Interfacing to the Parallel Port

While there is a choice of ports on the PC for amateur use and direct interfacing, the parallel port is probably the simplest. With eight data wires, several control wires and bidirectional capability, it offers a convenient way to get information in and out of a PC. The examples in the next two sections use an older software language, *BASIC* or *GWBASIC* to get information in and out. Newer languages can be used, but several varieties of *BASIC* are available on the Internet at no cost. The learning curve for someone who has never used a programming language is very short — usually a matter of a few minutes. The two examples that follow interface single chip analog-to-digital and digital-to-analog converters to the parallel port of a PC.

Single-Chip Dual-Channel A/D

In this analog world, often there is need to measure an analog voltage and convert it to a digital value for further processing in a PC. This single chip converter and accompanying software performs this task for two analog voltages.

Circuit Description

The circuit consists of a single-chip A/D converter, U2, and a DB-25 male plug (**Fig A**). Pins 2 and 3 are identical voltage inputs, with a range from 0 to slightly less than the supply voltage V_{CC} (+5 V). R1, R2, C3 and C4 provide some input isolation and RF bypass. There are four signal leads on U2. DO is the converted data from the A/D out to the computer; DI and CS are control signals from the computer, and CLK is a computer-generated clock signal sent to pin 7 of U2.

The +5-V supply is required. It may be obtained from a +12-V source and regulator U1. Current drain is usually less than 20 mA, so any 5-V regulator may be used for U1. The power supply ground, the circuit ground and the computer ground are all tied together. If you already have a source of regulated 5 V, U1 is not needed.

In this form the circuit will give you two identical dc voltmeters. To extend their range, connect voltage dividers to the input points A and B. A typical 2:1 divider, using 50-kΩ resistors, is shown in the figure. Resistor accuracy is not important, since the circuit is calibrated in the accompanying software.

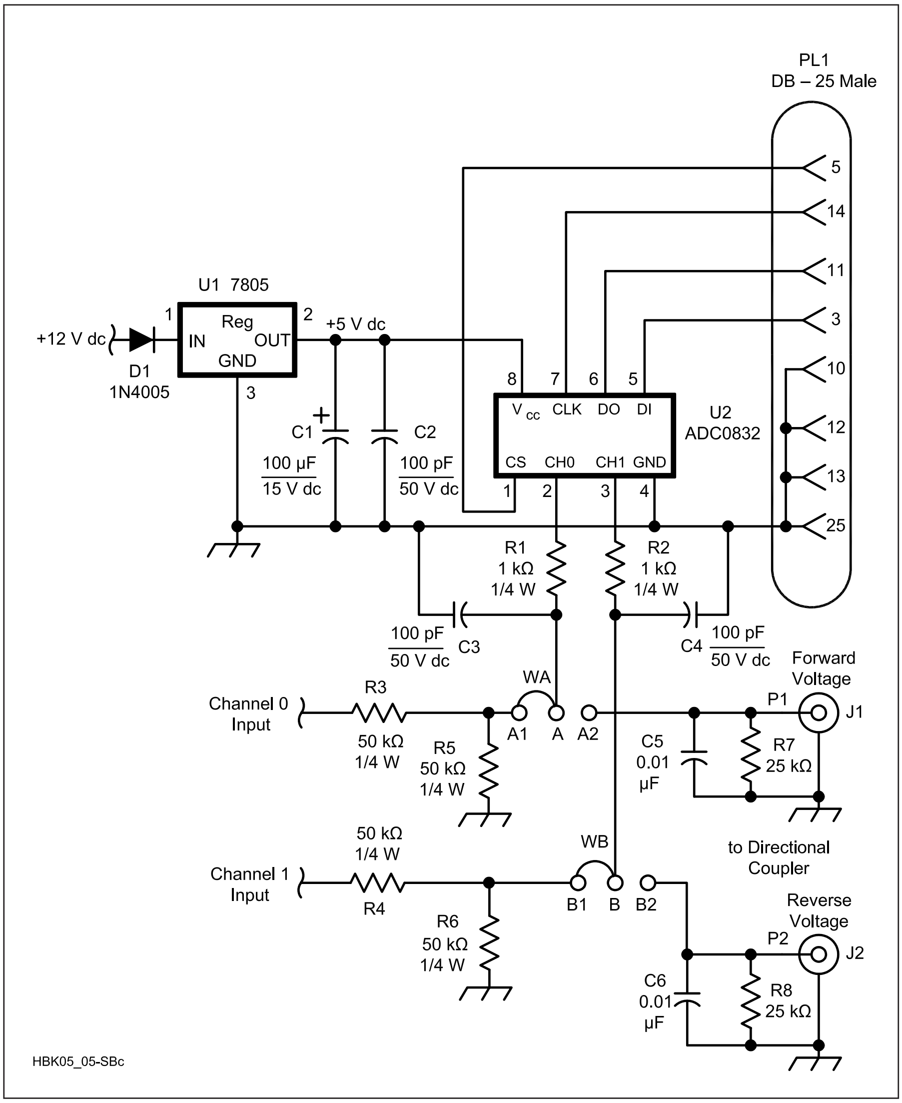


Fig A — Only two chips are used to provide a dual-channel voltmeter. PL1 is connected through a standard 25-pin cable to your computer printer port. U2 requires an 8-pin IC socket. All resistors are ¼ W. You can use the A/D as an SWR display by connecting it to a sensor such as the one shown in Chapter 19 of this *Handbook* (Tandem Match Wattmeter project). A few more resistors are all that are needed to change the voltmeter scale. The 50-kΩ resistors from 2:1 voltage dividers, extending the voltmeter scale on both channels to almost 10 V dc.

Software

The software, *A2D.BAS*, can be found on the *Handbook* CD. It includes a voltmeter function and an SWR function. It is written in *GWBASIC* and saved as an ASCII file. Therefore, you can read it on any word processor, but if you modify it, make sure you resave it as an ASCII file. It can be imported into *QBasic* and most other *BASIC* dialects.

The program was written to be understandable rather than to be highly efficient. Each line of basic code has a comment or explanation. It can be modified for most PCs. The printer port used is LPT1, which is at a hex address of 378h. If you wish to use LPT2 (printer port 2), try changing the address to 278h. To find the addresses of your printer ports, run *FINDLPT.BAS* (also included on the *Handbook* CD).

A2D.BAS was written to run on computers as slow as 4.7-MHz PC/XTs. If you get erratic results with a much faster computer, set line 1020(CD=1) to a higher value to increase the width of the computer-generated clock pulses.

The software is set up to act as an SWR meter. Connecting points A and B to the forward and reverse voltage points on any conventional SWR bridge will result in the program calculating the value of SWR.

Initially the software reads the value of voltage at point A into the computer, followed by the voltage at point B. It then prints these two values on the screen, and computes their sum and difference to derive the SWR. If you use the project as a voltmeter, simply ignore the SWR reading on the screen or suppress it by deleting lines 2150, 2160 and 2170. If the two voltages are very close to each other (within 1 mV), the program declares a bad reading for SWR.

Calibration

Lines 120 and 130 in the program independently set the calibration for the two voltage inputs. To calibrate a channel, apply a known voltage to input point A. Read the value on the PC screen. Now multiply the constant in line 120 by the correct value and divide the result by the value you previously saw on the screen. Enter this constant on line 120. Repeat the procedure for input point B and line 130.

D to A CONVERTERS — CONTROLLING ANALOG DEVICES

The complement to A/D converters is D/A (digital-to-analog) converters. Once there is a digital value in your PC, a D/A will provide an analog voltage proportional to the digital value. Normally the actual value is scaled. As an example, an 8-bit converter allows a maximum count of 255. If the converter is set up with a +5 V dc reference voltage, a maximum value digital value of 255 would result in a D/A output value of 5 V. Lower digital inputs would give proportionally lower voltages.

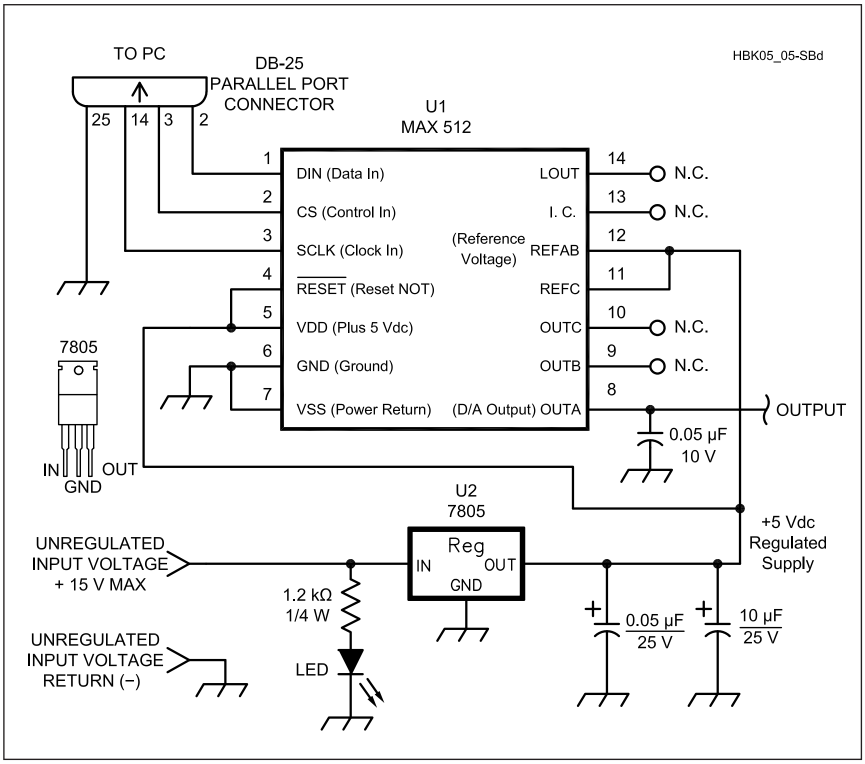


Fig B — Only three wires and a ground lead are needed to connect the converter to your PC.

Circuit Description

This project is the complement of the parallel port A/D converter described earlier. It takes a digital number from the computer, and converts it to a voltage from 0 to 5 V dc. Only one chip, the MAX 512, is required. It operates from a 5-V supply and is connected to the computer by a standard DB-25 parallel port connector. The chip may be ordered from Digi-Key, Allied Electronics and other ham suppliers as MAX512CPD-ND. The voltage regulator in **Fig B** provides the 5 V source required to power the chip.

Software

The software needed to run the chip, *D2A.BAS*, can be found on the *Handbook* CD. It is about 60 lines long, fully commented and written in *GWBASIC*, so it may be readily modified. The parallel port address is defined on line 105 as PORTO=&H378. Your computer may use a different address. To find the correct address, run *FINDLPT.BAS*.

The program takes the value AIN from the keyboard (line 230), converts it to a number between 0 and 255, and then sends it out as a serial word to the DIA chip. If you would like to use the project with another program, use your other program to set AIN to the value you want to generate, and then run this program as a subroutine.

At the end of the program is the clock pulse subroutine. In the event your computer is too fast for the converter chip, you can stretch the clock pulses by changing CD in line 5010 to a value greater than the default value of 1.

Applications

This circuit provides the capability of setting a voltage under computer control. It can be calibrated to match the power supply and the actual chip used. Tests with several chips showed an error of 25 mV or less over the range of 0 to 5 V dc output. — *Paul Danzer, N1II*

ARRL Periodicals Archive – Search Results

A membership benefit of ARRL and the ARRL Technical Information Service

ARRL Members: You may print a copy for personal use. Any other use of the information requires permission (see Copyright/Reprint Notice below).

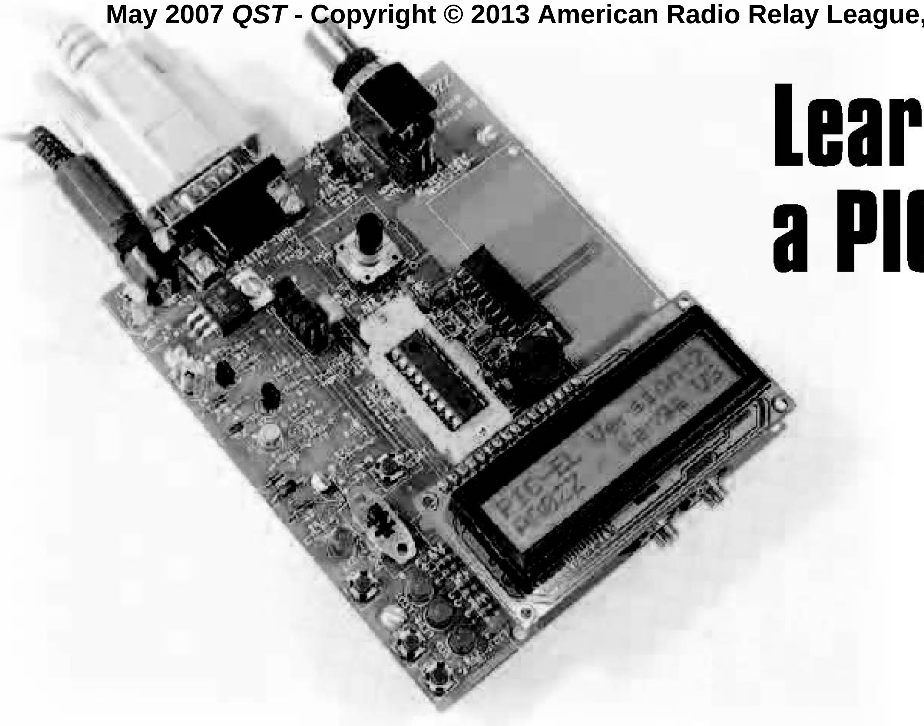
Need a higher quality reprint or scan? Some of the scans contained within the periodical archive were produced with older imaging technology. If you require a higher quality reprint or scan, please contact the ARRL Technical Information Service for assistance. Photocopies are $3 for ARRL members, $5 for nonmembers. For members, TIS can send the photocopies immediately and include an invoice. Nonmembers must prepay. Details are available at www.arrl.org/tis or email photocopy@arrl.org.

QST on CD-ROM: Annual CD-ROMs are available for recent publication years. For details and ordering information, visit www.arrl.org/qst.

Non-Members: Get access to the ARRL Periodicals Archive when you join ARRL today at www.arrl.org/join. For a complete list of membership benefits, visit www.arrl.org/benefits.

QST Issue: May 2007
Title: Learning to PIC with a PIC-EL -- Part 1
Author: Craig Johnson, AA0ZZ

Click Here to Report a Problem with this File



Learning to PIC with a PIC-EL — *Part 1*

This easy to duplicate prototype board for peripheral interface controllers will make it easy to get to know them!

Craig Johnson, AAØZZ

Have you ever wondered what makes all the little devices around your house that seem to have a mind of their own tick? You set the controls for your electric coffee pot, microwave oven, conventional oven, heating/cooling thermostat, DVD player, car's temperature controls, mileage indicators, trip odometer, etc. You look at your fancy new radio that you just bought and realize that there must be a lot of "brain power" hidden inside of it, too. How can you change bands, change operating modes, save a frequency in memory and change the speed of the CW keyer so quickly and effortlessly? In the past, devices like these were controlled by discrete electronics and later by integrated logic devices. Today many of these devices are controlled by little microcontrollers. They are simple and inexpensive. With a little effort you can learn how these microcontrollers work and how to put them to work in a new application that you dream up.

Ready to Try?

Peripheral inteface controllers (PICs) have been around for many years and are still very popular, in spite of the arrival of a number of new microcontroller families. One of the reasons PICs are so popular is because there are so many examples of projects that use them. Unfortunately, much of the code is complicated and hard to understand, especially for beginners. On the other hand, with a bit of determination and perseverance, trademarks of amateurs today, many have found PICs to be very useful in a variety of applications such as CW keyers, frequency counters, direct digital synthesis (DDS) controllers, receivers and transmitters controllers, repeater controllers, power meters, antenna controls and station control. The list seems endless.

What is the difference between a microcontroller such as a PIC and a microprocessor such as a Pentium in a personal computer (PC)? One major difference is that a microcontroller could be thought of as a complete stand-alone computer on a chip, since it contains a central processing unit (CPU), memory and input/output (I/O) interface. On the other hand, a Pentium microprocessor in a PC requires external memory modules and I/O processing components before it can function as a computer or controller. This is not to say that the PIC is better or more powerful, of course, but each type must be evaluated and used for its intended purpose. A PIC microcontroller is very sufficient and capable of controlling many wonderful Amateur Radio applications.

A wide variety of powerful, inexpensive PIC microcontrollers is readily available. They come in a variety of sizes, capabilities and configurations. Some have *flash* memory that can be programmed multiple times, while others are intended to be programmed just once. If there is a reason to protect the code from being read from the PIC by users, it can be protected by setting a configuration bit during programming.

This article is intended to show how easy it is to understand how to use a PIC microcontroller. When you buy a PIC microcontroller it has no instructions or code stored in it. You need to write the code (usually with an ordinary text editor) in one of several special programming languages, process the text with an *assembler* or *compiler* to create PIC machine instructions and store these machine instructions in the PIC.[1] The scope of how to write the PIC code is beyond the scope of this article. Later we will discuss an online course by John McDonough, WB8RCR, that shows how to write PIC code. It's called *Elmer 160*.

Once the PIC code is written it can be loaded into the PIC in the PIC-EL board. After you have compiled or assembled your text file into PIC machine instructions you are ready to load it into the PIC microcontroller in the PIC-EL board. This requires a computer program (called programming software) as well as specific hardware, connected to the computer via some sort of hardware interface, to control the PIC as the code is being loaded into it. A variety of programming software packages are available and described in Section 3 or in the *Elmer-160* course.

By looking at the various PIC-EL hardware components and the sample code or *Elmer-160* lessons, the amateur can learn how to use microcontrollers in many interesting applications. The rewards are great.

PIC-EL Hardware Description

Figure 1 shows the schematic of the PIC-EL board. The PIC-EL board has two parts — a PIC programmer and a test/demonstration portion.

PIC programmer

The PIC-EL board contains the hardware functionality that is required to interact with the programming software to transfer the machine language instructions from the computer to the PIC microcontroller. In the case of the PIC-EL, an RS-232 (DB9F) serial port connection connects them.

[1]Notes appear on page 42.

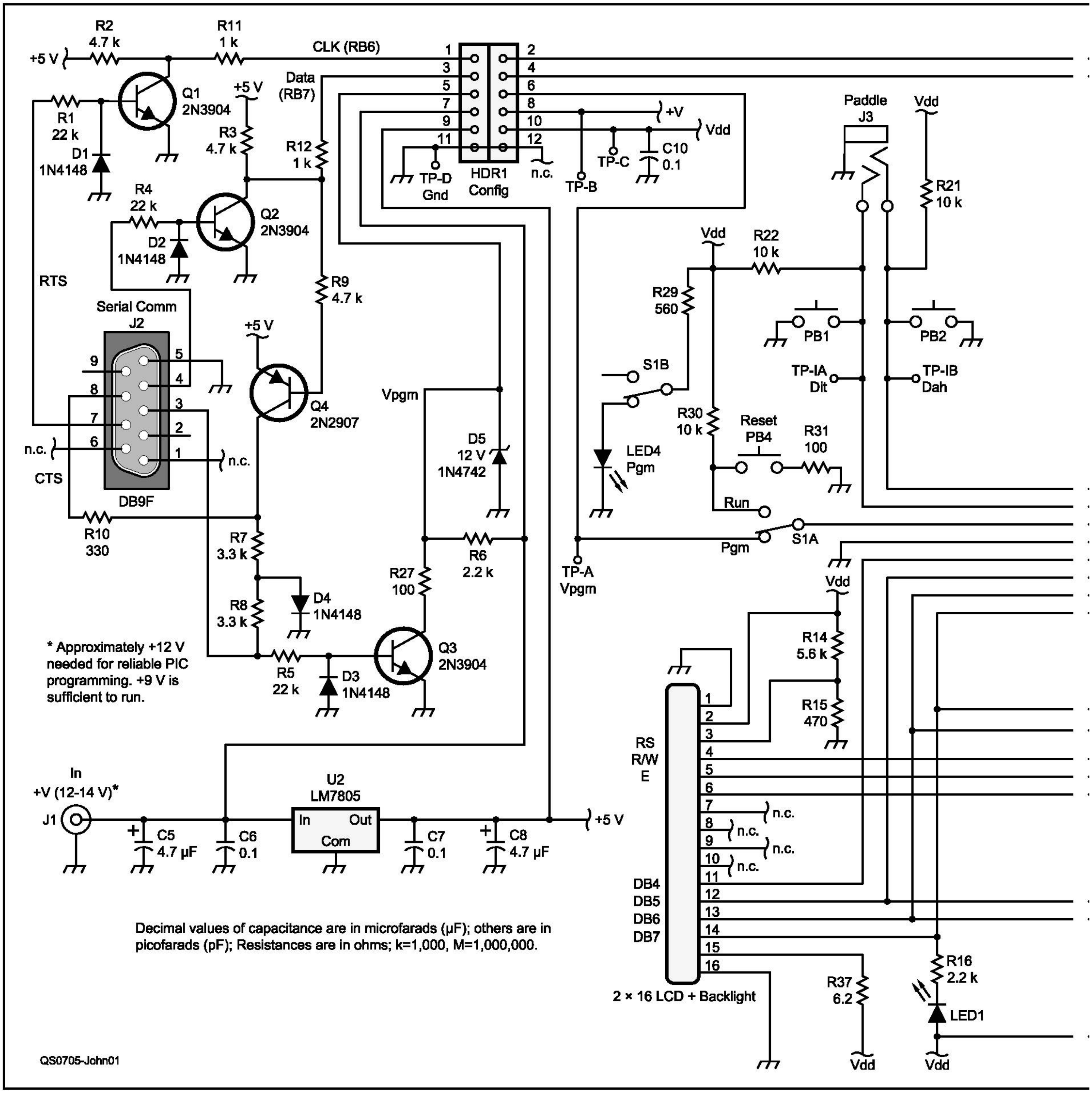
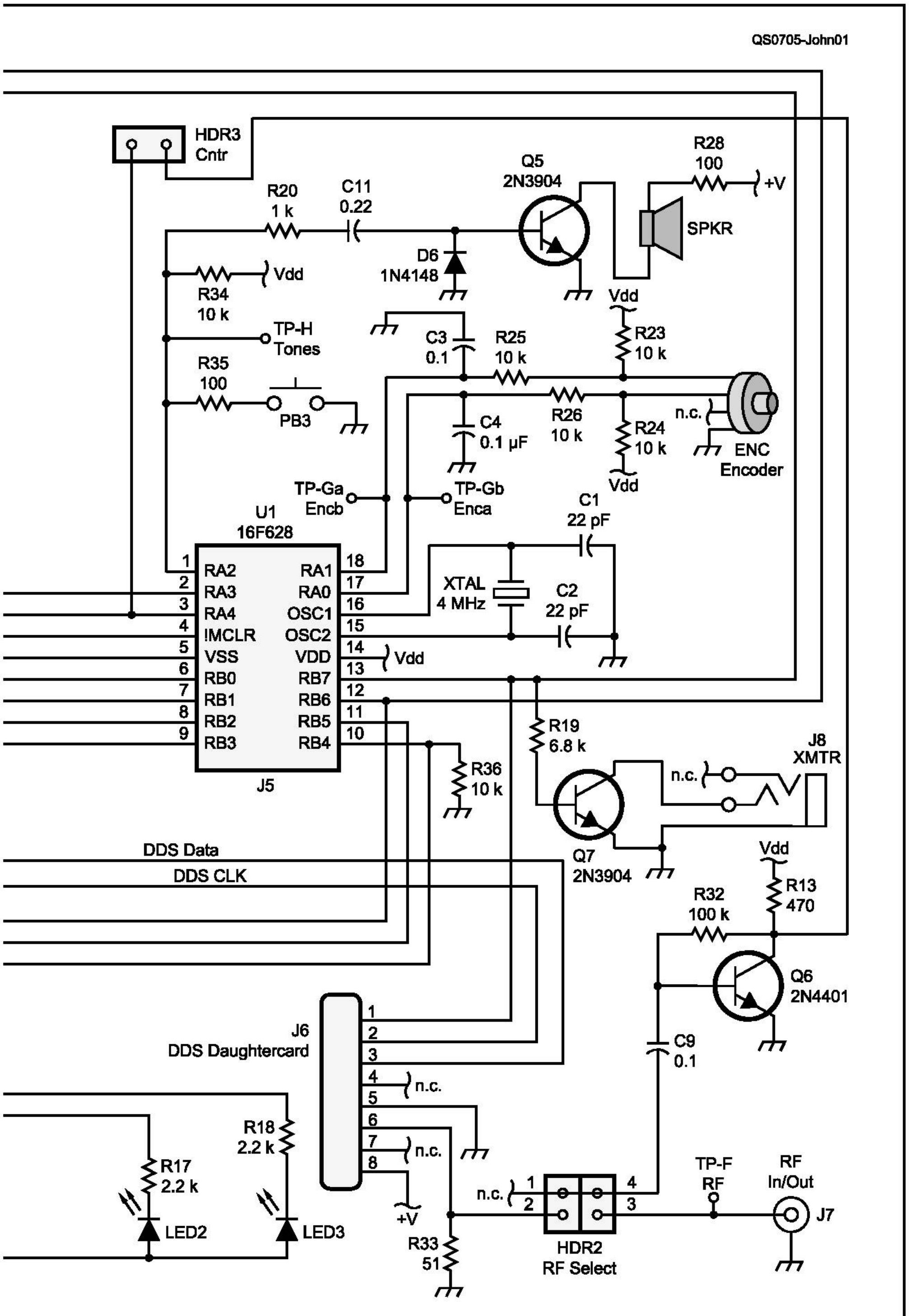


Figure 1 — Schematic of the PIC-EL board. The parts are listed in Table 1.

The PIC-EL's serial port programmer design is an adaptation of the classic programmer by David Tait. See Tait's PIC-related home page on the Internet, including downloadable software at **people.man.ac.uk/~mbhstdj/piclinks.html**. The software is also provided on the ARRL Web site with permission of the developer.[2] Details of how the RS-232 signals drive the PIC-EL hardware interface appear in Appendix A.[3]

No provision is made at this time for connecting a USB port or a parallel port to the programmer. However, external USB or parallel port programmers can be attached to the PIC-EL board via HDR-1. The PIC-EL board cannot be run directly from a PC parallel port because of the difficulty of getting the DATA IN signal to work properly. The parallel port's voltages are 0 V for a low level and +5 V for a high level. Since the TD signal never goes negative, the circuitry that was used to pull the Q4 collector below zero for a low level does not work. In fact, the low level only goes down to approximately 3.2 V when TD is asserted (0 V). Additional circuitry could be added, but it is not included in this PIC-EL board.

How about a USB to serial converter between the computer and the PIC-EL's serial interface? It would be nice, but so far



we have not found one that works in this application. The reason is that RS-232 ports are intended to perform asynchronous serial communications while we are directly controlling the serial port's control lines and doing synchronous communications to the PIC-EL. The PIC-EL requires a perfect one-to-one correspondence between signals generated by the computer programmer software and the signals presented to the PIC-EL serial interface. Since USB-to-serial interfaces do not allow this closely controlled synchronous activity, it cannot interface properly with the PIC programmer and does not work.

Project/Demonstration

The project/demonstration portion of the PIC-EL board was designed to allow the experimenter to understand how a PIC microcontroller can be used in a variety of applications. It allows the person to progress from controlling very basic components to more advanced components and projects.

In RUN mode, PIC experimenters have an opportunity to use and understand the following hardware functions:

- An 18-pin PIC microcontroller (16F84/A, 16F628/A, 16F88, etc). Includes a 4 MHz crystal.
- A 2×16 liquid crystal display (LCD with two lines of 16 characters).
- A rotary encoder (ENC-1).
- Three general-purpose push buttons, PB1 through PB3.
- A dedicated push button (PB4) for MASTER CLEAR (reset) of the PIC microcontroller.
- Three LEDs, LED1 through LED3.
- A speaker (SPKR-1) with transistor driver.
- All connections necessary to drive the NJQRP DDS daughtercard (DDS-30 or DDS-60).[4]
- A stereo jack for connection to CW paddles.
- A stereo jack with transistor driver for transmitter keying.
- A transistorized level conditioner for converting low-level signals to levels required for PIC input detection.
- A multi-purpose BNC connector.
- Selectable via a jumper at header HDR-2.
- Allow DDS output to be routed to the BNC
- Allow DDS output to be routed to a "conditioner" and then to a PIC input pin.
- Allow an outside signal source to be brought in to the "conditioner" and then to the PIC input pin.
- A 2×6 pin header block (HDR1 - CONFIG).
- Allows attachment of a "foreign programmer" to this PIC project board.
- Allows attachment of this programmer to a "foreign project board."

The PIC-EL schematic (Figure 1) may look quite complicated because many of the PIC pins have multiple usages. However, we can break down the schematic into its core pieces to understand the individual functions. This will also show how to use these basic components in other projects.

In Part 2 of this article we will discuss in detail the components connected to the PIC.

PIC-EL Computer Interface

Now you are ready to program a PIC using the PIC-EL. Before you can do this you need to have a program installed in your PC. There are many good options available (including *IC-PROG* and *WinPIC;* see Figure 2) for the *Windows* operating system but we will concentrate on one good one, *FPP* by David Tait,

Table 1
Parts for PIC-EL Board Project

Resistors are all ⅛ or ¼ W. Digi-key is at **www.Digi-key.com** *and Mouser is at* **www.mouser.com**.

C1, C2 — 22 pF, ceramic disc capacitor (Digi-key P4841-ND, Mouser 140-50N2-220J).
C5, C8 — 4.7 µF, electrolytic capacitor (Mouser 140-XRL16V4.7).
C3, C4, C6, C7, C9, C10 — 0.1 µF, monolithic capacitor (Digi-key P4910-ND, Mouser 80-C317C104M5U).
C11 — 0.22 µF, monolithic capacitor (Digi-key BC1149CT-ND).
D1, D2, D3, D4, D6 — 1N4148 diode (Digi-key 1N4148FS-ND, Mouser 625-1N4148).
D5 — 1N4742, 12 V Zener diode DO-41 (Digi-key 1N4742ADICT-ND, Mouser 78-1N4742A).
ENC — Rotary encoder (Digi-key P10860-ND).
HDR-1 — Pin header, 0.1″, 2 × 6 position (Mouser 571-1032406).
HDR-2 — Pin header, 0.1″, 2 × 2 position (Mouser 571-1032402).
HDR-3 — Pin header, 0.1″, 1 × 2 position (Mouser 571-1032392).
J1 — Coaxial power jack, 2.1 mm (Digi-key SC1153-ND, Mouser 163-5004).
J2 — DB9F serial connector (Jameco-104951).
J3, J8 — Audio jack, ⅛″, stereo (Mouser 161-3501).
J4 — SIP header, 16-pin (for PCB) (Mouser 571-16404526).
J5 — DIP socket, 18-pin (for PIC) (Digi-key ED3118-ND).
J6 — SIP socket, 8-pin, 90° (SamTec, SSQ-108-04-T-S-RA).
J7 — BNC jack, PCB mount (Mouser 523-31-5538-10-RFX).
LCD — Liquid crystal display, 2 × 16 character (08LCD9) (**www.ElectronixExpress.com**).
LED1, LED2, LED3, LED4 — Red LED, T1-3/4 (Digi-key P374-ND).
P4 — SIP socket, 16-pin (for LCD) (Mouser 517-974-01-16).
PB1, PB2, PB3, PB4 — SPST push button, momentary — P8079SCT-ND.
PCB — PC board (KangaUS).
Q1, Q2, Q3, Q5, Q7 — 2N3904 NPN transistor, TO-92 (Digi-key 2N3904D26ZCT-ND).
Q4 — 2N2907A PNP transistor, TO-18 (Digi-key 497-2577-5-ND, Mouser 610-2N2907A).
Q6 — 2N4401 NPN transistor, TO-92 (Digi-key 2N4401-ND, Mouser 610-2N4401).
R1, R4, R5 — 22 kΩ (Digi-key 22KEBK-ND, Mouser 291-22K).
R2, R3, R9 — 4.7 kΩ (Digi-key 4.7KEBK-ND, Mouser 291-4.7K).
R6, R16, R17, R18 — 2.2 kΩ (Digi-key 2.2KEBK-ND, Mouser 291-2.2K).
R7, R8 — 3.3 kΩ, (Digi-key 3.3KEBK-ND, Mouser 291-3.3K).
R10 — 330 Ω (Digi-key 330EBK-ND, Mouser 291-330).
R11, R12, R20 — 1 kΩ (Digi-key 1.0KEBK-ND, Mouser 291-1K).
R13, R15 — 470 Ω (Digi-key 470EBK-ND, Mouser 291-470).
R14 — 5.6 kΩ (Digi-key 5.6KEBK-ND, Mouser 291-5.6k).
R19 — 6.8 kΩ (Digi-key 6.8KEBK-ND, Mouser 291-6.8k).
R21, R22, R23, R24, R25, R26, R30, R34, R36 — 10 kΩ (Digi-key 10KEBK-ND, Mouser 291-10K).
R27, R28, R31, R35 — 100 Ω (Digi-key 100EBK-ND, Mouser 291-100).
R29 — 560 Ω (Digi-key 560EBK-ND, Mouser 291-560).
R32 — 100 kΩ (Digi-key 100KEBK-ND, Mouser 291-100K).
R33 — 51 Ω (Digi-key 51EBK-ND, Mouser 291-51).
R37 — 6.2 Ω (Mouser 291-6.2).
S1 — Slide switch, DPDT (Digi-key SW102-ND).
Shunts for HDR1, shunt-1 shunt, 0.1″, 2 position (Mouser 571-3828115).
SPKR — Speaker (Digi-key 433-1028-ND).
U1 — PIC16F628 microcontroller, with diagnostics pre-loaded (Digi-key PIC16F628-04/P-ND).
U2 — L7805 5 V voltage regulator (Digi-key 497-1442-5-ND, Mouser 511-L7805 ABV).
XTAL — Crystal, 4 MHz (Digi-key X405-ND, Mouser 520-HCU400-20).

GØJVY, since it is readily available and very easy to use. Similar packages are available for the Linux and Macintosh operating systems.

Installing FPP

You can download *FPP* directly from David Tait's Web site or obtain it from the ARRLWeb.[5,6] If you are using *Windows 95* or *98* you can run *FPP* without any additional drivers. However, if you are using *Windows 2000* or *Windows XP* you need to install two drivers, *directio* and *loaddrv*, to allow *FPP* to access the COM port. These two drivers are available at many Internet sites including the AmQRP Elmer-160 Web site.

Detailed instructions for installing *FPP* are available by downloading John's *Elmer-160* Lesson 10 from the AmQRP Elmer-160 Web site.

Using FPP

A serial cable is needed to connect the PC to the PIC-EL board. You need a standard straight-through 9-pin DB9M (male) to 9-pin DB9F (female) connector. One end of the cable plugs into the RS-232C serial (COM) port on your PC and the other end plugs into the PIC-EL board. You may need to disable other programs also using that same serial port.

Be sure that you have a sufficiently high voltage power supply connected to your PIC-EL board. Although the board can run off a 9 V battery, you will need to have at least +12 V at the PIC-EL CONFIG header (HDR1) in order to generate the minimum programming voltage, called V_{pgm}.

Setup

Once the serial cable is connected, you've got *FPP* loaded and turned on, your PIC-EL board is powered by at least 12 V, and you've manually wiggled the lines as described in Lesson 10, it should be a piece of cake to program a PIC on the PIC-EL board.

Obtain the .HEX Program

The HEX file is the new software you will be burning into the PIC. You can download the *PIC-EL V2* diagnostic program from my Web page or the FILES section of the PIC-EL Yahoo group. The *PICELv2diag.HEX* file contains the test program in HEX ASCII format. This file format is the standard output produced by PIC assembly programs and is a specific data format that is expected by the *FPP* program.

Load the PICELv2diag.HEX File into FPP

Click on the LOAD button and navigate to wherever you unzipped the PIC-EL V2 diagnostic files on your PC. You will see the *PICELv2diag.HEX* file listed there. Double-click on it and the HEX ASCII code will load into the *FPP* buffer. You will see that code in the *FPP* window as shown in Figure 3.

Slide Mode Switch S1 Down to PGM MODE

You need to move the slide switch S1 to the down position in order to put the PIC-EL board into the PGM mode. LED4, next to the switch, will turn on when you do this.

Erase the PIC Currently Plugged into the PIC-EL Board

You may want to erase, or clear out the software program currently in the PIC's flash memory before you burn a new program into the PIC. Click the ERASE button to do this, and if successful you will see a simple message pop up saying "PIC is erased."

Burn the New Code into the PIC

Now that the PIC memory is empty and you have the new program's HEX file in the *FPP* buffer window, you are all set to burn the program into the PIC. Click PROGRAM on the *FPP* application window and con-

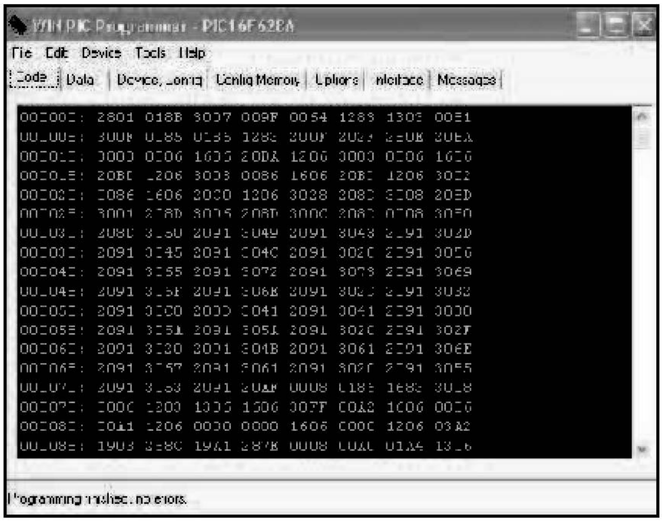


Figure 2 — PC screen snapshot running *WinPIC* software by DL4YHF to program a PIC in the PIC-EL II.

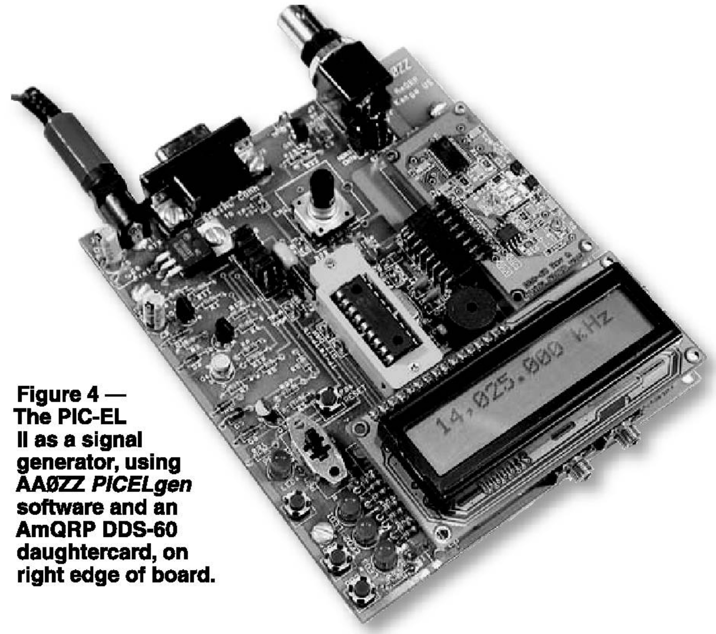


Figure 4 — The PIC-EL II as a signal generator, using AAØZZ *PICELgen* software and an AmQRP DDS-60 daughtercard, on right edge of board.

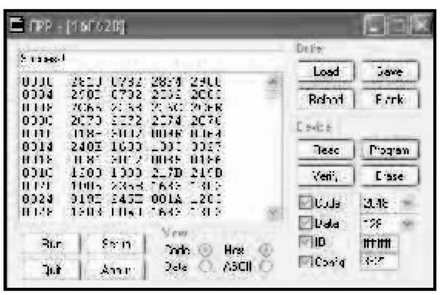


Figure 3 — PC screen snapshot running *FPP* software to program a PIC in the PIC-EL II.

firm your desire again in the pop-up window. It will take a few moments for this short program to be burned into the PIC, but when complete, *FPP* will display DEVICE PROGRAMMED! If it says PROGRAMMING FAILURE, you obviously have a problem such as the PIC was not first erased, the power supply wasn't connected or of sufficient voltage or the cable was not plugged in.

Slide the Mode Switch Up to Go Into RUN *Mode*

Now that the programming is complete, you next need to put the PIC-EL board back into RUN mode. Do this by sliding MODE switch S1 up, and the PGM LED will turn off once again.

Reset the Board to Start Up the New Program

Although not always necessary, just press the RESET button on the PIC-EL board to start the new program you just programmed into the PIC.

PIC-EL Diagnostic Program

A PIC-EL diagnostic program is preloaded on the 16F628 PIC that comes with the PIC-EL kit. The test symbolics file is also available on my Web site and on the PIC-EL Yahoo group FILES section. By looking at these symbolics and running the various tests you will be able to get an idea about how to activate the various elements of the PIC-EL. Note that the diagnostic program was written for demonstration purposes and not necessarily the way a finished program would be written.

When the test is started, the various test items will appear, one at a time, on line 2 of the LCD. When the test you want to run is displayed, press push button PB3. When you want to end a particular test you again press PB3 and hold it for ½ second. The main menu of tests will again start to appear, one at a time.

Test LEDs

When this test is selected the diagnostic will light the three LEDs in this sequence.

- LED1 turns on and then off.
- LED2 turns on and then off.
- LED3 turns on and then off.
- LED1 and LED2 and LED3 turn on and then all turn off.

The PGM/RUN LED is not exercised in this test. It is turned on when switch S1 is in PROGRAM mode and off when in RUN mode.

Test Push Buttons

This test will verify the operation of the three general-purpose push buttons. Pressing the push buttons in sequence will result in the following results:

- Pressing PB1 will cause LED1 to light. It will remain on until PB1 is released.
- Pressing PB2 will cause LED2 to light. It will remain on until PB2 is released.
- Pressing PB3 will cause LED3 to light briefly, but the test will then end.

This test does not exercise the RESET push button, PB4. Pressing PB4 resets the PIC and makes the test start over.

Test Speaker

Selecting this test will cause a series of tones to come from the PIC-EL speaker. The test plays the eight tones of a one-octave diatonic scale starting with C and ending at the next higher C. By the way, this test uses classical music theory, starting with middle C, ending with the next higher C and tuned such that the A above middle C is tuned to 440 Hz. The accuracy is limited by the tolerance of the 4 MHz crystal. By my ear it's pretty close.

Test Encoder

When this test is selected, the mechanical encoder is exercised. The test will start at zero with an eight digit number displayed on the LCD. Then, turning the encoder clockwise will increment the displayed number and turning the encoder counterclockwise will decrement the displayed number. The test uses the gray code signals generated by the encoder (see Figure 8) and determines the direction by the mechanism described.

Test Paddles

CW paddles can be connected to the PIC-EL via a patch cord that plugs into J3. The jack connects one of the paddle connections to the tip and the other to the ring. In the absence of CW paddles, you can exercise the equivalent functionality by pressing PIC-EL push buttons PB1 and PB2. When the DIT paddle (or PB1) is pressed the test will light LED1. When the DASH paddle (or PB2) is pressed the test will light LED2.

Test Transmitter Keying

This test will demonstrate how CW keying is accomplished via a PIC microcontroller. An oscillator or transmitter is connected to the PIC-EL board via a patch cord with an ⅛ inch mono phone plug. One end is plugged into J8 on the PIC-EL and the other is plugged into the oscillator/transmitter. The test activates transistor Q7, which effectively shorts the KEY LINE to ground and activates the oscillator/transmitter.

Test DDS-30

If you have a DDS-30 DDS daughtercard from AmQRP organization, you can test it by selecting this test. The DDS-30 is connected to J6 of the PIC-EL. The test program generates three frequencies — 7.040, 7.041 and 7.042 MHz. (They are displayed on the LCD in Hz as 7040000, 7041000 and 7042000.) The duration of each frequency is about ½ second and then the sequence repeats. You can observe the output on a spectrum analyzer (or oscilloscope) or by tuning your HF receiver to 7.040 MHz. See Figure 4.

Test DDS-60

The DDS-60 DDS daughtercard from AmQRP is slightly different from the DDS-30 in that the DDS-60 uses an AD9851 DDS while the DDS-30 uses an AD9850. The DDS-60 uses a 30 MHz clock oscillator and expects the 6× multiplier to be activated by driver software. Once again, the DDS-60 is installed in J6 of PIC-EL. When selected, this test generates the same three frequencies.

PIC-EL for Elmer-160 *Lessons*

I developed the PIC-EL board in late 2004, with consulting help from AmQRP club members, George Heron, N2APB; Joe Everhart, N2CX; John McDonough, WB8RCR; Earl Morris, N8ERO, and Jim Kortge, K8IQY. This was soon after John McDonough volunteered to teach an online course to help beginners learn how to use PICs. We wanted to make a project board the students could use to load the code they just wrote into the PIC and then to immediately try it out. (Immediate feedback does wonders in keeping the experimenter motivated!) Within a three-month period I developed several prototypes, tested them, and finally produced a printed circuit board that the AmQRP club could use in a kit. Hundreds of these kits were distributed by the club in early 2005 and the reception was very positive.

Since John called his online PIC course *Elmer-160,* I named this board the PIC-EL. The standard PIC for the PIC-EL is a 16F628, a very common, mid-range PIC, selected for this project because of its balance of architectural simplicity, power and low cost. It is one of the most common PICs used in amateur applications these days. The *Elmer-160* lessons start with an even lower-end PIC, the 16F84, because it is even easier for beginners to understand. The 16F628 is nearly compatible with the 16F84 in that only a few lines of code need to be changed. The PIC-EL can use either of these PICs interchangeably. However, the diagnostic program, preloaded on a 16F628 that is supplied with the kit, is slightly too large for a 16F84.

PIC-EL kit sales by the AmQRP club ended after a few months. Recently I updated the PIC-EL board to replace obsolete parts, correct a few deficiencies, and to make the kits available on a long-term basis. The result is the PIC-EL Version II. The kit is available for $55 plus $5 shipping from Bill Kelsey, N8ET, at Kanga US.[7]

If you are interested in the *Elmer-160* lessons from John McDonough, WB8RCR, you can find them at the AmQRP Web site.[8] John's lessons are available for downloading and you can contact him by using links provided on that site.

Questions and Support

For up-to-date details and documentation regarding this project, please see the author's Web page. For additional support questions, see the Yahoo group PIC-EL or e-mail the author directly at the address below.

Conclusion

It's very satisfying to be able to develop a PIC program that performs a task in exactly the manner you want it to. Yes, it takes a bit of effort, but the result is well worth it.

In the next part we will discuss in greater detail how to use the various hardware components attached to the PIC microcontroller in the PIC-EL.

Notes

[1]Assemblers or compilers are software programs provided to convert human readable instructions into the firmware needed by the PIC.

[2]**www.arrl.org/files/qst-binaries/Johnson0407.zip**.

[3]See Note 2.

[4]**www.njqrp.org/**.

[5]**www.people.man.ac.uk/~mbhstdj/piclinks.html**.

[6]See Note 2.

[7]**www.kangaus.com, n8et@arrl.net** or Kanga US, 3521 Spring Lake Dr, Findlay, OH 45840-9073.

[8]**www.amqrp.org/elmer160**.

Craig Johnson, AA0ZZ, holds a degree in electrical engineering from the University of Minnesota, and an MBA from the University of St Thomas. Since graduation he has worked on large-scale, mainframe computer design and development at Unisys Corporation in Roseville, Minnesota. He now works at Minnetronix, Inc in St Paul, Minnesota, developing microprocessor-based medical devices. He holds six patents related to computer disk technology and I/O architecture and has three more patents pending.

Craig got his first amateur license in 1964 at the age of 14 and his General class license six months later. He credits Amateur Radio as being a key factor in his desire to become an electrical engineer. After college he let his license lapse and concentrated on computers. In 1995, two of his three children expressed an interest in Amateur Radio. Today, all five family members have ham licenses.

Craig has a variety of ham radio interests including DXing and contesting. He operates almost 100% CW but he enjoys the digital modes as well. He enjoys the excitement of building and operating homebrew QRP radios. He has designed and developed DDS VFOs using PIC microcontrollers that are being used by hundreds of hams throughout the world.

You may contact Craig at 4745 Kent St, Shoreview, MN 55126 or **aa0zz@cbjohns.com**. *His Web site is* **www.cbjohns.com/aa0zz**.

New Products

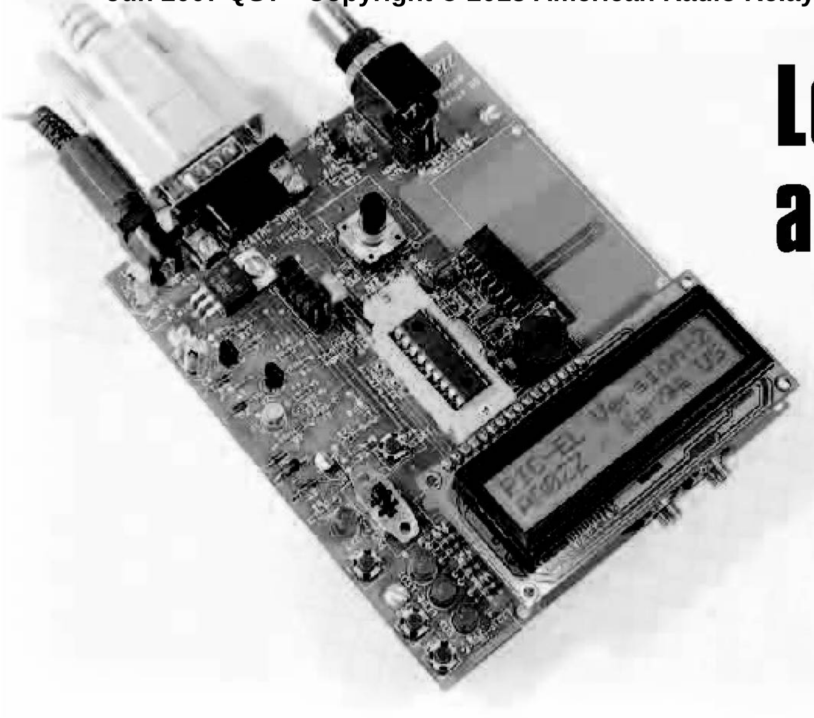
W2DGJ WIRE ANTENNA WIND DAMPER

◊The W2DGJ Wire Antenna Wind Damper is said to work like an automotive shock absorber, putting pressure on the damper and not the wire when the wind blows and the antenna supports move. The Wind Damper attaches between the insulator at the end of your antenna wire and the supporting tree or pole and is said to allow up to 10 inches of movement before stressing the wire. One Wind Damper is required for each end of a dipole antenna. Price: $29.95 each. For additional information, visit your favorite dealer or contact W2DGJ Insulators, 1300 Fatio Rd, Deland, FL 32720; email **tenpin11@hotmail.com**.

QSO WIZARD SE AND PRO SOFTWARE FROM ORCASTAR

◊*QSO Wizard* ham radio operating assistance software for the Macintosh is now available in *Pro* and *SE* versions. Both versions include logging features, a custom QSL card maker, beam heading path plots, multiple clocks, a schedule editor and other features for the active operator. The software is built around images of the Earth. Worldwide prefixes, time zones, grid locators and other information is available by clicking on the image or searching included databases. The SE version uses graphics centered on the middle of your state. The *Pro* version uses custom graphics centered on your city or town, has a contact time clock that shows local time anywhere in the world, displays a real time "night shadow" showing which parts of Earth are in darkness, and other features. Available for Mac *OS X* only, *QSO Wizard* is customized for the user's call sign and QTH (North American locations only) at time of order. Price: *Pro* version, $69.95; *SE* version, $39.95. A 30-day demo version is available for a $6.95 shipping/handling fee. More details are available at **www.orcastar.com**.

QST Issue: Jun 2007
Title: Learning to PIC with a PIC-EL -- Part 2
Author: Craig Johnson, AA0ZZ



Learning to PIC with a PIC-EL — *Part 2*

This easy to duplicate PIC proto board will make learning how to use these useful components easy!

Craig Johnson, AAØZZ

Last month I described the PIC-EL board that I developed for use as a companion board for an online course to teach beginners how to use peripheral interface controllers (PICs).[1] The course is written by John McDonough, WB8RCR, and additional lessons are still being developed. This article is intended to show how easy it is to understand how to use a PIC microcontroller. By looking at the various PIC-EL hardware components and the sample code or the Elmer-160 lessons, the amateur can learn how to use microcontrollers in many interesting applications.[2] The rewards are great.

PIC-EL Hardware

We looked at the PIC-EL computer interface last time and we also took a preliminary look at the various hardware components attached to the PIC microcontroller in the PIC-EL. Now we will take a closer look and show how you can make use of these components in your own projects. The project and demonstration portion of the PIC-EL board was designed to allow the experimenter to understand how a PIC microcontroller can be used in a variety of applications. It allows a person to progress from controlling very basic components to more advanced components and projects. PIC experimenters have an opportunity to use and understand the following hardware functions:

- An 18-pin PIC microcontroller (such as a 16F84/A, 16F628/A or 16F88).
- A 4 MHz crystal controlled clock.
- A two line (16 characters each) LCD display.
- A rotary encoder.
- Three general-purpose push buttons.
- A dedicated push button for master clear or reset of the PIC microcontroller.
- Three light emitting diodes (LEDs).
- A speaker with a transistor driver.
- All connections necessary to drive the NJQRP DDS daughtercard.[2]
- A stereo jack for connection to CW paddles.
- A stereo jack with transistor driver for transmitter keying.
- A transistor signal conditioner for converting low-level signals to levels required for PIC input detection.
- A multi-purpose BNC connector, jumper selectable to allow DDS output or signal input.
- A 2 × 6 pin header block to allow attachment of a foreign programmer.

What's Inside?

The PIC-EL schematic (see Figure 1 in Part 1) may look quite complicated because many of the PIC pins have multiple usages.[3] However, we can break down the schematic into its core pieces to understand the individual functions. This will also show how to use these basic components in other projects.

Last time we discussed the computer programming interface (the left side of the schematic). Now let's take a look at the various hardware components that are attached to the PIC and see how they work.

PIC System Clock

The system clock is generated by a 4 MHz crystal coupled with two 22 pF capacitors. A simple RC oscillator or the 16F628's internal oscillator could have been used instead. Since we are going to be experimenting with several timing-sensitive

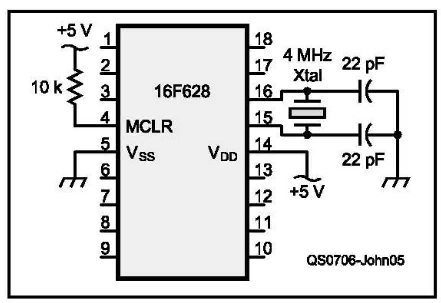


Figure 5 — The basic components necessary to run a PIC.

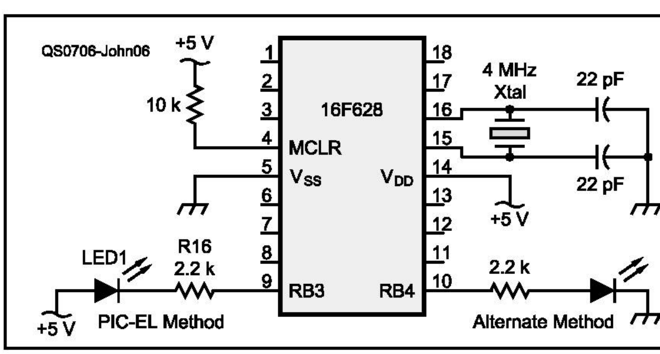


Figure 6 — Two ways of lighting an LED from a PIC microcontroller.

[1]Notes appear on page 36.

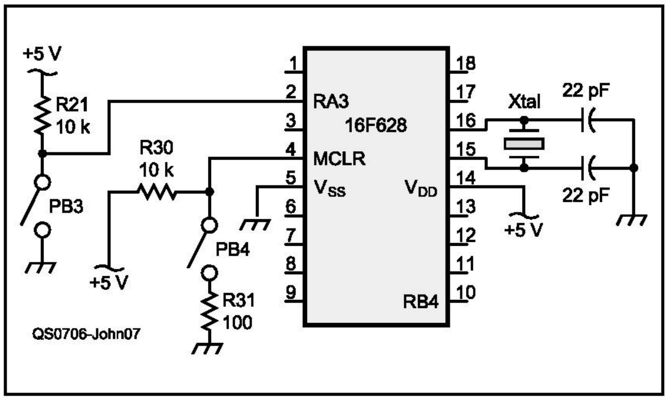


Figure 7 — The manner in which switches are used with a PIC and how PB3 is implemented in the PIC-EL.

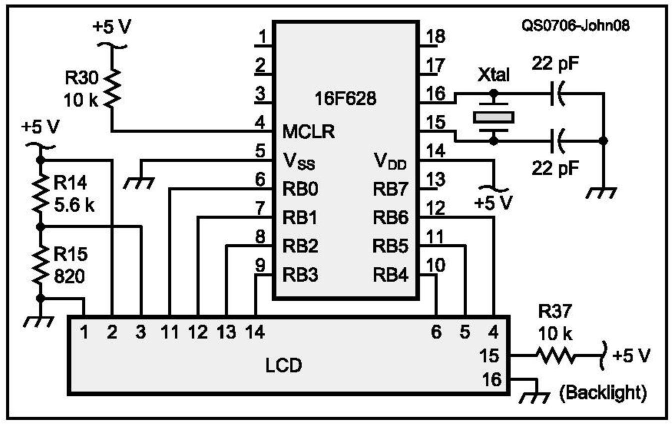


Figure 8 — Implementation of an LCD with the PIC-EL board.

projects such as frequency counters, an accurate clock is essential so a crystal was used.

The basic components necessary to run a PIC are shown in Figure 5. As you can see, it's really very simple.

LEDs

Two direct ways of lighting an LED from a PIC microcontroller are illustrated in Figure 6. The first is to connect a PIC output pin to a resistor and then to the anode of the LED with the cathode grounded. To light the PIC, the program needs to assert a logical high (+5 V nominal) on the output PIC pin. The PIC then provides the current to light the LED.

The other way is to connect a PIC output pin to a resistor and then to the cathode of the LED with the anode connected to +5 V. In this case, to illuminate an LED from the PIC, the PIC pin needs to be brought to a low level. The PIC is a current "sink." One minor drawback of this method is that the PIC programmer must remember that the logic is reversed. In this case the LED is illuminated when the PIC pin is set to a logical low, and it is dark when the PIC pin is at a logical high.

The method used in the PIC-EL board is to "sink" current with a PIC rather than to "source" the current.

Ideally, to illuminate an LED, the current flow through it should be between 1 mA and 20 mA. In this design the current flow is determined by the size of the series resistors. The series resistors (R16, R17 and R18) are each 2.2 kΩ. These values were selected in order to keep the circuit loading to a minimum, since the PIC pins to which they are connected are used for multiple functions. Since the voltage drop across each LED is about 1.8 V, the voltage drop across the 2.2 kΩ resistors is about 3.2 V. This means the current through the resistors and these LEDs is about 1.4 mA. This amount of current illuminates the LEDs sufficiently. In

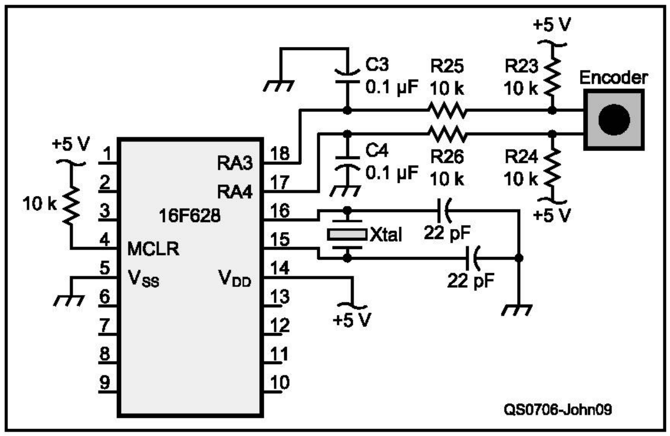


Figure 9 — A mechanical rotary encoder is attached to two PIC pins.

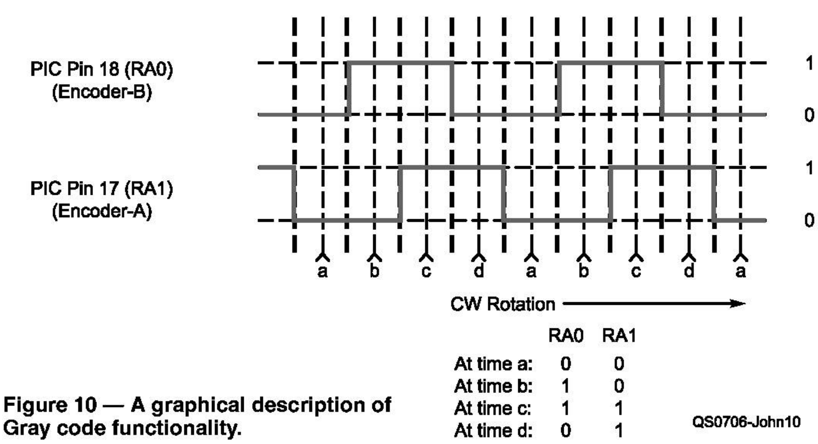


Figure 10 — A graphical description of Gray code functionality.

some cases you may want to increase this current for brighter illumination. You could use a 1 kΩ resistor (3.2 mA), for example.

Push Buttons

Figure 7 shows how these switches are used in a PIC and how push button PB3 is implemented in the PIC-EL. Three stand alone normally open SPST push buttons (PB1, PB2 and PB3) are connected to PIC pins RA4, RA3 and RA2 in the PIC-EL. They can be used for any type of control functions that the programmer wants to use them. One other normally open SPST push button (PB4) is connected to the PIC's MASTER CLEAR pin and is used to reset the PIC program (make it start over). The three PIC pins that have general-purpose push buttons (PB1, PB2 and PB3) also have 10 kΩ pull-up resistors (R22, R21 and R34 respectively) attached to V_{DD}

(+5 V). In general, using pull-up resistors is a good design principle and provides a good "stiff" pull-up. In some cases, no pull-up resistor is used because some PIC pins (Port B in a 16F628) can have internal weak pull-ups activated via PIC software instructions. (This is done by executing a PIC instruction that clears bit 7 of the PIC's OPTION register.) In this mode, the PIC in effect puts a 50 kΩ resistor between each of these pins and +5 V. This means the PIC is able to source 0.1 mA of current on each of those pins. This is sufficient for a simple push button operation.

Note that the PIC's MASTER CLEAR pin (pin 4) has a 10 kΩ pull-up resistor (R30) to +5 V and is switched via a normally open SPST push button (PB5) to "near" ground. This is also illustrated in Figure 8. The pull-up resistor is essential here, since the PIC needs +5 V on MCLR for normal PIC operation. The 10 kΩ resistor is sufficient here, since the MASTER CLEAR pin draws very little current. Push button PB4 also has a 100 Ω resistor to prevent voltage transients from locking up the PIC.

Liquid Crystal Display (LCD)

The LCD panel used in the PIC-EL demonstration board has two rows of 16 characters each. It is a standard 5 × 10 dot matrix LCD that uses a Hitachi 44780 controller. It is attached in such a way that it minimizes interaction with other functions of the PIC-EL. In particular, the PIC programmer (also using PIC pins 12 and 13 — RB6 and RB7) still works properly when the LCD is connected in this manner.

The values of the voltage divider resistors (R14, R15) were selected to put the proper voltage on the LCD's contrast pin (pin 3). Also, the LCD backlight is activated with the resistor to +5 V connected to LCD pin 15 along with the ground connection to LCD pin 16. The backlight of the 2 × 16 LCD used in the PIC-EL kit draws about 75 mA. If an LCD with a different level of backlight current is used, the size of this resistor must be adjusted. Figure 8 shows how the LCD is implemented in the PIC-EL board.

Rotary Encoder

A mechanical rotary encoder is attached to two PIC pins, RA3 and RA4, as shown in Figure 9. R23 and R24 are typical pull-up resistors, since the rotary encoder is essentially just a pair of switches that open and close as the shaft rotates. Capacitors C3 and C4 are filters for removing noise that comes from contact bounce. The series resistors, R25 and R26, help in the signal filtering. Without the noise filtering, operation could be erratic.

For the mechanical encoder included in the PIC-EL kit, each of the signal lines produce 24 pulses per revolution, so a total of 96 up or down voltage transitions per revolution are generated that can be detected by the PIC microcontroller. The pulses of the two data lines are encoded in an overlapping Gray code such that an algorithm allows the PIC program to determine which direction the shaft is being turned. Figure 10 provides a simple explanation of how Gray code works. Table 1 illustrates one way to determine the rotation direction.

Speaker

A miniature speaker (SPKR-1) is attached to a PIC pin by way of a simple transistor (Q5) driver, as shown in Figure 11. The transistor driver gives more "punch" to the speaker than could be attained by directly attaching it to the PIC speaker output pin. The capacitor and diode in the path to the base of the driver transistor would be optional in most PIC speaker applications but are very important in the PIC-EL board because they prevent the speaker from being inadvertently left "on" if the PIC-EL application happens to leave that pin in a high state. Q5 acts as a switch, allowing current to flow through the speaker when Q5 is turned on and not flow when Q5 is turned off.

Pulses are generated by the PIC software and pass through capacitor C11 to turn Q5 on and off. The PIC program produces different tones by changing the frequency of the pulses it generates. Since audio tones are relatively low frequency and the PIC executes an instruction every microsecond, accurate delay loops can be designed to produce pulses with the desired frequencies.

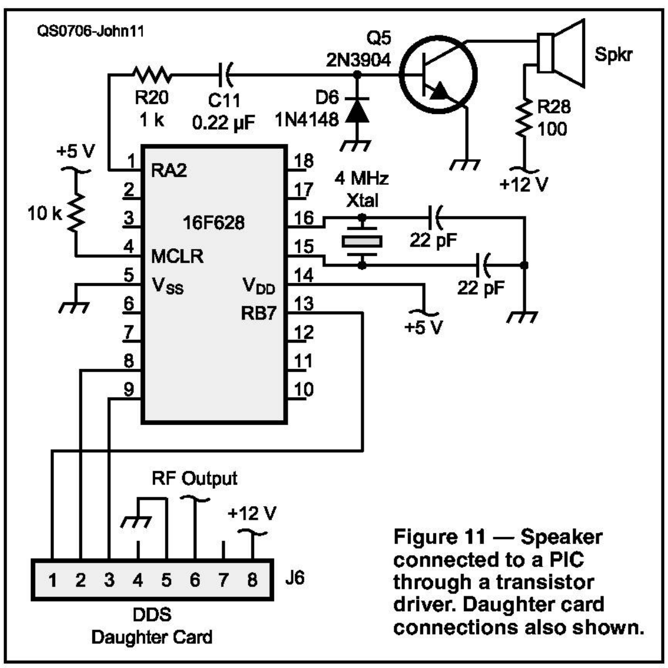


Figure 11 — Speaker connected to a PIC through a transistor driver. Daughter card connections also shown.

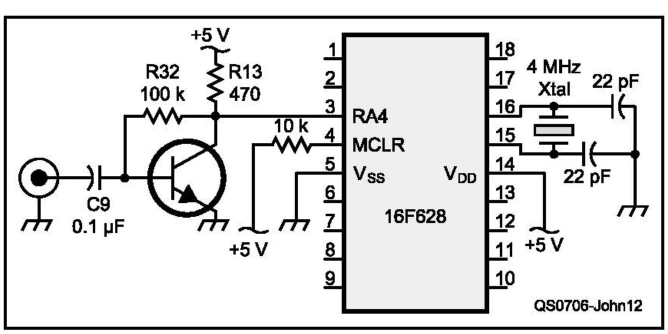


Figure 12 — Signal conditioner to increase small amplitude signals to voltage levels detectable by the PIC.

Signal Generation with the NJQRP DDS Daughtercard

The AmQRP DDS daughtercard can be plugged into the PIC-EL board by way of socket J6. Appropriate connections are made to the PIC and the required +12 V is also supplied to the daughtercard socket. Details of how the daughtercard operates can be found on the AmQRP Web page at **www.amqrp.org/kits/dds60/**. The PIC connections to the DDS daughtercard are illustrated on the bottom of Figure 11.

The output of the daughtercard is supplied back to pin 6 of socket J6. The PIC microcontroller can drive the DDS daughtercard to produce an amplitude of approximately 600 mV with a frequency within the range of 0 to 30 MHz or 60 MHz, depending on which version of the DDS daughtercard you have.

Signal Conditioner

A signal conditioner, shown in Figure 13, is provided to increase small amplitude signals to voltage levels detectable by the PIC. The output amplitude of the DDS daughtercard is too low to be fed directly back into a PIC pin for the demonstration of frequency counting. To make this work, the amplitude is increased by the signal conditioner circuitry. Notice that this conditioner is not a linear amplifier in that it does not attempt to keep a distortion-free sinewave output. For purposes of frequency measurement, a square wave is just as good as a sine wave.

Note that header HDR2 is used to select the source of the signal that goes into the conditioner. In one position, the output of the DDS daughtercard is fed into the conditioner while in another configuration a signal from an external source can be brought into the PIC-EL board via BNC connector J7 and routed through the conditioner before going to the PIC.

CW Paddle Input Via a Stereo Jack

CW paddles can be attached to the PIC by way of a ⅛ inch stereo jack as shown in Figure 13. The jack connects one of the paddle connections to the stereo plug's tip and the other to the ring. Both pins have pull-up resistors (R21 and R22) connected to +5 V. The PIC is then able to detect the paddle closures just as if they were two SPST switches. A demonstration example of a CW keyer is available on my Web site or the FILES section of the PIC-EL YAHOO group.

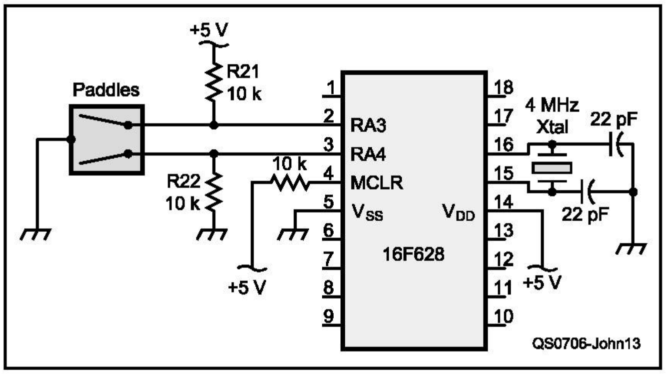


Figure 13 — CW paddles attached to the PIC by way of a ⅛ inch stereo jack.

Transmitter Keying Via a Stereo Jack

Figure 14 shows how to key a transmitter with the output of the demonstration keyer. Another ⅛ inch stereo jack is provided for this purpose. The output of a PIC pin goes to a transistor driver which then goes to the tip connection of the stereo jack. When keyed, the transistor driver drives the voltage at the tip connection from approximately 5 V to ground potential. When the PIC pin is not keyed the tip-to-ground connection looks like an open circuit so the tip remains at approximately 5 V. This keying mechanism will work for most modern rigs because they employ positive keyed transmitters. Some early transmitters (tube type in particular) used negative keying. Modern positive keyed transmitters have approximately +3 to +5 V on the tip connection with the radio keyed by connecting this pin to ground. Negative keying transmitters often have something on the order of –30 V on the tip connection. This keying circuit is for positive keying only, but if your radio requires a negative keying scheme, or uses high voltage cathode vacuum tube keying, a relay or driver circuitry will be required.

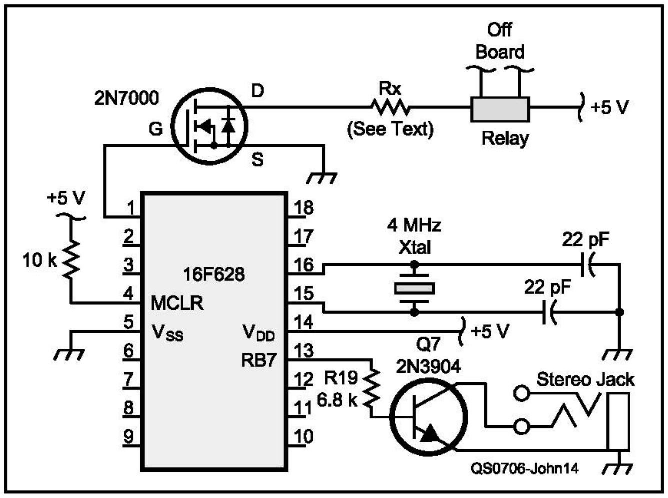


Figure 14 — Keying a transmitter with the output of the demonstration keyer. Relay driver also shown.

Table 1

A Method to Determine the Direction of Rotation of a Rotary Encoder

Going UP, the sequence is a,b,c,d,a,b,c,d,a, etc, so the sequence is:

00, 10, 11, 01, 00, 10, 11, 01, 00....

Going DOWN, the sequence is a,d,c,b,a,d,c,b,a, etc, so the sequence is:

00, 01, 11, 10, 00, 01, 11, 10, 00....

To determine if the sequence is UP or DOWN:

1) Take the "right-bit" of any pair.
2) XOR it with the "left-bit" of the next pair in the sequence.
3) If the result is 1, it is UP; if the result is 0, it is DOWN.

Frequency Counter

A frequency counter can be implemented in the PIC-EL by using the signal conditioner, described earlier. The conditioner feeds its output into PIC pin 3 (RA4/T0CKI). This PIC pin may be configured to be a general purpose input/output pin, but also has the unique characteristic of being configured as a counter input to PIC register TMR0. The TMR0 register is used by frequency counter applications.

How to Drive a Relay from a PIC

The PIC-EL is not set up to demonstrate relay control. In order to drive a device that requires more current than the 20 mA or so a PIC pin can deliver, a driver circuit is required. The top of Figure 14 shows an example of how it can be done. Resistor R_x in series with the relay coil must be sized to pass the proper amount of current. The 2N7000 MOSFET is a good general purpose device but in some applications, such as for driving relays or LCD backlight activation, an IRLML2502 is an even better choice, since its drain to source resistance when turned on is about 0.045 Ω. In contrast, the on R_{DS} of a 2N7000 is between 2 and 5 Ω.

Questions, Support?

For up-to-date details and documentation regarding this project, please see my Web page, **www.cbjohns.com/aa0zz**, the YAHOO group PIC-EL or e-mail me directly at **aa0zz@cbjohns.com**.

Conclusion

I hope you can see that getting started with PIC programming is not terribly difficult and that there are many useful things you can do with them. We have only scratched the surface, of course. The PIC-EL is a very convenient platform for experimenting but after that, it's up to you to develop your own projects with the components. Now it's time for you to try it!

Notes

[1]C. Johnson, AAØZZ, "Learning to PIC with a PIC-EL — Part 1," *QST*, May 2007, pp 37-42. Available on the ARRLWeb at **www.arrl.org/files/qst-binaries/Johnson0507.pdf**.
[2]**www.amqrp.org/elmer160**.
[3]See Note 1.
[4]**www.njqrp.org/**.

Craig Johnson, AAØZZ, can be reached at 4745 Kent St, Shoreview, MN 55126 or **aa0zz@cbjohns.com**. *His Web site is* **www.cbjohns.com/aa0zz**.

ARRL Periodicals Archive – Search Results
A membership benefit of ARRL and the ARRL Technical Information Service

ARRL Members: You may print a copy for personal use. Any other use of the information requires permission (see Copyright/Reprint Notice below).

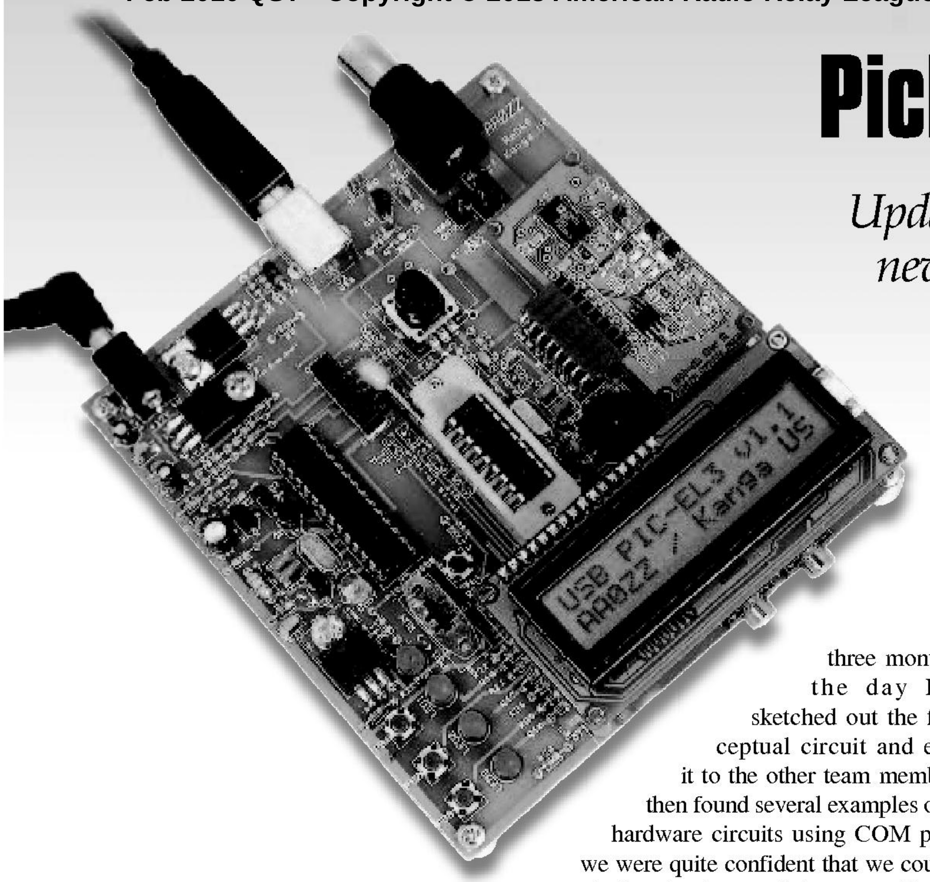
Need a higher quality reprint or scan? Some of the scans contained within the periodical archive were produced with older imaging technology. If you require a higher quality reprint or scan, please contact the ARRL Technical Information Service for assistance. Photocopies are $3 for ARRL members, $5 for nonmembers. For members, TIS can send the photocopies immediately and include an invoice. Nonmembers must prepay. Details are available at www.arrl.org/tis or email photocopy@arrl.org.

QST on CD-ROM: Annual CD-ROMs are available for recent publication years. For details and ordering information, visit www.arrl.org/qst.

Non-Members: Get access to the ARRL Periodicals Archive when you join ARRL today at www.arrl.org/join. For a complete list of membership benefits, visit www.arrl.org/benefits.

QST Issue: Feb 2010
Title: Pickle with USB I/O
Author: Craig Johnson, AA0ZZ

Click Here to Report a Problem with this File



Pickle with USB I/O

Update the PIC-EL to use with a new PC with a USB interface.

Craig Johnson, AAØZZ

"What do you mean it doesn't have a USB interface? I don't even *have* a COM port on my new computer!" Such was the cry from the very beginning from many people who wanted to learn to program PIC microcontrollers and then read about the COM port based PIC-EL board and the associated online PIC programming course that we developed.

The PIC-EL Revisited

In the May and June 2007 issues of *QST* I presented a two part article describing the PIC-EL. This project was developed to help hams get practical hands-on experience using PIC microcontrollers for some fun and useful ham applications.[1] As John McDonough, WB8RCR, began writing his lessons for the *Elmer-160* course, several of us who are associated with the AmQRP club began designing this companion board to allow the students to experiment with the material as they navigate through the lessons.[2] I decided to call it the PIC-EL ("pickle"), since it's for PIC microcontrollers and it goes along with the *Elmer-160* lessons.

Right from the beginning we knew that a USB interface would be a desirable interface for the tool. However, there were several problems. First of all, we had a very tight, self-imposed time schedule. We really wanted to get the board ready to go about three months from the day I hand-sketched out the first conceptual circuit and e-mailed it to the other team members. We then found several examples of simple hardware circuits using COM ports and we were quite confident that we could make similar circuitry work in our application. To develop the circuitry for a USB interface, whether it was one of the integrated circuits designed for this type of task or if the circuitry was designed from scratch, it was obvious that a USB solution would be much more complicated.

Second, we didn't have a simple PC application to drive the programmer with a USB interface that we could use to send the low level (HEX) code into the PIC. We had several nice, free, PC applications that could drive the COM port circuitry we were looking at, however.

We carefully evaluated these reasons and concluded that the risks with pushing for a USB interface were too great. We decided to go with the simple COM port for the first version, with the intent of continuing to look for a USB solution for the future.

In my PIC-EL YAHOO group, users of the PIC-EL board discuss their projects and ask questions about PIC programming issues as well.[3] One of the subjects that frequently comes up is how to get the PIC-EL to work with a USB interface. While we tried using various COM-to-USB converters (they don't work, since we drive the COM port pins directly) I quietly kept looking for a good solution.

I investigated many different methods but each had severe drawbacks, including the need for me to write new programming software to support it. Then, in late 2007, I bought a Microchip PICKit2, thinking it might be a way to quickly attach to the PIC-EL with a USB connection.[4] The PICKit2 would be connected such that it would bypass the PIC-EL's hardware programmer and connect directly into the PIC-EL's configuration header. My initial attempts to make this work failed, but several months later I discovered that Microchip had published a set of updates for the PICKit2 hardware. After I installed them, it worked perfectly on the PIC-EL II. I then discovered a number of "clones" on the Internet and, upon examining them, realized that a lot of the hardware in the Microchip PICKit2 is nice but not really needed for a PIC-EL environment. I then made and prototyped a stripped down version of the PICKit2 using ideas from other designs on the Internet but putting my own spin on it. After some extended debugging sessions the PIC-EL III was born.

PIC-EL III Hardware Description

The schematic of the new portion of the PIC-EL III board is shown in Figure 1. The "right side," the various hardware components that can be driven by the target PIC, has been changed very slightly since the PIC-EL II, but the "left side," the programmer portion, has been completely replaced and is shown in the figure. The complete schematic and parts list is available on the QST binaries Web site.[5]

PIC-EL III Computer (USB) Interface and Programmer

The programmer section has been replaced with circuitry to provide a USB interface. The circuitry is a simplified version of the PICKit2 by Microchip Technology (**www.microchip.com**) and has a Microchip PIC 18F2550 in its center. The code for the 18F2550 is produced and distributed (free of charge) by Microchip.

The PIC-EL III hardware programmer uses MOSFETs to drive the programming lines. It does not draw 5 V power from the USB con-

[1]Notes appear on page 42.

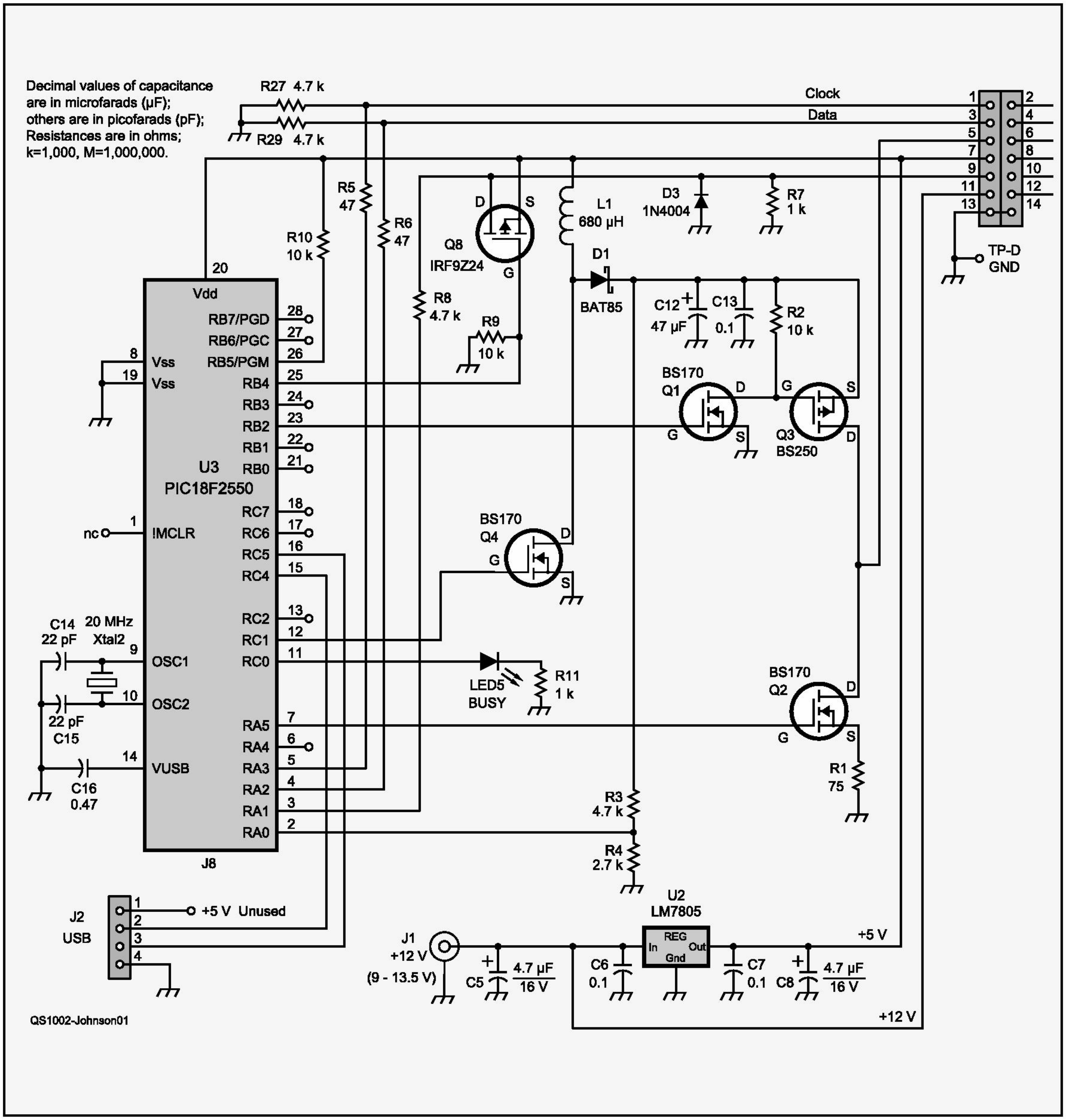


Figure 1 — Schematic diagram of the USB interface portion of the PIC-EL. Although the display subassembly is functionally equivalent to the version in May 2007 *QST*, there are a few differences in detail.

nection but instead runs on 5 V power from the PIC-EL's 12 V to 5 V regulator. The programmer hardware has charge pump circuitry to internally generate the +12 V programming voltage (V_{PP}) to be applied to the MCLR pin. Note that the PIC-EL expects 12 V power being supplied at all times and it does not use the 5 V power supplied by the USB. The PIC-EL's 12 V to 5 V voltage converter supplies all the 5 V power for the PIC-EL board.

Note that when the PIC-EL is in PROGRAM mode, the programmer hardware generates V_{DD} (+5 V) and V_{PP} (+12 V) with the appropriate timing to program the target PIC. Depending on the type of PIC being programmed, the programmer sometimes raises V_{PP} before V_{DD} and sometimes raises V_{DD} before V_{PP}. It's tricky but very important. (This removes a limitation found in the previous PIC-EL versions as well as all other *Tait* type programmers that use a COM interface. There simply aren't enough pins in a COM port to do this.)

Project/Demonstration Portion

The project/demonstration portion of the PIC-EL board was described in great detail in the 2007 *QST* article so I won't repeat it here. A summary is on the QST binaries Web page, however.

Programming a PIC with the PIC-EL III

Code to be loaded into the PIC in the PIC-EL can be developed in many different ways. If you follow the lessons in the *Elmer-160* course you will be able to understand the PIC architecture and command set well enough to write your applications in low level *machine language* code. Hey, it's not that hard. There are many examples of PIC code on the Internet, and you can easily collect pieces that are useful in your application. I also have several working examples on my Web page and in the FILES section of the PIC-EL YAHOO group of code that works on the PIC-EL.[6,7] There are a simple CW keyer, a frequency counter and a signal generator using the DDS-30 or DDS-60. The code for these sample applications is written in a form that makes it easy to understand (many comments) and can be used as a springboard for your own application. All are readable with a text editor. Go ahead and take a look.

Writing the Code into the PIC

One of the great benefits in having PICKit2 look-alike hardware is that it can be driven with the neat, stand-alone *Windows* application that Microchip freely distributes. Alternatively, it can be done with Microchip's freely distributed *MPLAB IDE* or their command-line version, *PK2CMD*. Note that Microchip also has versions of *PK2CMD* that run under the *Linux* and *Macintosh* operating systems as well.[8,9] With the PIC powered up and the USB cable connected, the PICKit2 application can be started. It will *connect* with the 18F2550 PIC in the PIC-EL and immediately report that it found a PICKit2. Figure 2 is a screenshot showing the PICKit2 application *connected* to the PIC-EL and ready to program a PIC. Programming a PIC is this easy:

- Power up the PIC-EL
- Connect the USB connector to the PC to the PIC-EL
- Start the *PICKit2* application
- Flip the PROGRAM / RUN switch in the PIC-EL to the PROGRAM position
- Insert the PIC to be programmed in the PIC-EL socket
- Select the PIC TYPE in the pull-down box in the *PICKit2* application
- Import the code (.HEX format) from the location on your PC where it is stored
- Press the WRITE tab
- Wait for a SUCCESS message. The verify operation is automatic.

Running the PIC Program on the PIC-EL III

Now you are ready to try out the PIC program that you just loaded into the PIC. It's just a matter of flipping the PROGRAM / RUN switch to RUN position and the PIC program will start up. Now you can see the results of your labor. Fun, isn't it?

How quick is it? I can literally change the source code of a program, assemble it, write it into the PIC, flip the switch and try it out in about 20 seconds! That's why developing code with a PIC-EL is so much fun.

Options

Microchip's *PICKit2* Application runs on the *Windows 98 SE* platform and beyond. I mentioned Microchip's stand-alone *PICKit2* application because it's perhaps the easiest way to go. There are a few other possibilities, however. Microchip's *MPLAB IDE* can also be used as well as Microchip's command-line interface program called *PK2CMD*. Microchip has versions of *PK2CMD* for *Windows* as well as Linux and *Mac OS X*.

Questions and Support

For up-to-date details and documentation regarding this project, please see my Web page, **www.cbjohns.com/aa0zz**. For additional support questions, see the PIC-EL YAHOO group or e-mail the author.

Conclusion

It's very satisfying to be able to develop a PIC program that performs a task exactly the way *you* want it to. Using the PIC-EL to develop and test code is a very convenient and enjoyable way to do this. With the low prices for PIC microcontrollers these days, it's really easy to think of lots of ways to use them. In simple configurations, you don't even need a crystal, so it's easy to throw a micro into a simple circuit to accomplish a task that you have in mind. Yes, it takes a bit of effort, but the end result is well worth it. Once you get going, you will be amazed how many more applications you will dream up.

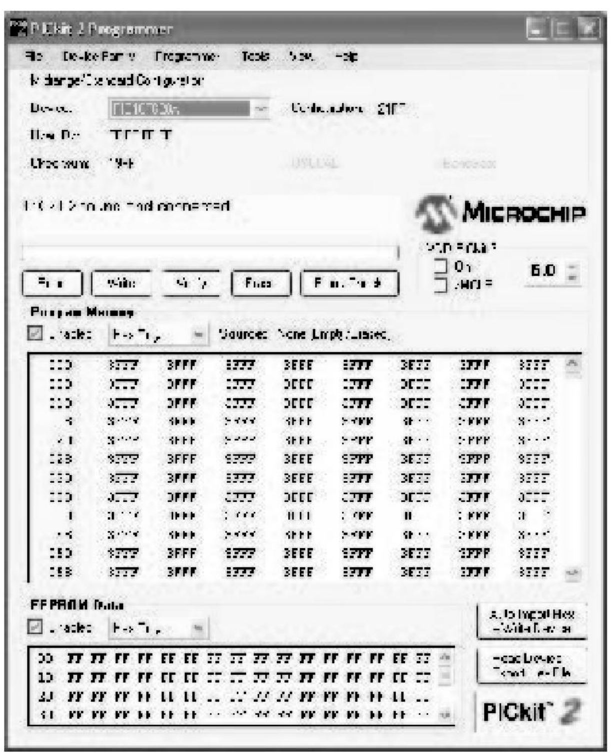


Figure 2 — Screenshot of Microchip *PICKit2* software screen.

Notes

[1]**www.amqrp.org/elmer160/index.html**.
[2]G. Heron, N2APB; J. Everhart, N2CX; J. McDonough, WB8RCR; E. Morris, N8ERO; J. Kortge, K8IQY, and the author.
[3]**www.groups.yahoo.com/group/PIC-EL**.
[4]Microchip Technology Inc (**www.microchip.com**).
[5]**www.arrl.org/files/qst-binaries**.
[6]**www.cbjohn.com/aa0zz**.
[7]See Note 3.
[8]**www.linux.com**.
[9]**www.apple.com/mac/**.

ARRL member and Extra Class licensee Craig Johnson, AA0ZZ, has earned BSEE and MBA degrees. He worked for Unisys for 35 years on the design and development of large computers and now works for Alliant Techsystems, a Defense Department contractor, developing microprocessor based products for the military. Craig holds seven US patents based on his work in computer hardware and software.

Craig got his first ham license in 1964 at the age of 14. He credits ham radio with sparking his interest in electronics and as a major factor in pointing him toward a career in electrical engineering. During and after his college years, however, he let his license lapse for several years and concentrated on computers.

For several years, Craig led a team of Volunteer Examiners (VE) and helped hundreds of people in the St Paul area get or upgrade their licenses. He still serves as a VE on occasion. He is an active member of the Minnesota QRP Society. Craig enjoys low power operating (QRP), DXing and contesting. He is happiest, however, when he is tinkering, building or experimenting with his new designs, circuits and software. His current interests are centered around projects that use microcontrollers, Direct Digital Synthesis and the new digital modes.

You can reach Craig at 4745 Kent St, Shoreview, MN 55126 or at **aa0zz@arrl.net**.