sending CW: a digital approach

A simple way to clean up your CW sending technique

International Morse code appears to be fairly simple on the surface. Many licensed Amateurs can testify to having mastered it with varying degrees of proficiency. When compared to modern digital codes, Morse code is really quite complex: two different on elements, dots and dashes, are used along with three different off elements to represent many letters, numbers, and special characters and symbols. Transmission speeds vary widely in a single transmission and dots, dashes, and spaces often become mashed in a stream of indecipherable information. This article details the use of digital circuits to accurately time and filter out-going Morse code sent with a straight key or other mechanical key so that dots and dashes are reproducible and well-timed. No new hand motions are required.

Someone who has listened to the CW ham bands for a short time will realize there are Amateurs who send code poorly. You may be unpleasantly surprised by your own CW-sending skills if you record one of your contacts and replay it. The main problem in sending good code is shown in **fig. 1**. A perfect transmission is shown at the top, and a typical on-the-air transmission is shown below. Note that real dots and dashes vary in length and the spacing is less than perfect. While the human brain is capable of adjusting for most of these changes, it is difficult to decipher a transmission in which dots and dashes are almost the same length and spaced in a semi-random fashion. Electronic paddle keyers, keyboards and computers are one answer, but they require mastery of new hand motions or skills and are frequently expensive. Many Amateurs would like to continue using their mechanical keys and improve their code-sending skills.

defining the circuit

The problem is to design a circuit that will accurately time dots, dashes, and spaces so they sound perfect on the air. Before the circuit can be designed, we need to specify what it is supposed to do. First we define the dot:dash:space ratio as being 1:3:1, the standard generally accepted for good Morse code transmissions. While many hams use weighting

By Jonathan Titus, KA4QVK, The Blacksburg Group, Inc., P.O. Box 242, Blacksburg, Virginia 24060

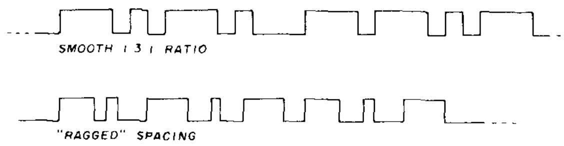

fig. 1. Perfect code versus real hand-sent code for CQ.

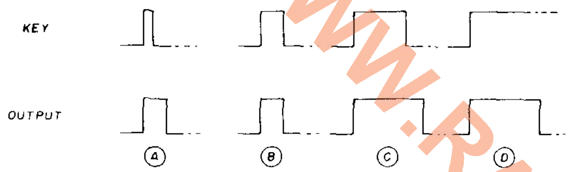

fig. 2. Dot and dash timing operations for the keyer circuit.

to vary the dot:dash ratio, it is better to use a well-known standard. Next, the dot/dash operation must be defined (shown in **fig. 2**). There are four actions:

A. If the key is closed for a very short time, a self-completing dot is generated.

B. If the key is closed and held closed up until the end of a dot, the same length dot is generated.

C. If the key is closed and held closed beyond the dot length, a self-completing dash is generated.

D. If the key is held closed for a long time, only a single dash is generated.

After each dot or dash, an off time of one dot-period is enforced to prevent code elements from being produced too tightly. If the key is activated during the off time, the action is remembered and acted upon as soon as the off time is finished. Perfectly-timed and spaced code can be generated by slightly leading the actions of the circuit.

using monostables

A simple timing circuit can be built using monostables, as shown in **fig. 3**. Monostable 1 is triggered by the key closure and generates a pulse one unit long. The negative-going edge of this dot pulse will trigger monostable 2 if the key is still pressed at this time. Monostable 2 generates a pulse that is two units long; when added to the length of the dot pulse, a dash pulse of three units is generated. The outputs of monostables 1 and 2 are gated to generate the Morse code output.

The output of monostable 2 is also used to disable (turn off) the key input to monostable 1 so it cannot be re-triggered during a dash. A third monostable is preset for one dot-time to generate the minimum required off period between dots and dashes. Monostable 1 is re-enabled so that the process may be repeated at the end of a dot or dash. While this circuit is useful in explaining the operations we would like, it does have some limitations: three monostables must be adjusted to change the speed of transmission, and key closures during the off period are not recognized. The circuit can be made to work but is impractical.

using a sequencer

A more reasonable approach uses a master clock

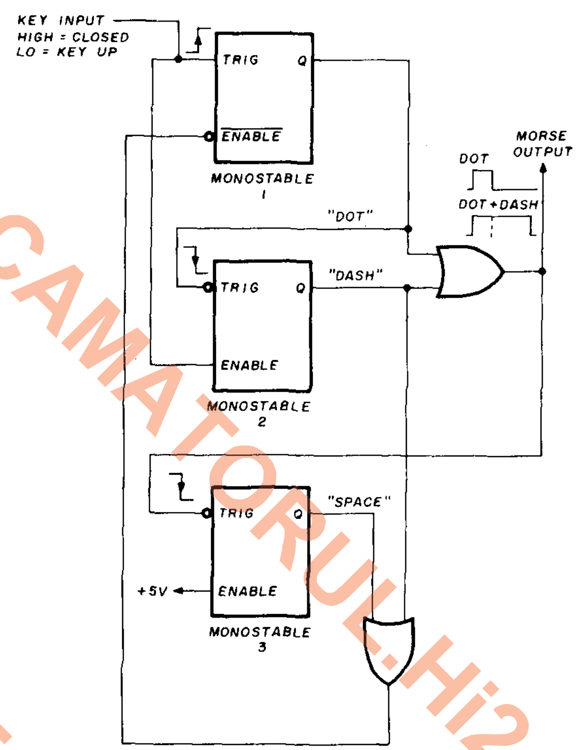

fig. 3. Using three monostables to filter hand-sent code.

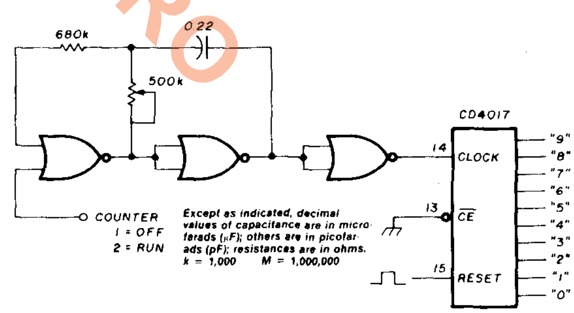

fig. 4. A counter/divider and gated clock used as a sequencer.

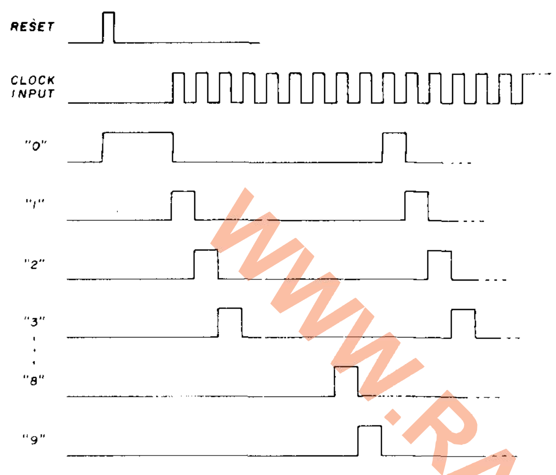

fig. 5. Timing diagram for a 10-stage sequencer circuit.

to sequence through a series of code-generating and condition-testing steps. Only the frequency of the clock need be adjusted to vary the code timing. The circuit described here was designed using digital complementary metal-oxide semiconductor (CMOS) integrated circuits to reduce power consumption.

As shown in **fig. 4**, the heart of the circuit is a CMOS CD4017 decimal counter/divider integrated circuit and a gated clock used as a sequencer. The CD4017 accepts input pulses at pin 14 and increments an internal count by one for each pulse. Only one of the ten outputs can be a logic one, indicating the state of the counter. Thus, ten external circuits can be turned on and off in a regular sequence, governed by the frequency of the clock signal applied to the CD4017. This is illustrated by the timing diagram in **fig. 5**. This sequencing circuit is used to generate the dot/dash/space times and to sequence various circuit elements that can test the key input for changing on and off conditions.

If the output of a free-running square-wave oscillator is controlled with a gate as shown in **fig. 6**, the first cycle output will be of arbitrary length. All subsequent cycles will be of equal length. This type of *gated waveform cannot be used for accurate timing*. The trick is to use an oscillator that is triggered or started by the enabling signal. The first clock cycle generated by a gated clock is always the same length as following cycles. When such a gated clock is used with the CD4017 sequencer circuit, the first sequence will be the same length as those that follow. The gated clock is simply a monostable that retriggers itself when gated on.

The Morse-generator portion of the circuit can be built from the sequencer and several CD4025 three-input NOR gates, shown in **fig. 7**. The dash input control line determines whether or not a dash is sent. If *DASH* = *1*, then a dot is sent. If *DASH* = *0*, the sequence is extended and the dot is stretched into a dash. A dot is always sent on sequence start.

generating a dot

The rest state of the sequencer is a logic one at the 0 output with logic zeros at all other outputs. This is the sequencer state when the gated clock is enabled. The first edge of the clock signal increments the internal sequencer count, moving the logic one to the 1 output. The logic one at the 1 output causes the lower NOR gate to output a logic zero. The sequencer moves the logic one to the 2 output on the next clock cycle and the output of the lower NOR gate goes back to the logic one state. This generates a logic-zero dot one clock cycle long. There is no other effect: any action through the upper NOR gates is blocked by the logic one on the dash control line. The time of the sequencer 2 output generates the off period (one clock cycle) at the end of each dot. Output 3 is the dot reset line (described in detail shortly). It resets the dot-generating sequence.

generating a dash

Let's see what happens when a logic zero on the dash control line allows the sequencer outputs to pass through the two upper NOR gates. The se-

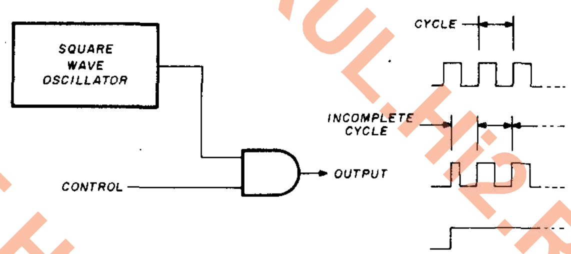

fig. 6. A simple gated free-running oscillator. Note that the first gated cycle may be incomplete.

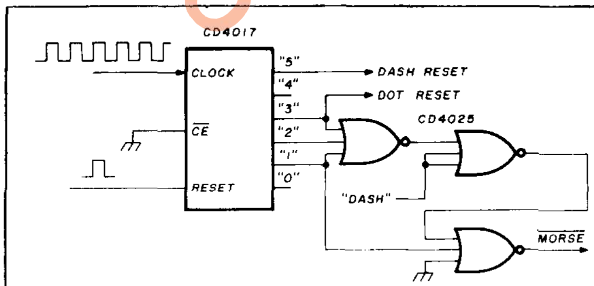

fig. 7. Using the sequencer circuit to generate perfectly timed dots and dashes.

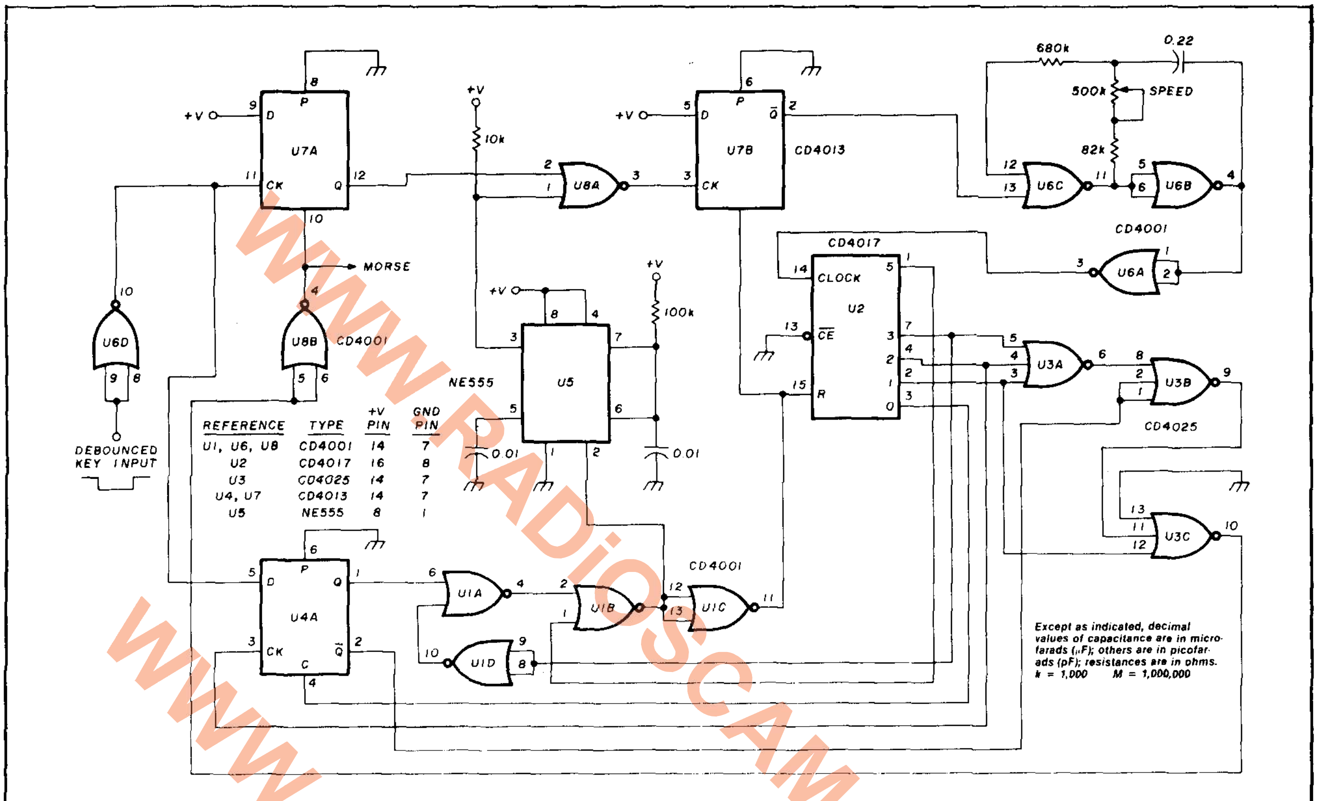

fig. 8. The complete keyer circuit.

quencer has been reset to 0 and the first clock pulse moves it to the 1 state. The output of the lower NOR gate goes to a logic zero once again. Since the two upper NOR gates are now enabled by the dash control line, the logic one outputs at 2 and 3 are passed through the lower NOR gate to generate a logic zero that is present for three clock cycles. This forms a dash exactly three times as long as a dot. Output 4 is not used and this sequence time provides the off period at the end of each dash. Output 5 is the dash reset line which resets the dash-generating sequence.

completing the circuit

Generating the dot and dash sequences is straightforward. Other circuits are used to detect the key closure and decide whether to send a dot or to extend the sequence and send a dash. A series of CD4013 D-type flip-flops and CD4001 two-input NOR gates are used for this, and shown in **fig. 8**.

The key input clocks a CD4013 flip-flop, U7A. The $\overline{Q}$ output from this flip-flop goes through a NOR gate to clock flip-flop U7B. U7B-2 turns on the gated clock to start the timing sequence. The Morse output from the sequencer/gating circuit is inverted and applied to the reset input of flip-flop U7A. This clears U7A and holds it in the cleared state for the length of any Morse element (dot or dash) being generated. The circuit is insensitive to any keying actions while it is generating a dot or dash. Since the clearing signal is the dot or dash, it is not present during the enforced off period and the circuit can detect another key closure during this time.

Several of the sequencer outputs are used to control another flip-flop, U4A, and four two-input NOR gates, U1. This part of the circuit determines whether or not to change from a dot to a dash sequence. Here is how it works: flip-flop U4A is used to determine whether or not the key is still pressed at the end of the dot now being generated. The key input provides the data (D input) signal to the flip-flop clocked at the start of the off period following a dot. If the key is open (as it would be for a properly sent dot) the state of the flip-flop will not be changed; the Q output remains a logic zero and the $\overline{Q}$ a logic one. So, when sequencer output 3 becomes a logic one, it will be passed through the four NOR gates (U1) and reset both the sequencer and flip-flop U7B. This ends the dot-generating sequence. Remember that the 0 output is a logic one when the sequencer is reset.

The key is held closed to send a dash. When the positive edge of sequencer output 2 appears, it clocks the key-closed condition into flip-flop U4A and the sequence is modified. The $\overline{Q}$ output (dash) from the flip-flop enables NOR gate U3B so the dash

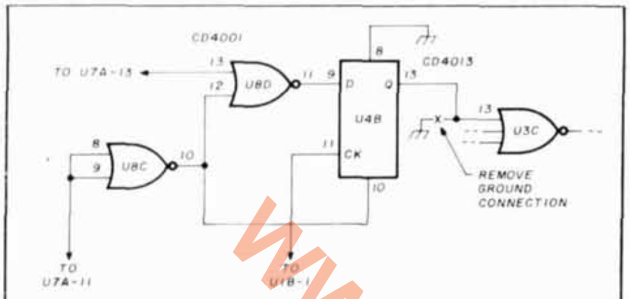

fig. 9. These circuits are added for the automatic tune-up feature.

sequence can be completed. The reset pulse normally generated by sequencer output 3 is blocked since the Q output from flip-flop U4A is now a logic one. This means that when the sequencer completes step 3, it will continue on to step 4 and then to step 5. The 5 output is now fed into NOR gates U1B and U1C to reset the system. When the sequencer returns to its reset condition, the logic one from the 0 output resets flip-flop U4A.

The same pulse that resets the sequencer counter/divider also resets flip-flop U7B, which controls the gated clock, turning the clock off. Since key-sensing flip-flop U7A may have detected a key closure during the enforced off time between elements, its condition must be tested and passed through to flip-flop U7B after the system has been reset. The sequence must be started again if a key closure is waiting. The testing operation is done by the NE555 monostable, U5, which is triggered by the reset pulse that clears the system. The pulse generated by monostable U5 is longer than the reset pulse. This allows the circuit to be completely reset before any new key-closure information is passed through to flip-flop U7B to restart the Morse-generating sequences. As mentioned previously, this allows the circuit to detect a key closure taking place during the enforced off time between dot and dash elements.

adding a tune-up circuit

Two unused gates and an unused flip-flop exist in the original circuit. These have been used to form an automatic tune-up circuit. Many hams like to make quick on-the-air adjustments to their transmitter or antenna, using key-down tuning for this. Most electronic keyer circuits use another switch or control function to constantly key the rig. If the keyer described here is used, you could only generate dots and dashes. The simple addition shown in **fig. 9** allows for constant keying.

This circuit checks to see if you still have your key closed at the end of a complete dash-generating cycle. If you depress your key and hold it closed, the keyer will generate a dash, a space and then go into a constantly-keyed mode so you can tune your rig. Releasing the key resets this operation so you can send code normally. No added tune control is needed when this tune-up circuit is used. Note: keep on-the-air tune-ups as short as possible! A complete keyer unit is shown in **fig. 10**.

The circuit described here follows your key operations instantaneously. There is no annoying dead-time or delay between your key closure and start of keying. The decision as to whether or not to send a dash is made on the fly. If you try and send too fast for the speed setting, you will immediately hear the result from your side-tone oscillator and can adjust your speed accordingly.

This circuit will generate accurately timed and spaced code for you and no new hand motions are required. Most hams take about 10 to 15 minutes to become accustomed to sending accurately-timed code with this keyer circuit. Since most of us are a bit inconsistent in our sending, the circuit will clean up the ragged edges of our code so it sounds almost perfect. Of course, it's up to you to generate the required spaces between characters and words.

learning the code

One of the reasons for designing this code-timing circuit was for hams to learn the sound of well-sent code and to learn sending good code with a straight key. Since you can only send code elements in the ratio of 1:3 with this keyer circuit, you quickly learn from the aural feedback whether you are sending good code or not. The ratio of 1:3:1 for dots, dashes and spaces is a bit difficult to master and this circuit can be used to great advantage in teaching newcomers the proper way to send code.

fig. 10. The completed keyer.

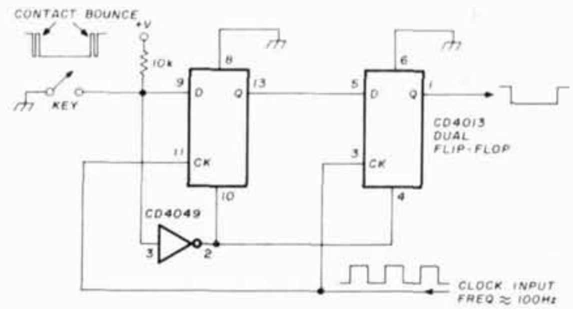

fig. 11. A sample switch debouncing circuit for single-pole single-throw switches.

finishing touches

The circuit in this article needs a key-debouncer as well as a transmitter keying circuit. A sidetone oscillator can be added if desired. The debouncer circuit can be a simple RC network or clocked circuit to filter multiple-contact closures characteristic of mechanical switches. A typical debounce circuit is shown in **fig. 11**. The external clock is set at about 100 Hz for 100 ms bounces of the key. The transmitter-keying circuit will depend on your rig; a small reed relay with appropriate contact rating and transistor oscillator can be used. A sidetone oscillator is easy to build with a 555 timer. This is recommended for off-the-air use, particularly if you're helping someone get started in ham radio.

CMOS devices in this design allow a power supply that provides 5 to 15 volts. I recommend using at least 9 volts. Most modern solid-state rigs use 12 to 14 Vdc, quite adequate. I don't recommend plug-in battery eliminators as a power source unless voltage regulating and filtering circuits are added.

Power consumption is low but will increase if you decide to add a sidetone oscillator driving a small speaker. When the automatic tune-up circuit is used with the keyer, only a speed control is needed. The upper sending speed may be increased by shorting the fixed series resistor with the Speed control in **fig. 8**. The values given provide speeds in the range of 3 to 30 words per minute. A sidetone oscillator volume control may be added, plus a power switch. CMOS circuits are fairly immune to electrical noise but a metal enclosure is recommended to protect the circuit from RFI.

ham radio