

The Smart Keyboard

Build a low cost keyboard keyer that works well and can work with your paddles, too!

Joe Lunsford, N4YG

CW is my favorite operating mode; in fact it is really my only operating mode. As I grow older I prefer a nice long chat with someone on a topic of common interest rather than chasing DX, as in my younger years. I also like good CW. I think there are many who have the same operating style and preferences. Many of the hams with whom I have contacts use keyboards, allowing them to send code with fewer errors and at higher speeds.

Available for many years, CW keyboards have gained in popularity in the last 20 years as computers have become widely used. Most keyboard programs make use of available programming languages and use the computer as the sole source of timing for the generation of Morse characters. These and other limitations described below caused me to develop my own keyboard Morse system.

The Smart Keyboard

Recently I began to look into the possibility of keyboards for my shack, just to try out and use occasionally along with the paddles. I use a laptop computer for logging and it seemed that I could acquire a good keyboard program and little would change at my operating position, other than the addition of a cable and an interface between the computer and the radio. As I began looking around and talking with others who had experience with keyboards, I noticed three things in the available keyboard programs that caused me to reject most of them.

- **Timing inaccuracy:** This is a concern because users are not provided with the flexibility to take advantage of the inherent capability of computers to generate accurately timed intervals. In most cases, timing can be guaranteed only to an accuracy of about 1 millisecond (some are timed to an accuracy of 10 ms). At 25 WPM a dot lasts 48 ms and would be timed to an accuracy of 2 percent. While this may be sufficiently accurate for some CW operators, at speeds above 25 WPM it becomes a problem.
- **Varied behavior among the many types of PCs and various versions of the Windows operating system:** This can cause a

keyboard program to perform very differently on the various PCs that may be found in ham shacks.

- **Port incompatibility:** Most keyboard programs, especially those available at no cost, use the serial or parallel port to interface between the computer and the transmitter keying line. My computer is about two years old and has only USB ports — neither a serial nor a parallel port.

These issues caused me to consider development of a keyboard program along with an interface unit using the USB port of my computer. The fact that I had already written the code to support a low-speed USB peripheral made the decision easier. Had I not already done this for a previous project, I never would have attempted to develop the Smart Keyboard. The time required to develop the firmware for the PIC16C745 and the code for the host computer would have been unreasonable for such a project. The task of researching the USB specification and the protocol for the interaction of the host computer and the USB peripheral is significant.

Requirements

The USB *interface unit* and the graphical user interface (GUI) program I decided to develop would have none of these shortcomings. These are the major features that I desired:

- An interface unit with a low-speed USB port.
- A GUI compatible with most *Windows* operating systems.
- Dot, dash and space timing accuracy of less than 10 μ s.



- An interface unit that can also serve as a fully functional stand alone keyer.
- A sidetone generator, with variable tone and volume, built into the interface unit.
- Keying of positive or negative keyed rigs.
- Selectable keying modes such as iambic, dot memory and dash memory.
- Keyer speed, weight and keying modes controlled by the host computer.
- User control of word and character spacing.

The design that satisfies each of these requirements also includes an interface unit with a low-speed USB port. This unit controls all timing and is also a fully functional electronic keyer. The USB port provides power for the operation of the interface unit. The host computer communicates with the interface unit whenever necessary. During keyboard operation, the host computer sends requests to the interface unit for Morse characters to be sent to the transmitter. Little information is sent from the interface unit to the host, only status information.

The Smart Keyboard

The *Smart Keyboard* solves all three of the problems I described. Timing is determined by the interface unit. The interface unit is a complete keyer that uses a microcontroller to generate dots and dashes to an accuracy of a fraction of a microsecond. This same microcontroller provides a USB port for com-

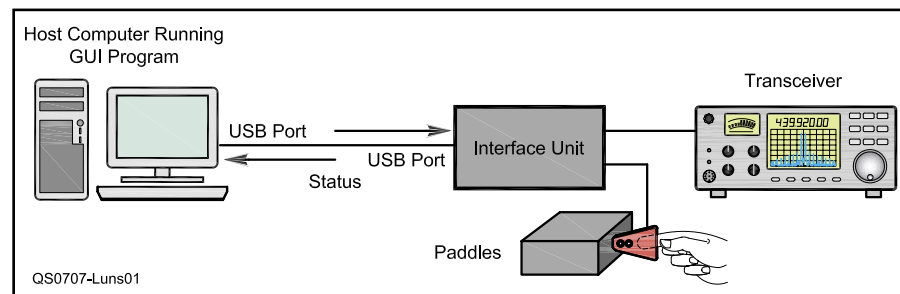


Figure 1 — Block diagram of the smart keyboard.

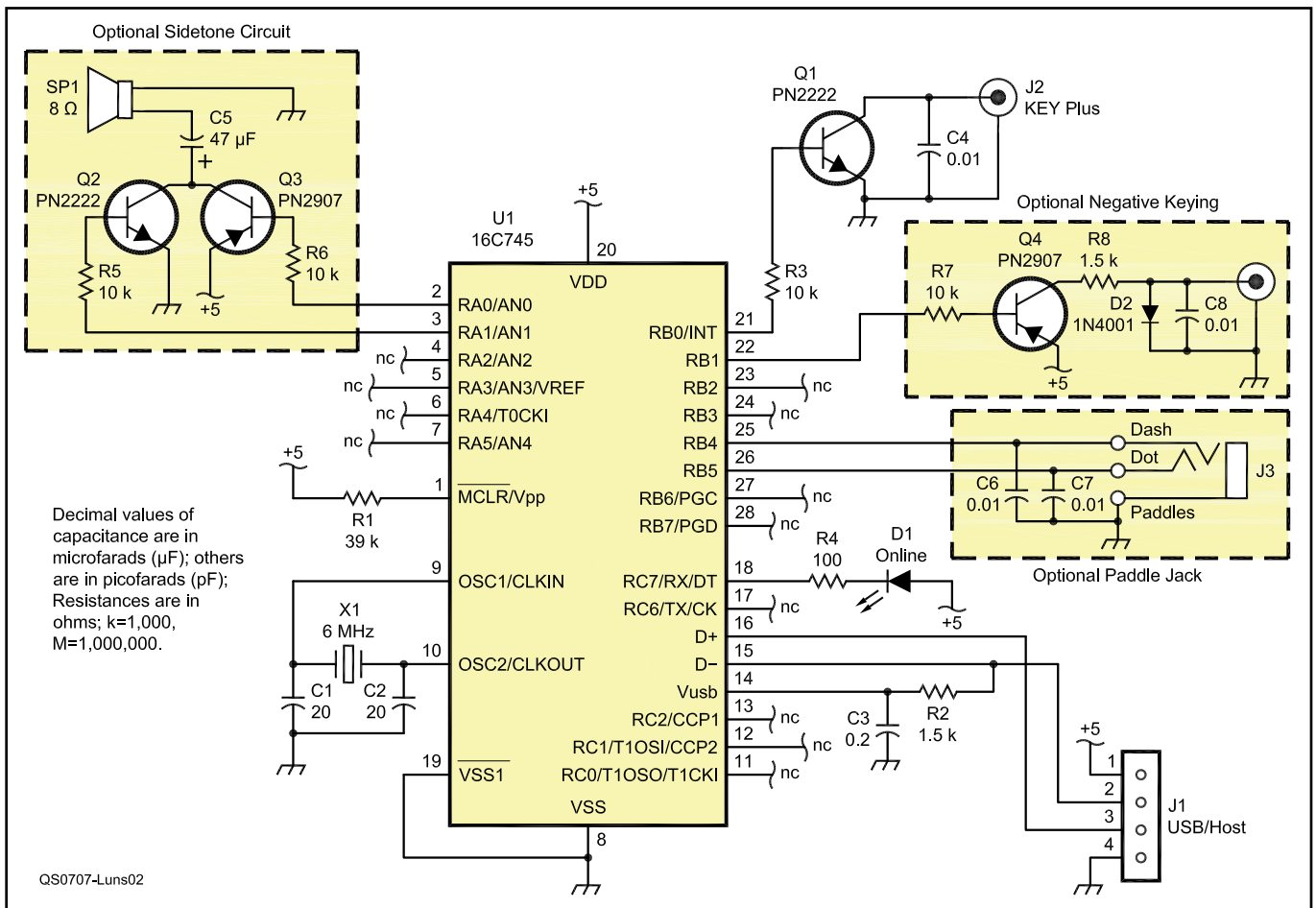


Figure 2 — Schematic diagram of the smart keyboard. Parts are listed in Table 1.

munication with the computer. Because the computer only asks the interface to generate dots, dashes and spaces, the behavior is independent of the computer used as the host. The *Smart Keyboard* program includes a GUI that allows the operator to control all the parameters of the keyboard and keyer. It also provides the operator with visual information on the status and operation of the keyboard.

Because the interface is also a keyer, paddles can be plugged into the interface and the operator can use the keyer or keyboard at any time interchangeably. A sidetone is provided in the interface and its tone and volume are controlled through the GUI, which also controls SPEED, WEIGHT, LETTER SPACING and WORD SPACING. The GUI program and the object code for the 16C745 microcontroller are provided. The interface unit hardware can be easily constructed for less than \$50.

The Interface Unit

Figure 2 is a schematic diagram of Smart Keyboard interface unit circuit. The unit is amazingly simple in terms of hardware. The 16C745 is the heart of the unit. A 6 MHz crystal provides the timing reference for the oscillator. From this an internal phase locked loop (PLL) provides the 24 MHz clock for

the processor. A very accurate 24 MHz clock is a requirement for operation of the USB module. The USB connector has four conductors, two of which supply 5 V with maxi-

mum current specified in 100 mA units.

Our interface unit requires much less than one unit of current. During the *enumeration* process the interface unit requests

Table 1

Parts List for Smart Keyboard*

C1, C2 — 20 pF, 50 V, $\pm 5\%$ disc capacitor (Mouser 140-50N5-200J).	C3 — 0.22 μF , 20% axial lead capacitor (Mouser 581-SA-105E224MAR).	C4, C6-C8 — 0.01 μF , 50 V, -20%, -80% disc capacitor (Mouser 140-50V5-103Z).	C5 — 47 μF radial lead electrolytic capacitor (Mouser 140-HTRL25V47-RC).	D1 — T-1 LED (Mouser 351-3102-RC).	D2 — 50 PIV rectifier diode (Mouser 621-1N4001).	J2, J4 — RCA phono jack (Mouser 161-2052).	J3 — $\frac{1}{4}$ " phone jack (Mouser 161-0023-E).	J1 — USB Type B connector PCB mount (Mouser 154-2442).	Q1, Q2 — PN2222A NPN transistor, TO92 (Mouser 511-PN2222A).	Q3, Q4 — PN2907A NPN transistor, TO92 (Mouser 511-PN2907A).	R1 — 39 k Ω , $\frac{1}{4}$ W carbon film resistor (Mouser 291-39K-RC).	R2, R8 — 1.5 k Ω , $\frac{1}{4}$ W carbon film
--	--	--	---	------------------------------------	--	--	--	--	---	---	--	---

resistor (Mouser 291-10K-RC).	R3, R5-R7 — 10 k Ω , $\frac{1}{4}$ W carbon film resistor (Mouser 291-1500-RC).	R4 — 100 Ω , $\frac{1}{4}$ W carbon film resistor (Mouser 291-47-RC).	SP1 — Mylar speaker, 1.18" diameter (Mouser 254-PS605).	U1 — 16C745 Microcontroller OTP programmed with firmware (Mouser 579-PIC16C745ISP).	X1 — 6 MHz Parallel Cut Crystal (Mouser 559-FOX060-20).	USB cable, 1.8 meters (Mouser 172-1024).	Enclosure, aluminum (painted grey) (Mouser 537-101-P).	Threaded spacers (4) (Mouser 534-1891).	4-40 machine screws (8) (Mouser 534-9300).	Prepared Vero Board 2 x 2 $\frac{3}{4}$ " (Ocean State 12-618).	28-pin DIP socket (Mouser 571-3902622).
-------------------------------	--	--	---	---	---	--	--	---	--	---	---

*All parts available from Mouser Electronics, www.mouser.com, or Ocean State Electronics, www.oselectronics.com. All listed items are included in the parts package available from the author for \$60.

1 unit of power from the host. Enumeration is a process that occurs when a USB device is attached to the host computer. At this time the host requests information from the peripheral device so that it can identify it and communicate with it. Data is transferred over the D+ and D- lines. A 1500 Ω resistor connected between D- and V_{USB} is required for the host to recognize the interface unit as a low speed USB peripheral.

A light emitting diode (LED), D1 provides a visual indication that the host computer has enumerated the peripheral, in this case our interface unit. When D1 is illuminated, the computer has recognized the interface unit and is ready to use the peripheral to transfer data. The keying transistor, Q1, for keying positive voltage, is connected to pin 21.

The remainder of the circuitry is optional and could be omitted. These circuit blocks include the sidetone, negative keying and paddle circuits (required to use the unit as a manual keyer). While these may be omitted, I suggest that it is just as easy to include them all because the parts cost is minimal.

The tough task was writing the code for the interface unit, but I had already completed the difficult part, the code for USB support. Code was needed to make the interface a keyer that could take commands from the USB module. A small portion of the required code was taken from the Smart Keyer Lite.¹

Graphical User Interface

The GUI program was written in *Visual Basic 6*. Figure 3 shows a screenshot of the program in operation. The window is resizable. I like to size the window so that it covers ¼ to ½ of the screen area so that other programs, such as my logging program, can be running and viewed simultaneously. At this size, there is ample area to view all that is going on with the keyboard and for the controls.

We will take a brief tour of the GUI window and discuss some of the important aspects of the window. The pull-down menus are at the upper left. The FILE menu allows you to select an operator whose call, personal data and preferences for SPEED, WEIGHT, SPACINGS, KEYSER MODES (current modes are shown in the four little boxes near the bottom) and messages have already been specified. You may also define new operators and their specific data. You can also save current settings so that any changes will be saved and applied when the program is restarted.

You may define as many operators as you wish. You and your spouse might use the same station, for example. You may define more than one version for the same operator or call. One person may have several opera-

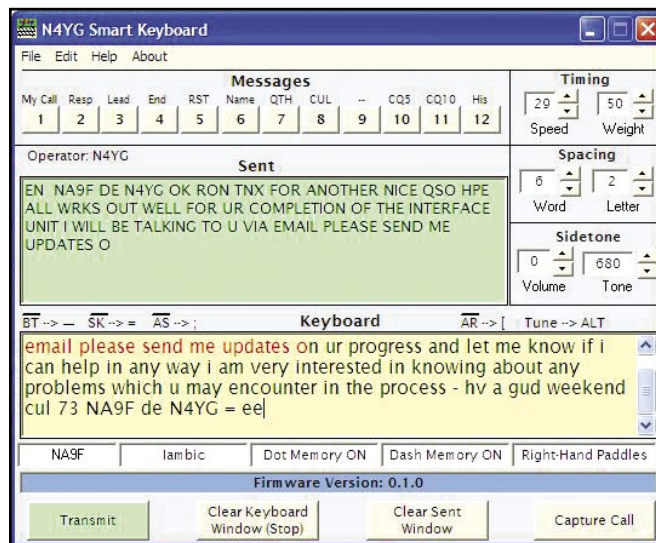


Figure 3 — Graphical user interface (GUI) of the smart keyboard's operating software.

tor configurations defined for himself such as, one for working DX, one for contesting, one for ragchewing and one for Field Day.

The EDIT menu allows you to edit keyer features and messages. The 12 message buttons are just beneath the pulldown menus. These buttons have the same effect as pressing the corresponding function keys. When clicked (or if the corresponding function key is pressed), the contents of the particular message are copied to the keyboard window. Each of the message buttons is labeled with a short title, selected by the operator, to remind him of the contents of the message. You will appreciate these if you have been frustrated by programs that provide nothing regarding the contents of the message, forcing you to make a list to keep nearby. One can view the actual contents of a particular message by placing the cursor over a message button. The message can also be viewed by clicking EDIT / MESSAGES / MESSAGE BUTTON.

The SENT window is located just below the message buttons and shows text that has been sent to the transmitter. This window is for viewing only and cannot be edited. Just above and to the left of this window, the current operator's call is shown, for information. The KEYBOARD window is the center of most of the activity and is located just below the SENT window. In normal operation any text placed in this window is sent to the interface unit and in turn to the transmitter in the form of dots and dashes at the proper time. As a character is sent, its color changes from black to red and the character is also copied to the SENT window. Once all the text in the window is sent and there is a three-second period of inactivity in which no additional text is entered into the window, the KEYBOARD window is cleared. The KEYBOARD window is also used for composing or editing messages. The area above the KEYBOARD window contains a few reminders of characters that are

surrogates for the prosigns \overline{BT} , \overline{SK} , \overline{AS} , \overline{AR} , meaning for example that when the character “=” is sent, you hear di-di-di-dah-di-dah, \overline{SK} . Pressing the ALT key causes a keydown condition for tuning.

An INFORMATION area is located beneath the KEYBOARD window and consists of one row of five boxes with a single wide box below. The far left box in the row of five boxes will contain the call of the station being worked, if it has been entered. When his call is entered, it is also copied to the Windows clipboard and can be pasted into other applications. I paste (by pressing CONTROL-V) the call into my logging program so I do not have to type it again. You can also paste it into qrz.com.

Your call, or that of the current operator, and the other station's call are available for reference in messages, so that your call as well as the call of the station being worked can be inserted at any place in a message. For example, if N4YG is the current operator and the entry in the HISCALL box is K4BFT, then when this text is ready for transmission K4BFT is sent instead. Similarly, if the message text contains MYCALL then N4YG is sent instead. The other four boxes in the row of five are for information only and can only be changed by editing keyer features from the EDIT menu.

The remaining box is just below the row of five and encompasses the entire width of the GUI window. This is the general information or ALERT window. In normal operation it gives an indication that the GUI has found the USB interface and shows the interface firmware version. When editing messages, it gives information on what action is appropriate during the editing process.

Four buttons are located below the alert window. The first is the TRANSMIT/PAUSE button. When clicked, this button changes color and its label changes to either

¹Notes appear on page 39.

TRANSMIT or PAUSE, depending on its initial condition. When green in color, its label is TRANSMIT. In the transmit condition, any text in the Keyboard window will be sent beginning with the leftmost character. If the TRANSMIT/PAUSE button is clicked while in the transmit mode, transmission will cease, the button color will change to beige and its label will be PAUSE and transmission will resume if the button is clicked again. This button is very useful. I select PAUSE when I am listening to the other station. I can then begin to compose my responses if I wish and when my turn comes, I can click again to select TRANSMIT and begin sending what I have entered. The second button clears the keyboard window and in doing so immediately stops any further sending.

The keyboard window is also cleared if a paddle is tapped while the keyboard is transmitting. The third button simply clears the SENT window. The fourth and final button is another means for entering the call of the station being worked. When clicked, a window pops up for entry of the call.

Finally, the controls for setting keyer timing parameters are located at the upper right corner. Using the scrolling controls, SPEED, WEIGHT, WORD SPACE, LETTER SPACE, SIDETONE VOLUME and TONE can be set. Table 2 shows a bit about how these are calculated. Each time any of these parameters changes, new values for all necessary parameters (24 bytes) are calculated and sent to the interface unit.

The relationship between length or duration of dots, dashes and spaces is taken from *The ARRL Handbook for Radio Communications* and is the simple relationship — WPM (words per minute) = $2.4 \times$ dots per second. This assumes a weight of 50%, or space and dot durations the same and dashes equal three dot intervals.²

Construction

Figure 4 is a photograph of the completed interface unit. Figure 5 shows the internal layout of my implementation. The unit is quite small, so small that I was sure that the wrong enclosure had been shipped to me when it arrived. The enclosure is made by LMB and measures $4 \times 2\frac{1}{4} \times 1\frac{1}{4}$ inches. My implementation of the interface unit has the USB jack on the front of the unit and the paddle and keying jacks on the rear. If a wider enclosure is used, all jacks could be on the rear panel.

My first interface circuit was constructed on a $2 \times 2\frac{3}{4}$ inch section of Vero board. The inter-

Table 2
Description of Timing Parameter Calculations

Parameter	Range	Units	Step Size	Meaning
Speed	5-50	WPM	1	WPM = $2.4 \times$ dots/s
Weight	30-70	Percent	1	Weight (dot + space) duration
Word Space	3-15	Dot Lengths	0.1	Number of 50% weighted dot lengths
Letter Space	2-11	Dot Lengths	0.1	Number of 50% weighted dot lengths
Volume	0-100	Percent	1	Percentage of log-weighted volume
Tone	400-1400	Hertz	10	

face unit shown uses a printed circuit (PC) board that measures 1.6×1.8 inches. The USB jack and the ONLINE LED are mounted on the PC board as well, but note that the LED is mounted on the bottom of the board. A PC board is included in the parts package, along with all the parts needed to construct the interface unit. Vero board works well and is another option for one-time projects such as this. The board has a rectangular grid of holes spaced 0.1 inch apart and conducting copper foil strips centered on the holes, running in one direction. With the copper foil strips running east west and most parts oriented in a north-south direction, wire jumpers are used to connect parts as desired. Before placing parts on the board, the copper foil strips are broken by removing foil to eliminate undesired conducting paths.

Before placing parts on either the PC or the Vero board, use the board and the other parts to mark the location and mounting holes on your enclosure. Pay particular attention to the placement of the holes

through which the USB jack and LED will protrude. Mounting holes for the board should be placed so that the USB jack will protrude only slightly from the surface of the enclosure. Cutting the rectangular hole for the USB jack will probably be the most difficult single task. You can drill holes for the other jacks. Mount the jacks and speaker first. You can glue the speaker as I did with silicone sealant if you wish. Assemble the circuit board while allowing the glue to dry. If you choose Vero board, solder carefully. I have found solder bridges to be much more likely with Vero board than with a printed circuit. Remember that the LED must be soldered on the bottom of the board. Do not mount the LED until you have made sure that everything else fits together properly.

Check the board thoroughly in good light before proceeding. Before mounting the LED, measure the height of the standoff on the board mounts. The LED should be mounted on the bottom of the board so that the distance from top of the small shoulder on the LED to the board is the same distance as the standoff height. Do not place the 16C745 into its socket until all soldering is completed.

Attach connecting wires to the completed board. I like to connect these to the bottom of the board. Make sure these have plenty of length and that they are marked appropriately. There should be seven wires attached as follows:

- For SPEAKER, two (one plus ground).
- For PADDLE JACK, three (dot, dash and ground).
- For KEYING JACKS, two (one each, share ground with PADDLE JACK).

Mount the circuit board to the enclosure using 4-40 hardware and threaded standoffs. Attach wires from the circuit board to the appropriate components. Insert the ICs into their sockets and verify pin 1 locations and proper insertion.

Operation

The smart keyboard interface



Figure 4 — Smart keyboard interface unit — rear view.

unit is now ready to use. All you need to do is connect it to your computer (running *Windows 98 Second Edition* or higher). Depending on the *Windows* version, the computer will probably tell you that new hardware has been found and may ask if you want to search for the best driver. Click YES and the computer should find the proper driver and install it. If you are running *Windows XP* or *Vista* all this will probably happen without your having to do anything. *XP* also gives you a *kerplunk* sound when a USB peripheral is found and enumerated. After the interface unit has been enumerated, the LED will also light.

You can see that the peripheral has been enumerated by looking at the hardware list in the *Windows Device Manager*. One way to do this is to right-click MY COMPUTER and select PROPERTIES. In the PROPERTIES window, click the HARDWARE tab and then click DEVICE MANAGER. In the hardware list, look under HUMAN INTERFACE DEVICES. There may be two entries for the interface unit, reading HID-COMPLIANT DEVICE and USB HUMAN INTERFACE DEVICES. You can identify the ones that apply to our interface unit by unplugging the USB cable. These entries should disappear but reappear when the unit is reattached.

The interface unit is now a fully functional keyer, with one exception. The keyer parameters are fixed (with default values I programmed) and cannot be changed until the GUI is up and running. Try out the keyer. Plug in paddles. Touch one of the paddles. You should hear dots or dashes. If the paddles are reversed (dots where dashes should be), don't worry, you can correct that when you run the GUI. The SPEED, VOLUME, TONE and WEIGHTING can also be changed from the GUI.

If all is well to this point, it is time to install and run the GUI program. When the SMART KEYBOARD window appears, it should have the message in the alert area at the bottom of the window as follows: FIRMWARE VERSION 0.2.0 over a blue background. Otherwise, the background will be red with alternating messages, INTERFACE UNIT NOT FOUND and PLEASE CONNECT THE SMART KEYBOARD INTERFACE TO THE USB PORT. Connecting the interface unit should result in the previous message. This message indicates that the GUI has found the interface unit. The GUI should now look like Figure 3, except that the KEYBOARD, SENT and CALL windows will be blank. N4YG will probably be the operator shown at the upper left. N4YG was one of the operator files in the setup package. If you click MESSAGE 1 button, you should hear my call, N4YG, in CW.

You will need to read the entire help message to see all of the things you can do with

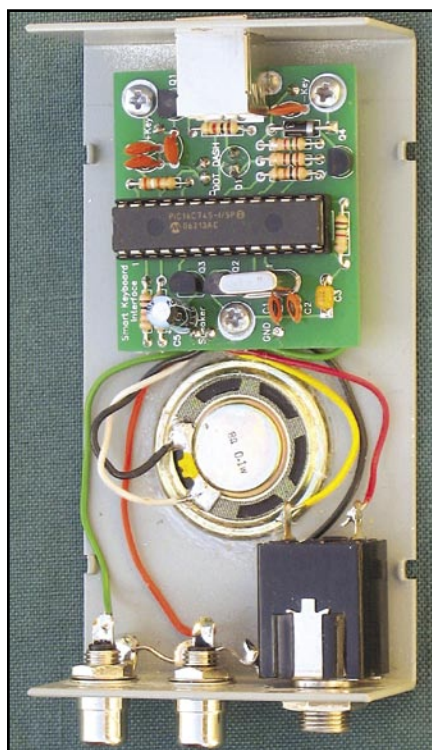


Figure 5 — Internal view of smart keyboard interface unit.

the smart keyboard. Try this short exercise just to get you started. Click FILE, then click NEW OPERATOR. Fill in your call, name and location (QTH) and click OK. Now click FILE and then click SAVE and respond YES to the message box. You have created an operator definition for yourself, but with the messages and options that are associated with N4YG. Your call now appears as the operator just above the SENT window. Press F1. You should hear your call instead of mine. You will find that in all the messages with which N4YG was heard, your call is now heard. You may edit all 12 messages to suit your needs. These will be saved. You can also change WEIGHT, SPEED, any other parameters, and then click FILE, then SAVE to save the new settings.

You will want to get a feel for typing into the KEYBOARD WINDOW while code is being sent. If the TRANSMIT/PAUSE button is labeled TRANSMIT, then as you begin typing, code will begin to be sent. You can type ahead if you type faster than the keying speed. As you type, the text displayed is black. Once a character is sent, it turns red and is copied to the SENT window. When you stop typing, code will continue to be sent until all text in the KEYBOARD WINDOW is red; then, if there is no activity for 3 seconds, the KEYBOARD WINDOW is cleared. The SENT window can only be cleared manually by clicking the button at the bottom of the window. The help file explains all the features of the GUI. You should read all of this file.

Final Remarks

I believe many of you will find the smart keyboard to be fun and easy to use. It should be relatively easy for you to construct the interface unit. I have been using mine for several months now and I love it. Ron, NA9F, Art, AB4RL, Joe, W4NKP, Dave, AA3EJ and Bill, W7NTA, have all completed this project without difficulty. I wish to thank them for their many excellent comments that have been invaluable in making the smart keyboard design much better than it would have otherwise been. Where do I go from here? Well, the *SMART Key-n-Log* is well underway. It is a GUI program that uses the same interface but includes a logging capability as well as the CW keyboard. Several of my friends are already using it.

Please report any bugs to me. The parts package is complete, including a PC board, the programmed 16C745 and even the enclosure and a USB cable. The only other items you will need in order to construct the interface unit are tools for preparing the enclosure, a soldering iron and solder. You may acquire the parts yourself if you wish. The code for the GUI and the firmware for the 16C745 are available for download.³ The cost of the parts package is \$60, which includes shipping within the USA. Have fun with it and I would love to hear from you.

Notes

¹J. Lunsford, N4YG, "The Smart Keyer Lite," *QST*, May 2004, pp 42-45.

²*The ARRL Handbook for Radio Communications*, 2007 Edition. Available from your ARRL dealer or the ARRL Bookstore, ARRL order no. 9760. Telephone 860-594-0355, or toll-free in the US 888-277-5289; www.arrl.org/shop/; pubsales@arrl.org.

³www.arrl.org/files/qst-binaries/Lunsford0507.zip

Joe Lunsford, N4YG, was first licensed as WN4RUF in 1969 and has held an Amateur Extra class license for 28 years. He has a Master's degree in electrical engineering. Joe recently retired from his job of over 36 years. He is an avid CW operator who in previous years chased DX regularly, but now enjoys a nice CW QSO. He has designed a dozen or more electronic keyers. The most notable of these is the smart keyer series. He is also the designer of the smart filter. These products were produced and sold in the '80s and '90s, and several hundred remain in service. This venture was discontinued in the late '90s. Since retiring, Joe continues to work part-time, but manages to spend more time operating. Most of his additional time is spent designing electronic products for ham radio and other applications, playing golf and enjoying his five grand-children. You can reach Joe at 1304 Toney Dr, Huntsville, AL 35802 or at n4yg@comcast.net. 