PIC Microcontroller

Revision sheet

1 Flowcharts

Flowcharts are a graphical way of representing flow through a program. Flowcharts are built from symbols, with each symbol having a different meaning. The symbols are outlined below.

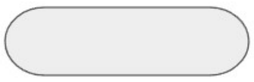


Figure 1: Terminator block

The terminator block indicates the start or the end of a program.

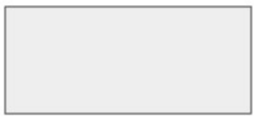


Figure 2: Process block

The process block indicates a calculation or delay.

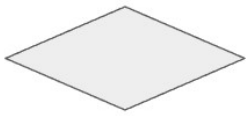


Figure 3: Decision block

The decision block tests a condition with a `yes` or `no` answer. The program will follow a different route depending on the answer.

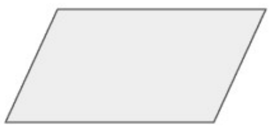


Figure 4: Input/Output block

The input/Output block indicates data being read into or written from the program.

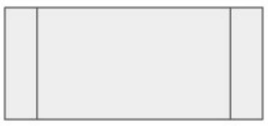


Figure 5: Subroutine block

The subroutine block indicates the program should run another sub-program then return to where it was called from (the location of the sub-routine block).

2 Parts Of The PIC Microcontroller

The PIC Microcontroller has a number of parts which are useful.

2.1 Pinout

The pinout of the PIC Microcontroller can be seen below.

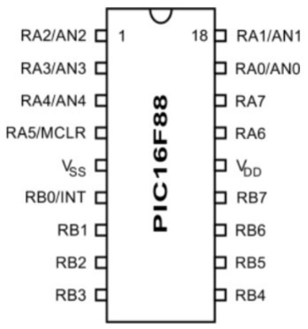


Figure 6: Pinout of the PIC16F88

2.2 Registers

A register is an 8-bit memory storage location. Each bit of the register is independantly controllable/ accessible. There are a number of different registers which are important.

2.2.1 The Working (W) Register

The working (W) register is the only register into which data can be entered from the program. This means that data which is to be placed into another register or data which has come from another register which the CPU needs to access for processing, has to go via the W register.

2.2.2 The Data Direction Registers

Each port has a data direction register. The contents of these registers sets the direction of the port bits. This generally only needs to happen once, at the beginning of the program. When setting the directions, 1 means input and 0 means output. The direction register for PORTA is called TRISA and the direction register for PORTB is called TRISB.

2.2.3 PORTA and PORTB registers

The PORTA and PORTB registers hold the actual data that is present on the two ports. If the port bits are set as outputs, the contents of the W register can be copied to the port registers - changing the outputs.

2.2.4 STATUS register

The STATUS register is used by the microcontroller for various functions. It's layout is shown on the formula booklet. The C bit (bit 0) will go logic high if an arithmetic operation resulted in a carry. The Z bit (bit 2) will go logic high if an arithmetic operation has resulted in zero.

3 Interrupts

3.1 Why Use Interrupts?

Interrupts are used as they are more efficient. The alternative to using interrupts, is to use polling which is where an input is continually checked to see what it is and reacting then. Interrupts work by reacting as soon as something changes. Once the interrupt has been dealt with, the microcontroller returns to whatever it was doing before.

3.2 Interrupt Service Routine

The steps below outline the ISR.

1. Save the contents of the W register to a variable (this is so that it can be re-filled after the ISR)
2. Check if teh right interrupt happened by checking the external interrupt flag
3. Execute the interrupt's code
4. Clear the External Interrupt Flag, so that another interrupt can be triggered
5. Restore the contents of the W register (copy the variable back into the W register)
6. Return from the ISR.

3.2.1 Example ISR

Shown below is an example ISR

```
interrupt

    movwf w_save
    ; copy w to save variable

    btfss INTCON, INT0IF
    ; check correct interrupt happened

    retfie
    ; if not, return to where was

    ; interrupt code here

    bcf INTCON, INT0IF
    ;clear interrupt flag

    movf w_save, W
    ;restore working register

    retfie
    ;return and re-set GIE bit
```

3.3 Defining The Interrupt

The PIC has to be told where to find the ISR. The interrupt vector address is 04, this means that when an interrupt is triggered whatever the code at memory location 04 is will be run. The code below shows the declaration of this.

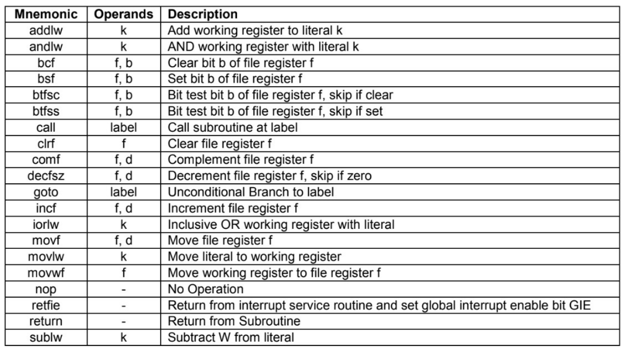
```
ORG
   h'04'
   goto interrupt
```

The code above should go near the start vector.

4 PIC Microcontroller Assembly Language

4.1 List Of Commands

This list is also available in the Data Booklet



Mnemonic	Operands	Description
addlw	k	Add working register to literal k
andlw	k	AND working register with literal k
bcf	f, b	Clear bit b of file register f
bsf	f, b	Set bit b of file register f
btfsc	f, b	Bit test bit b of file register f, skip if clear
btfss	f, b	Bit test bit b of file register f, skip if set
call	label	Call subroutine at label
clrf	f	Clear file register f
comf	f, d	Complement file register f
decfsz	f, d	Decrement file register f, skip if zero
goto	label	Unconditional Branch to label
incf	f, d	Increment file register f
iorlw	k	Inclusive OR working register with literal
movf	f, d	Move file register f
movlw	k	Move literal to working register
movwf	f	Move working register to file register f
nop	-	No Operation
retfie	-	Return from interrupt service routine and set global interrupt enable bit GIE
return	-	Return from Subroutine
sublw	k	Subtract W from literal

Figure 7: Assembly language commands for the PIC Microcontroller

4.2 Subroutines

The example below shows a subroutine which flashes a LED for a short amount of time. We will assume that the LED is connected to bit 0 of PORTA.

```
flash bsf PORTA, 0
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        call wait100ms
        bcf PORTA, 0
        return
```

The name of the subroutine is used at the top and then at the end the `return` keyword is used.

4.3 Setting port Directions

Example code to set the port direction is shown below. The code below will set all bits of PORTA to be input bits and all bits of PORTB to be output bits.

```
movlw b'11111111'
movwf TRISA
movlw b'00000000'
movwf TRISB
```