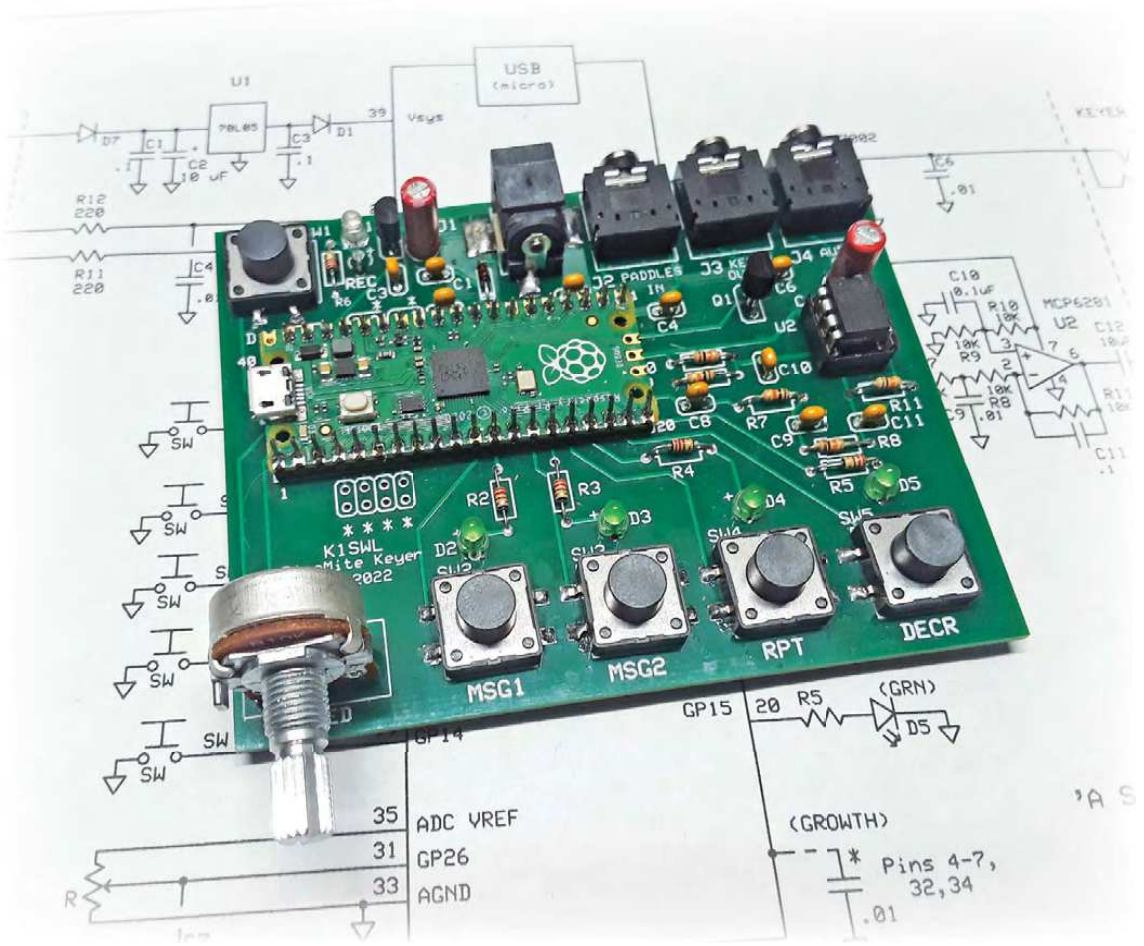
A Simple Morse Memory Keyer

Build this standalone, customized keyer for use during your contest operations.



Dave Benson, K1SWL

My modern radio includes a built-in keyer, but it requires an inconvenient number of button pushes to operate. I needed a standalone memory keyer that was dedicated to my contest operations. I based the design on the Raspberry Pi Pico, a tiny single-board computer with interesting features. Software for this device includes an *MMBasic* interpreter with plenty of memory space. With the interpreter loaded, it's referred to as the PicoMite. Some testing in *MMBasic* convinced me that it can generate CW characters at 40 WPM, which was indeed fast enough.

A Simple Design

I made a small printed circuit board (see the lead image) with tactile pushbuttons, LED indicators, and input/output provisions. I opted for four function buttons, which is a practical minimum for memory keyers.

The Pico Memory Keyer (PMK) schematic is shown in Figure 1. Five tactile switches control its operation — four initiate the message functions, and one starts the recording process in standalone mode. Each of the five inputs has a corresponding LED indicator.

The PMK tactile switches each have a dedicated purpose: (one) send call sign, (two) send contest exchange, (three) repeat contest exchange, (four) decrement exchange serial number, and (five) record.

Most of the good stuff is in function two. It's programmable by several methods. For many different contests, operators send a sequential serial number, and my design includes that feature. Function three repeats the exchange you just sent. This is needed when a CQing station sends "?" or "AGN" (again). If they never get the complete exchange and send "SRI" (sorry) or go back to CQing, it's a busted contact.

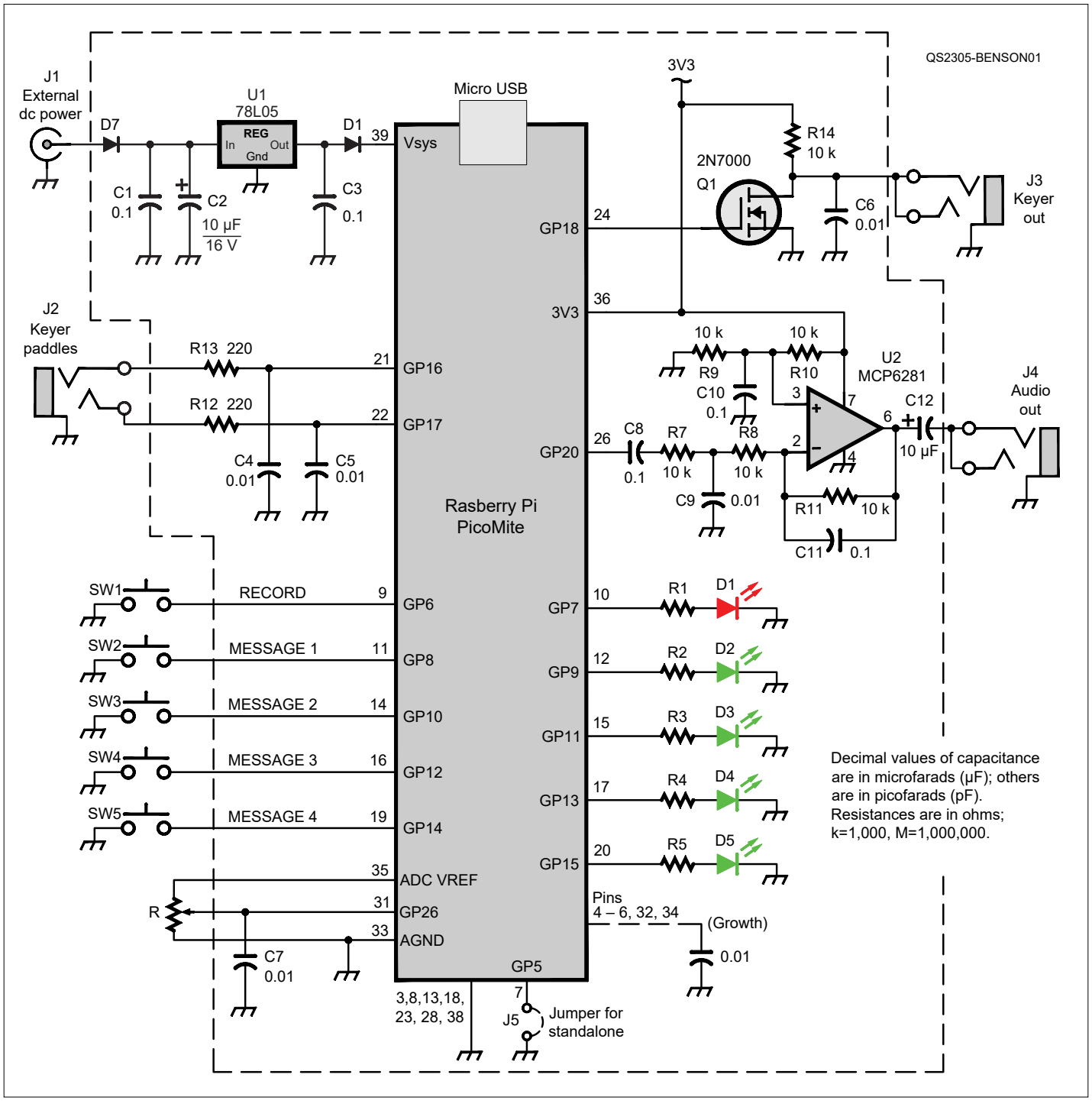


Figure 1 — Schematic diagram of the Pico Memory Keyer.

Function four rolls back the contact number. Function five enables you to record keyer paddle input for filling the message buffers. This function is useful in standalone mode.

There is some analog circuitry around operational amplifier U2 that takes a pulse-width modulation tone from the PicoMite and buffers and filters it into a nice-sounding sine wave for listening with earbuds. This is most useful during software development and usually isn't needed for on-the-air use, as your radio typically provides its own sidetone. The audio output also doubles as an off-air code practice oscillator. The PMK also has five spare breakout pins. Two of these can be used as analog inputs read by the PicoMite's analog-to-digital conversion feature. This could allow it to be used in monitoring battery voltage and SWR, for instance. These breakout signal paths can be taken off the board by plugging Adafruit jumpers onto the PicoMite header pins. There's room on the PMK board to add bypass capacitors to ground, if needed, to minimize RF pickup issues.

PicoMite Features at a Glance

- ✓ Comprehensive BASIC interpreter: floating point, 64-bit integer, strings, multidimensional arrays
- ✓ 133 MHz clock frequency
- ✓ Compiled C can be embedded
- ✓ Fast Fourier Transform and inverse Fast Fourier Transform included
- ✓ Memory: 7 blocks 124 K flash, 156 K RAM
- ✓ Extensive serial and parallel LCD support, including resistive-touch SPI/I2C/RS-232 support
- ✓ XMODEM protocol for program transfers and archiving
- ✓ 26 lines of general-purpose input/output (I/O)
- ✓ Three I/Os provide fast 12-bit A/D conversions
- ✓ Raspberry Pi Pico cost: $5

R12 and R13 were added in series with the paddle inputs to reduce the slew rate and eliminate the overshoot behavior, which could cause the Pico to crash. If your application includes bringing remote high-speed signals to the Pico, those resistors need to be included.

I developed software (available on the **www.arrl.org/qst-in-depth** web page) using the PicoMite's resident BASIC interpreter and a micro USB adapter cable. At the other end of the cable is a computer and a readily available (and free) terminal emulator program called *Tera Term* (**www.teraterm.org**). That environment allows editing and execution of programs without the need to compile code after each minor change. Details of the software installation process for the PMK can be found on the *QST* in Depth web page.

Two Operating Modes

The PMK has two distinct operating modes: terminal and standalone. In terminal mode, you rely on the virtual terminal running on your computer for setup and message entry. Upon connecting the USB cable, the PMK software comes alive with a user dialog. It prompts you to enter two messages and select three other features: reset serial number, paddle reversal, or iambic mode A/B. You can return to it at any time to make changes to the operational settings or customize the program for your own needs.

In standalone mode, only the keyer paddles are needed to initialize the PMK functions. Pressing the record button arms the programming mode for entering a new message. At the end of the message, press one of the two message pushbuttons to store it in your desired location, and the record LED will turn off. Your newly stored message will then be played back to you. This sequence is shorter because you won't be changing the keyer settings often, if at all. The program automatically starts in standalone mode upon application of power if a jumper has been installed at J5.

The last serial number is retained if the PMK is not reset. This allows the keyer to recover gracefully in the event of a power interruption. The other operational settings — paddle reverse and iambic A/B — are stored in non-volatile memory and are restored upon application of power. In both operating modes, a quick tap of either keyer paddle aborts an outgoing message if you've pressed the wrong message button. The keyer functions are available at all times.

Final Comments

This keyer satisfies my need for hunt-and-pounce operation in a contest. I'm also using it on a weekly basis for several sprint activities. If your Morse skills are rusty, or if you're new to operating CW, a memory keyer is a great option. You're able to put your best foot forward on the air, without the jitters that can arise from being relatively unfamiliar with Morse operation.

I'm grateful to George Heron, N2APB, for suggesting the PicoMite and for his steadfast support of this project. He's working on his own ham applications using this board, and we should expect to hear more about them in the future.

A printed circuit board for this project is available at minimal cost. Visit **https://midnightdesignsolutions.com** for details.

See *QST* in Depth for More!

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

- ✓ Software program and installation
- ✓ Bill of materials
- ✓ Additional images
- ✓ Assembly and driver installation

Lead photo by the author.

Dave Benson, K1SWL, is a retired electrical engineer. He contributed to a number of displays programs in the airborne military environment in Honeywell's Phoenix and Albuquerque divisions. This work included graphics overlay on sensor videos, as well as MIL-STD-1553B terminal design. He also holds several US patents. Dave left the corporate world to pursue amateur radio full-time as proprietor of Small Wonder Labs. He has frequently contributed to *QST* and is now undertaking a variety of homebrew projects. Dave designed and built his home in New Hampshire, where he enjoys gardening and woodworking. He can be reached at **k1swl@arrl.net**.

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