ARRL Periodicals Archive – Search Results
A membership benefit of ARRL and the ARRL Technical Information Service

ARRL Members: You may print a copy for personal use. Any other use of the information requires permission (see Copyright/Reprint Notice below).

Need a higher quality reprint or scan? Some of the scans contained within the periodical archive were produced with older imaging technology. If you require a higher quality reprint or scan, please contact the ARRL Technical Information Service for assistance. Photocopies are $3 for ARRL members, $5 for nonmembers. For members, TIS can send the photocopies immediately and include an invoice. Nonmembers must prepay. Details are available at www.arrl.org/tis or email photocopy@arrl.org.

QST on CD-ROM: Annual CD-ROMs are available for recent publication years. For details and ordering information, visit www.arrl.org/qst.

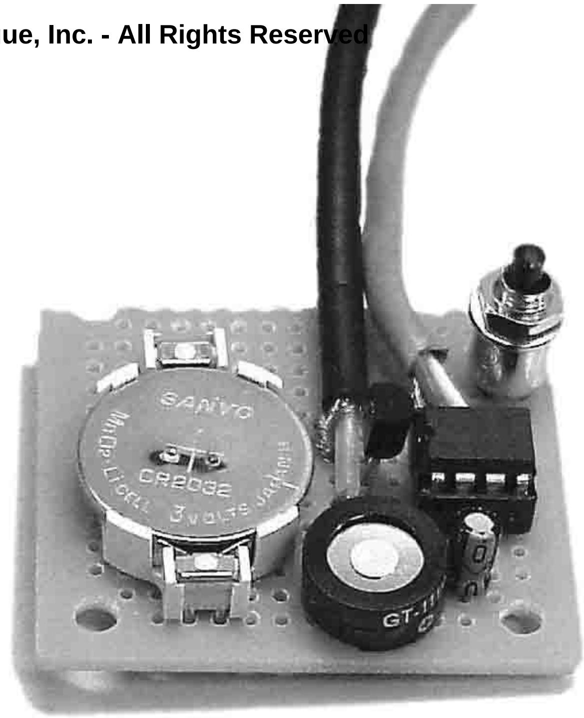
Non-Members: Get access to the ARRL Periodicals Archive when you join ARRL today at www.arrl.org/join. For a complete list of membership benefits, visit www.arrl.org/benefits.

QST Issue: Dec 2003
Title: The PicoKeyer--An Ultra Low Power CW Memory Keyer
Author: Dale L. Botkin, N0XAS

Click Here to Report a Problem with this File

By Dale Botkin, NØXAS

The PicoKeyer— An Ultra Low Power CW Memory Keyer



NØXAS redefines "low power" for us with a keyer that draws 4 nA (that's 0.004 microamperes!) at idle.

For most of the time I have been active on the HF bands, I have used various electronic keyers with a Vibroplex paddle. While I have spent time operating with a straight key, I have never really enjoyed it or gotten good at it. I have used built-in keyers in several brands of gear, including a few from well-known low-power kit manufacturers. All have worked well…some better than others.

I recently had the opportunity to upgrade my home HF station to a Kenwood TS-930S/AT. While I enjoy the rig, it lacks an internal CW keyer—so it was back to the straight key. I felt sorry, frankly, for some of those operators on 40 meters who were subjected to my fist during those first few weeks. I wasn't getting better, and I decided it was time to go back to the paddle and the keyer. Being a hacker at heart, however, I wasn't going to go out and buy what I could build in an evening! I decided I would build a very simple, barebones keyer to do nothing more than generate dots and dashes with proper timing and spacing. Like many other projects, this was a good idea until it got out of hand.

I knew I had all the parts I would need. One of my hobbies involves building projects with embedded controllers, mostly using Microchip's PIC processors.[1] For me, the perfect project is one that involves an absolute minimum of parts. First of all, I like to be able to build without ordering anything, and my "junk box" is pretty well stocked, but it doesn't have everything. Ideally, I like to use a small microcontroller with as close to zero external components as can be achieved. I am also a big fan of designing for as little current consumption as possible, mainly because I don't like changing (or buying) batteries and because power supplies are always a pain to build. I guess I've gotten lazy in my old age, but it does make for some interesting design challenges!

The Design

I wanted to use nothing more than one chip for the keyer, with a single pushbutton for any features that got added along the way. I also wanted to play with one of the latest 8-pin designs from Microchip, the 12F629. These little processors cram a lot of features into a very small package! 1024 words of program memory, 64 bytes of RAM, 128 bytes of nonvolatile EEPROM, interrupts, counter/timers, an internal oscillator, an analog comparator, and more. The 12F675 shares the 12F629's features and adds a 10 bit, four channel analog to digital converter. These chips are a major leap ahead compared to the older 8 pin 12C5XX-type devices and their power requirements show it as well. The new pair will operate down to 2.0 V, and their SLEEP mode lets them power down and run on not much more than a cool breeze.

The hardware design was the simple part, requiring nothing more than connections from the key to the PIC and a single transistor to key the rig. I added the SPST pushbutton switch and decided to use one pin for an LED during debugging, leaving one pin unused. I assembled the project on a solderless breadboard and wired up the paddle. Now came the tough part—the software—the heart of the project.

One of the first steps was to determine the timing to be used when sending Morse code. An hour or so of reading various pages found with a simple Internet search and I had the "magic numbers" needed to determine the dot and dash timing for any given code speed. A couple of hours of programming later and the chip was generating properly spaced code from 5 WPM to as fast as I could send, with the upper limit set at 60 WPM. The next step was to add a setup mode for changing the code speed. While I was at it, I added functions for setting the weight and selecting Mode A or Mode B timing.

When I was reasonably satisfied with the performance, it was time to move from the solderless breadboard to a "real" setup and try it out on the air. I used half of a small RadioShack perfboard and wired it up as a prototype assembly. On-air results were good, but there were a few key things still missing.

The project took on a life of its own

[1]Notes appear on page 40.

over the next couple of weeks. The LED was replaced with program-generated audio sidetone that could be turned on and off, and I managed to find a tiny speaker with low enough current requirements to drive directly from one of the PIC pins. I added message memory, beacon mode with variable repeat delay, then variable pitch for the sidetone, then paddle reversing for left-handed ops or miswired paddles. A few bugs were uncovered and fixed along the way, as I logged more operating hours with the keyer.

Redefining "Low Power"

Along the way, I got curious about the current consumption of the chip. I had the PIC in its low power sleep mode any time it was not actively doing something like keying the rig or generating a sidetone. Although it has now been updated, at the time, the data sheet for the 12F629 was pretty sketchy about sleep current. I knew it would be low, but I needed to determine whether the CR2032 PC motherboard clock lithium cell I was using would last long enough to be practical as I salvaged the battery from a defective old PC motherboard I was scrapping. While it was still reading a healthy 3.1 V, I was planning to switch to a couple of AAA alkaline cells if needed. After connecting my Fluke 77A meter in series with the battery I was surprised to see it reading 0.00 mA while the chip was idle. Thinking the meter was simply not accurate at low currents, I tried my Micronta bench meter on the 3 µA range. I was quite surprised to see it read 0.004 µA or 4 nA while sleeping! I abandoned the thought of alkaline cells—that little lithium battery would last quite a while! I've been using that same salvaged cell for a few months now, and it still reads a tad over 3 V when the keyer is idle.

Construction

I built my keyer on a small piece of RadioShack perfboard. The few connections required are pretty simple and straightforward. Power to the PicoKeyer chip, U1, is supplied on pin 1, while pin 8 is ground. The photo on the previous page shows the completed keyer as it appears before packaging. Figure 1 is a view of the finished product in a mini plastic case. Its size relative to a standard 1/4 inch phone plug reveals how small this keyer really is—and that's complete with power supply and speaker! The complete schematic and parts list is shown in Figure 2.

Pin 2 is the sidetone output. If you are building a standalone keyer you will probably want a small piezo or low current magnetic speaker connected to the chip through a capacitor. I used a small magnetic speaker with a drive requirement of only 15 mA. Other piezo or magnetic speakers can be used—but be careful not to exceed the 20 mA current capacity of the PIC output pin. If you are building the keyer into a rig, you can simply inject the sidetone signal into the radio's audio stages. A volume control might be a good idea. If you need more audio than the PIC can provide directly, a small audio amp like an LM386 could be used for more punch. This would of course require a more robust power source. The signal produced is a square wave; some filtering can be used to approximate a sine wave, if desired.

Q1 is a 2N7000 MOSFET, and is used to key the radio. You can use a 2N2222A or a similar NPN transistor with a current limiting resistor on the base or emitter. This works, but it requires more current. Since the MOSFET is a voltage, rather than current, operated device, it is ideal for super low power designs such

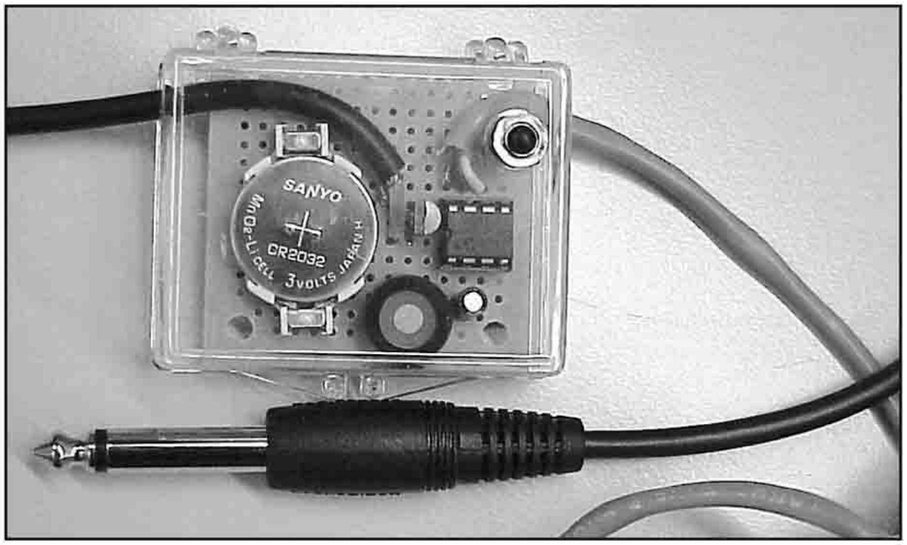


Figure 1—The completed PicoKeyer in a plastic case. A 1/4 inch phone plug is shown for comparison.

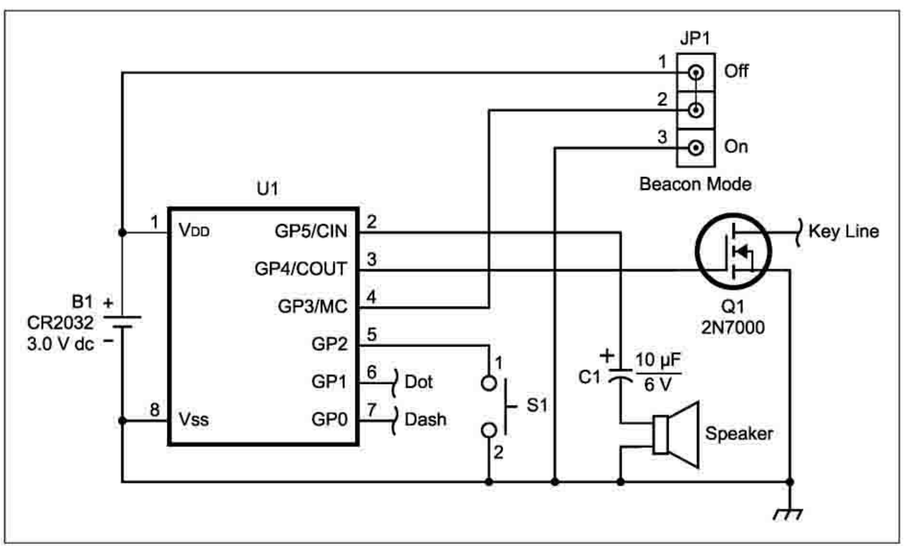


Figure 2—The schematic and parts list for the PicoKeyer. The battery is a PC clock motherboard cell, but any 3 V dc cell can be used.

B1—CR2032 lithium battery with holder or any 3-5 V dc cell (AA or AAA cells okay).
C1—10 µF, 6 V electrolytic capacitor.
JP1—Beacon mode switch or jumper (see text).
Q1—2N7000 MOSFET, Digi-Key 2N7000FS-ND (Digi-Key Corp, 701 Brooks Ave South, Thief River Falls, MN 56701; tel 800-344-4539; www.digikey.com).
S1—SPST pushbutton, momentary.
SPK—Soberton GT-111PS, Digi-Key 433-1023-ND (see text).
U1—Microchip PIC12F629 with PicoKeyer code. Available blank from Digi-Key (part no. PIC12F629-I/P-ND) or preprogrammed from the author.[2]

as this one. Q1's gate is connected to pin 3 of the keyer chip, with its source connected to ground. The drain lead is used to provide positive keying for the radio.

Pin 4 is used to select beacon mode. If you do not intend to use the beacon feature, you may simply connect pin 4 to the positive supply on pin 1. If you wish to be able to use beacon mode, install an SPDT switch or a jumper to connect pin 4 to either the positive supply or to ground.

The pushbutton switch is connected to pin 5 of the chip. Since the 12F629 has internal pull-up resistors, no external resistor is needed to keep the input from floating.

Pins 6 and 7 are the paddle inputs. When grounded (or driven low by other circuits), the keyer will produce a continuous stream of dits or dahs with proper spacing. If both inputs are grounded, alternating dits and dahs are sent.

Once all the connections are made, apply power to the keyer. You should hear the keyer send 73 in Morse code at 13 WPM via the sidetone only. This indicates that the chip is powered up and healthy. If you do not hear 73, recheck your wiring and reapply power. If everything is okay, you're all set to start using the keyer! The default settings are 13 WPM, normal weighting, message memory empty, Mode B timing and a beacon delay of five seconds.

If you have experience programming PIC processors, the HEX code is available at the ARRL Web site: **www.arrl.org/files/qst-binaries/picokeyer.zip**. A programmed version of the processor with the latest revisions is available from the author.[2]

Operation

Operating the PicoKeyer is simple and straightforward. To use it as a normal iambic keyer, simply apply power and connect the paddles. Dual or single lever paddles can be used. To send the contents of the message memory, press and quickly release the pushbutton once. While the message is being sent, any paddle or pushbutton input will immediately end the transmission.

To enter setup mode, press and hold the pushbutton. The keyer will step through the setup menu choices, with about a one second delay in between. When you hear the character for the item you want to check or change, release the button immediately. You may then verify or change that item. A short press of the button will exit setup mode, or you may press and hold the button to keep cycling through the choices. When you exit the menu the keyer will end with the Morse prosign SK.

Menu Selections

The menu selections are as follows:

U (tUne/straight key mode): In this mode, either paddle input will act as a straight key. This is useful for sending a steady carrier for tuning.

S (Speed): Use the paddles to raise or lower the speed. After each paddle hit the keyer will send the new setting.

T (Tone): Turns the sidetone output ON (Y), OFF (N), or sets RIG (R) mode. In either the ON or OFF settings, the rig is not keyed when in setup mode. RIG mode is useful if you wish the transmitter output to be active even in setup mode.

M (Message): Hitting either paddle will play the current message, followed by the Morse prosign AR. To record a new message, hit either paddle again. The keyer sends K and waits for your input. Enter the message, making sure to exaggerate word spacing. When you're finished sending the message, press and release the pushbutton, and the keyer will send R to confirm. You may then replay the message by hitting either paddle. It may take a couple of tries to get the hang of the timing, but it's not too picky unless you send the characters too slowly.

W (Weight): Adjusts the weight. Adjustment range is from 0 (50% "light") to 9 (50% "heavy"), with 5 being the normal weight setting.

C (Curtis mode): Selects Mode A or Mode B timing. This determines the behavior of the keyer when the paddles are released after a "squeeze." In Mode A, the keyer will complete the element (dit or dah) currently being sent. In Mode B, the keyer will "remember" seeing the other paddle, and will send one last element. For example, say you are sending the letter C. In Mode A, you would not release the dit paddle until the start of the last dit. In Mode B you would need to release both paddles as soon as the second dah has started. The keyer will "remember" to send the last dit.

P (Paddle): Selects the dit paddle. Simply hit the paddle you wish to use for dits. Note that in the menu mode, the "normal" dit and dah paddle inputs are used regardless of whether the paddles have been reversed or not.

B (Beacon delay): Sets the delay between message transmissions while in beacon mode, from 0 to 99 seconds.

A (Audio tone): Sets the audio sidetone frequency. After each paddle hit, a slightly long dash is sent.

When you are finished altering the keyer settings, press and release the pushbutton. The keyer will save all settings to its internal nonvolatile EEPROM memory and send SK. At that point you are back in iambic keyer mode, ready to go!

I have been using this keyer for a few months now and several other hams have built them as well. One of the things I like is the ability to change features very quickly; it took about an hour of revising the program code to make a version pin compatible with the popular RockMite transceiver.[3] Another couple of hours and I had a version that used a potentiometer to set the speed instead of the menu; it was even less time to produce one to replace another 8-pin keyer chip that used a different pinout. While I still enjoy soldering, I also like this facet of our wonderfully diverse hobby. Besides, I can write code while in an airline seat—an environment in which a soldering iron is somewhat impractical!

Notes

[1]**www.microchip.com**.

[2]A pre-programmed version of the 12F629 PIC processor with the author's latest HEX code is available from the author for $8.95 (**www.hamgadgets.com**).

[3]D. Benson, K1SWL, "The RockMite—A Simple Transceiver for 40 or 20 Meters," *QST*, Apr 2003, pp 35-38.

Photos by the author.

Dale Botkin, NØXAS, was first licensed in 1981 as KA5MSS, although that call and HL9CA both lapsed without ever having been used. Dale was relicensed as a Technician in 1994, with subsequent upgrades to General and Amateur Extra. He enjoys casual operation using low power CW and PSK31 on the HF bands. He also operates VHF and UHF mobile and makes an occasional FM satellite contact. As is obvious, Dale is an avid radio experimenter, especially with embedded microcontrollers. You can contact him at 16624 Elm St, Omaha, NE 68130 or at **n0xas@arrl.net**.

STRAYS

I would like to get in touch with . . .

◊ amateurs who hold an elected office. —*Charlie Trice, K8IJ;* **k8ij@arrl.net**

◊ amateurs who became licensed after retirement.—*Bruce Atchison, VE6XTC;* **ve6xtc@telusplanet.net**

◊ anyone with a photocopy of a schematic for a Pace Communicator II synthesized 2 meter mobile radio.—*Ron Eckton, W7RE, PO Box 508, Maple Falls, WA 98266-0508;* **ecktonrj@wmconnect.com**

◊ members, former members and friends of the Texas A&M University Radio Club, W5AC, for information on a Club reunion.—*Glen Reid, K5FX;* **k5fx@arrl.net**