

Raven Education Board User's Guide

Copyright 2000 © Raven Digital

Table of Contents

INTRODUCTION	1
FEATURES	2
PIC16F877 FEATURES.....	2
INSTALLATION	2
ADDITIONAL MATERIALS REQUIRED	3
SUGGESTED ADDITIONAL MATERIALS.....	3
BOARD LAYOUT	4
SCHEMATIC	5
POWER SUPPLY	5
POWER SUPPLY	6
PROCESSOR.....	7
DEMULTIPLEXER.....	8
THREE COLOR LEDs	9
LED BANK	10
SEVEN SEGMENT LEDs	11
ALPHANUMERIC DISPLAY	12
LIQUID CRYSTAL DISPLAY.....	13
PUSHBUTTONS	14
ROTARY ENCODER.....	15
KEYPAD	16
DIP SWITCH	17
RS-232 INTERFACE.....	18
PC DEVICES	19
A/D AND D/A CIRCUITRY	20
EXPANSION PORT.....	21
PROGRAMMER APPLICATION	23
PROGRAM OPERATION	24
READ OPERATION	24
VERIFY OPERATION	25
CONFIGURATION OPTIONS	26
SERIAL PORT MONITOR	27
DEMO PROGRAM	28
BOOTLOADER PROGRAM	29
SUGGESTED PROJECTS	32
EXPANSION PROJECTS.....	32

Introduction

Get on the fast track to real world embedded microcontroller programming experience. This platform provides practical training in embedded microcontroller concepts using the PIC16F877 microcontroller. The circuitry provides a rich bed of devices for learning both the software and hardware features of this microcontroller. The in-circuit programming capability allows for quick and easy development. An expansion port is available for adding custom circuitry. The board was designed with the hobbyist, student, and developer in mind.

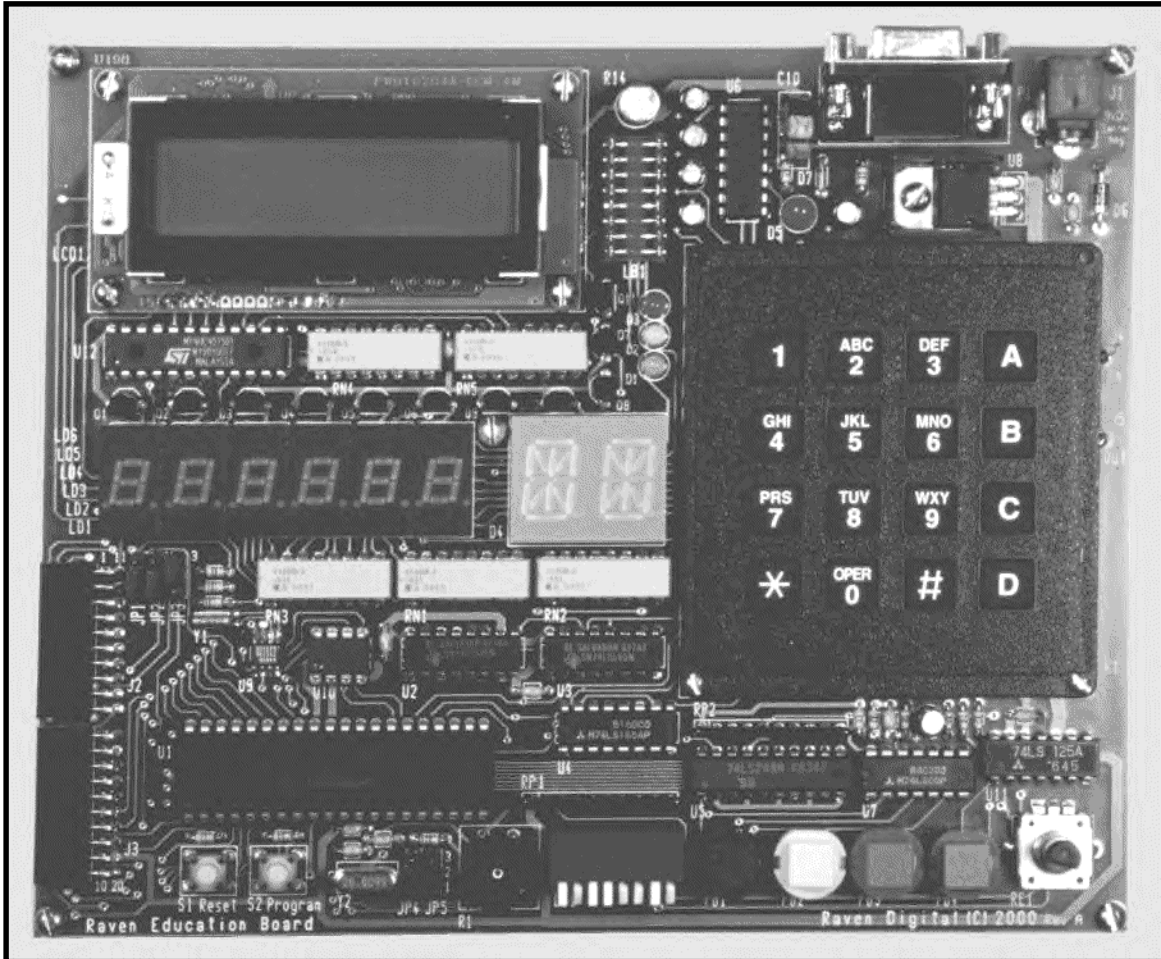


Image of Board

Contents of the Education Board Kit includes:

- Raven Education Board
- 3.5-inch diskette
- 9-volt DC power adapter (Input: 120VAC, 60Hz)
- 20 pin right angle male header x 2
- 6 foot RS-232 DE9 cable
- Printed manual

Features

- Socketed PIC16F877 included
- In-circuit programmable with included Windows® program
- 20 pin header x 2 expansion port
- 16 x 2 LED backlit LCD module
- Six 7-segment LEDs
- Two character alphanumeric LED display
- Socketed I²C 16K byte EEPROM (24LC128)
- I²C RTC/temperature sensor (DS1629)
- 16 key keypad
- 8 position DIP switch
- 4 pushbuttons
- Mechanical rotary encoder
- Phone jack for audio input (monophonic)
- Phone jack for audio output (monophonic)
- 8 LED bank for byte value display
- 3 (red, yellow, green) color LEDs
- Potentiometer for analog input
- RS-232 interface
- Reset and program pushbuttons
- Socketed microcontroller crystal
- Small size (7 1/8 by 5 7/8 inches)
- Includes power adapter
- Mounted on a sturdy acrylic panel
- Includes demo/test program
- Included RS-232 cable
- All of the above listed parts are included

PIC16F877 Features

Multi-channel 10 bit A/D, 10 bit PWM D/A, I²C capability, Brown out detection, USART, 16-bit timer, 8K FLASH program memory, Only 35 instructions to learn, Single cycle instructions, Harvard architecture, 256 bytes of EEPROM data memory, High sink/source current, Fully static design, Two capture/compare/PWM modules, In-circuit programmable, Multiple I/O ports, 368 bytes of RAM.

Installation

Host Computer System Requirements

- Microsoft® Windows® 95/98/ME or Windows® NT 3.51/4.0/2000
- One available RS-232 serial port (DB9)
- Mouse or other pointing device
- Available 3.5 inch disk drive

To install the files located on the disk, follow these steps:

1. Insert installation disk in Drive A:
2. Choose File | Run on the Windows Program Manager.
3. Type A:SETUP, and click OK.
4. Follow the on-screen prompts to finish the installation.

These files should appear on your hard drive after a successful installation

- Raven Programmer.exe Application for in-circuit programming of the PIC
- Raven Programmer.hlp Help file for Raven Programmer.exe
- Raven Education Board.pdf The file you are viewing
- Demo.asm Demo/Test program assembly code
- Demo.hex Demo/Test program hex code
- Uninstall files Files to uninstall application from your computer system
- Bootloader.hex Hex program code allowing the PIC to be programmed in-circuit

The PIC microcontroller is shipped with both the *Bootloader.hex* and *Demo.hex* code programmed into the chip.

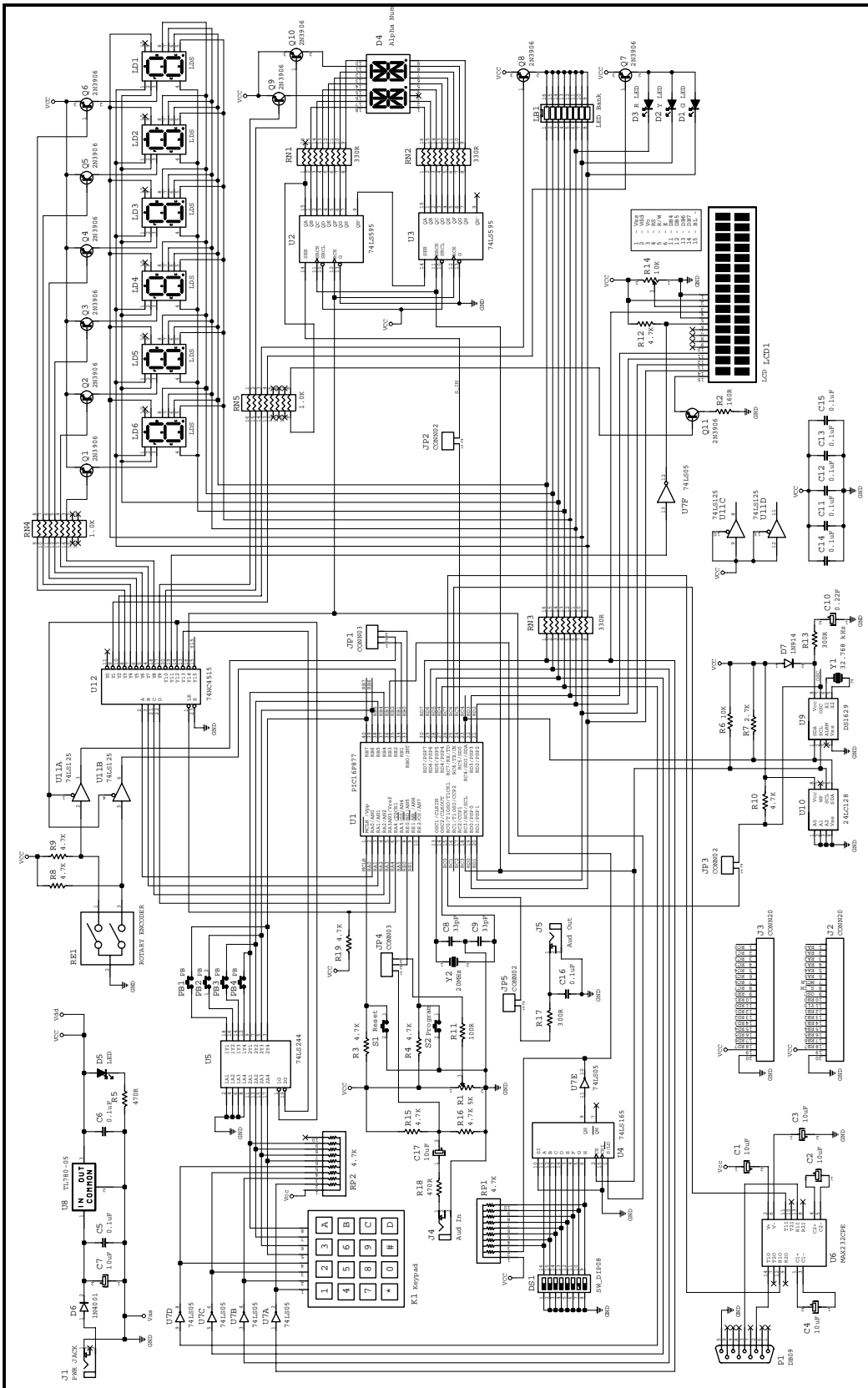
Additional Materials Required

- MPLAB – Assembler available free from the Planet Microchip® web site
- PIC16F87X Data Sheet - Available free from the Planet Microchip® web site

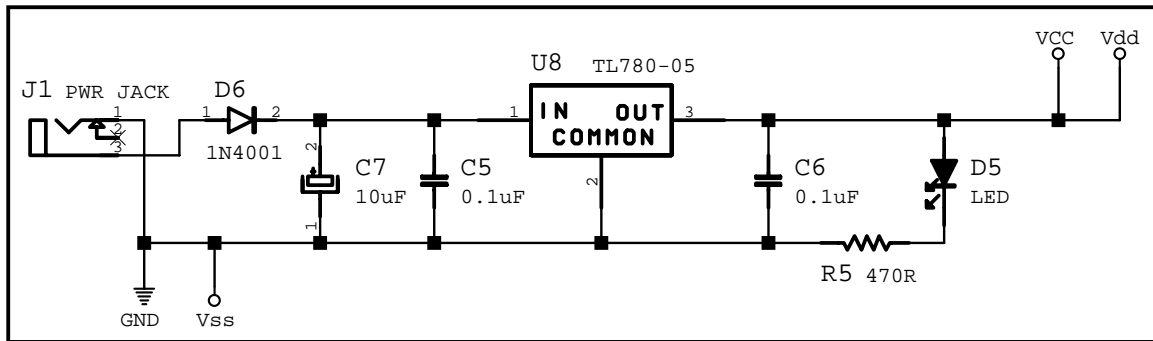
Suggested Additional Materials

- Dallas Semiconductor® DS1629 I²C RTC/temperature Sensor Data Sheet
- Microchip® 24LC128 I²C 16K byte EEPROM Data Sheet
- Microchip® Application Notes
- 74HC4515 1-of 16 Decoder/Demultiplexer Data Sheet
- 74LS595 Serial-in, Parallel-out Shift Register Data Sheet
- 74LS165 Parallel-in, Serial-out Shift Register Data Sheet
- 74LS244 3-State Bus Driver/Receiver Data Sheet
- Optrex® LCD Data Sheet
- ‘Design with PIC Microcontrollers’ book by John B. Peatman.
- ‘Programming and Customizing the PIC Microcontroller’ book by Myke Predko

Schematic



Power Supply



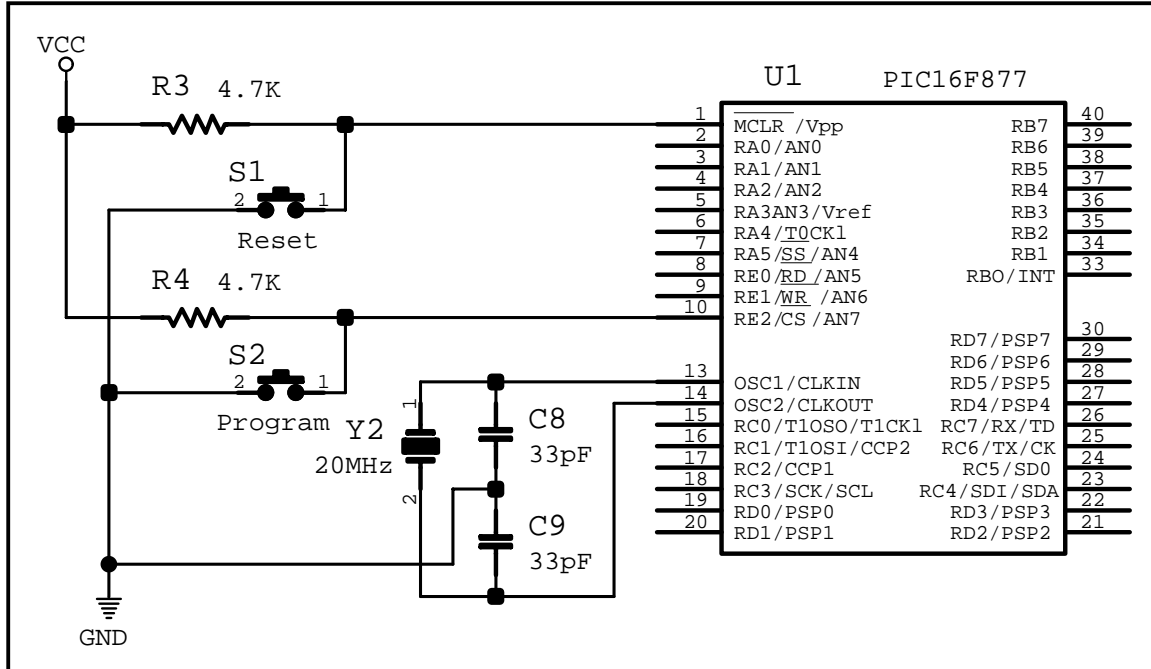
The power supply circuitry converts an unregulated DC voltage from a wall power adapter (suggested 9V and $\geq 500\text{mA}$) into a regulated 5-volt DC voltage needed for the rest of the circuit. The 1N4002 diode provides for polarity protection. The 7805 IC supplies the regulated voltage. The LED provides an indication that power is being supplied to the circuit. Since the power supply circuit does not contain an on/off switch, the wall power adapter plug should be disconnected from the power jack to power down the circuit.

Note: The power jack is a female 2.1mm x 5.5mm co-axial female type. A mating wall power adapter with negative center polarity should be used.

Note: Use a separate power supply with your expansion project circuitry when powering high current devices such as motors and the like.

Note: The power supply circuitry contains voltages that can damage other sections of the board if a jumper wire is accidentally connected to one of the exposed connections.

Processor



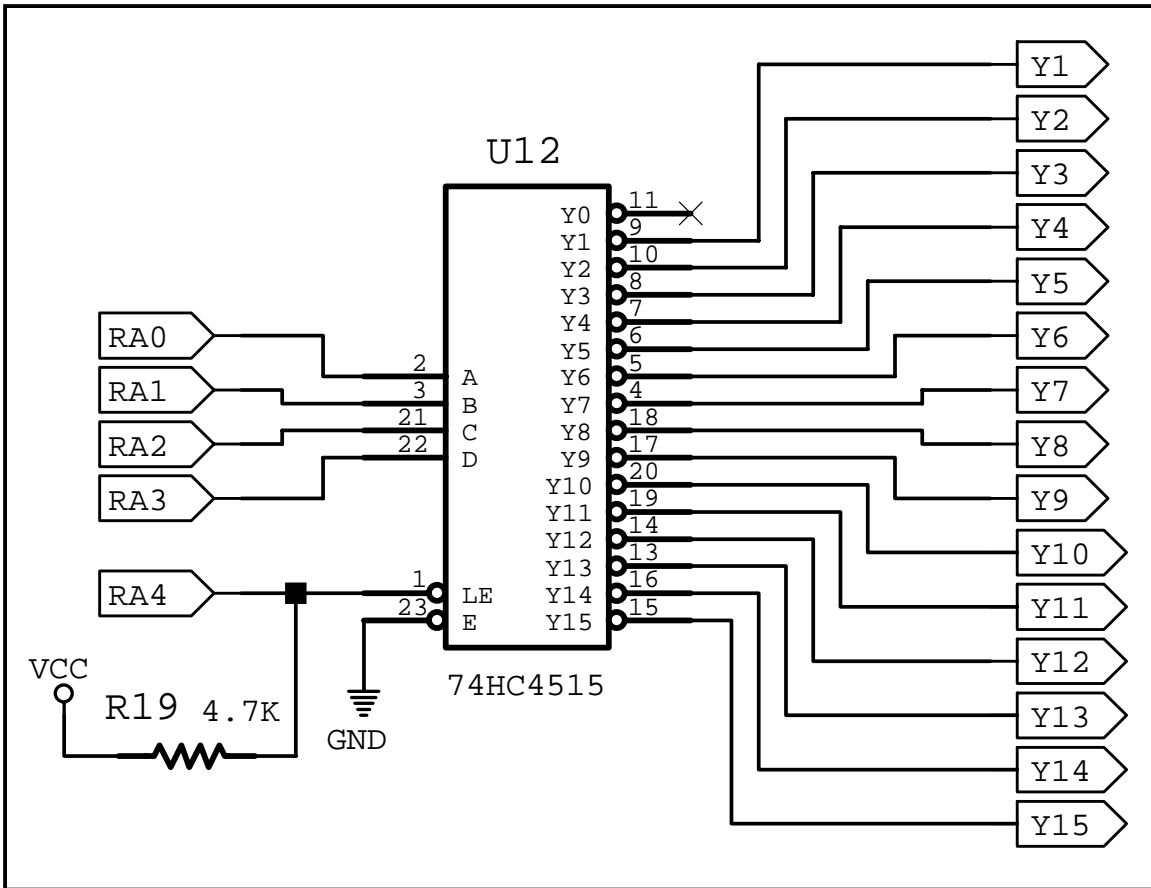
The processor circuitry contains the PIC16F877 microcontroller and some support circuitry. The crystal supplies the heartbeat to the chip, allowing it to operate at a precise rate. The *Reset* pushbutton allows for a manual reset of the microcontroller. The *Program* pushbutton allows for the in-circuit programming of the microcontroller, with the use of the internal bootloader program. Crystals other than the 20MHz crystal can be used to clock the microcontroller, but the 20MHz crystal must be used when performing in-circuit programming, reading, and verifying operations with the supplied Windows® program. PORTA is used mainly for control of the demultiplexer chip. PORTB is used with input devices. PORTC is used for its special functions. PORTD is used with output devices. PORTE has one pin connected to the *Program* pushbutton. The other two pins of PORTE are free for expansion projects.

The PIC microcontroller is shipped with both the *Internal.hex* and *Demo.hex* data programmed into the chip.

The PIC microcontroller configuration bits and device ID can only be accessed using an external programmer. The bootloader program cannot access the configuration bits and device ID. The PIC microcontroller is shipped with the configuration bits set to the following conditions: Oscillator: HS, Watchdog Timer: Off, Power Up Timer: On, Code Protect: 1F00-1FFF, Brown Out Detect: On, Low Voltage Program: Disabled, Data EE Protect: Off, Flash Program Write: Enabled, Background Debug: Disabled. The device ID is programmed with a value of \$7F7F7F7F.

Note: Do not deliberately configure PORTB as an output port due to the contention that will be created with some of the input devices. The devices can handle the contention, but the power supply circuit can be overloaded if it is also powering expansion circuitry.

Demultiplexer

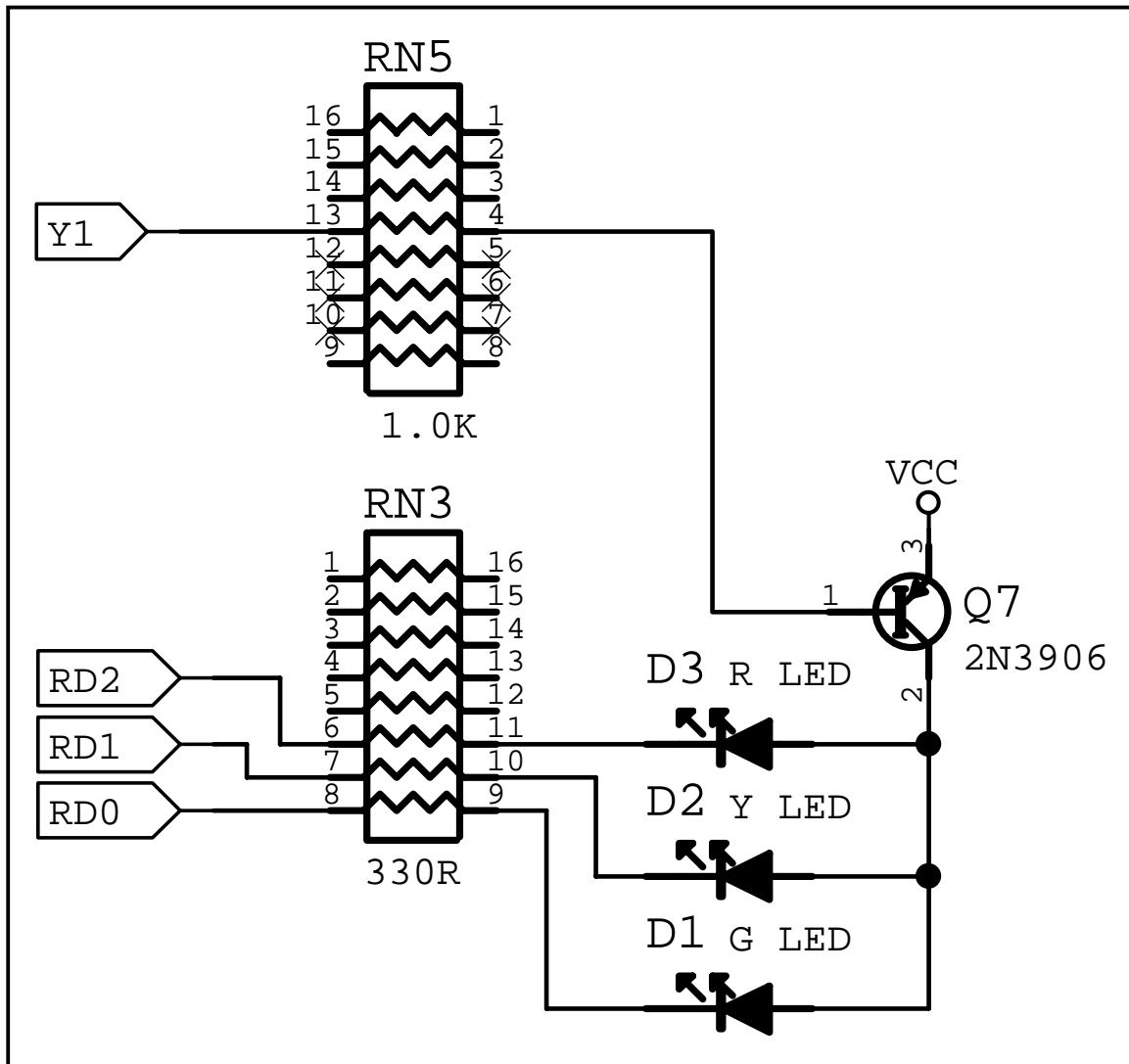


The 74HC4515 1-of-16 demultiplexer is used to control the mutually exclusive selection of the following devices: one of the 7 segment LEDs, DIP switch latching, three color LEDs, LED bank, rotary encoder, LCD, one character of the alphanumeric display, keypad, and pushbuttons. Only one of these devices can be selected at any particular moment. The Y15 output of the demultiplexer is available for expansion projects. The LE signal input to the demultiplexer allows latching of the demultiplexer output state.

Output Line	Selects	Output Line	Selects
Y0	No Connection	Y8	7 Seg Digit 6 (MSD)
Y1	Three LEDs	Y9	Alphanum Character 1
Y2	LED Bank	Y10	Alphanum Character 2
Y3	7 Seg Digit 1 (LSD)	Y11	LCD Enable
Y4	7 Seg Digit 2	Y12	Pushbuttons and Rotary
Y5	7 Seg Digit 3	Y13	Keypad
Y6	7 Seg Digit 4	Y14	Serial Registers
Y7	7 Seg Digit 5	Y15	Free

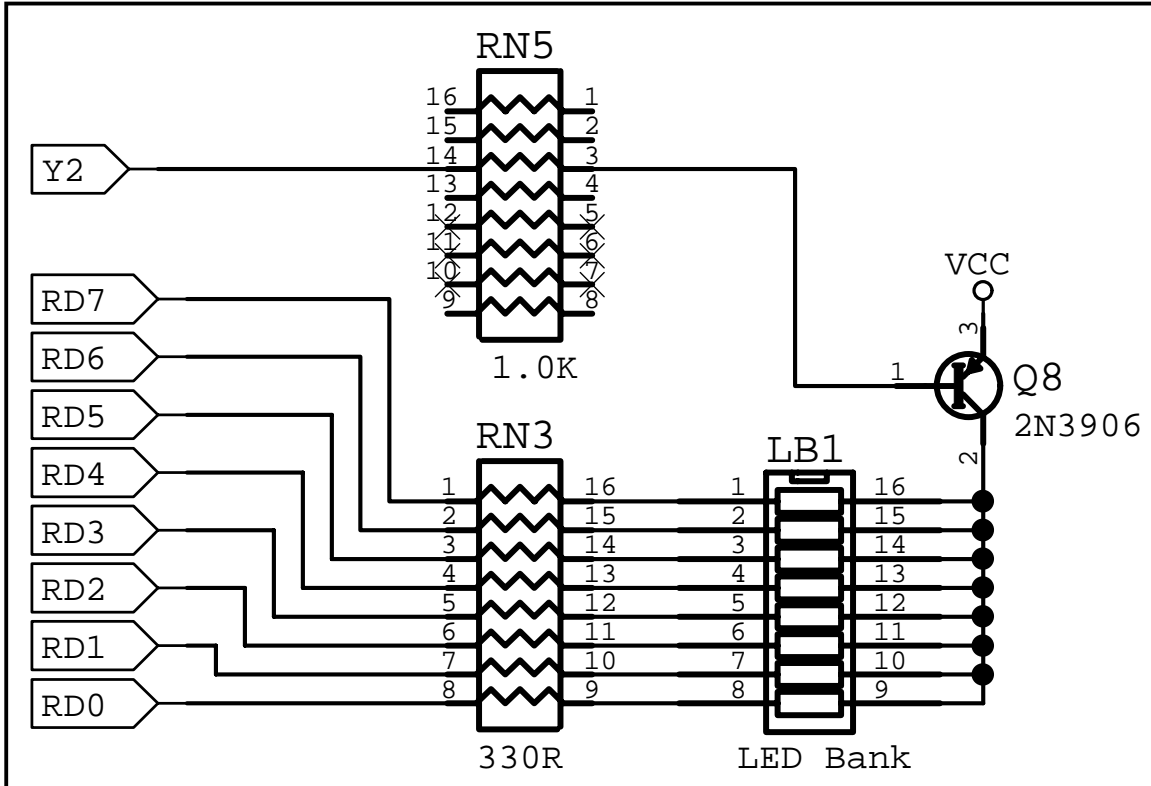
Demultiplexer Table

Three Color LEDs



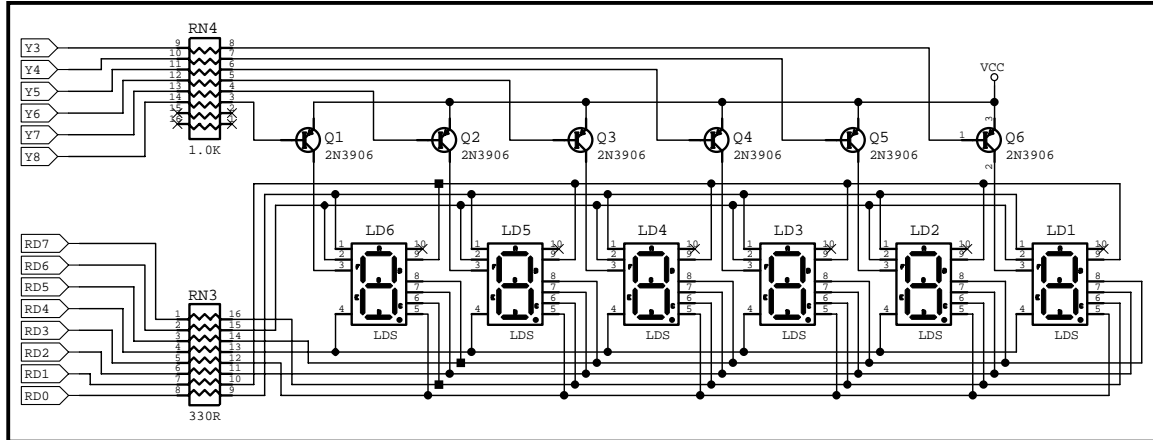
The three color LEDs (red, yellow, green) provide for a simple but colorful output indicator. These LEDs are useful status indicators. They are also useful in beginning programming projects where the control of an output port line is being tried.

LED Bank



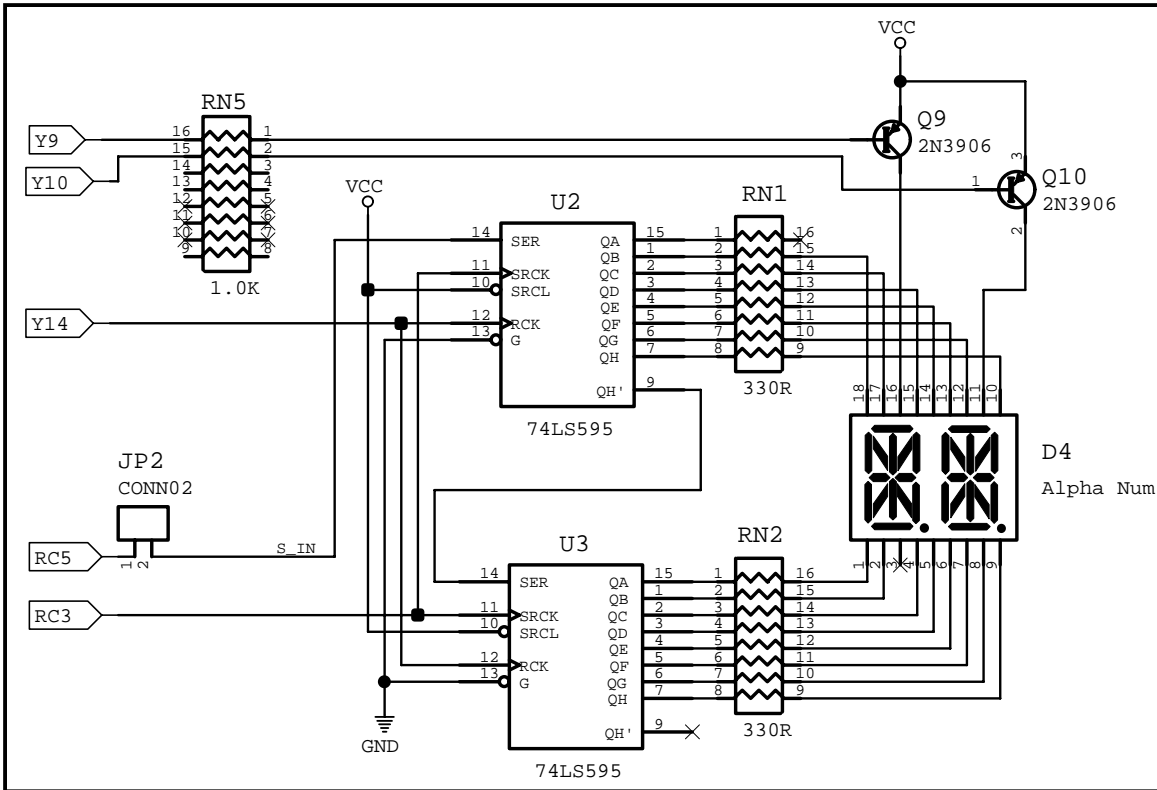
The LED bank is useful for displaying the value of a particular byte in the PIC RAM memory. The LED bank is also useful for viewing the effects of the arithmetic and logic instructions. It contains eight individually controllable LEDs.

Seven Segment LEDs



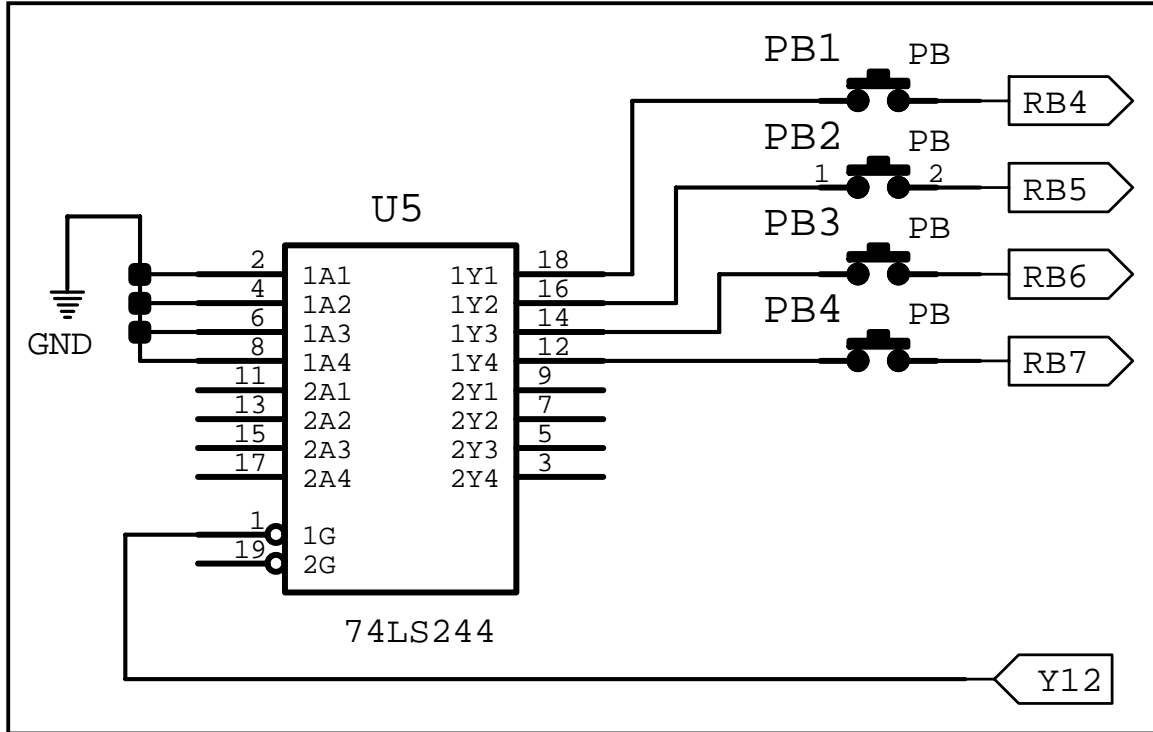
The six 7 segment LEDs are useful for displaying numeric data. All segments are individually controllable along with the right hand decimal points. The transistors control the supply to the common anode of each individual 7 segment LED.

Alphanumeric Display



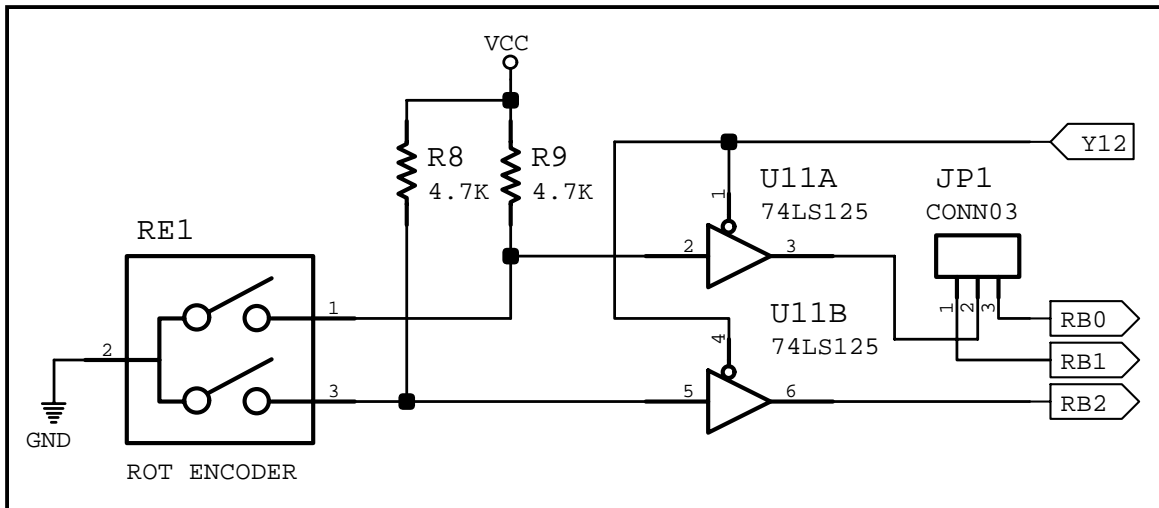
The alphanumeric display is useful for displaying two characters of data. Two double-buffered serial-in, parallel-out shift registers are used for transferring data from the PIC to the alphanumeric display. The Serial Peripheral Interface (SPI) should be used when communicating with the shift registers. Once the data has been sent to the registers, a strobe signal (Y14) is sent to transfer the data from the 595's shift register to the 595's output latch. Jumper JP2 can be left open to allow for the addition of off board shift registers for output expansion projects. One of the outputs from the shift registers is used to control the backlight of the LCD module (see LCD schematic).

Pushbuttons



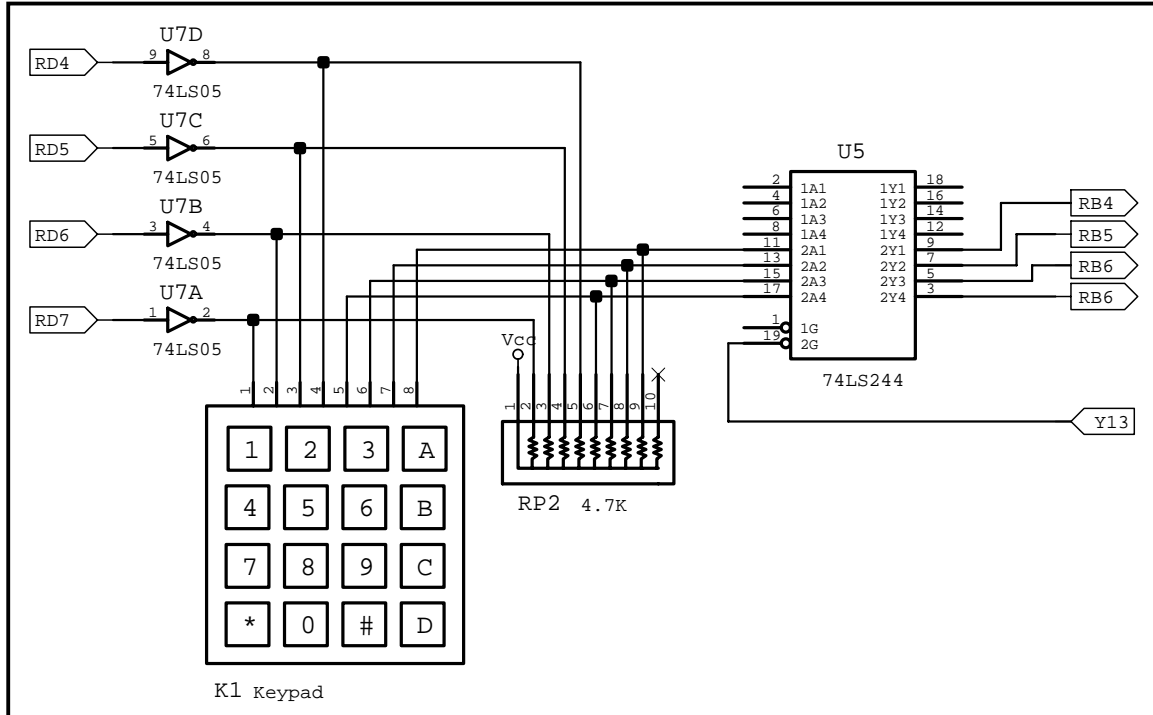
The four pushbuttons are useful for changing program flow based on one of the buttons being pressed. A 3-state buffer is used to isolate the buttons from each other. This 3-state buffer also allows expansion circuitry to be isolated from the four buttons.

Rotary Encoder



The mechanical rotary encoder is a useful input device when incrementing or decrementing a value. The device has 24 detented positions per revolution. The direction of rotation can be determined by detecting the phase difference between the two outputs of the encoder. The outputs from the rotary encoder are fed through 3-state buffers before reaching the PIC. Jumper JP1 allows one of the outputs of the encoder to be connected to the external interrupt input pin (RB0) of the PIC for illustrating external interrupt concepts. Connect the output to RB1 if RB0 will be used for expansion projects.

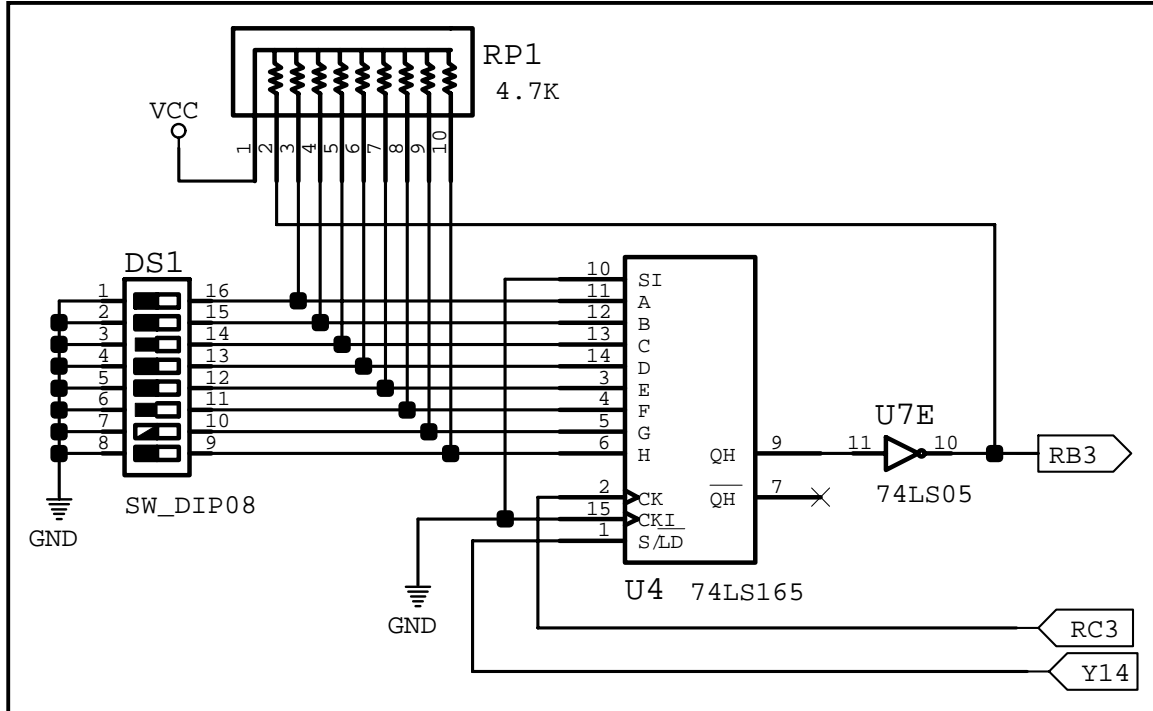
Keypad



The 4x4 keypad can be used for inputting numeric data or can be used as individual pushbuttons. Four lines leading to the keypad select an individual column. Four lines leading out of the keypad are used to sense for a key being pressed. Each key is isolated from the others through the use of open collector invertors and 3-state buffers.

Note: A delay may be required with the use of the keypad due to the slow operation of the open collector invertors.

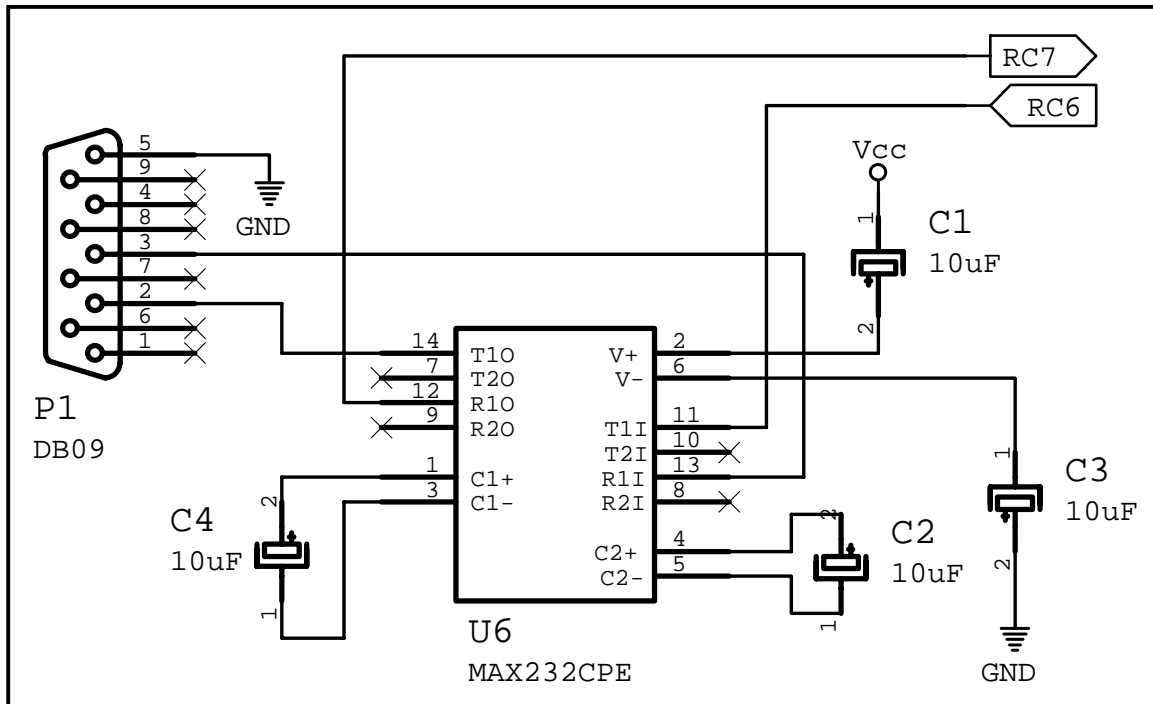
DIP Switch



The DIP switch circuit uses a parallel-in, serial-out shift register. The output of the shift register is connected through an open collector inverter before reaching the PIC. Bit banging programming should be used to read the state of the DIP switch. This circuit is useful for illustrating bit banging and shift register interfacing concepts.

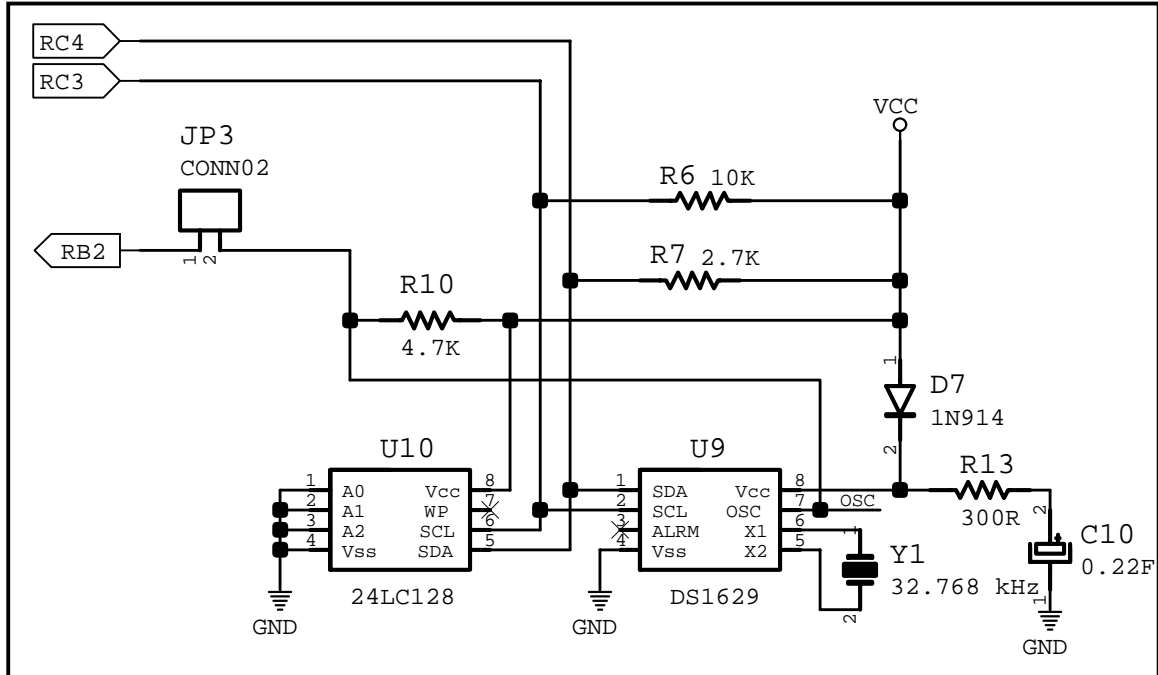
Note: A delay may be required with the use of the DIP switch (along with other devices on the board) due to the slow operation of the open collector inverter.

RS-232 Interface



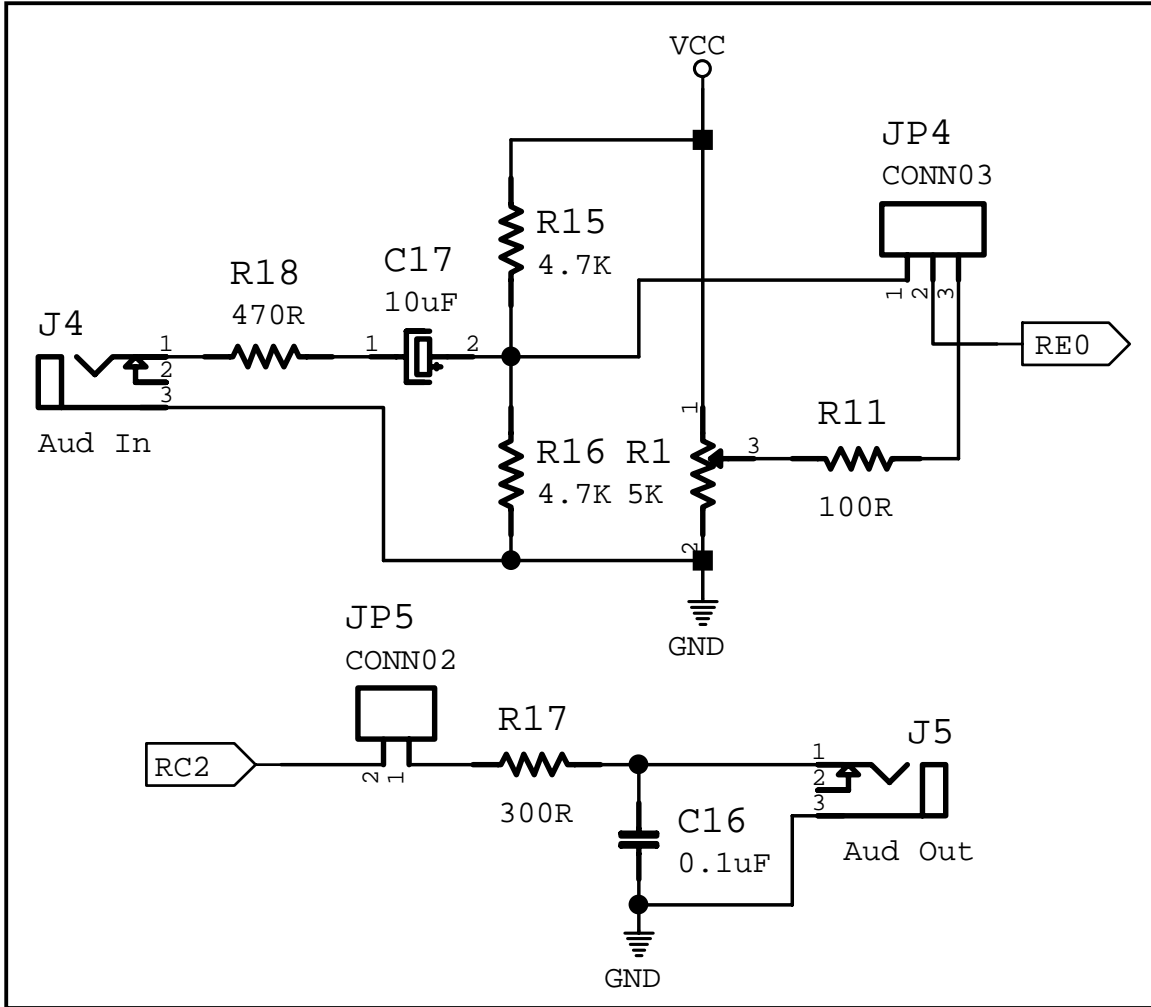
The MAX232 chip and its associated capacitors perform the level shifting required by the RS-232 specification. No hardware flow control (handshaking) is supported with this circuit. All flow control should be done in software. The RS-232 interface is also used for the in-circuit programming of the microcontroller with the use of the supplied Windows® program.

I²C Devices



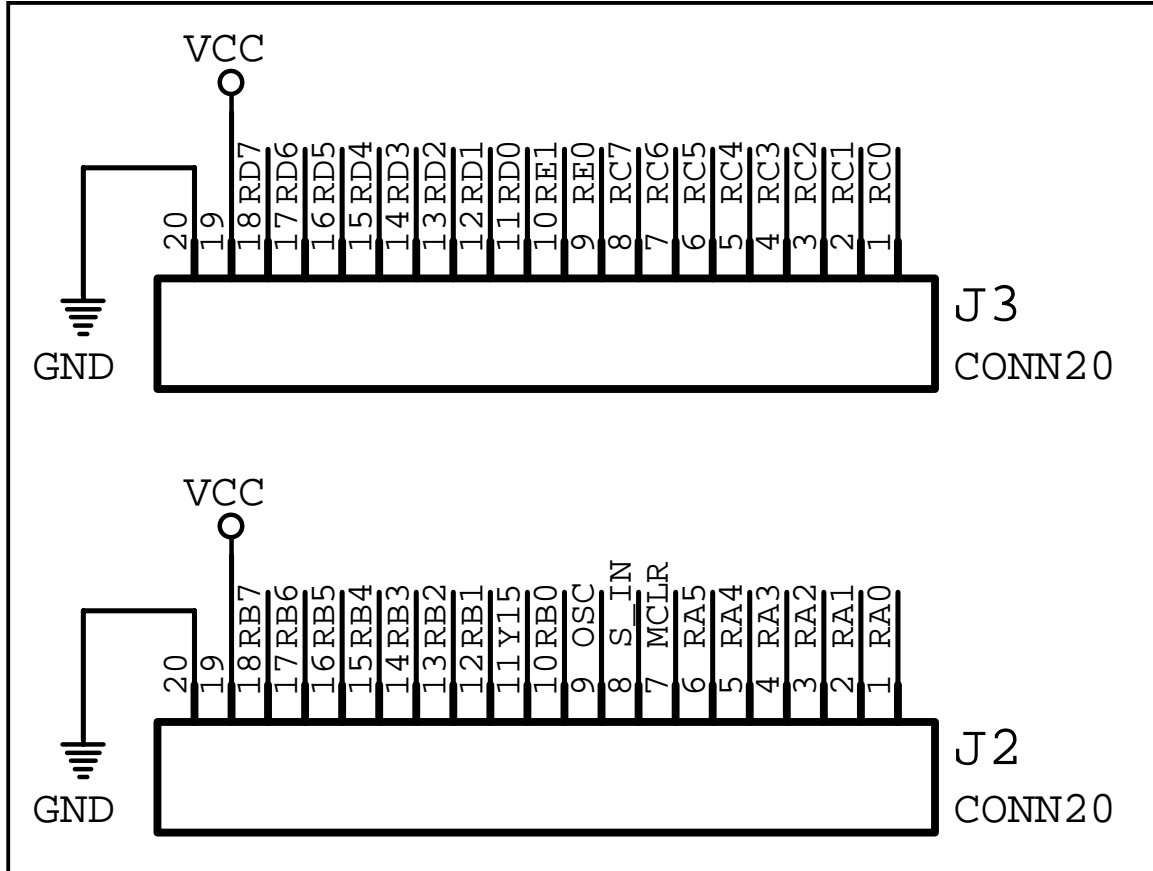
Two devices are included on the board to illustrate I²C bus concepts. The two I²C devices used are a Microchip 24LC128 16K byte serial EEPROM chip and a surface mount DS1629 RTC (Real Time Clock)/temperature sensor chip. The DS1629 chip also supplies a clock signal to an input of the PIC. This clock signal input can be connected or disconnected from this pin with jumper JP3. A 32.768 kHz cylinder-type crystal clocks the DS1629. A 0.22F capacitor allows for the retention of the DS1629 registers. Be sure to use low power modes of the DS1629 to extend the retention time. The EEPROM chip is socketed so a different capacity device can be used if desired.

A/D and D/A Circuitry



The 3.5mm phone jacks and 5K potentiometer are used to illustrate A/D and PWM D/A concepts. The audio input can be connected to the headphone output of a computer or portable audio player. Due to the lack of amplification in this circuit, the volume of the headphone output should be at or near maximum. The audio output can be connected to an amplified mini speaker system usually connected to the audio output of the computer. Do not connect headphones to the audio output due to the loud volume level that may occur. Stereo cables (available from Radio Shack®) should be used for the audio input and output connections. Although the circuitry does not provide for stereo capability, the audio connections use stereo jacks with the two channels shorted together (this is not shown on the schematic). The two 4.7K resistors are used to center the audio input voltage between 0V and 5V. Jumper JP4 selects between the 5K potentiometer and the audio input source. Jumper JP5 allows expansion circuitry to use the associated PWM output without interference from the audio output circuitry. C16 and R17 comprise a low-pass filter.

Expansion Port



The expansion port consists of two 20 pin female headers that allow for the addition of custom circuitry to the main Education Board circuit. Most of the pins of the microcontroller are available on the expansion port. Four of the pins allow for connection to the power supply. Several other connections are available. The expansion port can be connected to a custom board using the supplied mating male headers. Individual pins of the expansion port can also be connected to a solderless breadboard using solid 22 A.W.G. connection wire.

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	RC0	6	RC5	11	RD0	16	RD5
2	RC1	7	RC6	12	RD1	17	RD6
3	RC2	8	RC7	13	RD2	18	RD7
4	RC3	9	RE0	14	RD3	19	Vcc
5	RC4	10	RE1	15	RD4	20	Gnd

J3 Connections Table

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	RA0	6	RA5	11	Y15	16	RB5
2	RA1	7	MCLR	12	RB1	17	RB6
3	RA2	8	S_IN	13	RB2	18	RB7
4	RA3	9	OSC	14	RB3	19	Vcc
5	RA4	10	RB0	15	RB4	20	Gnd

J2 Connections Table

Programmer Application

This section discusses how to use the programmer application to interact with the Education Board. The application uses a serial port on the host computer. It supports COM1 through COM4. The supplied serial cable should be connected between the computer and the board before the application is started. When the application program is started, the selection of the port to use can be made by pressing the *Configure* button.

The programmer application allows for the programming, reading, and verifying of the flash program memory and data EEPROM memory located within the PIC chip supplied with the Education Board. The main window of the programmer allows for the selection of these functions.



All operations can be performed without removing the PIC from the Education Board circuit due to a bootloader program located in the upper 256 bytes of the PIC program memory. In addition to the upper 256 bytes, the bootloader also relies on the first three program memory locations (at addresses 0-2) to allow for the jump necessary to access the upper program memory locations. What action the code in the upper memory performs is determined by the state of the program button (S2) located on the Education Board during a reset. If the button is held down during a reset, the Education Board is prepared for communication with this application. If the button is not held down during a reset, the user code starting at address 3 will be executed.

Note: For your program code to be programmed successfully, make sure the code begins at address 3 (use `ORG 3` assembler directive) and make sure the code does not make use of the upper 256 bytes of program memory.

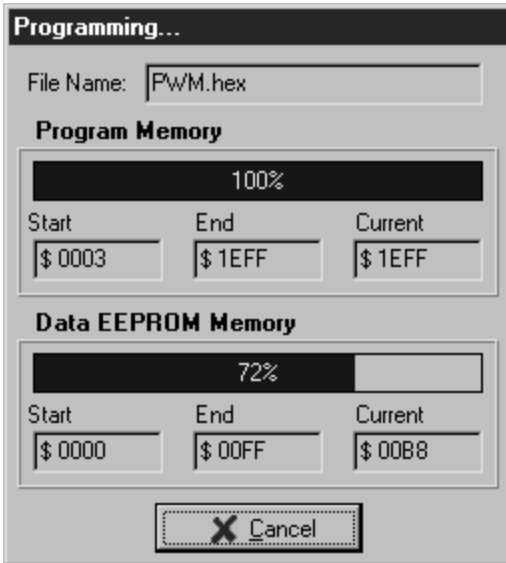
Note: This Application only supports the Intel Hex file format (INHX8M), not the Split Hex format (INHX8S) or the Hex 32 format (INHX32).

Note: A 20MHz crystal must be used with the microcontroller in the Education Board for proper operation with this application due to a required baud rate of 9600bps. A crystal of a different frequency can be used after the required operations are performed with the use of this application.

Note: The configuration bits and ID bits cannot be accessed by this application. Access to these bits requires an out-of-circuit programmer. The PIC microcontroller is shipped with the configuration bits set to the following conditions: Oscillator: HS, Watchdog Timer: Off, Power Up Timer: On, Code Protect: 1F00-1FFF, Brown Out Detect: On, Low Voltage Program: Disabled, Data EE Protect: Off, Flash Program Write: Enabled, Background Debug: Disabled.

Program Operation

The program operation transfers the instruction data contained in a hex file to the flash program memory area of the PIC. The program operation also transfers the data EEPROM data contained in a hex file to the data EEPROM memory area of the PIC.



First a hex file must be selected when the open file dialog box appears. Next a window indicating that the board must be prepared for programming will appear. To prepare the Education Board for programming, press both the reset button (S1) and the program button (S2) simultaneously. Then release the reset button before releasing the program button. If this is done properly, three LEDs on the Education Board should be blinking. Next a window indicating the progress of the programming process will appear. The *Start* indicator displays the lowest memory address of the PIC that can be programmed. The *End* indicator displays the highest memory address of the PIC that can be programmed. The

Current indicator displays the address of the instruction or data EEPROM being programmed. If the program operation is successful, a window will appear indicating this condition.

Read Operation

The read operation displays the data contained in the PIC flash program memory in either a hex listing format or a disassembly listing format. The read operation also displays the data contained in the PIC data EEPROM memory in a hex listing format.

First a window indicating that the board must be prepared for reading will appear. To prepare the Education Board for reading, press both the reset button (S1) and the program button (S2) simultaneously. Then release the reset button before releasing the program button. If this is done properly, three LEDs on the Education Board should be blinking. Next a window indicating the progress of the reading process will appear. The *Start* indicator displays the lowest memory address of the PIC that will be read. The *End* indicator displays the highest memory address of the PIC that will be read. The *Current* indicator displays the address of the instruction or data EEPROM being read.

When the flash program memory data is read from the PIC the data can be viewed in either a hex listing format or a disassembly listing format. The columns of the hex listing contain the following items: hex address, 8 hex formatted instructions, character representation of the least significant byte of the 8 instructions. The columns of the disassembly listing contain the following items: line number, hex instruction address, hex formatted instruction, disassembled instruction, instruction operand(s). The columns of

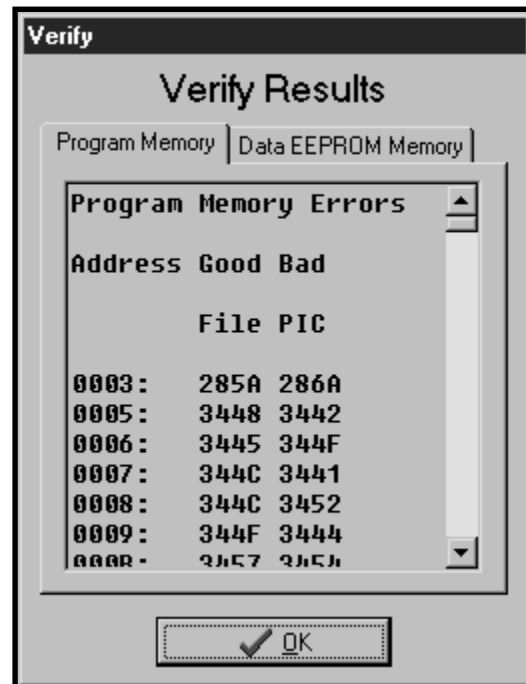
the data EEPROM listing contain the following items: data EEPROM address, 8 hex formatted data items, character representation of the 8 data items.



Verify Operation

The verify operation compares the data contained within a hex file to the data contained in the flash program memory of the PIC. The verify operation also verifies the data contained within a hex file to the data contained in the data EEPROM memory of the PIC.

First a hex file must be selected when the open file dialog box appears. Next a window indicating that the board must be prepared for verifying will appear. To prepare the Education Board for verifying, press both the reset button (S1) and the program button (S2) simultaneously. Then release the reset button before releasing the program button. If this is done properly, three LEDs on the Education Board should be blinking. Next a window indicating the progress of the reading process will appear. The *Start* indicator displays the lowest program memory address of the PIC that will be read. The *End* indicator displays the highest memory address of the PIC that will be read. The *Current* indicator displays the address of the instruction being read.

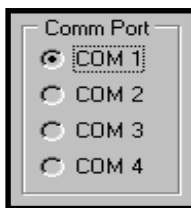


When the data is done being read and the verification is a success a window will appear indicating this condition. If the hex file does not agree with the PIC program memory, a window will appear indicating the addresses with discrepancies.

Configuration Options

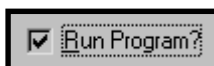
The configuration options of the application allow for the following selections:

- Selection of the serial communication port used to communicate with the Education Board.
- Selection offering the automatic running of the PIC user program after a program operation is completed.
- Selection offering the programming/verification of all flash program memory locations or only the program memory locations specified within the hex file.
- Selection offering the suppression or non-suppression of warnings relating to the hex file.
- Selection offering the suppression or non-suppression of the data EEPROM programming.



The communication port selection allows for the selection of the proper communication port connected to the Education Board. Selection of the correct communication port is necessary before communication with the Education Board can occur.

When the *Run Program?* checkbox is checked the PIC will automatically start running the user program after a program operation is complete. If this checkbox is left unchecked the Education Board will have to be reset manually before the user program can be run.



The *Program All?* checkbox allows unspecified hex file flash program memory locations to be programmed to a \$3FFF (all ones) value. If the checkbox is left unchecked, only the flash program memory locations specified in the hex file (that your program actually occupies) will be programmed. This checkbox also controls how the verify operation is performed. If checked, unspecified hex file flash program memory locations are set to a \$3FFF (all ones) value. If the checkbox is left unchecked, only the flash program memory locations specified in the hex file will be verified. Selecting this checkbox reduces the amount of programming time required when programming short programs.



When the *Hex File Warnings?* checkbox is checked the warnings associated with the hex file will be display. If unchecked the hex file warnings associated with the hex file will not be displayed.



When the *Program Data EEPROM?* checkbox is checked the data EEPROM will be programmed during a program operation. If unchecked the data EEPROM will not be programmed during a program operation.



Serial Port Monitor

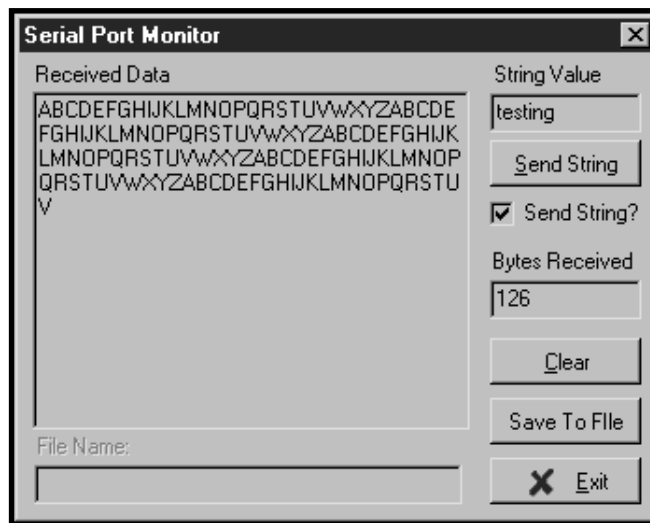
The serial port monitor is a simple utility capable of reading and writing data over the RS-232 interface. It can be used with custom serial port projects and can also be used to verify serial port operations when running the demo program.

Either single byte data can be sent or a string of bytes can be sent, depending on the state of the *Send String?* checkbox. When sending string data, a backslash character followed by two hexadecimal characters will send a byte with the hexadecimal value over the serial port. For example: sending 'This is a test\0d' will send the string 'This is a test' followed by an ASCII carriage return character.

Pressing the *Save to File* button will cause subsequent data received over the serial port to be written to a file. After selecting the name for the file, the file name will appear in the *File Name* field.

Press the *Clear* button to clear both the *Received Data* field and the *Bytes Received* field.

This utility operates at a fixed 9600bps baud rate.



Demo Program

The included demo program demonstrates the operation of most of the devices on the Education Board. It is also useful for performing a functional test of these devices. The PIC comes shipped with the demo program programmed into its program memory. The file demo.hex can be used to reprogram the demo program into the PIC after it is overwritten by other programs. The assembly file demo.asm contains the assembly code for the program. Use this file as a last resort if you have trouble programming a particular device. You will learn the most by overcoming programming difficulties on your own.

After powering up the board or pressing the reset button the demo program will perform the following operations:

- Display the scrolling text 'BOARD TEST' on the LCD.
- Display a three bit binary count using the three color LEDs.
- Output ASCII alphabetical characters over the RS-232 interface (A thru Z).
- Output a decimal count using the 7 segment LEDs.
- Display on the LED bank an XOR value dependent on the DIP switch settings.
- Display alphabetic characters on the alphanumeric display. JP2 should connect pins 1 and 2.
- Display a keypad key code on the 7 segment LEDs when a key on the keypad is pressed.
- Turn all LEDs on (7-segment segments, three color LEDs, LED bank LEDs, and alphanumeric segments) when PB1 is pressed and held down. The LCD backlight will turn off.
- Displays a received RS-232 byte value on the LED bank when PB2 is pressed and held down. Also displays the A/D converted potentiometer (R1) value on the 7 segment LEDs. JP4 should connect pins 2 and 3.
- Test the rotary encoder when PB3 is pressed. The red LED should light when turned one direction. The green LED should light when turned the other direction. JP1 should connect pins 1 and 2.
- Test the audio connections when PB4 is pressed. An audio source should be connected to J4 and an audio amplifier should be connected to J5 in order to perform this test. JP4 should connect pins 1 and 2. JP5 should connect pins 1 and 2.
- Tests the I²C devices when S2 is pressed. The LED bank should show a binary count of the address being accessed if the test is successful. If the test fails, the three LEDs will be lit.

After pressing PB3, PB4, or S2 the Education Board must be reset to exit the associated operation.

Bootloader Program

The bootloader code contained within the PIC is used to allow an external application to perform in-circuit program, read, and verify operations on the PIC. The included Windows® program uses the bootloader code for performing these operations. A custom application can be programmed on a different platform for performing these operations. A baud rate of 9600bps must be used. The bootloader program uses the following byte commands to control its operation:

HEX Code	Command	Operation
F0	Send Checksum	Returns the internal byte checksum the bootloader has been computing.
F1	Load Address	After sending this command the bootloader will expect the next two bytes to be an address that points to the location where an instruction or data EEPROM value will be programmed.
F2	Read Program	After this command is sent the bootloader will return byte after byte (low byte first) representing the internal program instructions, starting at address 0.
F3	Read EEPROM	After this command is sent the bootloader will return byte after byte representing the internal data EEPROM values, starting at address 0.
F4	Write Program	After sending this command the bootloader will expect the next two bytes to be a flash program memory instruction. The instruction will then be programmed. The address pointer is automatically incremented after this operation.
F5	Write EEPROM	After sending this command the bootloader will expect the next two bytes to be a data EEPROM value. The data EEPROM will then be programmed. The address pointer is automatically incremented after this operation.
F6	Run Program	Runs program starting at address 3
F7	End Programming	The bootloader is sent into an endless loop. It will not respond to further commands.
F8	Identification	Returns '877 x'. 'x' indicates the revision of the bootloader code. It is currently revision A. This command also initializes the address pointer and checksum to zero.

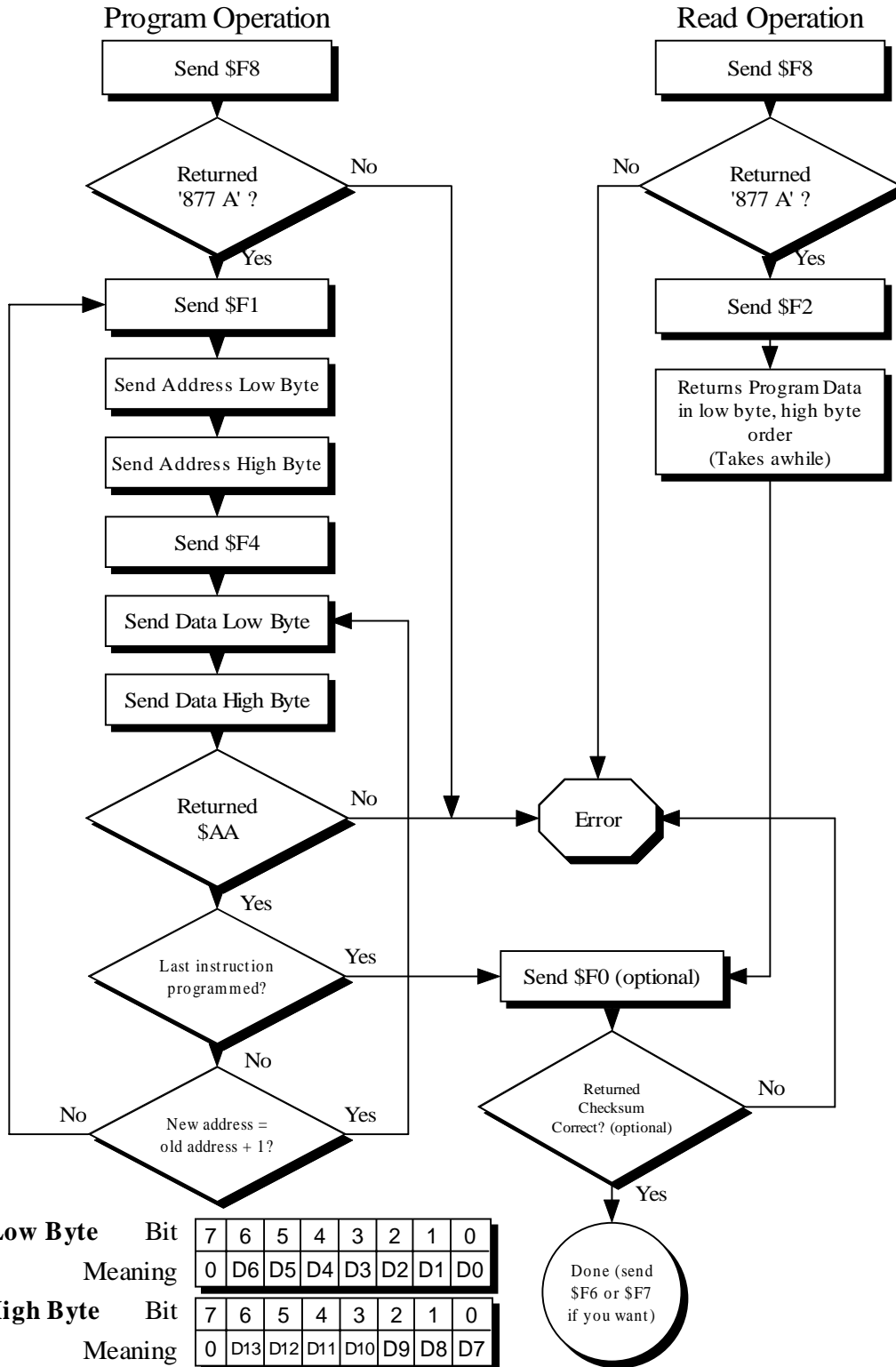
Bootloader Commands Table

- *Send Checksum* - After sending this command the bootloader returns a byte that is the byte sum of all the previous data sent to and received from the bootloader. The checksum is cleared after sending the Identification command and when the

bootloader code is first entered. The checksum can be used to verify proper communication with the bootloader program. Use of the checksum is optional.

- *Load Address* – After this command is sent two bytes should be sent. The first byte contains the 7 least significant bits of the address. The second byte contains the most significant bits of the address. The most significant bit of both bytes should be clear.
- *Read Program* – After this command is sent the bootloader will return byte after byte (low byte first) representing the flash program memory starting at address \$0 and ending at address \$1EFF.
- *Read EEPROM* – After this command is sent the bootloader will return byte after byte representing the internal data EEPROM memory values starting at address \$0 and ending at address \$FF.
- *Write Program* - After sending this command two bytes should be sent. The first byte contains the 7 least significant bits of the instruction. The second byte contains the most significant bits of the instruction. The most significant bit of both bytes should be clear. The instruction is then programmed. The write cycle takes a few milliseconds. The programmed instruction is verified after the program operation is complete. If the verification is successful the byte \$AA is returned. If the verification is unsuccessful the byte \$55 is returned. Programming of the instructions at addresses 0 through 2 is not allowed. These instructions are reserved for the bootloader code.
- *Write EEPROM* - After sending this command two bytes should be sent. The first byte contains the 7 least significant bits of the data EEPROM value. The second byte contains the most significant bit of the data EEPROM value. The most significant bit of both bytes should be clear. The data EEPROM value is then programmed. The write cycle takes a few milliseconds. The programmed value is verified after the program operation is complete. If the verification is successful the byte \$AA is returned. If the verification is unsuccessful the byte \$55 is returned.
- *Run Program* – After sending this command the bootloader code will exit and the instruction starting at address 3 will be executed.
- *End programming* – After sending this command the bootloader is sent into an endless loop. It will not respond to further commands.
- *Identification* – After sending this command the bootloader will return the string '877 x'. 'x' indicates the revision of the bootloader code. It is currently revision A. This command also initializes the address pointer and checksum to zero.

The following flowchart shows the sequence of operations required to perform a program operation and read operation on the flash program memory of the microcontroller from a host program.



Suggested Projects

- 1) LED blinker
- 2) LED flip flop
- 3) LED dimmer (vary duty cycle)
- 4) Binary counter (use LED bank)
- 5) Bit shifter (using LED bank)
- 6) Turn on all LED devices
- 7) Display DIP switch value using LED bank
- 8) Variable LED blink rate using DIP switch
- 9) Parity indicator (use DIP switch and LED)
- 10) 7-segment LED counter
- 11) 7-segment LED counter with zero blanking
- 12) Display keypad key code on 7-segment LEDs
- 13) Use TMR0 interrupt to produce 7-segment multiplexing
- 14) Decode DIP switch value to hexadecimal
- 15) Decode DIP switch value to decimal
- 16) Pushbutton debounce routine
- 17) Use PORTB interrupt on change to detect pushbutton press
- 18) Pseudo Random number generator. Display 0 – 65535 on 7-segment LEDs
- 19) Dancing LEDs (random patterns on 7-segment LEDs)
- 20) Digital dice
- 21) Craps game
- 22) Count from 0-FF (hex) using alphanumeric display
- 23) Display alphabet on alphanumeric display
- 24) Create a macro that uses arguments
- 25) Use direct, indirect, and relative addressing modes in a program
- 26) Create program that uses all four pages of program memory
- 27) Detect external interrupt using rotary encoder connected to RB0
- 28) Detecting PIC Interrupts (Light a LED for each interrupt case)
- 29) Use sleep instruction, and then wake processor
- 30) Create a table longer than 256 bytes
- 31) Write accurate millisecond delay routine (1-255 millisecond delay based on W register setting)
- 32) Stop watch (millisecond accuracy)
- 33) Reflex game – how fast can you press the button
- 34) Base number converter (bin, hex, dec)
- 35) Simple calculator (+, -, *, /)
- 36) Display potentiometer setting on 7-segment LEDs
- 37) Send characters over RS-232 interface and display on computer
- 38) 'Hello World' - using LCD
- 39) LCD character select – use DIP switch
- 40) Scrolling marquee – using LCD
- 41) Display keypad key press on LCD
- 42) Receive characters over RS-232 interface and display on LCD
- 43) Light red or green LED based on rotary encoder direction
- 44) LCD character select – use rotary encoder
- 45) Inc/Dec 7-segment value using rotary encoder
- 46) Scroll LCD cursor using rotary encoder
- 47) Write custom characters to LCD
- 48) Combination lock using keypad
- 49) Combination lock using rotary encoder
- 50) Count pulses from the RTC clock signal
- 51) Use capture and compare modes of CCP module
- 52) Audio frequency sweep – from low to high
- 53) Siren – like a police siren
- 54) Metronome
- 55) Audio white noise generator
- 56) Tone generator – sine, triangle, square, sawtooth
- 57) Music Box – have it play a tune
- 58) VU meter using LED bank
- 59) VU meter using LCD
- 60) Audio peak level meter
- 61) DTMF tone encoder using keypad
- 62) Interface to internal EEPROM (read/write data)
- 63) Power up counter – using internal EEPROM
- 64) Interface to external EEPROM (read/write data)
- 65) Digital thermometer
- 66) Temperature data logger – use external EEPROM
- 67) Simple clock (display hour, minutes, seconds) using RTC
- 68) Clock with alarm (hours, minutes, seconds) using RTC
- 69) Configure RTC chip to use minimal power. Determine how long RTC data can be retained without power.
- 70) Electronic organ – using keypad and audio output
- 71) Smart electronic organ with playback capability – using keypad and external EEPROM
- 72) Feed audio from audio input jack to audio output jack
- 73) Feed audio from audio input jack to audio output jack with rotary encoder as volume control
- 74) Morse code keyer – computer as keyboard
- 75) Memo pad – using LCD, keypad, and external EEPROM
- 76) Create computer program that reads/writes internal EEPROM data over RS-232 port
- 77) Transmit data over RS-232 port to computer using highest rate possible without dropping data
- 78) Read flash program memory and send data over RS-232 interface. Display data using custom computer program
- 79) Frequency counter
- 80) Simple voltmeter (one range)
- 81) Audio oscilloscope – using computer program to display waveform received from RS-232 port

Expansion Projects

- 1) Interface to additional LEDs
- 2) Interface to a relay driver
- 3) Interface to a stepper motor
- 4) Interface to an infrared device
- 5) Interface to a graphic LCD module
- 6) Interface to an additional PIC device
- 7) Interface to a digipot
- 8) Interface to a serial-in, parallel-out shift register using the SSP
- 9) Interface to a SPI device
- 10) Interface to a Microwire device
- 11) Interface to a USB device
- 12) Interface to a dynamic RAM chip
- 13) Interface to a 12-bit A/D device
- 14) Interface to two potentiometers
- 15) Interface to a computer keyboard