

A Simple and Inexpensive Keyer for Morse Keyboard Use

This modest keyer uses just four components, yet includes features normally found in more sophisticated keyers.

Ray Kauffman, AJ4YN

This keyer consists of a Raspberry Pi Pico, a 330 Ω resistor, a solid-state relay, and a connector for the transceiver's key line. The Raspberry Pi Pico is an inexpensive microcomputer with some very useful hardware peripherals built in. It can generate frequencies from audio through HF, operate relays, measure voltage, and much more — all on a 1 \times 2-inch printed circuit board. And the software tools for programming and code development are free.

The total cost to build the keyer is less than \$20, but you'll also need a laptop and a USB type B cable. The macro creation, code speed, and CW transmission are controlled from your laptop keyboard, and the keyer is powered by USB. Its features include 10 macros for contest or casual contacts, a type-ahead buffer for CW transmissions, a code practice oscillator that can drive earphones or a speaker, a keyed carrier on 7020 kHz for receiver monitoring, and an opto-isolated keying relay. The parts list is shown in Table 1, and the schematic is illustrated in Figure 1.

The simplest version of the keyer is shown in Figure 2. The solid-state relay is mounted and superglued to the Raspberry Pico, dead-bug style. A solderless breadboard version with a speaker and antenna is shown in Figure 3.

Table 1 — Parts List

Quantity	Description	Source
1	*Seeed Studio Raspberry Pi Pico	Amazon
1	330 Ω ¼ W resistor	Amazon, DigiKey
1	Opto-isolated solid-state relay TLP592J	DigiKey
1	Piezoelectric speaker Murata PKM22EPPH2001-B0	Digikey
1	Phone plug to match your radio's keying jack	Amazon, DigiKey

*Purchase with headers if you use a breadboard. No headers are used for the simpler version of the keyer.

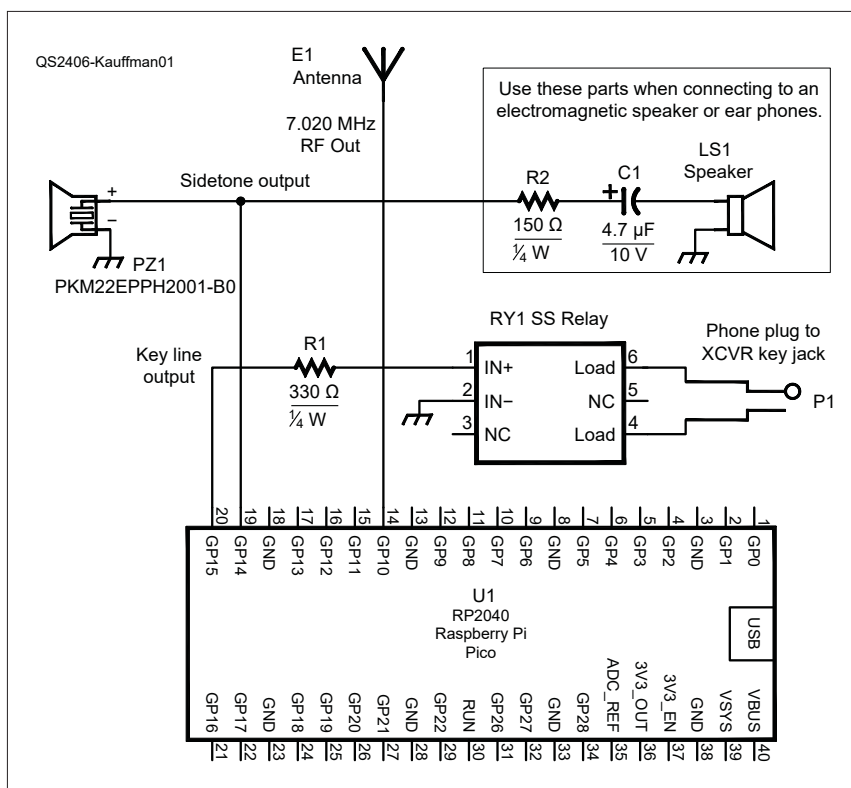


Figure 1 — A schematic of the simple keyer.

THIS SHOULD BE A LEFT



Figure 2 — The simplest version of the keyer.

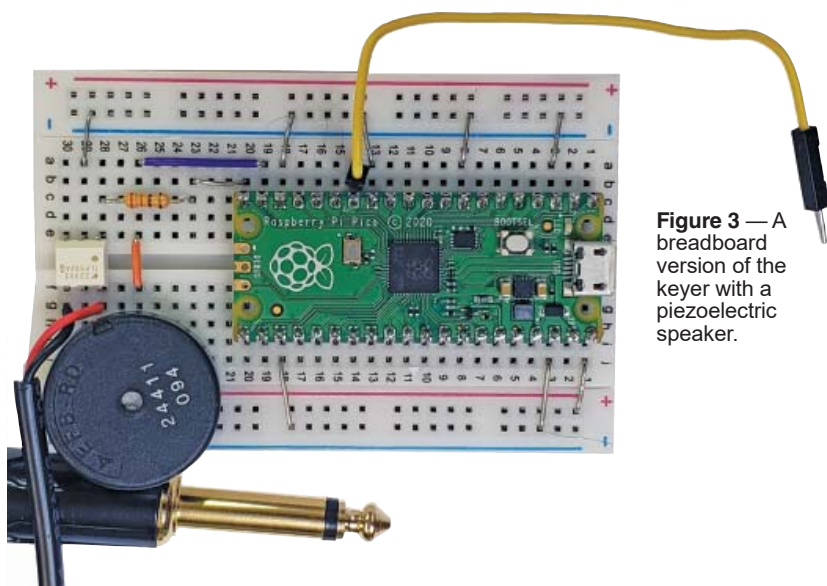


Figure 3 — A breadboard version of the keyer with a piezoelectric speaker.

I packaged the simplest version of the keyer in a $\frac{3}{4}$ -inch section of PVC pipe that was squeezed into a vise, making a slightly elliptical shape (see Figures 4 and 5). I center-drilled the end cap for the key line and passed it through the hole. I slit the opposite end cap so that the USB cable could slide into the slit.

The Pico also transmits CW on 7020 kHz. This can be useful for adjusting a receiver's automatic gain control, filters, and noise blanker for optimum CW reception. The power level is about 1 mW, and a short wire will radiate enough signal to be heard. A small piezoelectric speaker can be added to produce a sidetone for use as a code practice oscillator. The sidetone's 3.3 V square wave output is capable of providing about 25 mA of drive current. If you use an earphone or speaker for sidetone monitoring, place a 150 Ω resistor and a 4.7 μ F 10 V capacitor in series with either to prevent damaging the Pico.

Programming the Pico

A PDF of *Get Started with MicroPython on Raspberry Pi Pico* is available at no cost from <https://hackspace.raspberrypi.com/books/micropython-pico>. This is an excellent book for those who are new to the Pico. You won't need to read the whole book, as the first few chapters provide enough information to get started. The keyer is controlled using the free *Tera Term* terminal program (<https://sourceforge.net/projects/tera-term>)



Figure 4 — The keyer installed in a PVC tube.



Figure 5 — The fully packaged keyer.

that should be installed on your computer. More details on the actual programming of the keyer, the *Tera Term* installation and setup, and the program listings are available on the QST in Depth web page (www.arrl.org/qst-in-depth).

Operating the CW Keyer

The keyer program is not case sensitive. Ten messages can be stored and recalled, and the code speed can be set from 5 to 50 WPM. Any-

thing typed on the keyboard is instantly sent at the selected code speed with full buffering for rapid-touch typists. For code practice, an ASCII file can be transmitted with practically no limit on the length. If you're learning to receive CW, you can write the characters you want to practice learning in a text file using Windows *Notepad*. To send the text file, just drag and drop it to the *Tera Term* terminal window and it will be converted into Morse code (the keyer must be in CW mode when you do this). An example of the keyer operation is detailed on the *QST* in Depth web page.


This is a fun project that not only results in a simple, inexpensive, and quite functional keyer, but it will also give you experience working with the Raspberry Pi Pico processor.

See *QST* in Depth for More!

Visit www.arrl.org/qst-in-depth for the following supplementary materials and updates:

- ✓ More details on how to program the keyer
- ✓ *Tera Term* installation and setup
- ✓ An operation example of the keyer
- ✓ The program listings

???

 In the digital edition of *QST* (www.arrl.org/qst), we take you through a build of this code practice oscillator. See how easy it can be to complete this on your own workbench!

All photos provided by the author.

Ray Kauffman, AJ4YN, was first licensed in 2010, after many years as an avid SWL operator. He has worked in the electronics field since the early 1960s. In 1978, Ray started a small company that manufactured test equipment for the marine electronics industry. He holds several patents, as well as an FCC First Class Radiotelephone Operator's Certificate with Radar Endorsement. Ray sold his business in 2011 and has been retired and enjoying ham radio ever since. He can be reached at rayk2@cox.net.

For updates to this article, see the *QST* Feedback page at www.arrl.org/feedback.

VOTE

If you enjoyed this article, cast your vote at www.arrl.org/cover-plaque-poll

This Month in **QEX**

QEX magazine is a forum for the free exchange of ideas among communications experimenters. All ARRL members can access the digital edition of *QEX* as a member benefit (www.arrl.org/magazines). Print subscriptions are available and sold separately; see www.arrl.org/qex.

Coming up in the May/June issue of *QEX*:

- David J. Ahlgren, K1BUK, elucidates the possibility of using standard-impedance cables to make stepped line transformers that match a wide range of impedances.
- Anthony Le Cren, F4GOH/KF4GOH, describes a method of using inexpensive modules to send SSTV images from a sounding balloon.
- Gwyn Griffiths, G3ZIL, details how spectral spread at the transmitter and receiver can be less than 30 MHz, using modestly priced equipment.

- Jacques Audet, VE2AZX, simulates and tests popular 49:1 unun designs and suggests how they may be improved.
- In his essay series, Eric P. Nichols, KL7AJ, discusses how the behavior of hardware demonstrates the truth of the mathematical concepts hams deal with.

QEX is edited by Kazimierz "Kai" Siwiak, KE4PT (ksiwiake@arrrl.org), and is published bimonthly.

Would you like to write for *QEX*? We pay \$50 per published page for full articles and *QEX* Technical Notes. Get more information and an Author Guide at www.arrl.org/qex-author-guide.