Auto-bug Keyer with Message Memory

Many CW fans enjoy iambic keyers, but if you don't want your keying to sound too automatic, you now have another choice: the "auto-bug keyer."

Taka K. Shimazu, JA3KAB

I found myself missing the distinctive sound of code sent from a "bug" by a good operator. "Manual" CW can have more personality than code sent using an "automatic" keyer.. Sometimes they have a bit longer dash, sometimes a classic "swing" rhythm that can be music to the ears. However, intensive practice is necessary to make the bug sound "ear-pleasing." My bug generated CW that couldn't be called ear-pleasing!

So What's the Difference Between Good Bug Keying and My Bug Signals?

The major difference is in the spacing. Good operators maintain the same spacing between dashes and between a dash and dot. Compare the paddle manipulations and signals of A and B in Figure 1. Significant bug practice is required mainly to make the spacing constant.

Fortunately, the problem can be solved in an electronic implementation by adjusting the spacing using the keyer's logic. See the difference between signals B and C in Figure 1. The paddle operations are the same for each.

This is the auto-bug keyer concept. It's electronics, not magic, that allow good sounding bug signals *without* the need for lots of practice.

Auto-bug Mode Message Memories

Your bug-like signal can be recorded just as it is sent. Variable length dashes are recorded in the message memory with a length of dash that can be as short as a dot or up to about eight dots in length. Spacing is also variable, except the minimum length is one dot — important for a good-sounding auto-bug. Variable space memory makes the recording more like the way you heard it from the monitor when it was recorded.

This key also simulates the early "sideswiper" configuration. Just select reciprocal mode. You can use the paddle for reciprocal key operation, both in real time and for recording. *Spaces* are again adjusted to keep a minimum of a single dot length.

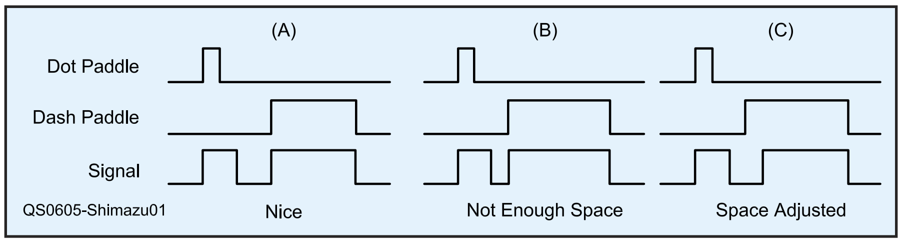

Figure 1 — At A an experienced bug operator's spacing, at B the spacing of a typical beginner, at C the results with an auto-bug keyer

The auto-bug features are:

- Simple to build.
- Four operational modes — iambic, auto-bug, ultimatic and reciprocal.
- Controls for SPEED and MODE are on the front panel.
- Four non-volatile message memories that can be used in all operational modes.
- Low power consumption — two AA batteries will last for months.

Circuit Description and Assembly

The circuit diagram is shown in Figure 2. For ease of construction as well as power conservation, a PIC16F819 microprocessor is used. A serial electrically erasable-programmable read only memory (EEPROM) 24LC64 is used to provide large capacity for multimode message memories.

The circuit is fairly simple. The programmable integrated circuit (PIC) micro processor and EEPROM are connected by just two wires. The output device is a 2N7000 field effect transistor (FET). This is a good device for this application since it offers low current consumption and high noise margin. It does not, however, tolerate the high voltage keying potential found in most tube type transmitters. In that case an additional driver stage or keying relay must be used.

To minimize the likelihood of damage, I suggest the use of sockets for the PIC and the other integrated circuits (IC)s.

The three FB-101 ferrite beads shown are included to prevent RF interference from the transceiver. I installed these because I experienced interference once while the project was under development. While they may not be needed in all applications, they do provide inexpensive insurance against a frequently encountered problem.

You may use other ferrite beads, but not larger ones. Energy accumulated in the beads can cause damage to the active devices. Install the PIC and ICs as the last assembly step to

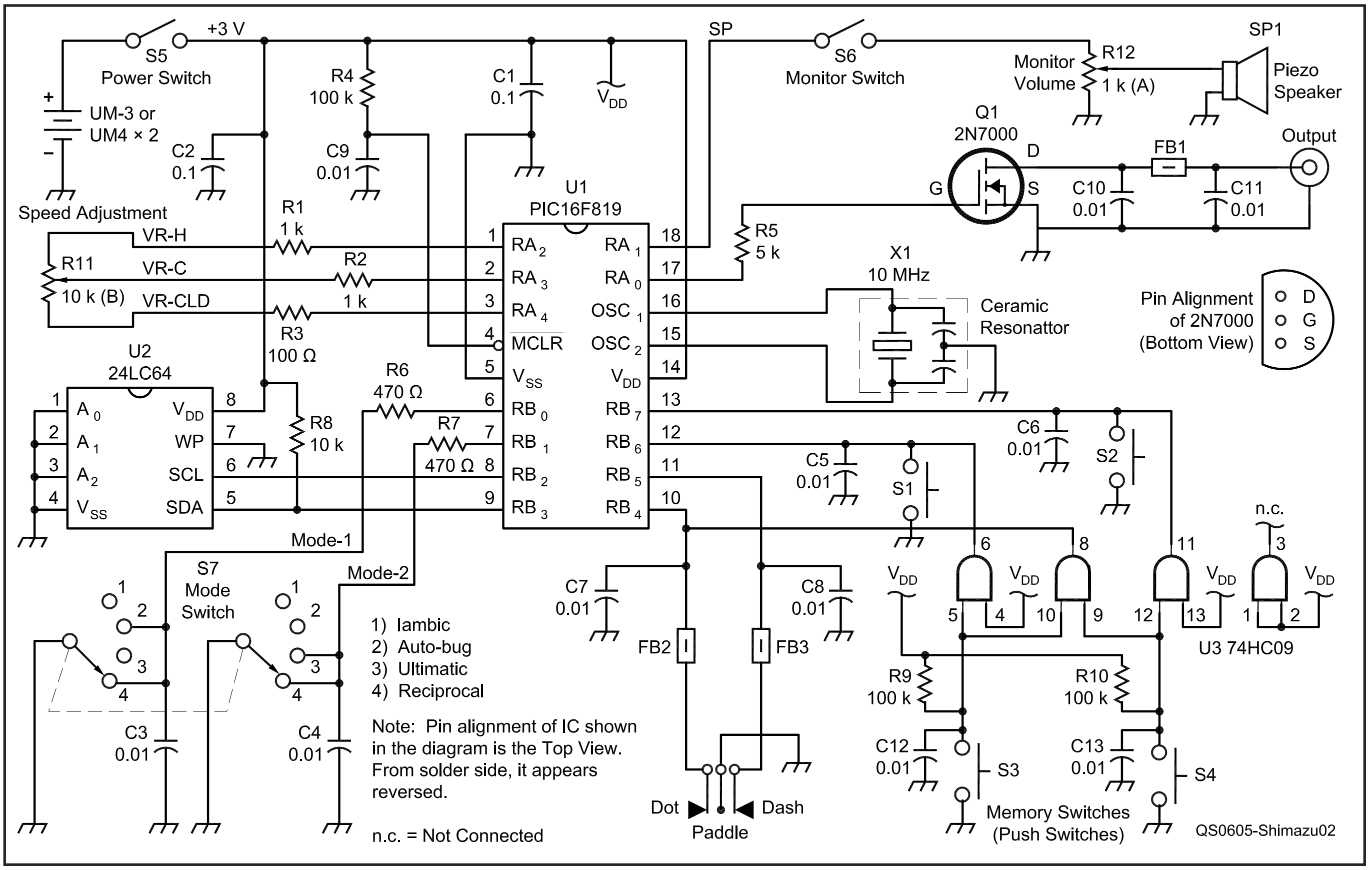

Figure 2 — The schematic diagram of the auto-bug keyer.

C1, C2 — 0.1 µF, 25 V ceramic capacitor.
C3-C13 — 0.01 µF, 25 V ceramic capacitor.
FB1-3 — FB-101 ferrite bead.
Q1 — 2N7000 FET.
R1, R2 — 1 kΩ, ¼ W carbon film resistor.
R3 — 100 Ω, ¼ W carbon film resistor.
R4 — 100 kΩ, ¼ W carbon film resistor.
R5 — 5 kΩ, ¼ W carbon film resistor.
R6, R7 — 470 Ω, ¼ W carbon film resistor.
R8 — 10 kΩ, ¼ W carbon film resistor.
R9, R10 — 100 kΩ, ¼ W carbon film resistor.
R11 — 10 kΩ, ¼ W linear potentiometer.
R12 — 1 kΩ, ¼ W audio taper potentiometer.
S1-S4 — Momentary contact SPST push button.
S5, S6 —SPST toggle switch.
S7 —2P4T rotary switch.
SP1 — Miniature piezoelectric loudspeaker.
U1 — PIC16F819 programmable integrated circuit.
U2 — 24LC64 serial electrically erasable-programmable read only memory (EEPROM).
U3 — 74HC09 quad AND-gate.
X1—10 MHz ceramic resonator.

avoid any damage from static electricity.

The printed circuit board (PCB) pattern (print the pattern, adjusting the size to 55 mm × 55 mm for photo etching) and the top view from the parts side of PCB are available on the ARRLWeb.[1] See Figure 3 for an interior view of the completed unit.

Using the Auto-bug Keyer

Connect your paddle and turn S5 and S6 on. Set R11 and R12 to the center position. First try iambic keying with S7 set to position 1 and check the speed variation by turning R11. Set the monitor volume to your preference with R12.

Next, check the *ultimatic* mode with S7 set at position 3. When the paddle is squeezed, the last paddle pushed will overtake the former one. This mode effectively converts a double lever paddle to the action of a single lever paddle.

Now turn S7 to position 2. The keyer is now set to auto-bug mode. First, push the dash paddle and check that the output stays on while you have it closed. Next, push the dot paddle. Dots should continue to be sent as long as the key is pressed. Now manipulate the paddle and send characters mixing dashes and dots, K, C, X, etc. Does it feel like a bug? Next, try characters with repeated dashes, Y, M, O, 1, etc. You have to push the dash lever repeatedly to get the number of dashes you want. You should find that the minimum allowed spaces between code elements are appropriate and that the signals sound, in a way, "natural." That is the result of auto-bug's design. When both paddle levers are pushed, the last one will dominate, as in the ultimatic mode.

The last test is the reciprocal key mode. Turn S7 to position 4. You should find that the functions of dash lever and dot lever are the same with this mode. Spacing is flexible with minimum spacing provided, just as in auto-bug mode.

Using the Message Memory

To enable the message memory, push any channel of the memory switch for about 2 seconds. The Morse characters R and $\overline{\text{BT}}$ will be heard from the monitor. The speed of R is fixed and that of $\overline{\text{BT}}$ is set by R1. Release the switch before $\overline{\text{BT}}$ finishes. This operation erases any previously recorded message.

Now, whatever signal you send will be recorded. The operation and the recording mode will be as set by S7 and will be maintained until you finish the recording. The recording starts as soon as you start sending. Any time lag between $\overline{\text{BT}}$ and the start of paddle operation will be eliminated from the recording.

To finish recording, push the memory switch briefly. $\overline{\text{AR}}$ will be heard from the monitor and the keyer will return to normal operational mode. The time lag between the

[1]**www.arrl.org/files/qst-binaries/Shimazu0506.zip**.

end of the message and your pushing the memory switch will also be eliminated from the recording. If the memory becomes full while recording, the keyer automatically signals with an $\overline{\text{AR}}$ and stops recording. I have never experienced this because the memory capacity is at least 500 characters for each memory.

To play back and send the message you recorded, briefly push the memory switch for the corresponding memory channel. The current operational mode is ignored and the recorded message is transmitted and sent to the monitor. To stop memory playback before completion of the message, just push the dash paddle.

Repeat Mode

Turn power switch S5 on. The monitor sends R, and the keyer enters into repeat mode. Repeat mode operates normally during regular keyer transmission; however, during playback, the message will be repeated until you stop it by pushing the dash paddle. If you wish to send a message repeatedly, the corresponding memory must be recorded in repeat mode. When you record messages in repeat mode, the space after the end of the message will not be ignored. It will be recorded until you push the memory switch. The length of this space may be up to the equivalent of 40 dots. To disable repeat mode, switch the keyer off and turn it on normally.

I regularly used an early type of auto-bug keyer while I operated 9V1DJ in Singapore. Sometimes I was asked what kind of bug I was using. The operators on the other end did not suppose they were hearing the sound of an auto-bug keyer!

The object file to program the processor, PIC16F819 will be provided by e-mail, for your personal use. The programmed PIC processor is also available. For details, send an e-mail to **shimazu@m7.dion.ne.jp**.

Figure 3 — The inside of the completed auto-bug keyer. Note the PIC microprocessor.

Taka K. Shimazu received his first Amateur Radio license in 1965 and upgraded to the (Japanese) First-class license in 1975. While working for his employer in the US in 1986 he held the call KE6QO. Business travel took him to Singapore from 2000 to 2003 where he operated as 9V1DJ. He enjoys building his own transceivers, transverters, linear amplifiers and peripheral equipment including keyers. Some of his homebrew equipment still uses tubes. The auto-bug is the 19th keyer he has built since 1968.

Taka's professional life is as an environmental engineer. Sometimes he has had the opportunity to work in overseas assignments on projects such as air pollution control facilities and wastewater treatment plants. Amateur Radio and local hams have always made the stay more enjoyable. You may contact Taka at 10-36 Kizuri 3-chome, Higashiosaka, Osaka 577-0827, Japan, or at **shimazu@m7.dion.ne.jp**. *Check out his Web page at* **www.h5.dion.ne.jp/~shimazu**.

New Products

RF CURRENT METERS FROM MFJ

◊MFJ has announced a new line of RF current meters that can be used for adjusting transmitters, amplifiers and antenna tuners, determining feed point impedance, and troubleshooting or comparing antennas and tuners.

The MFJ-834 coaxial RF ammeter uses SO-239 connectors and has three calibrated ranges — 0.3, 1 and 3 A. It's rated for 1.8-30 MHz. A higher power version, the MFJ-834H, has 3, 10 and 30 A ranges.

For open wire or ladder line, try the MFJ-835 balanced line RF ammeter. This version uses a cross-needle meter and displays the current on each side of the feed line. When currents are in balance, the dual needles cross in the center. The MFJ-835 reads 0.3, 1 and 3 A ranges.

Add an SWR/wattmeter to the MFJ-834 and you have the MFJ-836. This version displays RF current (0.3/1/3 A or 3/10/30 A for the "H" version), plus forward power (300/3000 W), reflected power (60/600 W) and SWR on a separate cross-needle meter. Power measurements can be average or peak, and the unit uses SO-239 connectors.

All units require a 12 V dc supply. Price: MFJ-834, $69.95; MFJ-834H, $79.95; MFJ-835, $119.95; MFJ-836, $129.95; MFJ-839H, $139.95. To order or for your nearest dealer, call 800-647-1800 or see **www.mfjenterprises.com**.