



Microchip Technology

## **Introductory - PIC 101**

### **Introduction to PIC® MCUs Hands-on Workshop**

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

### **In this workshop you will have a chance to:**

- Explore different application areas where microcontrollers are used
- Develop a real world working application
- Work with Microchip's development tools
  - PICSTART Plus programmer
  - MPLAB software/application development manager
  - Workshop board

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## What you will do...*continued*

- Develop application code
- Test your design
  - Workshop board
  - Scale model prototype
- Develop higher order functions
- Evaluate your design on a workshop board

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Agenda Day 1

- Review of architecture and instruction set
- Break
- Basic tools usage
- Lunch
- Switching
- Timing
- Break
- Timing (*continued*)
- Motor/motion control

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Agenda Day 2

- Motor/motion control *(continued)*
- Sensing inputs
- Break
- Sensing inputs *(continued)*
- State logic and higher order functions
- Lunch
- State logic and higher order functions *(continued)*
- Break
- Proof on scale model prototype

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Why Microchip?

- High reliability
- Feature creep solution
- Speed of implementation
- Development tools suite
- Lower cost
- Field upgrades

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Final project goal

- Automatic door controller (commercial type)



© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## What feature set?

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## What feature set

- Open/close
- Switch controlled
- Held open to allow passage through door
- Automatic close (store type)
- Safety cutoffs (if time permits)
- Independent motor cutoff (if time permits)
- Double tap (if time permits)
- Lighting control (optional)
- Sound control (optional)

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Microcontroller applications involved

- Switching
- Timing
- Movement/motor control
- Sensing
- State decisions/logic

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC 101

# Architecture

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture RISC-like Features

**The high performance of the PIC MCU family can be attributed to the following architectural features:**

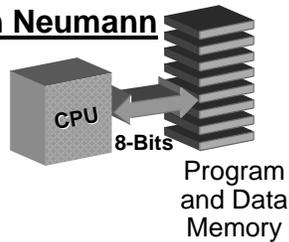
- Harvard architecture
- Register file concept
- All instructions single-word
- LWI (Long Word Instruction)
- Instruction pipelining
- Single-cycle instructions
- Reduced instruction set
- Orthogonal instruction set

© 1999 Microchip Technology Incorporated. All Rights Reserved.



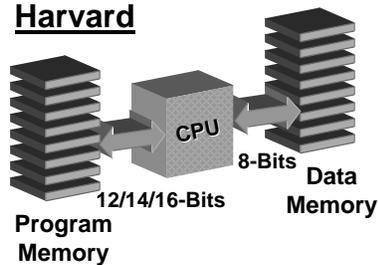
## PIC Architecture Harvard Architecture

### Von Neumann



- Fetches instructions and data from one memory.
  - Limits Operating Bandwidth
- Two separate memory spaces for instructions and data.
  - Increases throughput
  - Different program and data bus widths are possible

### Harvard



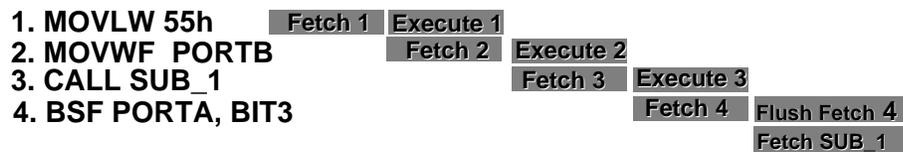
© 1999 Microchip Technology Incorporated. All Rights Reserved.



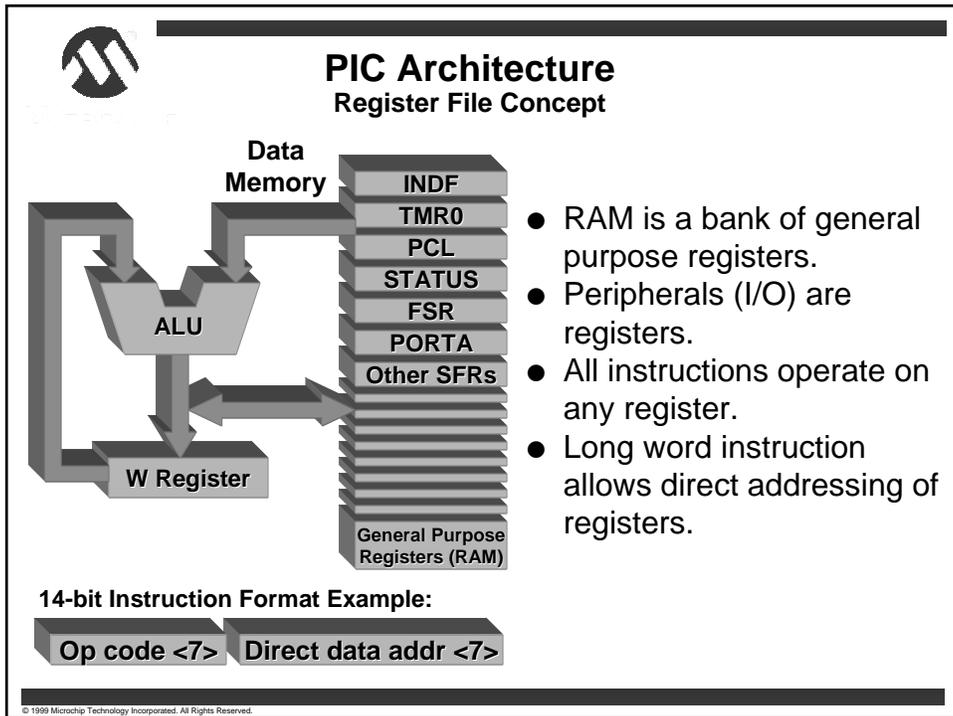
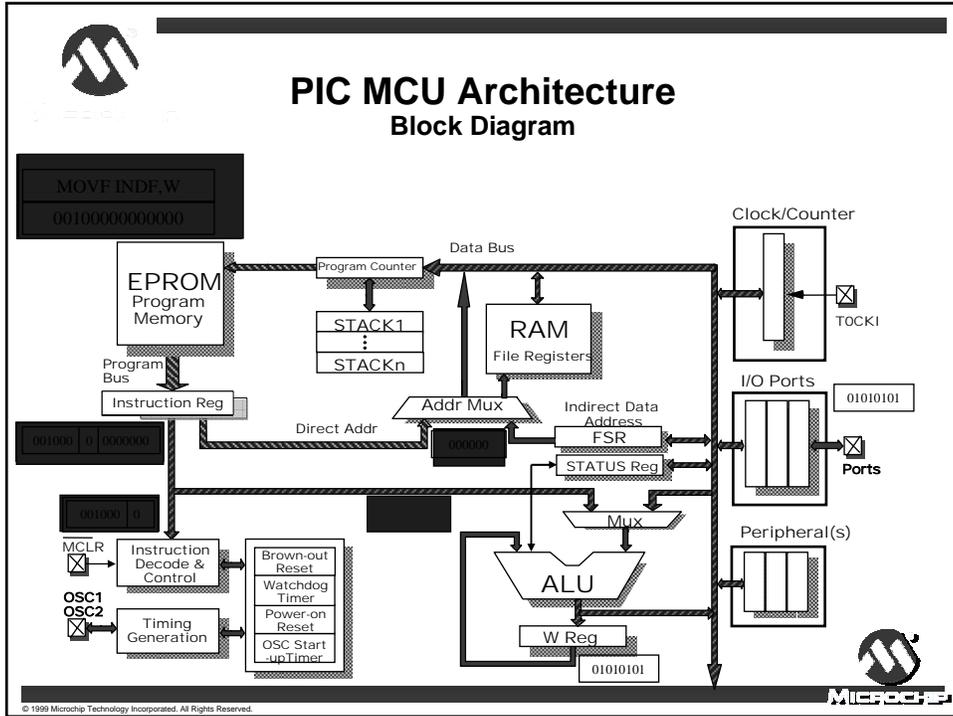
## PIC MCU Architecture Pipelining

- In most microcontrollers, instructions are fetched and executed sequentially.
- Allows overlap of fetch and execution.
- Makes single cycle execution.
- Program branches (e.g. GOTO, CALL or Write to PC) takes two cycles.

Tcy0 Tcy1 Tcy2 Tcy3 Tcy4



© 1999 Microchip Technology Incorporated. All Rights Reserved.





Microchip Technology

## Quiz 1

- 1) All of Microchip's PIC instructions are \_\_\_\_.
- Single Cycle
  - Single Byte
  - Single Word
- 2) The PIC pipeline must be flushed whenever \_\_\_\_ is modified.
- W register
  - PortB
  - PC

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## PIC Architecture

# Memory

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture Memory

- 2 types of memory
  - Program
  - Data
- Organization
  - Pages (program)
  - Banks (data)

© 1999 Microchip Technology Incorporated. All Rights Reserved.



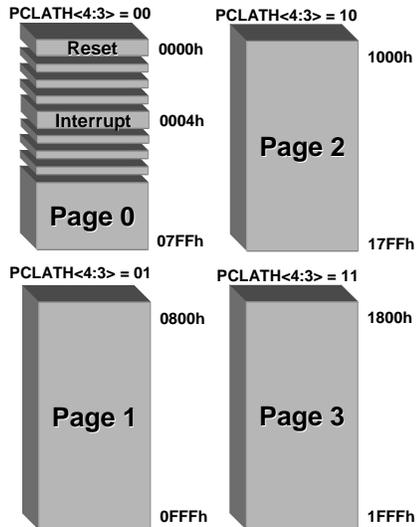
## PIC Architecture Paging / Banking Boundary Summary

PIC16/17 Family	Word Size (Bits)	Code Page Boundary (Words)	Current Max code Size (Words)	Register Bank Boundary (Bytes)	Current Max RAM Size (Bytes)
16F5X	12	512	2K	32	73
16FXXX	14	2048	8K	128	368
17CXXX	16	8192	16K/64K	256	902

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC16FXX Architecture Program Memory



- Maximum 8K Bytes (13 bits) of program memory space
- Four Pages each 2K bytes (11 bits)
- Page access using PCLATH<4:3>
- Reset Vector at 0000h
- Interrupt Vector at 0004h

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture Paging program memory

- Paging needs consideration ONLY when executing program CALL or GOTO operation
- Modify page bits ONLY when jumping to different page than last jumped
  - page bits define desired page execution
- Instructions invoking page bits:
  - GOTO <address>
  - CALL <address>
  - <Instruction> PCL,F ; e.g. ADDWF PCL,F
- There are no paging considerations when returning

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture Paging program memory

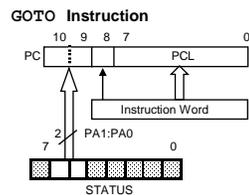
- ADDWF PCL,F has eight bits (256 words) of address
  - In PIC16F5X PA0, PA1 determine final page
    - Bit 8 of PC is cleared by hardware
    - Destination address must be first 256 of page
  - In PIC16FXXX/17CXXX, PCLATH is used for upper address
    - Destination address can be anywhere
    - Look-up tables that cross a 256 word boundary require pre-conditioning of PCLATH

© 1999 Microchip Technology Incorporated. All Rights Reserved.

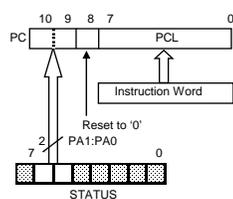


## PIC Architecture Paging program memory

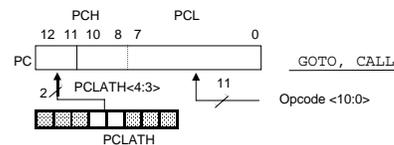
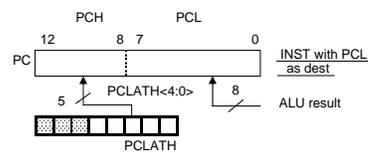
### PIC16F57



### CALL or Modify PCL Instruction



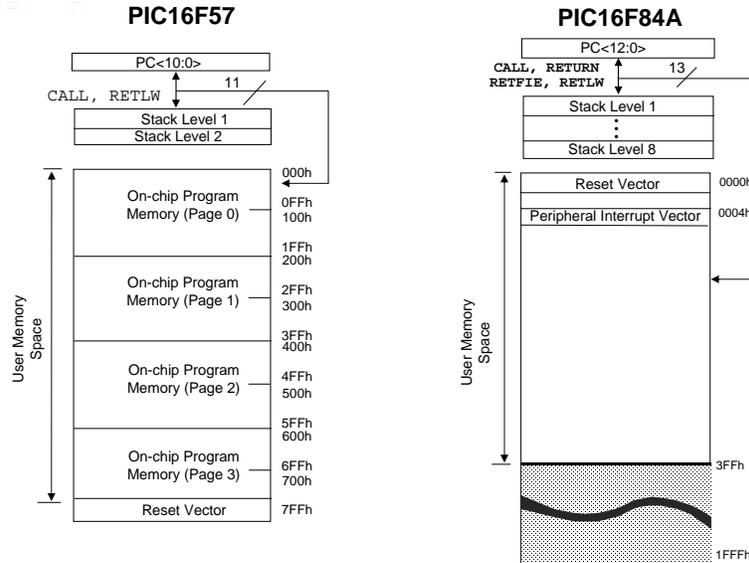
### PIC16F84A



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Program Memory Map



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture

### Program Memory: Immediate Operation

- 8-bit constant (literal) value included in instruction word.
- Used by literal instructions such as movlw, addlw, retlw, etc.

#### 14-bit Instruction for Literal Instructions



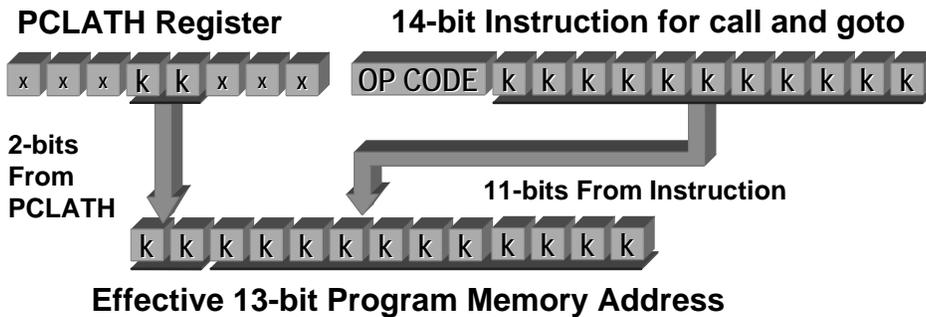
© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture

### Program Memory: PC Absolute Addressing

- Used by control instructions CALL and GOTO to modify the PC (Program Counter)



© 1999 Microchip Technology Incorporated. All Rights Reserved.

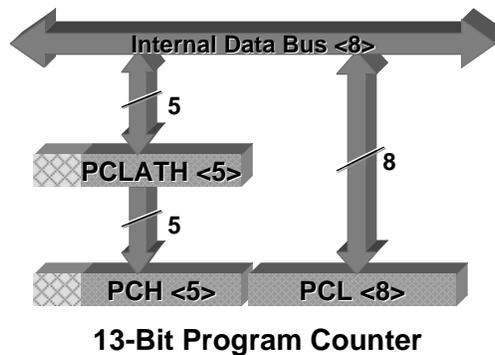


## PIC Architecture

### PC Relative Addressing (14-bit core)

- First write high byte to PCLATH.
- Next write low byte to PCL, this loads the entire 13-bit value to PC.

movlw HIGH Delay  
movwf PCLATH  
movlw LOW Delay  
movwf PCL



Note: PCH cannot be read

© 1999 Microchip Technology Incorporated. All Rights Reserved.



# PIC MCU Architecture

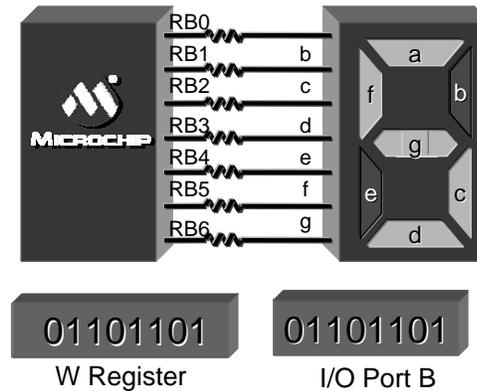
## PC Relative Addressing

```

movlw HIGH Decode
movwf PCLATH
movf  DisplayValue,W
call   Decode
movwf PORTB
goto  Continue
Decode
addwf PCL,F
retlw B'00111111';decode 0
retlw B'00000110';decode 1
retlw B'01011011';decode 2
retlw B'01001111';decode 3
retlw B'01100110';decode 4
retlw B'01101101';decode 5
retlw B'01111101';decode 6
retlw B'00000111';decode 7
retlw B'01111111';decode 8
retlw B'01101111';decode 9
Continue

```

- Look-up Table Example



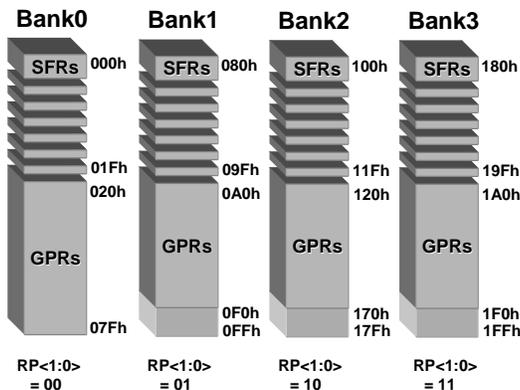
Continue

© 1999 Microchip Technology Incorporated. All Rights Reserved.



# PIC16FXX Architecture

## Data Memory



- Four banks each of 128 bytes of Data Memory
- Special Function Registers (SFRs) are mapped in top 32 locations
- Banks selected by RP0,1 and IRP in Status register

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture

### Methods for accessing data memory

- Direct
- Indirect

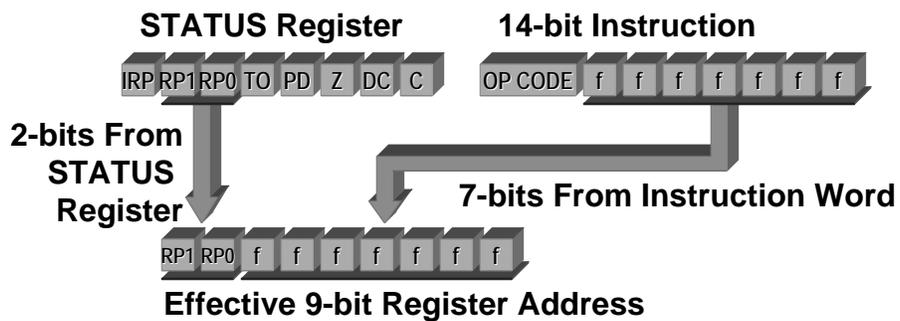
© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture

### Data Memory: Direct Addressing

- 7-bit direct address from the instruction
- 2-bits from STATUS register



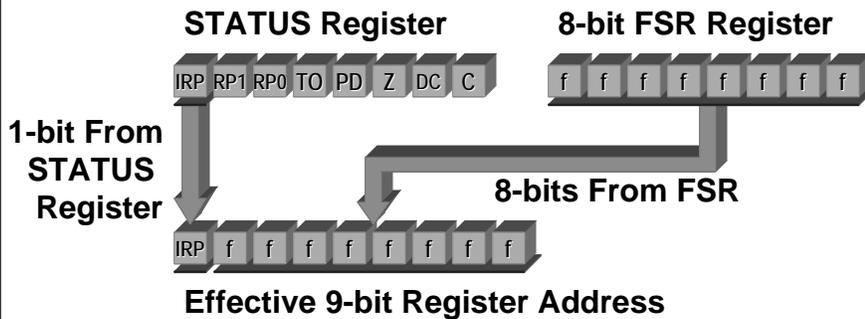
© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture

### Data Memory: Indirect Addressing

- 8-bit indirect address from the FSR (File Select Register).
- 1-bit from STATUS register.



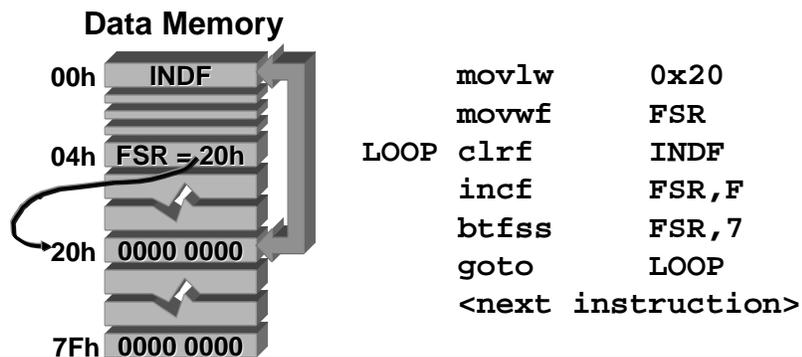
© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Architecture

### Data Memory: Indirect Addressing

- Clear all RAM locations from 0x20 to 0x7F.
- Indirect address is loaded into FSR.
- Every time INDF is used as operand, register pointed to by FSR is actually used.



© 1999 Microchip Technology Incorporated. All Rights Reserved.





Microchip Technology

## Quiz2 (cont.)

- 4) For the 12-bit core, what SFR register and respective bits are needed to access different pages in program memory ?  \_\_\_\_\_  
\_\_\_\_\_
- 5) For the 14-bit core, what SFR register and respective bits are needed to access different pages in program memory?  \_\_\_\_\_  
\_\_\_\_\_
- 6) What bits are needed to access different RAM banks?  1) \_\_\_\_\_  
2) \_\_\_\_\_  
3) \_\_\_\_\_  
4) \_\_\_\_\_

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## PIC MCUs

# Instruction Set

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

- 33 instructions → 12-bit core
- 35 instructions → 14-bit core
- 58 instructions → 16-bit core
  - Easy to learn
  - High compaction
  - Very powerful single-word instructions
  - All single-cycle except program branches
  - Upward compatibility of instructions

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Instruction Set 14-bit core Summary

Byte-Oriented Operations			Bit-Oriented Operations		
NOP	-	No Operation	BCF	f,b	Bit clear f
MOVWF	f	Move W to f	BSF	f,b	Bit set f
CLRWF	-	Clear W	BTFSC	f,b	Bit test f, skip if clear
CLRF	f	Clear f	BTFSS	f,b	Bit test f, skip if set
SUBWF	f,d	Subtract W from f	<b>Literal and Control Operations</b>		
DECF	f,d	Decrement f	SLEEP	-	Go into standby mode
IORWF	f,d	Inclusive OR W and f	CLRWDTC	-	Clear watchdog timer
ANDWF	f,d	AND W and f	RETLW	k	Return, place literal in W
XORWF	f,d	Exclusive OR W and f	RETFIE	-	Return from interrupt
ADDWF	f,d	Add W and f	RETURN	-	Return from subroutine
MOVF	f,d	Move f	CALL	k	Call subroutine
COMF	f,d	Complement f	GOTO	k	Go to address (k is 9-bit)
INCF	f,d	Increment f	MOVLW	k	Move literal to W
DECFSZ	f,d	Decrement f, skip if zero	IORLW	k	Inclusive OR literal with W
RRF	f,d	Rotate right f through carry	ADDLW	k	Add literal with W
RLF	f,d	Rotate left f through carry	SUBLW	k	Subtract W from literal
SWAPF	f,d	Swap nibbles of f	ANDLW	k	AND literal with W
INCFSSZ	f,d	Increment f, skip if zero	XORLW	k	Exclusive OR literal with W

f = File Register, k = literal value (8-bit), b = bit address <0,7>, d = destination (0=W, 1=f)

© 1999 Microchip Technology Incorporated. All Rights Reserved.

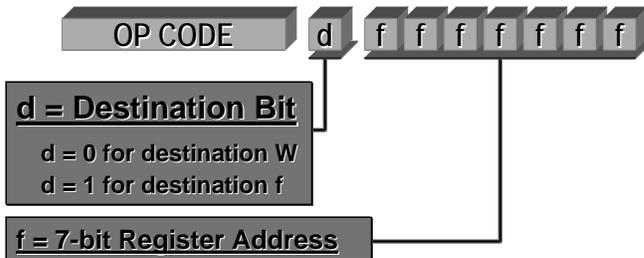


## PIC Instruction Set

### Byte-Oriented Operations

Byte-Oriented Operations	
NOP	-
MOVWF	f
CLRWF	-
CLRF	f
SUBWF	f,d
DECWF	f,d
IORWF	f,d
ANDWF	f,d
XORWF	f,d
ADDWF	f,d
MOVF	f,d
COMF	f,d
INCF	f,d
DECFSZ	f,d
RRF	f,d
RLF	f,d
SWAPF	f,d
INCFSZ	f,d

#### 14-bit Instruction for Byte Oriented Operations



Example:  
**ADDWF**    **REG, W**  
*ADDWF*    *f, d*

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

NOP    No Operation  
 Syntax: NOP  
 Operands:        None  
 Operation:        No operation  
 Status: None  
 Encoding:        00 0000 0000 0000  
 Words: 1  
 Cycles: 1

- Example:  
*NOP*

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**MOVWF** Move W to f

Syntax: **MOVWF** f

Operands: 0 <= f <= 127

Operation: (W) -> (f)

Status: None

Encoding: 00 0000 1fff fff

Words: 1

Cycles: 1

- Example:

**MOVWF** FSR

Before Instruction

FSR = 0xFF

W = 0x4F

After Instruction

FSR = 0x4F

W = 0x4F

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**CLRW** Clear W

Syntax: **CLRW**

Operands: None

Operation: 00h -> (W)

1 -> Z

Status: Z

Encoding: 00 0001 0000 0000

Words: 1

Cycles: 1

- Example:

**CLRW**

Before Instruction

W = 0x4F

After Instruction

W = 0x00

Z = 1

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**SUBWF** Subtract W from f

Syntax: **SUBWF** f,d

Operands:  $0 \leq f \leq 127$   
 $d = \{0,1\}$

Operation:  $(f) - (W) \rightarrow \text{dest}$

Status: C,DC,Z

Encoding: 00 0010 dfff ffff

Words: 1

Cycles: 1

- Example:

**SUBWF** FSR,W

Before Instruction

FSR = 0x03

W = 0x02

C = ?

Z = ?

After Instruction

FSR = 0x03

W = 0x01

C = 0

Z = 0

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**DECf** Decrement f

Syntax: **DECf** f,d

Operands:  $0 \leq f \leq 127$   
 $d = \{0,1\}$

Operation:  $(f) - 1 \rightarrow \text{dest}$

Status: Z

Encoding: 00 0011 dfff ffff

Words: 1

Cycles: 1

- Example:

**DECf** FSR,F

Before Instruction

FSR = 0x01

Z = 0

After Instruction

FSR = 0x00

Z = 1

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**ANDWF** AND W with f

Syntax: **ANDWF** f,d

Operands:  $0 \leq f \leq 127$

$d = \{0,1\}$

Operation: (W).AND.(f) -> dest

Status: Z

Encoding: 00 0101 dfff ffff

Words: 1

Cycles: 1

- Example:

**ANDWF** CNT,F

Before Instruction

CNT = B'00010111'

W = B'11000010'

After Instruction

CNT = B'00000010'

W = B'11000010'

Z = 0

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**XORWF** Exclusive OR W & f

Syntax: **XORWF** f,d

Operands:  $0 \leq f \leq 127$

$d = \{0,1\}$

Operation: (W).XOR.(f) -> dest

Status: Z

Encoding: 00 0110 dfff ffff

Words: 1

Cycles: 1

- Example:

**XORWF** FSR,W

Before Instruction

FSR = B'10101111'

W = B'10110101'

After Instruction

FSR = B'10101111'

W = B'00011010'

Z = 0

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**MOVF** Move f

Syntax: MOVF f,d

Operands: 0 <= f <= 127  
d = {0,1}

Operation: (f) -> dest

Status: Z

Encoding: 00 1000 dfff ffff

Words: 1

Cycles: 1

- Example:

*MOVF FSR,F*

Before Instruction

FSR = 0x00

Z = 0

After Instruction

FSR = 0x00

Z = 1

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**INCF** Increment f

Syntax: INCF f,d

Operands: 0 <= f <= 127  
d = {0,1}

Operation: (f) + 1 -> dest

Status: Z

Encoding: 00 1010 dfff ffff

Words: 1

Cycles: 1

- Example:

*INCF REG,F*

Before Instruction

REG = 0xFF

Z = 0

After Instruction

REG = 0x00

Z = 1

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**DECFSZ**      Dec. f, Skip if 0  
**Syntax:** DECFSZ f,d  
**Operands:**    0 <= f <= 127  
                  d = {0,1}  
**Operation:**    (f) - 1 -> dest  
                  skip if result = 0  
**Status:**      None  
**Encoding:**    00 1011 dfff ffff  
**Words:**      1  
**Cycles:**      1(2)

**Example:**  
*Loop*    DECFSZ CNT,F  
          GOTO    *Loop*  
*Continue*  
Before Instruction  
PC = address *Loop*  
After Instruction  
CNT        = CNT - 1  
if CNT     = 0,  
PC = address *Continue*  
if CNT != 0,  
PC = address *Loop*+1

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

**SWAPF**      Swap Nibbles in f  
**Syntax:**      SWAPF f,d  
**Operands:**    0 <= f <= 127  
                  d = {0,1}  
**Operation:**    f<3:0> -> dest<7:4>  
                  f<7:4> -> dest<3:0>  
**Status:**      None  
**Encoding:**    00 1110 dfff ffff  
**Words:**      1  
**Cycles:**      1

**Example:**  
*SWAPF*    *REG,W*  
Before Instruction  
REG        = 0xA5  
After Instruction  
REG        = 0xA5  
W          = 0x5A

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Byte-Oriented Operations

INCFSZ Inc. f, Skip if 0

Syntax: INCFSZ f,d

Operands: 0 <= f <= 127  
d = {0,1}

Operation: (f) + 1 -> dest  
skip if result = 0

Status: None

Encoding: 00 1111 dfff ffff

Words: 1

Cycles: 1(2)

Example:

```
Loop  INCFSZ CNT,F
      GOTO  Loop
```

*Continue*

Before Instruction

PC = address *Loop*

After Instruction

CNT = CNT + 1

if CNT = 0,

PC = address *Continue*

if CNT != 0,

PC = address *Loop*+1

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Bit-Oriented Operations

#### Bit-Oriented Operations

BCF f,b  
BSF f,b  
BTFSC f,b  
BTFSS f,b

#### 14-bit Instruction for Bit Oriented Operations

OP CODE    b b b    f f f f f f f f

b = 3-Bit Address  
(Bit Number)

f = 7-bit Register Address

Example:

```
BTFSC    STATUS, C
BTFSC    f, b
```

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Bit-Oriented Operations

**BCF** Bit Clear f

Syntax: BCF f,b

Operands: 0 <= f <= 127  
0 <= b <= 7

Operation: 0 -> (f<b>)

Status: None

Encoding: 01 00bb bfff ffff

Words: 1

Cycles: 1

- Example:

*BCF FSR,4*

Before Instruction

FSR = 0011 0000

After Instruction

FSR = 0010 0000

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Bit-Oriented Operations

**BSF** Bit Set f

Syntax: BSF f,b

Operands: 0 <= f <= 127  
0 <= b <= 7

Operation: 1 -> (f<b>)

Status: None

Encoding: 01 01bb bfff ffff

Words: 1

Cycles: 1

- Example:

*BSF FSR,4*

Before Instruction

FSR = 0010 0000

After Instruction

FSR = 0011 0000

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Bit-Oriented Operations

**BTFSK** Bit Test f, Skip if 0

Syntax: **BTFSK** f,b

Operands:  $0 \leq f \leq 127$   
 $0 \leq b \leq 7$

Operation: skip if (f<b>) = 0

Status: None

Encoding: 01 10bb bfff ffff

Words: 1

Cycles: 1(2)

Example:

*Here* **BTFSK** CNT,1  
*False* **GOTO** Done  
*True*

Before Instruction

PC = address *Here*

After Instruction

if CNT<1> = 0,

PC = address *True*

if CNT<1> = 1,

PC = address *False*

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Bit-Oriented Operations

**BTFSK** Bit Test f, Skip if 1

Syntax: **BTFSK** f,b

Operands:  $0 \leq f \leq 127$   
 $0 \leq b \leq 7$

Operation: skip if (f<b>) = 1

Status: None

Encoding: 01 11bb bfff ffff

Words: 1

Cycles: 1(2)

Example:

*Here* **BTFSK** CNT,1  
*False* **GOTO** Done  
*True*

Before Instruction

PC = address *Here*

After Instruction

if CNT<1> = 1,

PC = address *True*

if CNT<1> = 0,

PC = address *False*

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

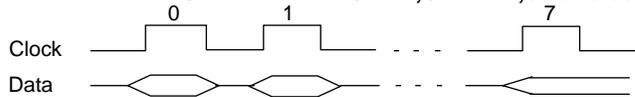
### Example: Bit Manipulation

- Synchronous serial transmission of eight bits of data from file register XDATA to I/O Pin:

```

XMIT:      MOVLW      0x08      ;Bit count = 8
           MOVWF      bit_count

XM_LOOP:   BCF        PORTB,DT  ;preset clock & data lines to 0
           BCF        PORTB,CLK ;preset clock & data lines to 0
           RRF        XDATA,F   ;rotate data right thru Carry
           BTFSC     STATUS,C   ;test carry bit
           BSF        PORTB,DT  ;set 1 → Data pin
           BSF        PORTB,CLK ;set 1 → Clock pin
           DECFSZ    bit_count,F ;decrement count
           GOTO      XM_LOOP    ;Not done then repeat
           BCF        PORTB,CLK ;clear clock line and exit
  
```



© 1999 Microchip Technology Incorporated. All Rights Reserved.



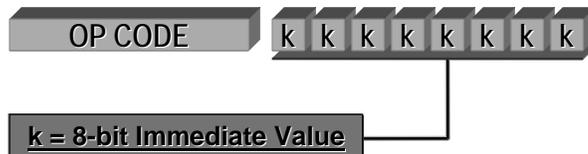
## PIC MCU Instruction Set

### Literal Operations

#### Literal Operations

MOVLW	k
IORLW	k
ADDLW	k
SUBLW	k
ANDLW	k
XORLW	k

#### 14-bit Instruction for Literal Operations



Example:

```

MOVLW 0x2F
MOVLW  k
  
```

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Literal Operations

**MOVLW** Move Literal to W

Syntax: **MOVLW** k

Operands: 0 <= k <= 255

Operation: k -> (W)

Status: None

Encoding: 11 0000 kkkk kkkk

Words: 1

Cycles: 1

- Example:

*MOVLW 0x5A*

After Instruction

W = 0x5A

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Literal Operations

**SUBLW** Subtract W from Literal

Syntax: **SUBLW** k

Operands: 0 <= k <= 255

Operation: k - (W) -> (W)

Status: C,DC,Z

Encoding: 11 1100 kkkk kkkk

Words: 1

Cycles: 1

- Example:

*SUBLW 0x02*

Before Instruction

W = 2

C = ?

Z = ?

After Instruction

W = 0x00

C = 0

Z = 1

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Literal Operations

**ANDLW** AND Literal with W

Syntax: **ANDLW** k

Operands: 0 <= k <= 255

Operation: (W) .AND. k -> (W)

Status: Z

Encoding: 11 1001 kkkk kkkk

Words: 1

Cycles: 1

- Example:

*ANDLW B'01011111'*

Before Instruction

W = B'10100011'

After Instruction

W = B'00000011'

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set

### Literal Operations

**XORLW** Exclusive OR Literal with W

Syntax: **XORLW** k

Operands: 0 <= k <= 255

Operation: (W) .OR. k -> (W)

Status: Z

Encoding: 11 1010 kkkk kkkk

Words: 1

Cycles: 1

- Example:

*XORLW B'10101111'*

Before Instruction

W = B'10110101'

After Instruction

W = B'00011010'

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set Control Operations

Control Operations	
SLEEP	-
CLRWDT	-
RETLW	k
RETFIE	-
RETURN	-
CALL	k
GOTO	k

### 14-bit Instruction for RETLW



k = 8-bit Immediate Value

### 14-bit Instruction for CALL and GOTO



k = 11-bit Immediate Value

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set Control Operations

SLEEP Enter SLEEP mode

Syntax: SLEEP

Operands: None

Operation: 00h -> WDT

1 ->  $\overline{TO}$

0 ->  $\overline{PD}$

Status:  $\overline{TO}$ ,  $\overline{PD}$

Encoding: 00 0000 0110 0011

Words: 1

Cycles: 1

- Example:  
*SLEEP*

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC Instruction Set Control Operations

**CLRWDT** Clear Watchdog

- Example:  
*CLRWDT*

Syntax: CLRWDT

Operands: None

Operation: 00h -> WDT  
0 -> WDT prescaler  
1 ->  $\overline{TO}$   
1 ->  $\overline{PD}$

Status:  $\overline{TO}$ ,  $\overline{PD}$

Encoding: 00 0000 0110 0100

Words: 1

Cycles: 1

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set Control Operations

**RETLW** Return with Literal in W

- Example:  
*RETLW 0x5A*

Syntax: RETLW k

Operands:  $0 \leq k \leq 255$

Operation: k -> (W)  
TOS -> PC

After Instruction  
W = 0x5A

Status: None

Encoding: 11 0100 kkkk kkkk

Words: 1

Cycles: 2

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set Control Operations

RETFIE Return from Interrupt

Syntax: RETFIE

Operands: None

Operation: TOS -> PC

1 -> GIE

Status: None

Encoding: 00 0000 0000 1001

Words: 1

Cycles: 2

- Example:

*RETFIE*

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set Control Operations

RETURN Return from Subroutine

Syntax: RETURN

Operands: None

Operation: TOS -> PC

Status: None

Encoding: 00 0000 0000 1000

Words: 1

Cycles: 2

- Example:

*RETURN*

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set Control Operations

**CALL** Call Subroutine  
Syntax: **CALL k**  
Operands:  $0 \leq f \leq 2047$   
Operation:  $(PC) + 1 \rightarrow TOS$   
 $k \rightarrow PC\langle 10:0 \rangle$   
 $PCLATH\langle 4:3 \rangle \rightarrow PC\langle 12:11 \rangle$   
Status: None  
Encoding: 10 0kkk kkkk kkkk  
Words: 1  
Cycles: 2

Example:  
*Here CALL There*  
Before Instruction  
PC = address *Here*  
After Instruction  
PC = address *There*  
TOS = address *Here+1*

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## PIC MCU Instruction Set Control Operations

**GOTO** Unconditional Branch  
Syntax: **GOTO k**  
Operands:  $0 \leq f \leq 2047$   
Operation:  $k \rightarrow PC\langle 10:0 \rangle$   
 $PCLATH\langle 4:3 \rangle \rightarrow PC\langle 12:11 \rangle$   
Status: None  
Encoding: 10 1kkk kkkk kkkk  
Words: 1  
Cycles: 2

Example:  
*GOTO There*  
After Instruction  
PC = address *There*

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Assembler Directives

- LIST
  - Syntax: list [<list\_option>,...list\_option>]
  - Ex: list p=16F84A, f=INHX8M, r=DEC
  - Description: Select various assembler options
- INCLUDE
  - Syntax: include "filename.\*" or <filename.\*>
  - Ex: include <p16f84a.inc>
  - Description: Assembles the indicated file as if it were in-line code in the source file. Commonly used to create modular code segments.

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Assembler Directives

- ORG
  - Syntax: ORG <address>
  - Ex: ORG 0x00
  - Description: Start assembling the lines below this statement at location 0x00.
- EQU
  - Syntax: <label> equ <expr>
  - Ex: SECONDS equ 0x24
  - Description: Define a text substitution for a constant or variable. Seconds is a constant of 24h or is a register @ address 0x24.
    - movlw SECONDS ;const = 24h
    - movf SECONDS,W ;value in seconds reg.

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Assembler Directives

- CBLOCK
  - Syntax: cblock [<expr>]
  - Example:

```
cblock 0x20 ;Start point of cblock
      hours
      minutes
      seconds
      halfseconds:2

      endc ;Indicates end of cblock
```
  - Description: Declare Symbol Constant

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Assembler Directives (cont)

- END
  - Syntax: end
  - Example: Indicates end of source code.
  - \* You must have this statement after the last instruction you want assembled.

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Relocatable Code

### What is it?

- Code written to work at any address
- In source, code and data are organized into blocks called 'sections'
- The linker assigns sections into memory regions

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Relocatable Code

### Why use it?

- Organization - You can organize your project into multiple files
- Reusability - Source files can have a more focused purpose, encouraging code reuse
- Faster Development - Recompile of entire project is not required each time a change is made
- Convenience - User does not need to organize and manage individual memory locations

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Relocatable Code in MPASM

- CODE - Declares the beginning of a program code section
- UDATA - Declares the beginning of an uninitialized data section
- BANKSEL - Selects the correct bank for a RAM access
- PAGESEL - Selects the correct bank for a ROM access
- EXTERN - Indicates a variable that is declared in another module
- GLOBAL - Indicates that a variable is referenced in another module

© 1999 Microchip Technology Incorporated. All Rights Reserved.



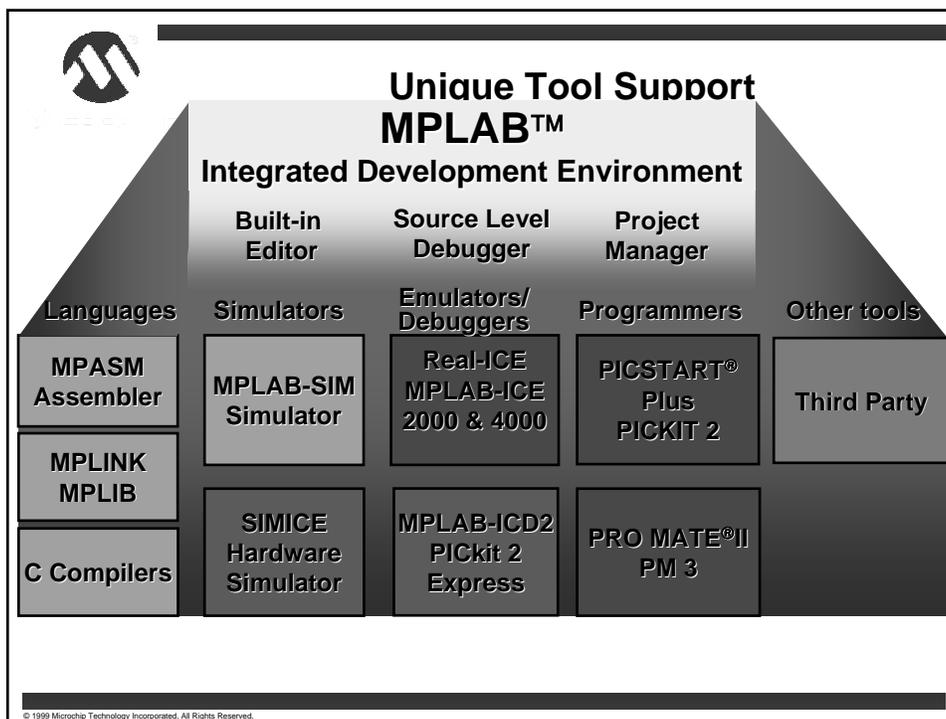
MICROCHIP

# *Devtools*



© 1999 Microchip Technology Incorporated. All Rights Reserved.

MICROCHIP




## Development Systems MPLAB-ICE

- Universal in-circuit emulator for PIC® MCUs
- Windows™ compatible
- MPLAB compatible
- Parallel port interface
- Complete source-level debugging
- Real-time in-circuit emulation of PIC MCUs
- Low voltage emulation
- Program memory emulation and memory mapping up to 64K words

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Development Systems

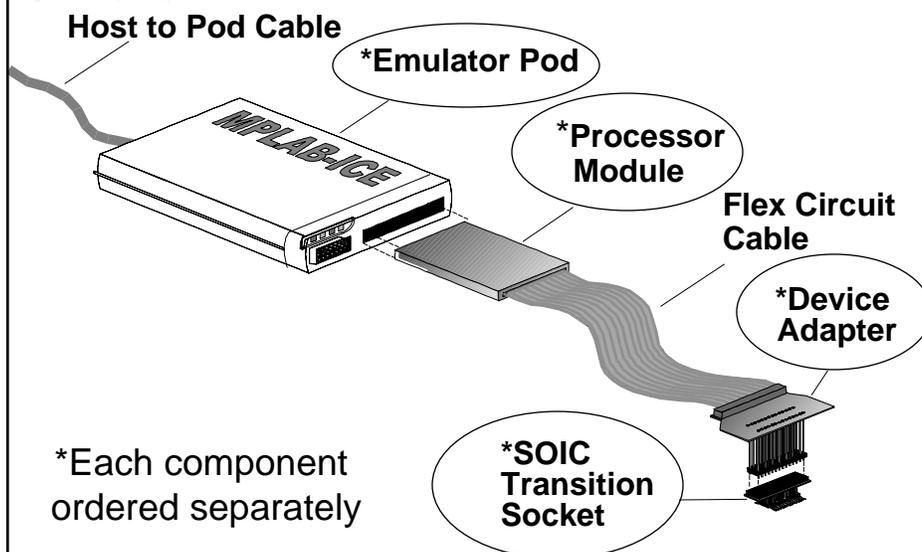
### MPLAB-ICE (continued)

- Break/trace/trigger on program address/data, internal register address/data, external inputs, or bus cycle type (mid-range, hi-end, golden gate only)
- Real-time trace with up to 32K x 128 buffer
- Time stamp trace
- External trigger input and output for logic analyzer/scope interface
- Complex breakpoints of up to four levels. Sequential events, AND/OR events, filtered trace, time between two events, and pass counts.

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## MPLAB-ICE 2000



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Development Systems ICEPIC

- Low cost in-circuit emulator for PIC MCUs
- Developed by RF Solutions (U.K.)
- Windows-compatible
- Complete source-level debugging
- EIA232 serial interface
- Unlimited breakpoints

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Development Systems MPLAB-SIM

- Universal simulator for PIC MCUs
- Windows and MPLAB-compatible
- Discrete event instruction-based simulation
- Complete source-level debugging
- Support interrupts and most peripheral functions (not A/D or serial I/O)
- Stimulus injection and file input capability
- Free

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Development Systems MPLAB-ICD 2

### ● Features

- USB (Full Speed 2 M bits/s) & RS-232 interface to host PC
- Real time background debugging
- MPLAB IDE GUI (free copy included)
- Built in over-voltage/short circuit monitor
- Firmware upgradeable from PC
- Totally enclosed
- Supports low voltage to 2.0 volts. (2.0 to 6.0 range)

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Development Systems MPLAB-ICD 2

### ● Features Continued

- Diagnostic LEDs (Power, Busy, Error)
- Reading/Writing memory space and EEDATA areas of target microcontroller
- Programs configuration bits
- Erase of program memory space with verification
- Peripheral freeze-on-halt stops timers at breakpoints

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Development Systems MPLAB-ICD 2

- **There is some shared overhead expense that includes:**
  - one stack level,
  - some general purpose file registers
  - a small area of program memory when in the debug mode

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

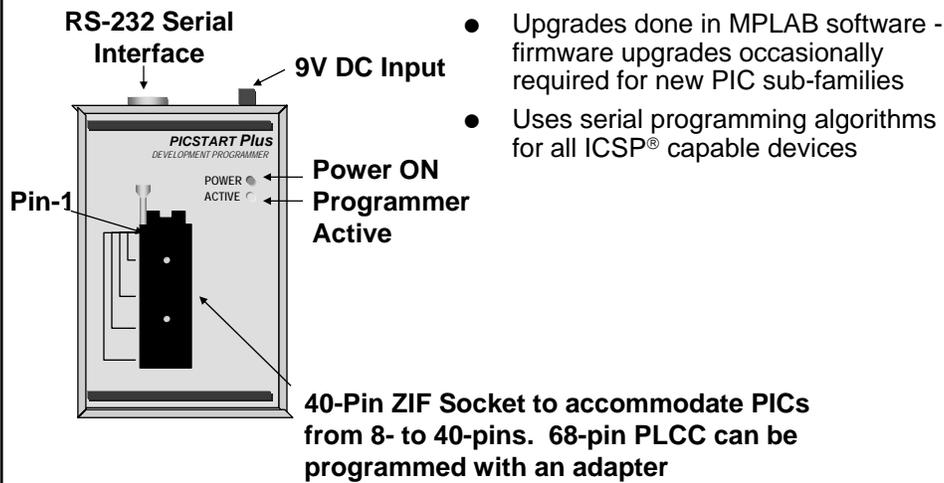
## Development Systems PICSTART Plus

- Development programmer kit
- Supports all PIC MCUs
- Windows- and MPLAB-compatible
- CD-ROM literature

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Development Systems PICSTART Plus Development Programmer



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Development Systems PRO MATE<sup>®</sup> II

- Universal device programmer for PIC MCUs
- Stand-alone or PC DOS-compatible, MPLAB-compatible
- EIA232 PC host interface
- Interchangeable socket modules
- Environment "save and restore"
- Programmable voltages for verify
- Serialized programming capabilities
- ICSP support
- Serial EEPROM support
- KEELOQ<sup>®</sup> support

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Development Systems

### PICDEM-1, -2, -3, -14, -17

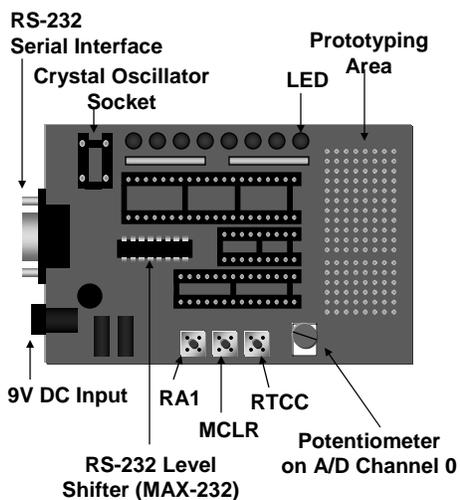
- Demonstration board for PIC MCUs
- On board +5V regulator
- EIA232 interface
- 5K pot to simulate analog input
- 3 push buttons for external stimulus and reset
- LED's connected to PORTB
- LCD display panel connection
- Socket for crystal oscillator
- Prototype area

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Development Systems

### PICDEM-1 Demo Board



- Demo board accommodates all 18-pin PIC devices, 28-pin PIC16C5X and 40-pin PIC17C4X devices
- 8-LEDs connected to PORTB
- 5k $\Omega$  Pot connected to AN0
- PB switches connected to RA1, RTCC, and MCLR

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Development Systems MPASM

- Universal macro assembler for all PIC MCUs
- DOS- and Windows-compatible
- Compatible with MPLAB
  - Symbolic and source-level debugging
  - Can generate relocatable code
- Multiple output formats
- Licensed by several third party vendors
- Free

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Development Systems COMPILERS

- Microchip MPLAB-C17
- Microchip MPLAB-C18
- Microchip MPLAB-C30
- Hi-Tech
- CCS
- IAR
- Byte Craft
- Micro Engineering Labs (Basic)

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Development Systems Reference Materials

- Application notes
  - Embedded Controls Handbook
- Reference designs
- Website
  - <http://www.microchip.com>
  - Conference groups
  - Mailing lists
- Microchip CD ROM
- Template subdirectory under MPLAB (code templates)
- Seminars
- Workshops

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Development Systems Third Party Guide

- Microchip encourages third-party support
  - 312-page reference manual to aid in selection of appropriate tools
  - 106 companies and 208 products
    - Emulators
    - Programmers and gang programmers
    - 'C' compilers, software, demo boards, etc.
  - Design consultant reference

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Books For Further Research (English)

- *Design with PIC Microcontrollers*, John Peatman, Prentice-Hall, 0-13-759259-0
- *Programming and Customizing the PIC Microcontroller*, Myke Predko, McGraw-Hill, 0-07-913646-X
- *Easy PIC'n*, David Benson, Square 1 Electronics, 0-9654162-0-8
- *PIC'n Up The Pace*, David Benson, Square 1 Electronics, 0-9654162-1-6
- *A Beginners Guide to the Microchip PIC*, Nigel Gardner, Bluebird Technical Press Ltd., 1-899013-01-6

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Books For Further Research (English)

- *PIC Cookbook Vol 1 & Vol 2*, Nigel Gardner, Bluebird Technical Press Ltd., 1899013-02-4, 1-901631-00-1
- *The Greatest Little PIC Book*, Gordon McNee, Bluebird Technical Press Ltd., 1-901631-01-X
- *PIC Microcontroller Operation and Applications*, DN de Beer, Cape Technikon, daandb@norton.ctech.az.za
- *Getting started with PIC Microcontrollers*, Al Stevens, Ziggurat Technologies, ziggurat@global.co.za

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Books For Further Research (French)

- *Les Microcontrôleurs PIC et mise en oeuvre*, Christian Tavernier, Dunod, 2-10-002647-X
- *Micontrôleurs PIC a structure RISC*, C.F. Urbain, Publitrionic, 2-86661-058-X
- *Pratique des Microcontrôleurs PIC*, Francesco Volpe/Safinaz Volpe, Publitrionic Elektor, 2-86661-077-6

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Books For Further Research (German)

- *Mit dem PIC-Controller Erfolgreich Arbeiten*, Dr. Anne Koenig/Manfred Koenig, Markt & Technik Verlag, 3-8272-5168-0
- *Mikrokontroller mit RISC Struktur*, C.F. Urbain, Elektor Compact, 3-928051-83-0
- *Mikroprozessor PIC16C5X*, Michael Rose, Huthig, 3-7785-2169-1
- *Mikroprozessor PIC17C42*, Michael Rose, Huthig, 3-7785-2170-5
- *PIC uC Praxis*, Francesco Volpe/Safinaz Volpe, Elektor Verlag, 3-89576-030-7

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Books For Further Research (Italian/Russian/Spanish)

- *La Programmazione dei Microcontrollori PIC*, Andrea Sbrana, Italy +39 544 464070
- *New Possibilities with the Microchip PIC*, RIGA, Latvia +371 935 0550
- *MicroControladores PIC La Solution en un Chip*, Martinez Usategui, Paraninfo, 84-283-2371-2
- *MicroControladores PIC*, Christian Tavernier, Paraninfo, 84-283-2373-9

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Books For Further Research (Chinese)

- *Parts 1-5 PIC16C5X/71/84 Development and Design*, United Tech Electronic Co. Ltd,
  - 957-21-0807-7
  - 957-21-1152-3
  - 957-21-1187-6
  - 957-21-1251-1
  - 957-21-1257-0
- *PIC16C84 MCU Architecture and Software Development*, ICC Company, 957-8716-79-6

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Microchip Technical Services Engineering Web Site

- Web address: [www.microchip.com](http://www.microchip.com)
- Latest versions available
  - Datasheets
  - User's Guides
  - Application Notes
  - Device Errata
  - Development tools
- Frequently-Asked Questions (FAQs)
- Latest product announcements
- Recent press releases

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Microchip Technical Services

- Workshops
  - Hands on workshops using Microchip development tools
  - Various topics in many cities
  - Watch the web site for details
- Address: [www.microchip.com](http://www.microchip.com)



© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

# On to the Labs

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab 1

**PIC12C672**  
**14-bit core**

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab1

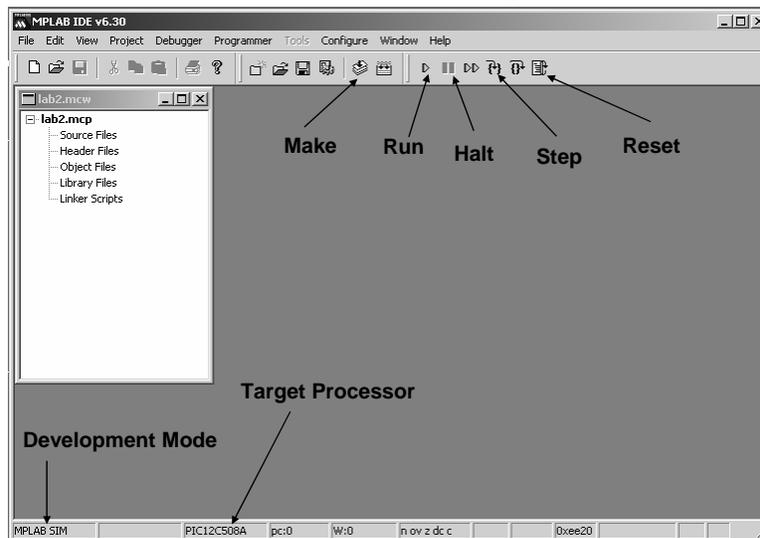
### Objective:

- Program an OTP and plug it into your PIC & Stick take home demo board
  - 1st time experience with tools
  - Will use PIC & Stick board with non-coded PIC's

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab 1 MPLAB Layout



© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology Inc.

## Lab 1

### Selecting Hardware Options

- Use Debugger > Select Tool > MPLAB SIM... to select execution environment
- Use Configure > Select Device to select processor
- Use Debugger > Settings > Clock TAB to select oscillator frequency
  - MPLAB-ICE uses a programmable clock
  - MPLAB-SIM only uses frequency for time calculations

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology Inc.

## Lab 1

### MPLAB Projects

- MPLAB projects manage:
  - Installed build tools
  - Development mode
  - Hardware settings
  - Source files
  - Dependencies (\*.inc, \*.h)
  - Compile, link, and assembly options

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab 1 Creating a New Project

- Choose Project > New Project... from the menu, and select a filename for your project

The 'New Project' dialog box has a title bar with a close button (X). It contains two text input fields: 'Project Name' with the text 'Lab1' and 'Project Directory' with the text 'C:\pic101'. To the right of the 'Project Directory' field is a 'Browse...' button. At the bottom of the dialog are three buttons: 'Help', 'OK', and 'Cancel'.

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab 1 Setting Build Options

The 'Build Options' dialog box has a title bar with help (?) and close (X) buttons. It features three tabs: 'General', 'MPASM Assembler', and 'MPLINK Linker'. The 'General' tab is active. A 'Categories' dropdown menu is set to 'General'. Under 'Generate Command Line', there are two checkboxes: 'Disable case sensitivity' (unchecked) and 'Extended mode' (unchecked). The 'Default Radix' section has three radio buttons: 'Hexadecimal' (selected), 'Decimal', and 'Octal'. The 'Macro Definitions' section contains an empty list box and three buttons: 'Add..', 'Remove', and 'Remove All'. Below this are two checkboxes: 'Inherit global settings' (unchecked) and 'Use Alternate Settings' (unchecked), along with a 'Restore Defaults' button. At the bottom are 'OK', 'Cancel', and 'Apply' buttons.

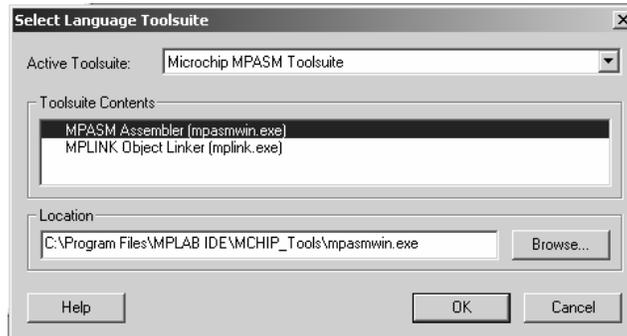
© 1999 Microchip Technology Incorporated. All Rights Reserved.



MPLAB IDE v6.30

## Lab 1 Configuring Build Tools

- Microchip's tools are configured for use during installation
- For other tools, use Project > Select Language Tool Suite...



- Use the Active Toolsuite: pulldown bar for other Language toolsuits

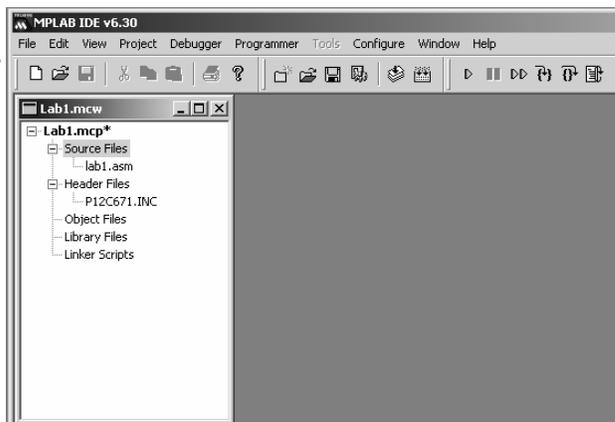
© 1999 Microchip Technology Incorporated. All Rights Reserved.



MPLAB IDE v6.30

## Lab 1 Using the Project Window

- Quick access to project summary
  - Source files
  - Dependencies



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Setting Up MPLAB IDE Projects

### Program Your Part

- Verify that your program is listed in the “Program Memory Window”
- If it is not, build your project again to update the “Program Memory Window”
- Verify device settings in the PICSTART Plus Device Programmer window
- Insert part to be programmed with correct pin 1 orientation
- Click on Program to program part

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab notes

- Must make a project first, prior to programming part
- Make sure that the .asm file is listed in the project as a file to be built
- Must build the .asm file to create a .hex file prior to programming part
- Make sure that correct part is selected on programmer
- Must set configuration bits for the operating mode of the part

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## What feature set (reminder)

- Open/close
- Switch controlled
- Held open to allow passage thru door
- Automatic close (store type)
- Safety cutoffs (if time permits)
- Independent motor control (if time permits)
- Double tap (if time permits)
- Lighting control (optional)
- Sound control\* (optional)

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab 2

### Using An I/O Port (Switching) PIC12F508 12-bit Core

© 1999 Microchip Technology Incorporated. All Rights Reserved.



© 1999 Microchip Technology Incorporated. All Rights Reserved.

## Lab notes: Peripheral Used: Digital I/O Ports

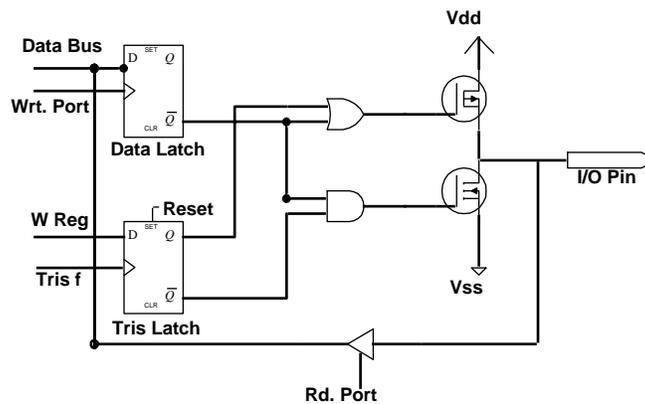
- 6 - 20 I/O pins
- Each I/O pin software configurable as input or output
- Direct bit manipulation (single-word/single-cycle):
  - Bit set
  - Bit clear
- High drive capability
  - 25mA sink max (all parts)
  - 60mA sink max (PIC17CXXX)
  - 25mA source max
- Can directly drive LEDs:

© 1999 Microchip Technology Incorporated. All Rights Reserved.



© 1999 Microchip Technology Incorporated. All Rights Reserved.

## Peripherals: Digital I/O Port (Cont.)



© 1999 Microchip Technology Incorporated. All Rights Reserved.





Microchip Technology

## Lab2 Using An I/O Port

### Objective:

- Drive port pin GP4 to a logic 1
- Verify code with the MPLAB Simulator
- Program a part and test it in your workshop board

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

**Make a project file review**

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Setting Up MPLAB IDE Projects

### Make A .asm File

- File -> New
  - This opens a text window where you can begin entering your code
- Type in program header
- File -> Save As
  - Save the file as lab2.asm
- File -> Close

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Setting Up MPLAB IDE Projects

### Make A Project File

- Project -> New
  - Type Lab2 in Project Name Box
  - Click Browse: Find Folder: C:\PIC101 Click the Select button
    - Click on OK
  - Project Files
    - Right Click on Source files
      - Select Add Files
      - Find Lab2.asm Click Open

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Setting Up MPLAB IDE Projects

### Make A Project File (cont.)

- Project Files Continued
  - Right Click on Header files
    - Select Add Files
    - Find `c:\Program Files\MPLAB IDE\MCHIP_tools\P12C508A.inc`
    - Click Open

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Setting Up MPLAB IDE Projects

### Make A Project File (cont.)

- Project Files cont.
  - Debugger > Select Tool > MPLAB SIM
  - Configure > Select Device
    - Use Device: pull down Bar
      - Select the appropriate part [12C508A]
      - Click on OK (Close Select Device Dialogue box)

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Setting Up MPLAB IDE Projects

### Verify Correct .asm File

- Verify that the “Node: lab2.asm” file is the correct file you are working on.
- Double click on lab2.asm in the Lab2.mcw Project window
  - Lab2.asm will be used for this project

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Code Ex2 - Light an LED

```
*****  
;  
;This program drives a 1 onto port pin GP4  
*****  
;  
List P=12F508, R=hex  
include <P12F508.inc> ;load the include file  
__CONFIG _IntRC_OSC & _WDT_OFF & _CP_OFF & _MCLRE_ON  
  
Start      org      0x00      ;indicate the start location  
           clrf      GPIO      ;load output latch w/zeros  
           movlw    0x00      ;Configure port to outputs  
           tris     GPIO      ;  
           bsf      GPIO,4    ;set pin 4 on GPIO (GP4) to logic 1  
  
Loop       nop  
           goto     Loop      ;$ = current address... infinite loop  
           end
```

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Create a Watch Window

- View -> Special Function Registers
- View -> Watch -> New Watch Window
  - Use pull down menu to select registers/variables to watch
  - Click Add FSR button after each one or Click Add Symbol button

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Use simulator to step thru the code

- Debugger -> Run
  - F6 Reset
  - F7 Step
  - F8 Step Over
  - F9 Run
  - F5 Halt
  - Ctrl + F9 Animate

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab notes

- Must declare `org` statements so assembler knows code/memory organization
- Don't forget to declare variables if needed
- Must configure port pins prior to writing to them

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab3

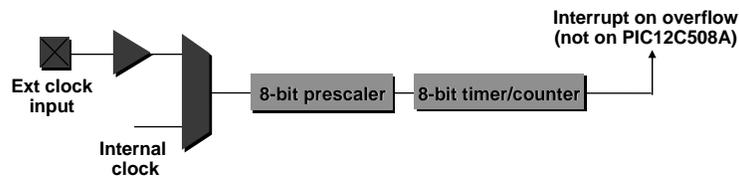
Timing  
PIC12F508  
12-bit Core

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab notes: Peripheral Used: TMR0

- Readable and writable
- Generates interrupt on overflow from FFh to 00h
- Prescaler is programmable
- In timer mode: fastest increment rate is OSC/4 (5 MHz @ 20 MHz oscillator frequency)
- In counter mode: configurable to increment on either edge
- External counter mode has Max. input = 50 MHz (using prescaler)
- Writing to TMR0 causes an error of 2 Tcy

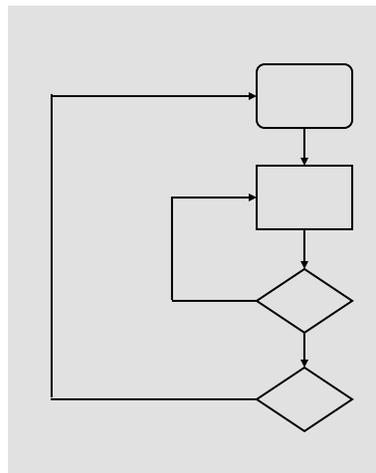


© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab Notes:

Timing Loop  
Counting Loop  
Nested Loop



Branch Control  
BTFSS  
BTFSC  
DECFSZ  
INCFSZ

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## OPTION REG PIC12F508

W-1	W-1	W-1	W-1	W-1	W-1	W-1	W-1
GPWU	GPPU	TOCS	TOSE	PSA	PS2	PS1	PS0
bit7	6	5	4	3	2	1	bit0

W= Writable bit  
U= Unimplemented bit  
-n= Value at POR reset  
Reference Table4-1 for other resets.

bit 5: **TOCS**: Timer0 clock source select bit  
1 = Transition on T0CKI pin  
0 = Transition on internal instruction cycle clock, Fosc/4

bit 4: **TOSE**: Timer0 source edge select bit  
1 = Increment on high to low transition on the T0CKI pin  
0 = Increment on low to high transition on the T0CKI pin

bit 3: **PSA**: Prescaler assignment bit  
1 = Prescaler assigned to the WDT  
0 = Prescaler assigned to Timer0

bit 2-0: **PS2:PS0** : Prescaler rate select bits

Bit Value	Timer0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab3 Timing

### Objective

- Write an approx. 2 ms delay loop program using TMR0
  - Typically used to
    - provide timing for delaying/controlling events
    - provide timing for tracking events
- Do not use the prescaler
- Pre-load TMR0 to count 200 micro seconds
- Light 2 LEDs to indicate status
  - GP4 LED2 should light to indicate that your program is running
  - GP5 LED1 should light when the approx. 2 ms delay has occurred
- Verify using MPLAB simulator with stopwatch

© 1999 Microchip Technology Incorporated. All Rights Reserved.



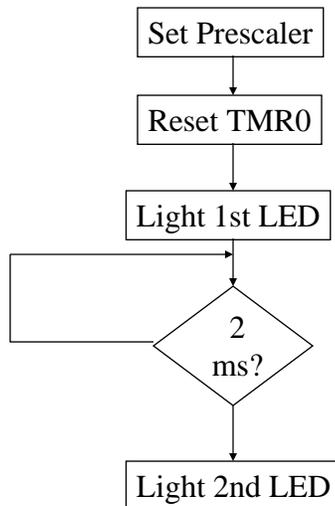
## Lab notes: Calculate # of TMR0 Roll Overs

- Part uses a 4Mhz clock
- TMR0 clocks @ a rate of  $F_{osc}/4$ 
  - if prescaler not used
- TMR0 rolls over from FFh to 00h
- How many counts does a s/w counter need to get approx. 2 ms?
- How many counts does a s/w counter need to get approx. 2 ms if tmr0 only counts 200 micro seconds?

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Code Ex3 - 2 ms delay (cont.)



© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab notes

- The PIC12F508 has an 8 bit timer (TMR0) with a selectable prescaler (up to 8 bits)
- You may need to pre-load TMR0 to get the desired time

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab notes: Sample window

- What happens if the FFh -> 00h roll over occurs after you have sampled?
- Use a Sample Window

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab3 Timing

- Review sample code lab3.asm provided by the instructor.

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab4

### Timing

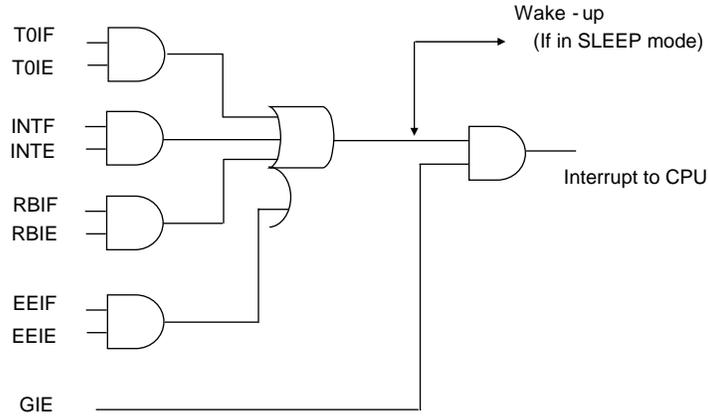
### PIC16F84A

### 14-bit Core

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Feature Used: Interrupts



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## INTCON REG PIC16F84A

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x	
GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	R= Readable bit W= Writable bit U= Unimplemented bit, read as '0' - n= Value at POR reset
							bit7	bit0
<p>bit 7: <b>GIE</b>: Global Interrupt Enable bit 1 = Enables all un-masked interrupts 0 = Disables all interrupts</p> <p>bit 5: <b>TOIE</b>: TMR0 Overflow Interrupt Enable bit 1 = Enables the TMR0 interrupt 0 = Disables the TMR0 interrupt</p> <p>bit 2: <b>TOIF</b>: TMR0 Overflow Interrupt Flag bit 1 = TMR0 has overflowed (must be cleared in software) 0 = TMR0 did not overflow</p>								

© 1999 Microchip Technology Incorporated. All Rights Reserved.





## Why Interrupts

- Better use of CPU resource
- Faster response time
- Multi-tasking
- Fixed or known interval handler
- Exception handler

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab notes: Interrupt Service Routines; Context Changes

Push macro

```
movwf    W_TEMP           ;push W and STATUS
swapf    STATUS,W         ;
movwf    STATUS_TEMP     ;
```

endm

Pop macro

```
swapf    STATUS_TEMP,W   ;pop W and STATUS
movwf    STATUS           ;
swapf    W_TEMP,F        ;
swapf    W_TEMP,W        ;
```

endm

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab4 Timing

### Objective

- Write an approx. 2 second delay loop program using TMR0 and interrupts
  - typically used to
    - provide timing for delaying/controlling events
    - provide timing for tracking events
- Use prescaler
- Light 2 LEDs to indicate status
  - RB2 LED1 should light to indicate that your program is running
  - RB3 LED2 should light when the approx. 2 sec delay has occurred
- Verify using MPLAB simulator and on your workshop board when finished

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

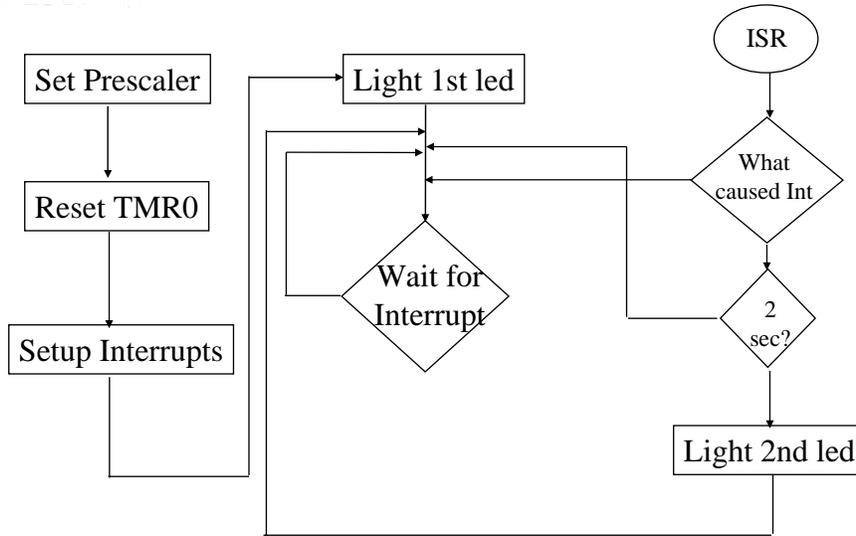
## Lab notes: Calculate # of TMR0 Roll Overs

- Part uses a 4Mhz clock
- TMR0 clocks @ a rate of  $F_{osc}/4$ 
  - if prescaler not used
- TMR0 rolls over from FFh to 00h
- Prescaler set to max
- How many counts does a s/w counter need to get approx. 2 sec.?

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Code Ex4 - 2 sec delay (cont.)



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab notes

- The 16F84A has an 8 bit timer (TMR0) with a selectable prescaler (up to 8 bits)
- You may need to pre-load TMR0 to get the desired time
- The main program should be an endless loop (goto \$)
- The interrupt vector is address 0x04
- Assert RB3 in the Interrupt Service Routine as status indicator
- Keep ISR as simple as possible. Fill with calls as needed.

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab4 Timing

- Review sample code lab4.asm provided by the instructor.

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

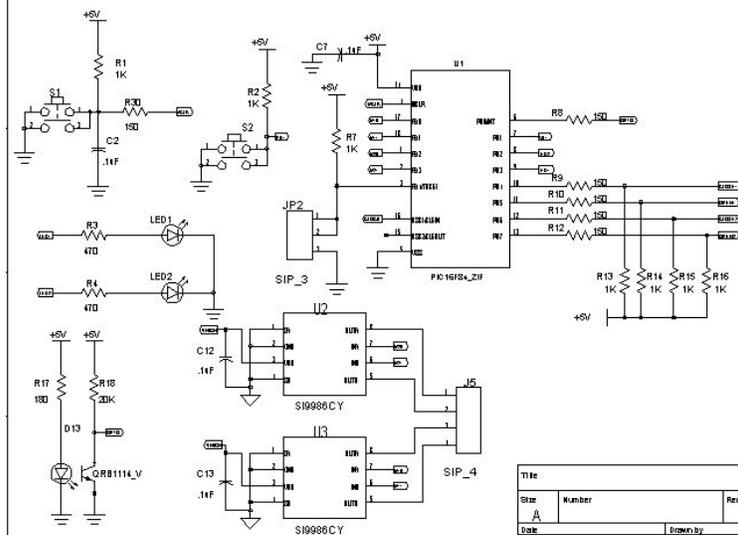
## Lab5

**Motor Control  
PIC16F84A  
14-bit Core**

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Motor Connection Schematic



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab5 apply to simple motor control

### Objective

- Apply labs 2 and 4 to control door motors
- Motor control to drive doors both open and close
- Toggle opening and closing of door in a repeating cycle
- Validate using MPLAB simulator, your workshop board, and the scale model prototype when finished

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab notes

- Motor biasing is set for standard logic
- 2 control bits are required for each motor
- You will need to stop the motion of the motor before reversing it's direction to avoid stripping motor's gears
- Use the timing and I/O port control modules you developed earlier to help you
- PORTA is tied to the 2 door motors

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab 6

**Sensing an Input**  
**PIC16F84A**  
**14-bit Core**

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab6 add sensing/inputs (2 part lab)

- Mechanical catch/releases vs electrical switches
- Floor pad/wall switch
- Input techniques
  - polling
  - external interrupt on INT pin
  - interrupt/wake up on change

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Input techniques

- Polling
  - sample window roll over
  - power consumed
  - ringing input
  - debouncing of input signal
- External interrupt on INT pin
  - multiple interrupts
  - sequential servicing
  - simultaneous interrupts
- Interrupt/wake up on change
  - can put part to sleep prior to receiving interrupt input
  - battery type apps

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab 6 (a)

### Polling Input Pin

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Why Debounce?

- Ringing inputs
- Noisy inputs
- Glitch on input

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Debounce push button/key Ringing or noisy inputs

```

Check_key
    ;poll for key press
    btfsc    PORTB,1
    ;
    goto    Check_key
;
    call    Debounce_delay    ;debounce signal
    btfsc    PORTB,1
    ;check for bouncing input
    goto    Check_key
    ;
    return

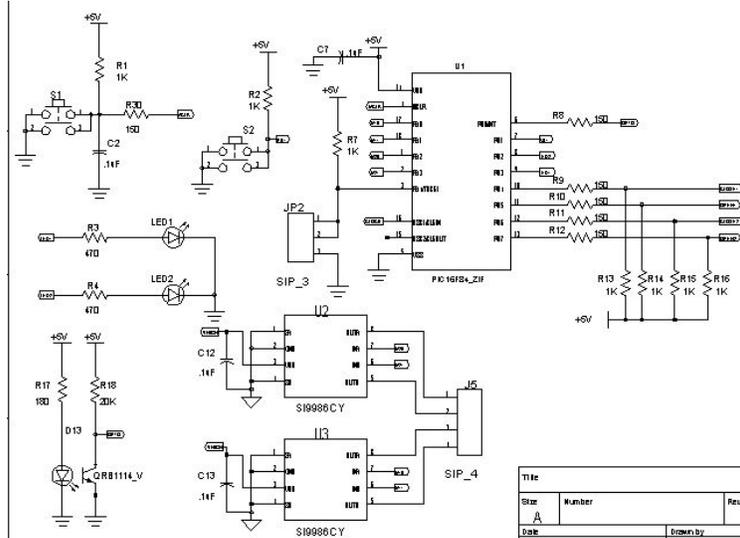
;routine to give delay for key bounce routine
Debounce_delay
    .
    .
    delay routine here
    .
    return
  
```

AN 566 contains debounce

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Polling Input Schematic



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab 6 (a) of 2-step lab

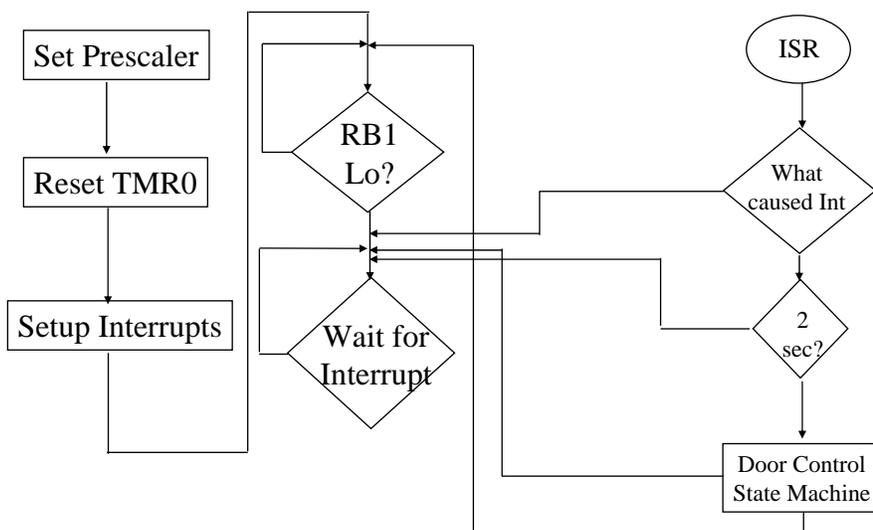
### Objective

- Read input by polling, (foot pad) to activate door “opening/closing sequence” module you developed earlier
- Momentary switch on port pin RB1
- RB1 is an active low input
- Validate using MPLAB simulator, your workshop board, and scale model prototype when finished

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Code Ex6a - Polling Input



© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab notes

- Call a input checking routine from your main loop
- Might need a state machine to track the different phases (open, close, etc.) and coordinate with your timing routines
- Must loop back to look for next input

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab 6 (b)

### Interrupt on INT Pin

© 1999 Microchip Technology Incorporated. All Rights Reserved.





## Lab6 (b) of 2 step lab

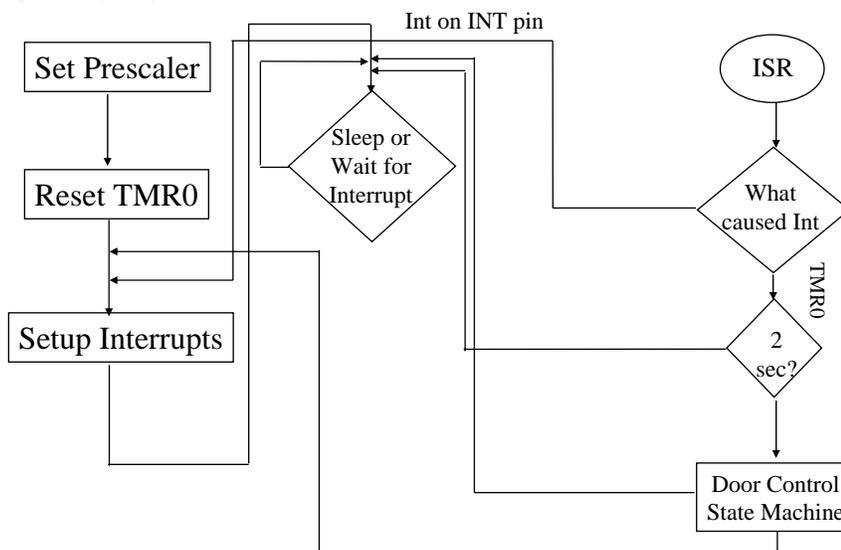
### Objective

- Read input by interrupt on INT pin (optical sensor) to activate door opening/closing module you developed earlier
- Light sensor on port pin RB0
- RB0 is active low sensor input
- Validate using MPLAB simulator, your workshop board, and scale model prototype when finished

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Code Ex6b - INT Pin Input



© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab notes

- Use different setup routines for the different interrupt modes
- Might need a state machine to track the different phases (open, close, etc.) and coordinate with your timing routines
- Don't forget to put part to sleep or in a dead loop while waiting for interrupt on INT pin input (don't overrun into unwanted areas)
- Must loop back to look for next input

© 1999 Microchip Technology Incorporated. All Rights Reserved.



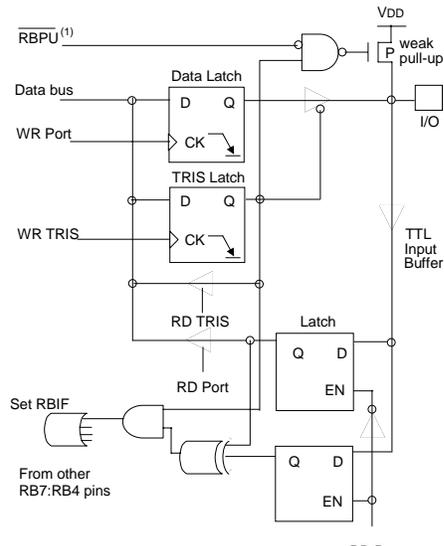
Microchip Technology

## Lab7 State Logic/Higher Order Functions PIC16F84A 14-bit Core

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Feature Used: Interrupt On Change PIC16F84A



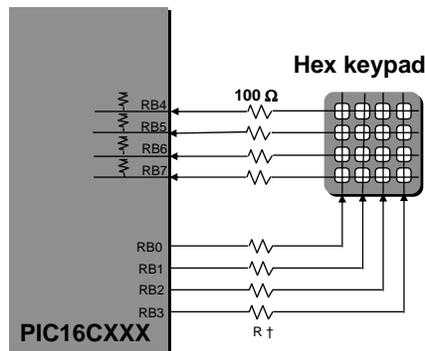
Note 1: TRISB = '1' enables weak pull-up  
(if RBPU = '0' in the OPTION\_REG register).  
2: I/O pins have diode protection to V<sub>DD</sub> and V<sub>SS</sub>.

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Peripheral Used PortB Interrupt on Change example

### Keypad interface:



† Resistance for ESD protection

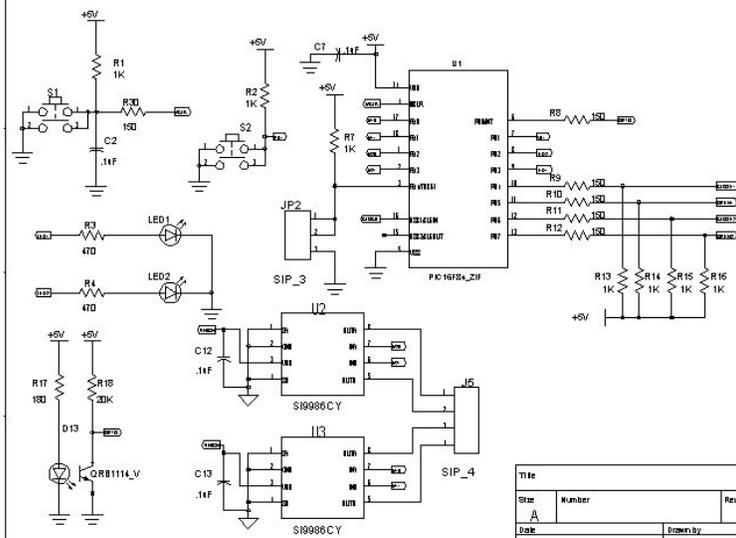
- Internal pull-ups (software option) keep RB4-RB7 pins high
- RB0-RB3 pins drive '0's
- Any key pressed pulls down an RB pin and generates an interrupt
- This interrupt wakes up processor from SLEEP
- Saves timer resources

AN557

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Interrupt On Change Schematic



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab7 State logic/higher order functions

### Objective

- Delay to hold door open approx 2 seconds before auto closing
- Use "interrupt on INT pin" input method from Lab6(b) to start door sequence
- Option 1
  - Motor cutoffs
  - Position feed back
  - Use interrupt on change

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab7 State logic/higher order functions

### Objective (cont.)

- Option 2
  - Do not close doors if INT pin sensor still active
  - Safety feature
  - People still in the doorway
- Option 3
  - Independent motor cutoff
  - Different motor speeds
- Validate using MPLAB simulator, your workshop board, and scale model prototype when finished

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Lab notes

- Might need a state machine to track the different phases (open, hold open, close, etc.) and coordinate with your timing routines
- Use different setup routines for the different interrupt modes
- If adding motor cutoffs (option1), don't use timing delay module while motors running (open or close). But use timing delay to hold door open for 2 sec.
- Safety check (option2) only applies when closing doors

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab8 Proof Final Functionality PIC16F84A 14-bit Core

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Lab8 combine labs 6 & 7

### Objective

- Demonstrate final functionality of automatic door controller on workshop prototype board
  - Combine all of the developed modules
  - Demonstrate optional features if developed (i.e. emergency cutoffs, access light, double tap logic)

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

# MCU Robust Design Techniques

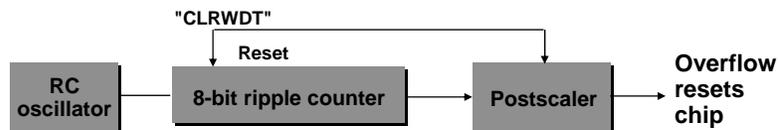
© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Robust Design Techniques Watchdog Hardware

- Helps recover from software malfunction
- Uses its own free-running on-chip RC oscillator
- WDT cannot be disabled by software
- WDT overflow resets the chip
- CLRWDT instruction clears WDT
- Programmable time-out period: 18 ms to 2.5 seconds
- Operates in SLEEP
- On time-out wakes up CPU



© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Robust Design Techniques

### Watchdog Software Strategies

- Watchdog effectiveness is only as good as the software controlling it
- Use one CLRWDT instruction for the entire program
- Place CLRWDT in the main loop
- Do not place CLRWDT in ISR or any sub-routines
- Select the minimum WDT timeout period that main loop timing can tolerate
- Initialize unused memory as GOTO wdtreset (self-loop) to force timeout

© 1999 Microchip Technology Incorporated. All Rights Reserved.



## Robust Design Techniques

### Robust Software - Filling Unused Memory

- When using watchdogs, fill unused memory locations with a GOTO wdtreset
- This self loop is normally never executed
- If the PC is corrupted, it is likely it will execute this instruction
  - Force a WDT reset to re-initialize the core
- FILL can be used to insert these instructions (Substitute <last location+1> for 400h):

```
FILL    (GOTO    wdtreset), (400h-$)  
ORG 3FFh
```

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Robust Design Techniques

### Robust Software - Clean WDT reset

- To force WDT reset on power-up:
  - Check for RAM pattern at power-up
  - If pattern is not there:
    - Initialize RAM pattern
    - Force WDT reset
- Force WDT reset if code jumps to unused program memory

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Robust Design Techniques

### Robust Software - Subroutine Counting

- Maintain subroutine call and execution counters
- Increment call counter for every subroutine call
- Increment execution counter at the top of every subroutine
- At the top of main loop, check if call and execution counters are equal
- If not equal, force a WDT reset

**Code Example: ADVRELSW.ASM**

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

**The End  
Thanks for attending!**

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

**Appendix**

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

## Glossary

- TMR0 - 8-bit timer peripheral w/8 bit prescaler
- Prescaler - peripheral which delays the TMR0 count by a prescale ratio i.e. 1:1, 1:2, 1:4
- Page - memory organization for program memory
- Bank - memory organization for data memory
- Pipeline - hardware architecture for prefetching an opcode while executing the previous opcode
- Orthogonal instruction set - instructions work on ports and registers the same way
- OPTION\_REG - register of control bits for configuring tmr0, INT, and pull ups
- INTCON - register of control bits for configuring peripheral interrupts
- STATUS - register of bits indicating the results of an operation
- RP1&RP0 - direct addressing bank selection control bits
- IRP - indirect addressing bank selection control bit
- PORTB - I/O port B
- GPIO - I/O port on the 8 pin parts

© 1999 Microchip Technology Incorporated. All Rights Reserved.



Microchip Technology

The Microchip name, logo, The Embedded Control Solutions Company, PIC, PIC, PICSTART, PICMASTER, PRO MATE, SEEVAL, KEELOQ and the KEELOQ logo are registered trademarks and MPLAB, fuzzyLAB, In-Circuit Serial Programming, ICSP, are trademarks of Microchip Technology Incorporated in the USA and other countries.

Windows is a registered trademark of Microsoft Corporation.

SPI is a trademark of Motorola.

PC is a registered trademark of Philips Corporation.

Microwire is a registered trademark of National Semiconductor Corporation.

fuzzyTECH is a registered trademark of Inform Software Corporation.

All other trademarks herein are the property of their respective companies.

© 1999 Microchip Technology Incorporated. All rights reserved.

\*Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Inc. with respect to the accuracy of such information, or infringement of patents arising from any such use of otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.\*

© 1999 Microchip Technology Incorporated. All Rights Reserved.