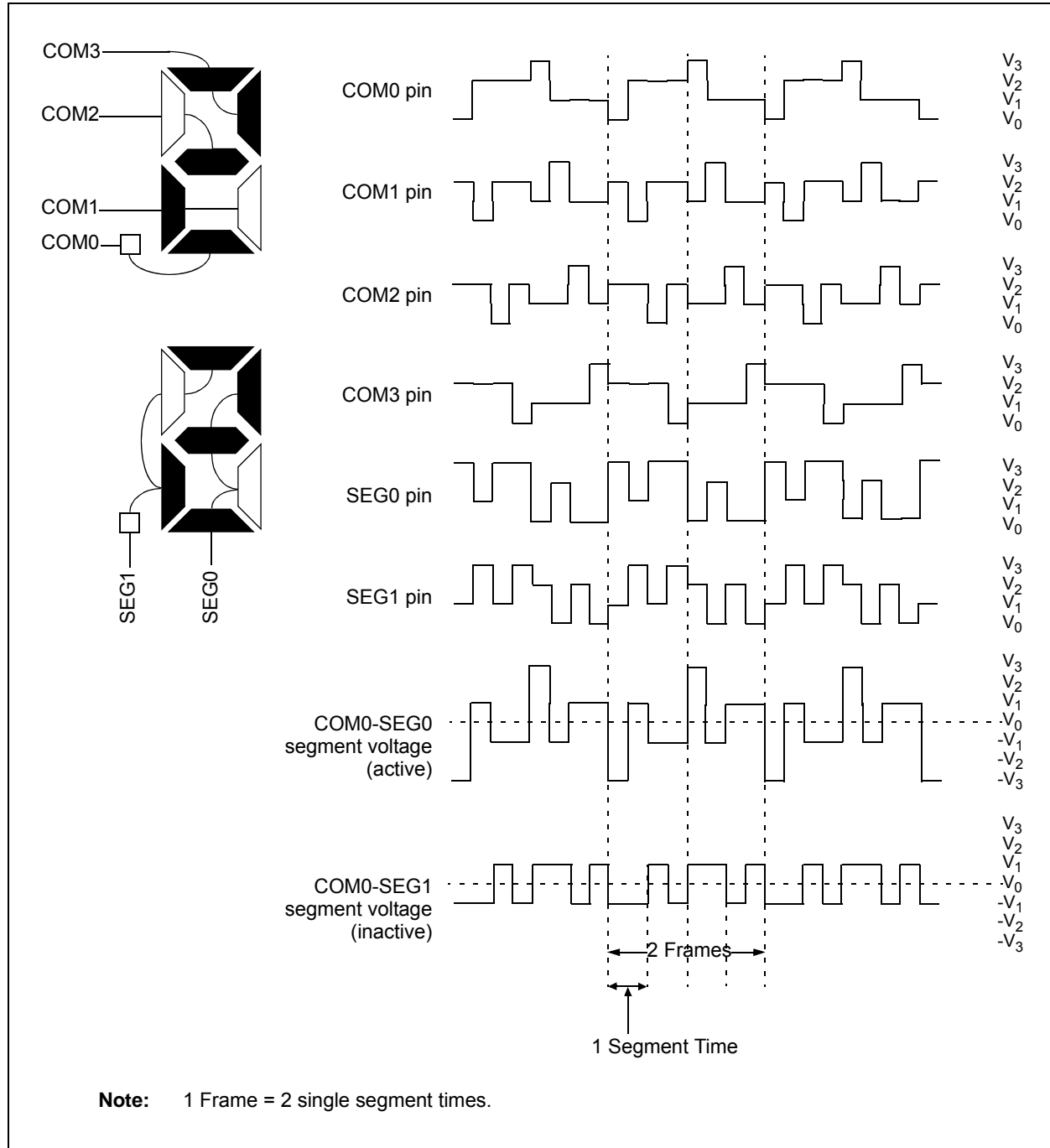
18.4.2 POWER MODES

The internal reference ladder may be operated in one of three power modes. This allows the user to trade off LCD contrast for power in the specific application. The larger the LCD glass, the more capacitance is present on a physical LCD segment, requiring more current to maintain the same contrast level.

Three different power modes are available, LP, MP and HP. The internal reference ladder can also be turned off for applications that wish to provide an external ladder or to minimize power consumption. Disabling the internal reference ladder results in all of the ladders being disconnected, allowing external voltages to be supplied.

Whenever the LCD module is inactive (LCDA = 0), the internal reference ladder will be turned off.

FIGURE 18-18: TYPE-B WAVEFORMS IN 1/4 MUX, 1/3 BIAS DRIVE



18.12 Configuring the LCD Module

The following is the sequence of steps to configure the LCD module.

1. Select the frame clock prescale using bits LP<3:0> of the LCDPS register.
2. Configure the appropriate pins to function as segment drivers using the LCDSEn registers.
3. Configure the LCD module for the following using the LCDCON register:
 - Multiplex and Bias mode, bits LMUX<1:0>
 - Timing source, bits CS<1:0>
 - Sleep mode, bit SLPEN
4. Write initial values to pixel data registers, LCD-DATA0 through LCDDATA21.
5. Clear LCD Interrupt Flag, LCDIF bit of the PIR2 register and if desired, enable the interrupt by setting bit LCDIE of the PIE2 register.
6. Configure bias voltages by setting the LCDRL, LCDREF and the associated ANSELx registers as needed.
7. Enable the LCD module by setting bit LCDEN of the LCDCON register.

18.13 Disabling the LCD Module

To disable the LCD module, write all '0's to the LCDCON register.

18.14 LCD Current Consumption

When using the LCD module the current consumption consists of the following three factors:

- Oscillator Selection
- LCD Bias Source
- Capacitance of the LCD segments

The current consumption of just the LCD module can be considered negligible compared to these other factors.

18.14.1 OSCILLATOR SELECTION

The current consumed by the clock source selected must be considered when using the LCD module. See **Section 21.0 “Electrical Specifications”** for oscillator current consumption information.

18.14.2 LCD BIAS SOURCE

The LCD bias source, internal or external, can contribute significantly to the current consumption. Use the highest possible resistor values while maintaining contrast to minimize current.

18.14.3 CAPACITANCE OF THE LCD SEGMENTS

The LCD segments which can be modeled as capacitors which must be both charged and discharged every frame. The size of the LCD segment and its technology determines the segment's capacitance.

PIC16LF1902/3

DECFSZ Decrement f, Skip if 0

Syntax: [*label*] DECFSZ f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(f) - 1 \rightarrow (\text{destination})$;
 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.

INCFSZ Increment f, Skip if 0

Syntax: [*label*] INCFSZ f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(f) + 1 \rightarrow (\text{destination})$,
 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.

GOTO Unconditional Branch

Syntax: [*label*] GOTO k

Operands: $0 \leq k \leq 2047$

Operation: $k \rightarrow PC<10:0>$
 $PCLATH<6:3> \rightarrow PC<14:11>$

Status Affected: None

Description: GOTO is an unconditional branch. The 11-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.

IORLW Inclusive OR literal with W

Syntax: [*label*] IORLW k

Operands: $0 \leq k \leq 255$

Operation: $(W) .OR. k \rightarrow (W)$

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.

INCF Increment f

Syntax: [*label*] INCF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(f) + 1 \rightarrow (\text{destination})$

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'.

IORWF Inclusive OR W with f

Syntax: [*label*] IORWF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(W) .OR. (f) \rightarrow (\text{destination})$

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'.

22.0 DC AND AC CHARACTERISTICS GRAPHS AND CHARTS

Graphs and charts are not available at this time.

23.0 DEVELOPMENT SUPPORT

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

- Integrated Development Environment
 - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
 - MPLAB XC Compiler
 - MPASM™ Assembler
 - MPLINK™ Object Linker/
MPLIB™ Object Librarian
 - MPLAB Assembler/Linker/Librarian for
Various Device Families
- Simulators
 - MPLAB X SIM Software Simulator
- Emulators
 - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
 - MPLAB ICD 3
 - PICKit™ 3
- Device Programmers
 - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards,
Evaluation Kits and Starter Kits
- Third-party development tools

23.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

23.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

23.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

23.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

23.9 PICkit 3 In-Circuit Debugger/Programmer

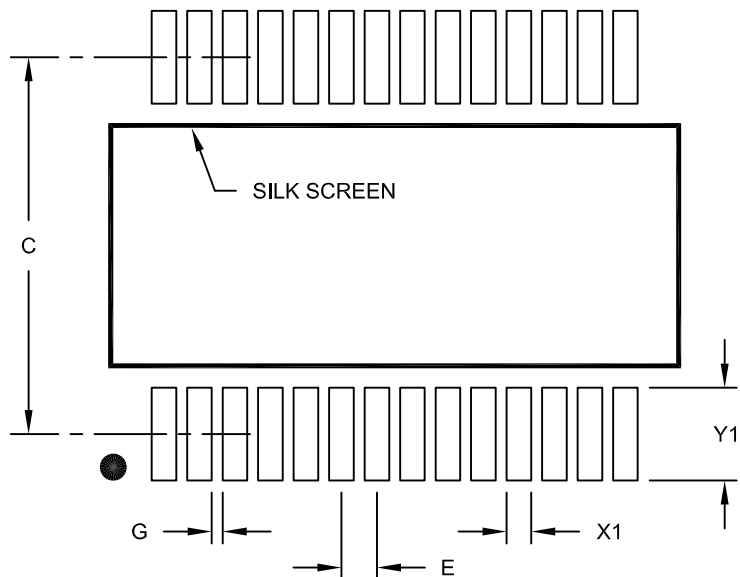
The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

23.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Contact Pad Spacing	C		7.20	
Contact Pad Width (X28)	X1			0.45
Contact Pad Length (X28)	Y1			1.75
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

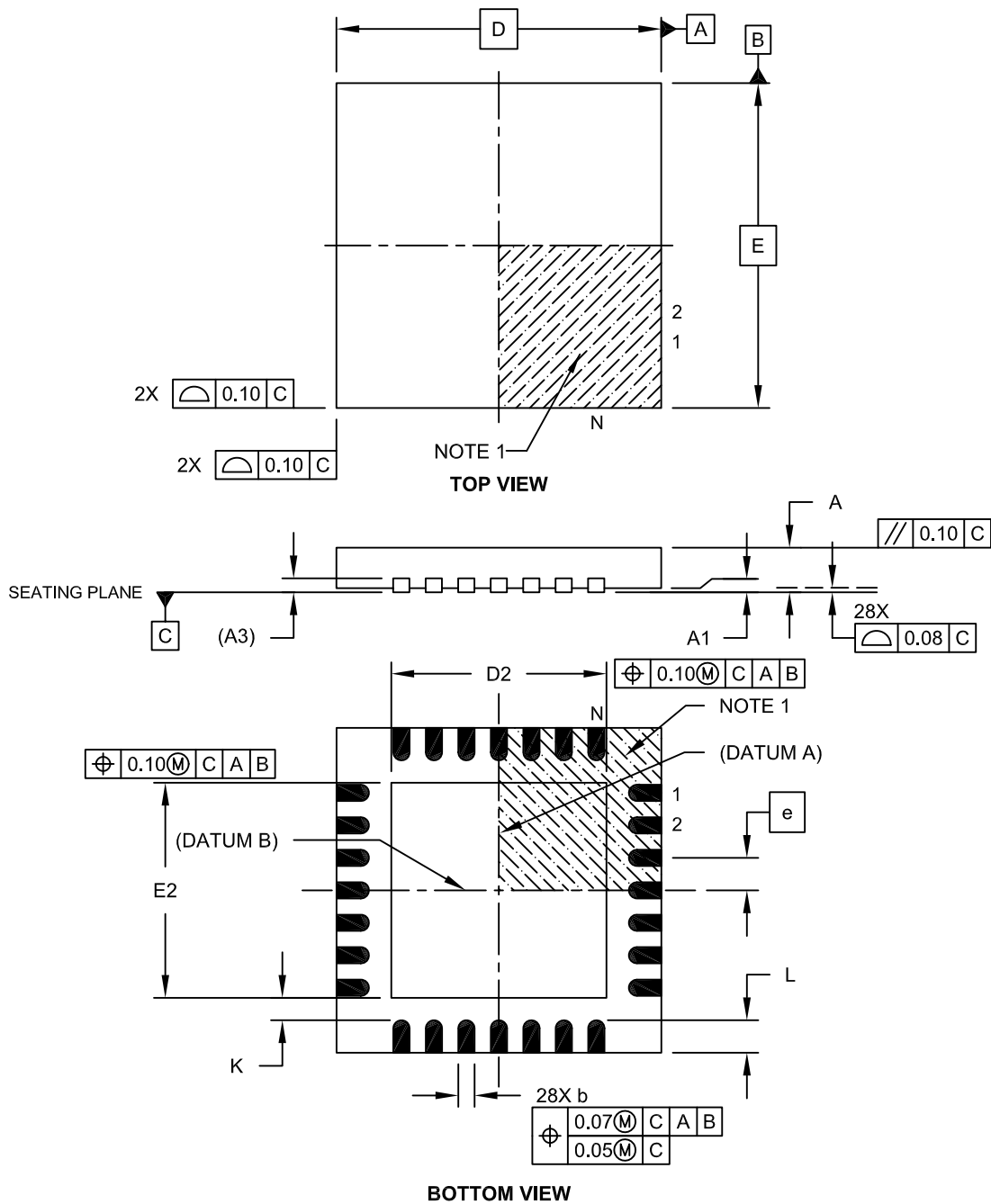
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

PIC16LF1902/3

28-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) – 4x4x0.5 mm Body [UQFN]

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



Microchip Technology Drawing C04-152A Sheet 1 of 2