Introduction to Using the PIC16F877

Justin Rice

IMDL Spring 2002

Basic Specs:

- 30 pins capable of digital I/O
- 8 that can be analog inputs
- 2 capable of PWM
- 8K of nonvolatile FLASH memory
- 386 bytes of RAM
- 2 built in serial communications pins

Startup Costs:

The Board:

I was able to order all of these parts through Jameco (http://www.jameco.com)

Part	Cost
PIC16F877	$9.49
PIC Proto 64 Board	$16.95
8 MHz crystal oscillator	$0.79
40 pin wire wrap socket	$2.66
40 pin ZIF socket	$11.95

You will also need some resistors, capacitors, an LED, and a 5V regulator. You can get all of these parts in lab for free.

Software/Programming:

The programmer I used is the EPIC Programmer ($59.95). The compiler I used was the PIC BASIC PRO ($249.95). Both of these products are available at Jameco. I was able to borrow both of these from a friend and save a lot of money.

Microchip provides a free development environment and compiler for its assembly language and there are a few freeware C compilers. I did not play with any of these, so I am not really sure how good they are. I think that the senior design lab has a copy of PIC BASIC PRO that you may be able to use on their computers.

Setting up the board:

The instructions provided in documentation that comes with the PIC Proto 64 board make the initial setup quite easy. There are just a few parts to solder in.

The ZIF socket is more expensive than the PIC itself, but you really need to have it. I soldered the 40-pin wire wrap socket to the board and put the ZIF into that socket. With the EPIC programmer, you have to physically remove the PIC and put it in the programmer each time you program it. Without the ZIF socket, I would have been very worried about breaking or bending pins on the PIC each time.

ZIF Socket

The board provides a power bus along the top and a ground bus along each side, saving a bit on wire wrapping. To connect each device to the PIC I would solder some male headers on the prototyping area. I would wire wrap underneath the board to connect to the PIC socket.

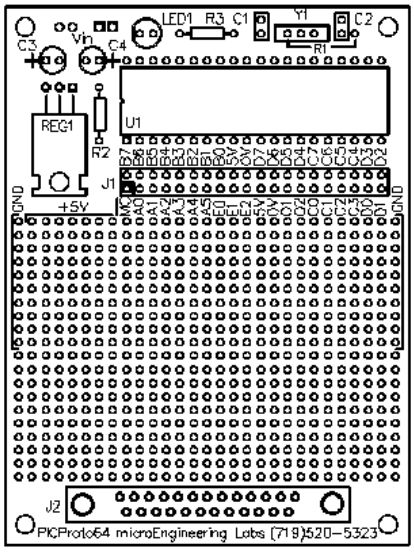
I had plenty of room at the bottom of the prototyping area to fit in a 555 timer and a microphone circuit.

PICProto64 Prototyping Board

$16.95

CO residents add .51 sales tax

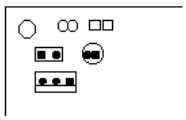
- High quality double-sided board
- Solder mask both sides
- More than 700 plated-through holes
- 4 mounting holes
- Overall dimensions 3" X 4"



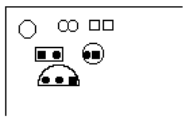
U1 - PIC16C64, 65, 67, 661, 662, 74 or 77
Y1 - crystal or ceramic resonator
C1, 2 - crystal capacitors
C3 - bypass capacitor
C4 - input capacitor
REG1 - 5 volt regulator
LED1 - LED
R1 - RC oscillator resistor
R2 - Master Clear resistor
R3 - LED series resistor
J1 - PIC I/O connector
J2 - DB9, 15, or 25

Vdd - plus 5 volt buss
GND - ground buss

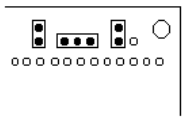
PARTS PLACEMENT:



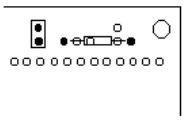
TO-220 Regulator
REG1 = 7805T
C3 = .1 - 10uf
C4 = .01 - .1uf



TO-92 Regulator
REG1 = 78L05
C3 = .1 - 10uf
C4 = .01 - .1uf



Crystal or Ceramic Resonator
Y1 = DC - 20MHZ
C1, 2 = 5 - 22pf



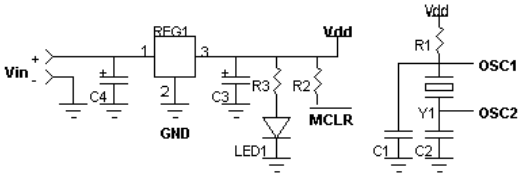
RC Oscillator
5k ≤ R ≤ 100K
C1 ≥ 20pf
C2 = none

ASSEMBLY NOTES:

Pin 1 of U1 is marked with a square pad.
Note polarity of Vin, REG1, LED1 and any polarized capacitors.

Don't forget to pull-up Master Clear to Vdd.
All unused inputs should be tied to Vdd or ground.

SCHEMATIC:



SOURCES:

PICmicro documentation is available from:
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler AZ 85224-6199
(602) 786-7200
(602) 786-7627 fax

microEngineering Labs, Inc.

Box 7532 Colorado Springs CO 80933
(719) 520-5323 (719) 520-1867 fax
http://www.melabs.com
email: support@melabs.com

Rev598

PIC Proto 64 Board Layout and Instructions

Serial Communication:

Serial communications on the PIC is very easy. Even though standard serial communications runs on +-12 volts, the PIC is able to communicate with most devices just using 0V and 5V. All that you need are two current limiting resistors.

To connect to my computer, I went to Radioshack and got a female DB-9 connector. I soldered 3 wires to the end of the connector that ran out to my PIC board. You can then plug in the female end to a standard serial cable and plug the other end of that to your computer.

The three wires you need to solder are Ground, Transmit, and Receive. Ground is pin 5 on the DB-9 connector, and can connect directly to the ground on your board. The transmit and receive lines both need a 1k current limiting resistor. Transmit is pin 2 on the DB-9, and I connected it to PortC.6. Receive is pin 3 on the DB-9, and I connected it to PortC.7.

PIC BASIC has several built in serial communication commands. The most important and easiest to use is Debug. Debug sends ASCII code to your serial transmit line. Here is some example code showing how to set up your PIC to use Debug:

```
' define crystal speed at 8 MHz
DEFINE OSC 8

' define the serial transmit pin to PortC bit 6
DEFINE DEBUG_REG PORTC
DEFINE DEBUG_BIT 6

' Define baud rate for serial debug
DEFINE DEBUG_BAUD 9600

' Define serial debug mode for inverted
DEFINE DEBUG_MODE 1
```

To send data to a computer you must use inverted mode. This interprets 0V as true and +5 as false. This is because in standard serial communications –12V is considered true and +12V is false. Setting the debug mode to 1 automatically inverts the data you send so you don’t need to worry about it.

Once you have Debug properly initialized, it is very easy to use:

```
Debug "Center= ", DEC IR_Center, 10, 13
```

This command will send the string “Center = “ followed by the decimal value of the variable IR_Center. The 10 and 13 that follow are the ASCII codes for carriage return and line feed, and are used as end of line characters.

A/D Conversion:

The PIC BASIC Pro language also has a built in command to read and convert an analog value. PortA and PortE are the 8 pins that can support analog to digital conversion. The following code sets up the analog ports for reading and converting.

```
'*****************************************************************************
' A/D System Definitions

' A/D Clock Selection
'       00 = FOSC/2
'       01 = FOSC/8
'       10 = FOSC/32
'       11 = FRC (clock derived from the internal A/D module RC oscillator)

' A/D Channel Selection – used in the ADCIN command
'       000 = channel 0, (RA0/AN0)
'       001 = channel 1, (RA1/AN1)
'       010 = channel 2, (RA2/AN2)
'       011 = channel 3, (RA3/AN3)
'       100 = channel 4, (RA5/AN4)
'       101 = channel 5, (RE0/AN5)
'       110 = channel 6, (RE1/AN6)
'       111 = channel 7, (RE2/AN7)
'*****************************************************************************

' define number of bits in result – max of 10
DEFINE ADC_BITS 8

' set clock source
DEFINE ADC_CLOCK 0

' sampling time in microseconds
DEFINE ADC_SAMPLEUS 1

' Set PORTA.0 - 4 as inputs
TRISA = %00001111

' Set PORTA to analog
ADCON1 = 2
```

To read in and convert an analog value on PortA.0 and store it in the variable IR_Left you would use the following command:

ADCIN 0, IR_Left

The channel definitions are listed above, and decimal values may be used to specify them.

PWM Commands:

PWM is very easy to generate on the PIC by simply modifying a few register values. Here is some example code for 38 kHz generation that I used to modulate IR. This PWM will run completely in the background and not effect the timing of other code.

```
TRISC.2 = 0             ' CCP1 (PortC.2 = Output)
PR2 = 52                ' Set PWM Period for approximately 38KHz
CCPR1L = 26             ' Set PWM Duty-Cycle to 50%
CCP1CON = %00001100     ' Select PWM Mode
T2CON = %00000100       ' Timer2 = ON + 1:1 prescale
```

Calculating PR2:

PR2 = ((clock speed) / (4 * TMR2 prescale value * (desired frequency))) - 1

For a %50 percent duty cycle, CCPR1L should be half of PR2. The other prescaler values can easily be found in the datasheet.

The actual PWM command in the PIC BASIC language is nearly useless since nothing else runs on the processor while you are generating that PWM.