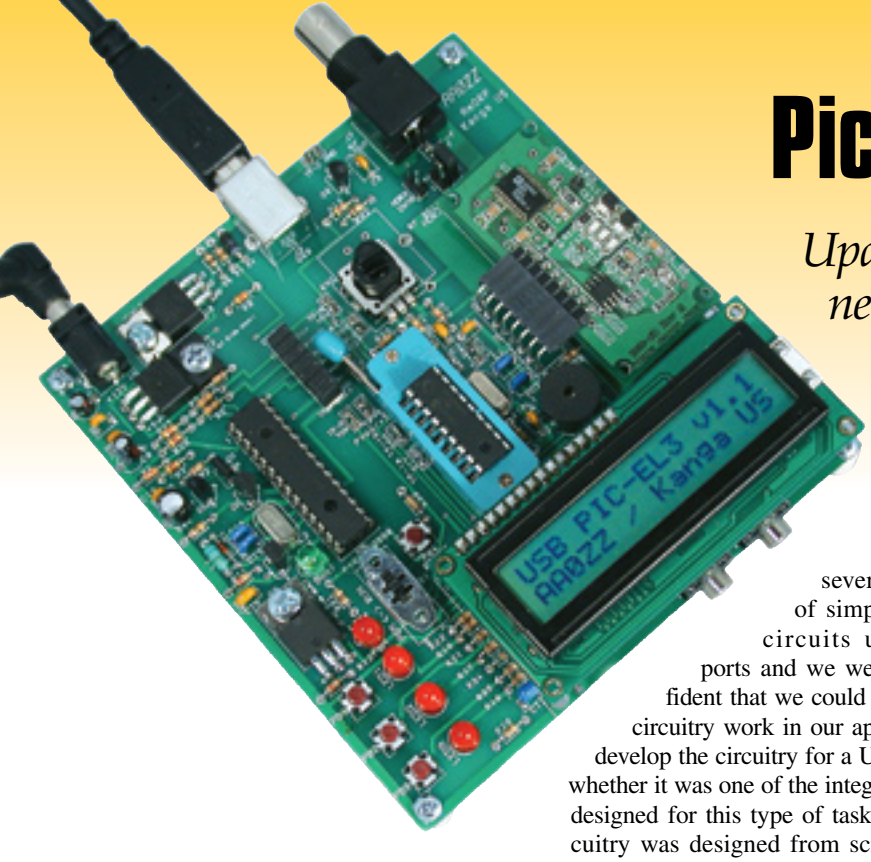
Pickle with USB I/O

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 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.

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 connection 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

[1]Notes appear on page 5.

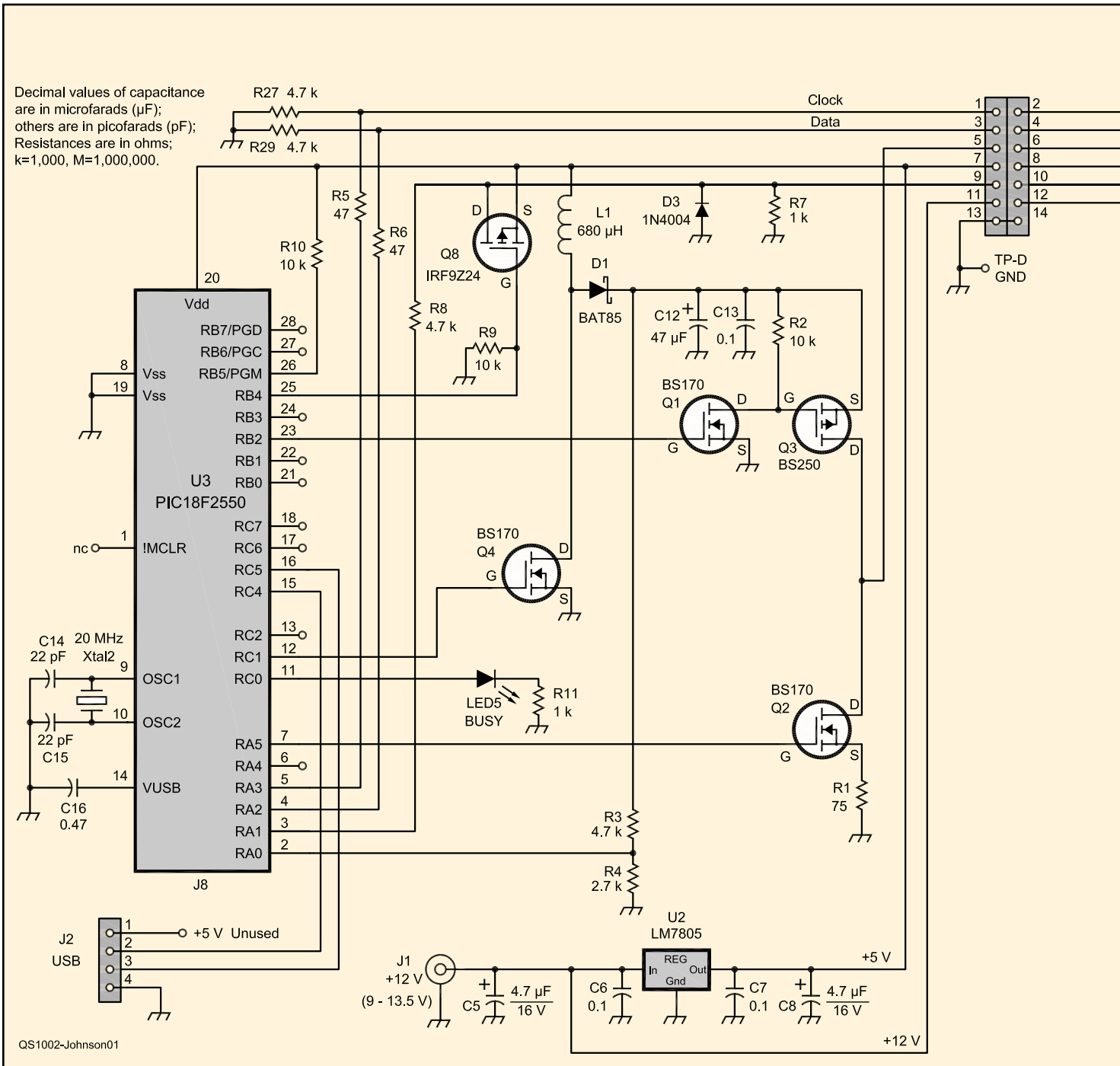
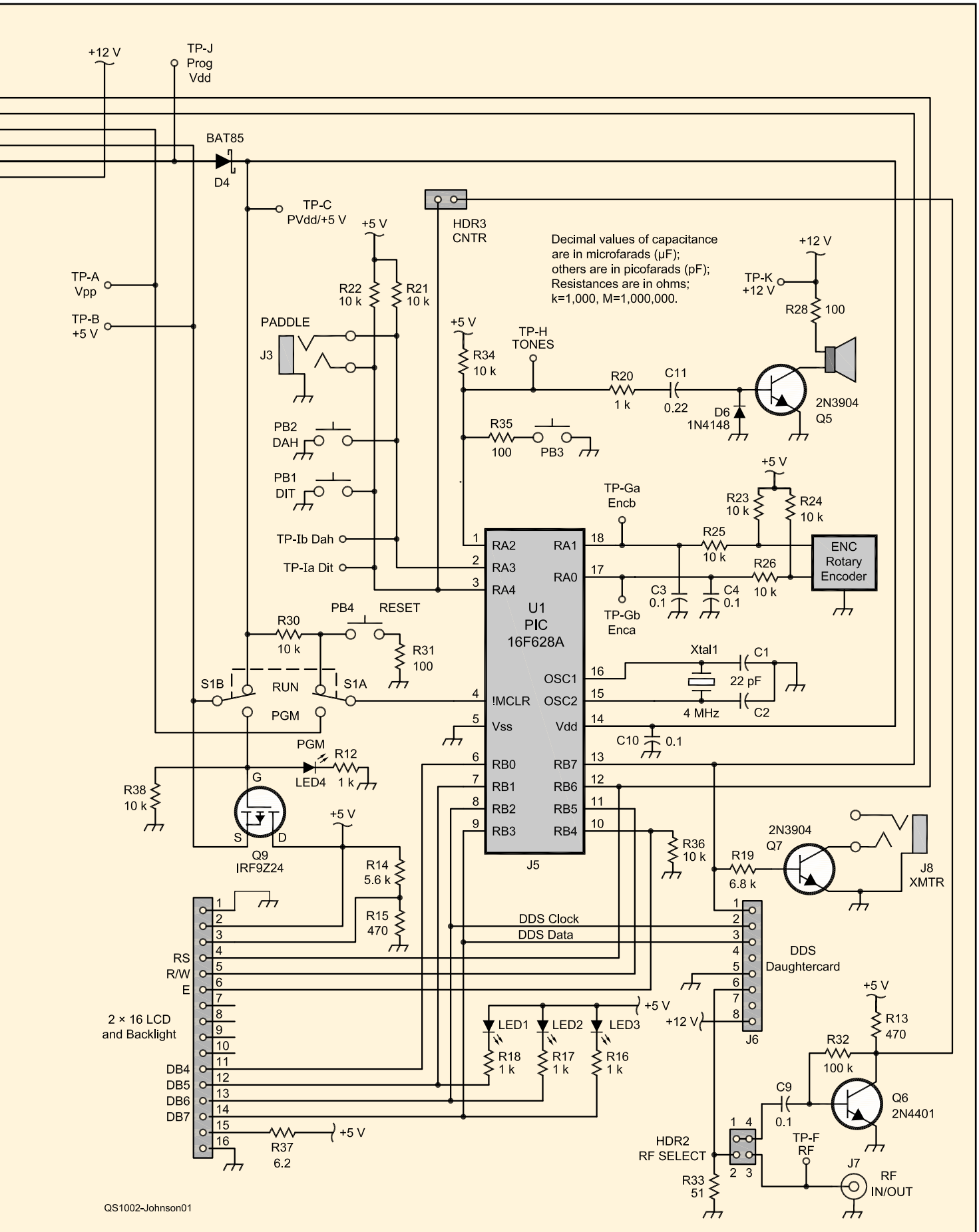


Figure 1 — Schematic diagram of the PIC-EL with USB interface. The right side is very similar to the previous versions. See parts list on page 6.


+12 V
TP-J
Prog
Vdd
BAT85
D4
TP-C
PVdd/+5 V
+5 V
HDR3
CNTR
Decimal values of capacitance
are in microfarads (µF);
others are in picofarads (pF);
Resistances are in ohms;
k=1,000, M=1,000,000.
TP-A
Vpp
TP-B
+5 V
R22
10 k
R21
10 k
PADDLE
J3
+5 V
TP-H
TONES
R34
10 k
R20
1 k
C11
0.22
D6
1N4148
+12 V
TP-K
+12 V
R28
100
2N3904
Q5
R35
100
PB3
PB2
DAH
PB1
DIT
TP-Ib Dah
TP-Ia Dit
+5 V
TP-Ga
Encb
R23
10 k
R24
10 k
R25
10 k
R26
10 k
ENC
Rotary
Encoder
C3
0.1
C4
0.1
TP-Gb
Enca
U1
PIC
16F628A
RA2
RA3
RA4
RA1
RA0
!MCLR
Vss
OSC1
OSC2
Vdd
RB0
RB1
RB2
RB3
RB7
RB6
RB5
RB4
Xtal1
C1
22 pF
4 MHz
C2
C10
0.1
PB4
RESET
R30
10 k
R31
100
S1B
RUN
S1A
PGM
PGM
R12
LED4
1 k
R38
10 k
G
S
D
Q9
IRF9Z24
+5 V
R14
5.6 k
R15
470
J5
R36
10 k
2N3904
Q7
R19
6.8 k
J8
XMTR
DDS Clock
DDS Data
DDS
Daughtercard
J6
RS
R/W
E
2 × 16 LCD
and Backlight
DB4
DB5
DB6
DB7
LED1
LED2
LED3
+5 V
+12 V
R18
1 k
R17
1 k
R16
1 k
R37
6.2
+5 V
+5 V
R13
470
R32
100 k
C9
0.1
Q6
2N4401
HDR2
RF SELECT
TP-F
RF
J7
RF
IN/OUT
R33
51
QS1002-Johnson01

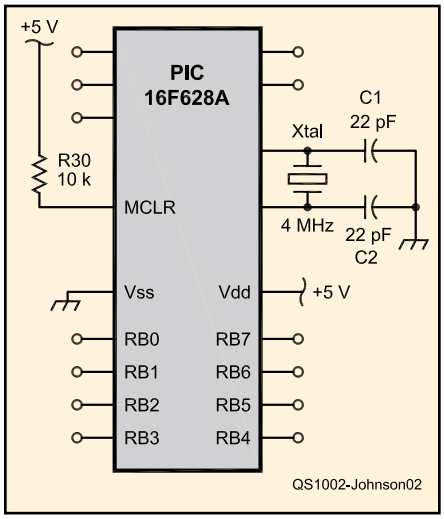


Figure 2 — Schematic of basic PIC circuit showing interconnections.

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. In brief, PIC experimenters using a PIC-EL have an easy way to use and understand the following hardware components:

- 18 pin PIC microcontroller (16F84A, 16F628/A, 16F88, 18F1320, and others)
- 4 MHz crystal oscillator
- 2 × 16 LCD (two lines of 16 characters)
- 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 mount and drive an NJQRP DDS daughtercard (DDS-30 or DDS-60)
- A stereo jack for connection to CW paddles.
- A stereo jack with transistor driver for transmitter keying
- A transistor "conditioner" for converting low level signals to levels required for PIC input detection.
- A multi-purpose BNC connector
- Selectable via a jumper at header HDR2
- Allow DDS output to be routed to the BNC
- Allow DDS output to be routed through the "conditioner" circuit 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 × 7 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

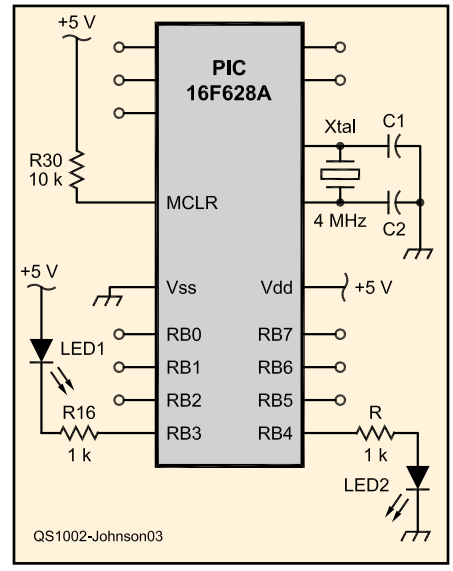


Figure 3 — Two methods of illuminating an LED.

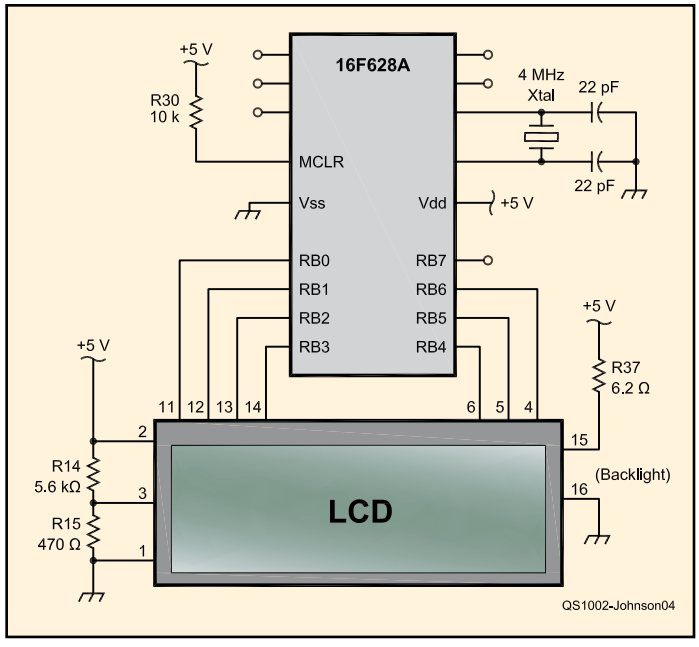


Figure 4 — Schematic of control circuitry for a backlighted LCD display panel.

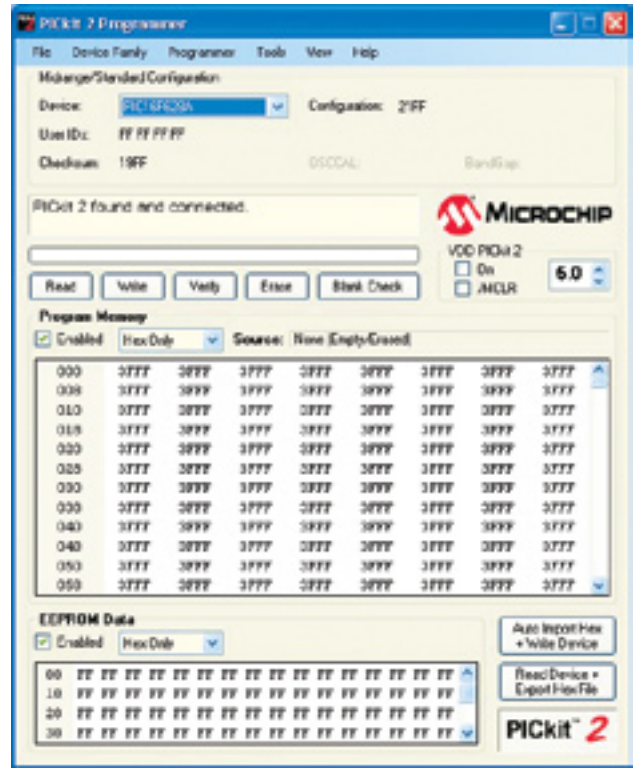


Figure 5 — Screenshot of Microchip *PICKit2* software screen.

PIC pins have multiple usages. It's much easier to understand the individual functions when they are isolated. Here are a couple of examples. Figure 2 shows the basic connections and components that are required to run any PIC. (The crystal is optional but is used in the PIC-EL.) Figure 3 shows the basic connections and components needed to light an LED, while Figure 4 shows one way of connecting an LCD. Many more examples were shown in the previous *QST* articles and are shown in the user manual.[5]

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 5 is a screen-shot 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.cbjohn.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.

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]PIC-EL III kits are now available from Bill Kelsey, N8ET, at Kanga. See his Web page, **www.kangaus.com** for details, or e-mail him at **n8et@arrl.net**.
[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, AAØZZ, 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**.

Parts list for Figure 1 (pages 2 and 3)

C1, C2, C14, C15 — 22 pF, ceramic disc (Digi-Key 490-3639-ND, Mouser 140-50N2-220J-RC).
C3, C4, C6, C7, C9, C10, C13 — 0.1 µF, monolithic (Digi-Key P4910-ND, Mouser 80-C317C104M5U).
C5, C8 — 4.7 µF, 16 V, electrolytic (Digi-Key P996-ND, Mouser 140-XRL16V4.7-RC).
C11 — 0.22 µF, monolithic (Digi-Key 445-2849-ND).
C12 — 47 µF, 16 V, electrolytic (Digi-Key P969-ND).
C16 — 0.47 µF, ceramic radial (Digi-Key BC1150CT-ND).
D1, D4 — BAT85 (Digi-Key 568-1617-1-ND).
D3 — 1N4004 (Digi-Key 1N4004-E3/73GI-ND).
D6 — 1N4148 (Digi-Key 1N4148FS-ND, Mouser 625-1N4148).
ENC — Rotary encoder (Digi-Key P10860-ND, or Digi-Key P12334-ND or Digi-Key P12335-ND).
HDR-1 — Pin header, 0.1", 2 × 7 position (Mouser 571-4-103328-3).
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 CP-102AH-ND).
J2 — USB B receptacle (Digi-Key WM17131-ND).
J3, J8 — Audio jack, ⅛", stereo (Digi-Key CP1-3513N-ND, Mouser 161-3507-E).
J4 — SIP header, 16 position (LCD) (Mouser 571-16404526).
J5 — DIP socket, 18 position (PIC) (Digi-Key ED3118-ND).
J6 — SIP socket, 8 position, 90° (SamTec SSQ-108-04-T-S-RA).
J7 — BNC jack, PCB mount (Mouser 523-31-5538-10-RFX).
J9 — DIP socket, 28 position (PIC) (Digi-Key ED3128-ND).
L1 — 680 µH inductor (Digi-Key M8156-ND).
LCD — Liquid crystal display, 2 × 16 character (ElectronixExpress.com 08LCD9).
LED1-LED4 — LED, T1-¾ (red) (Digi-Key 67-1110-ND).
LED5 — LED, T1-¾ (green) (Digi-Key 67-1109-ND).
P4 — SIP socket, 16 position (PCB) (Digi-Key ED7150-ND, Mouser 517-974-01-16).
PB1, PB2, PB3, PB4 — SPST push-button, momentary (Digi-Key P8079SCT-ND).
Q1, Q2, Q4 — BS170, TO-92 (Digi-Key BS170-ND).
Q3 — BS250, TO-92, with two sides flat (Digi-Key BS250P-ND).
Q5, Q7 — 2N3904 transistor, NPN, TO-92 (Digi-Key 2N3904D26ZCT-ND).
Q6 — 2N4401 transistor, NPN, TO-92 (Digi-Key 2N4401-ND, Mouser 610-2N4401).
Q8, Q9 — IRF9Z24, TO-220 (Digi-Key IRF9Z24PBF-ND).
R1 — 75 Ω, ¼ W (Digi-Key 75EBK-ND).
R2, R9, R10, R21-R26, R30, R34, R36, R38 — 10 kΩ, ¼ W (Digi-Key 10KEBK-ND, Mouser 291-10K-RC).
R3, R8, R27, R29 — 4.7 kΩ, ¼ W (Digi-Key 4.7KEBK-ND).
R4 — 2.7 kΩ, ¼ W (Digi-Key 2.7KEBK-ND).
R5, R6 — 47 Ω, ¼ W (Digi-Key 47EBK-ND).
R7, R11, R12, R16-R18, R20 — 1 kΩ, ¼ W (Digi-Key 1.0KEBK-ND, Mouser 291-1K-RC).
R13, R15 — 470 Ω, ¼ W (Digi-Key 470EBK-ND, Mouser 291-470-RC).
R14 — 5.6 kΩ, ¼ W (Digi-Key 5.6KEBK-ND, Mouser 291-5.6K-RC).
R19 — 6.8 kΩ, ¼ W (Digi-Key 6.8KEBK-ND, Mouser 291-6.8K-RC).
R28, R31, R35 — 100 Ω, ¼ W (Digi-Key 100EBK-ND, Mouser 291-100-RC).
R32 — 100 kΩ, ¼ W (Digi-Key 100KEBK-ND, Mouser 291-100K-RC).
R33 — 51 Ω, ¼ W (Digi-Key 51EBK-ND, Mouser 291-51-RC).
R37 — 6.2 Ω, ¼ W (Digi-Key 6.2EBK-ND, Mouser 291-6.2-RC).
S1 — Slide switch, DPDT (Digi-Key SW102-ND).
Shunts for HDR1-HDR3 — shunt, 0.1", 2 position (Digi-Key S9000-ND, Mouser 571-2-382811-1).
SPKR — Speaker (Digi-Key 433-1028-ND).
U1 — PIC16F628A microcontroller with diagnostics pre-loaded (Mouser 579-PIC16F628A-E/P with AAØZZ diagnostics).
U2 — L7805 voltage regulator, 5 V, TO-220 (Digi-Key 497-1443-5-ND, Mouser 511-L7805 ABV).
U3 — PIC18F2550 microcontroller with PICKIT2 code pre-loaded (Digi-Key PIC18F2550-I/SP-ND with code).
XTAL1 — Crystal, 4 MHz (Digi-Key X405-ND, Mouser 520-HCU400-20).
XTAL2 — Crystal, 20 MHz (Digi-Key CTX416-ND).