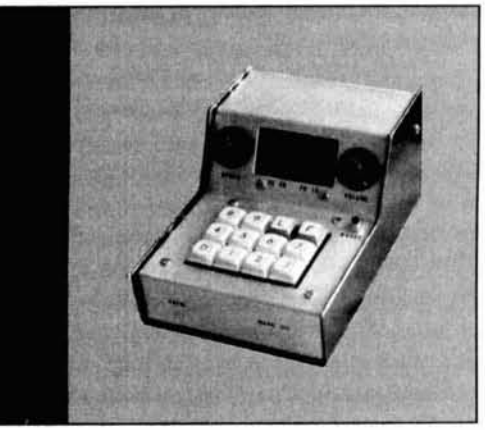

# microcomputer-based contest keyer

**During the last CW contest,** did you wish you had a memory keyer to send the contest exchange and give you more time for log and dupe-sheet entry? Did you wish you could automatically transmit the proper QSO number and generate any contest exchange? The microcomputer-based contest keyer may be the answer. This fully iambic keyer has ten programmable messages, a four-digit QSO counter, and the flexibility to transmit almost any contest exchange. An added feature is the display of the current code speed setting.

Several years ago I used tape-recorded tones to drive a relay and key my transmitter during a CW sweepstakes. Since this first attempt at automation, I built several memory keyers,[1] a TTL CW QSO number generator, and an 8080 microprocessor keyer that required three circuit boards. Because of the decreasing cost of microcomputer chips, I thought the time had arrived to design a state-of-the-art contest keyer.

Several operational objectives were set for the design. First, it should be an iambic keyer having dot and dash memory plus the optional forced-letter space of the popular WB4VVF *Accu-Keyer.*[2] For simple programming, the user should be able to load messages directly in code from the keyer paddle. Next, there should be several options such as automatic QSO number and RST generation. Last and most important, the keyer should be simple and natural to use but be flexible enough for any contest exchange. The result is the microcomputer keyer presented here.

## circuit description

Keyer construction is shown in the lead photo and **fig. 1.** Instead of a large keyboard, I selected a simple, twelve-button keypad for control. A potentiometer is included to adjust code speed and a four-digit LED display is included for QSO number and other alpha-numeric keyer messages. Simplicity results from using an Intel 8748 programmable microcomputer. The advantage of such a device is the ability to perform complex tasks with minimum circuitry. Software may be reprogrammed for other features or options. The following sections describe the design, discuss operation, and outline software routines for the keyer functions.

The keyer circuit, **fig. 2,** requires only seven integrated circuits and six transistors. The 8748 microcomputer, U1, contains 1024 bytes of electrically programmable read-only memory (EPROM), 64 bytes of random access memory (RAM), a programmable counter, and three 8-bit input/output ports. Since a complete description of the 8748 was recently published, it will not be repeated here. Readers should review the article by N6TY[3] and the Intel *MCS-48 Microcomputer Users Manual.*[4]

The 8748 has limited RAM for this design. U2, an Intel 8155, is used to expand the message buffer area. The 8155 contains 256 bytes of RAM, three input/output ports, and a 14-bit counter; only the RAM section is used. While this may seem a waste of internal functions, the 8155 will directly interface with the 8748's multiplexed data and address lines, DB0 to DB7. Additional interface circuitry is not required, saving at least two additional chips and considerable wiring.

U3 is an Intel 8279 programmable keyboard/display interface.* This versatile device can drive up to thirty-two multiplexed 4-bit display digits and simultaneously decode a 64-button keyboard. This design uses the interface to drive four 7-segment LED displays for alpha-numeric messages to the user. It also scans the 12-button keypad, decoding the key and signalling U1's IRQ (Interrupt Request) input with a logic 1. An internal first-in, first-out (FIFO) buffer stores up to eight keypresses; the user can "type ahead" without losing inputs.

Display multiplexing of the common-anode 7-segment display reduces the number of interconnect wires from 32 to 11. Anodes are switched by Q3 to Q6 from U6 inverting SL0 to SL3 from U3. Scan out-

*U2, U3 *functions* are programmed by commands from U1; the EPROM within U1 contains the actual program for the entire keyer.

**By Andrew B. White, K9CW,** 31 Quebec Road, Marlboro, New Jersey 07746

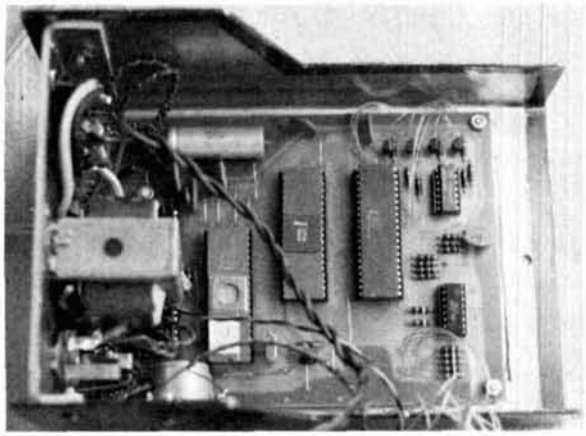

**fig. 1. The keyer electronics are contained on one single-sided printed circuit board.**

puts SL0 to SL3 are also used for the switch rows in the keypad. Segment driving is accomplished by A0 to A2, B0 to B3 from U3 through U7. Scanning is invisible to the user.

Keypad switch columns are connected to RL0, RL2, RL4, and RL6 of U3 for key decoding. U3 provides debouncing and key rollover detection internally. The keypad may be a conventional telephone type with separate switch contacts.

## command and control

U1, U2, and U3 are all connected to the same 8-bit data/address bus, DB0 to DB7, and the same Read (RD) and Write (WR) control lines. Data input or output for the microcomputer will be enabled by a logic 0 on either U1-36 (chip select for U3) or U1-37 (chip select for U2). U1-35 is the Control/Data (C/D) input for U3; Control mode (logic 1) is for display scanning, Data mode (logic 0) reads the keypad and loads the display.

The keyboard/display interface requires a clock input for display multiplexing and other timing operations. This is supplied by U1's Address Latch Enable (ALE) and is equal to the crystal frequency divide by 15. This signal eliminates a separate display oscillator. It should also be noted that U1 contains an internal clock oscillator circuit, requiring only an external crystal.

Dual timer U4 functions as both code-speed oscillator and sidetone generator. The code-speed section, U4A, is enabled by a logic 0 on U1-34. The same enable signal energizes clock indicator CR1 to indicate that the code clock is on. Code speed output, U4-5, is connected to U1's internal 8-bit counter. The microcomputer will set the length of one dot as 100 code speed pulses during CW operations. U4A oscillates at 100 times dot rate and is set by R1; speed range may be altered from the keypad.

Transmitter keying occurs when U1-33 is logic 0 (U5-10 at logic 1). The Q1, Q2 circuit is used with grid-block keying on my transmitter. Key-up voltage at the TX line is about −30 Vdc. Key-down voltage is close to ground. Other keying methods will require changing the Q1, Q2 circuit.

The microcomputer interrogates the key dot and dash paddle switches through I/O ports at U1-27 and U1-28. Internal pull-up resistors allow direct key connection.

U1 will automatically reset to program start when turned on. A manual reset push button, S1, is included for convenience. CR3 and CR4 are program status indicators and are described later.

Sidetone oscillator U4B is optional. The speaker may be any small unit with 8-ohm impedance or higher. CR2 indicates key down and should be retained; a keyer option function allows holding key down for tuning.

## software routines

Microcomputer hardware is useless without a program to make it "come alive". Software routines are described, but the detailed program is too long to include here. A complete program listing is available from the author.†

A flowchart of the main software polling loop appears in **fig. 3**. At power on or RESET, the keyer status flags are initialized and -HI- appears on the 4-digit display. The program enters a loop that continually checks the key paddle switches and keypad switch status, waiting for some request. Any switch closure will enable a branch to a routine that will generate the proper message or code element.

For example, assume the dot paddle switch is closed. The microcomputer turns on the transmitter for one dot period. It also checks the dash paddle switch and sets the dash memory flag if the dash switch was closed before the dot was completed. If so, a dash will begin after the dot space. Holding either dot or dash paddle closed will generate successive dots or dashes with proper spacing.

If no key paddle switches are closed, the microcomputer checks the letter space option. If enabled, no paddle inputs are processed for two dot periods, corresponding to one letter space interval. Key paddles are still tested during this interval. If either is closed during the interval, the proper flag is set and the appropriate dot or dash is sent after interval com-

†A copy may be obtained from the author for $3.00, which covers reproduction and postage costs.

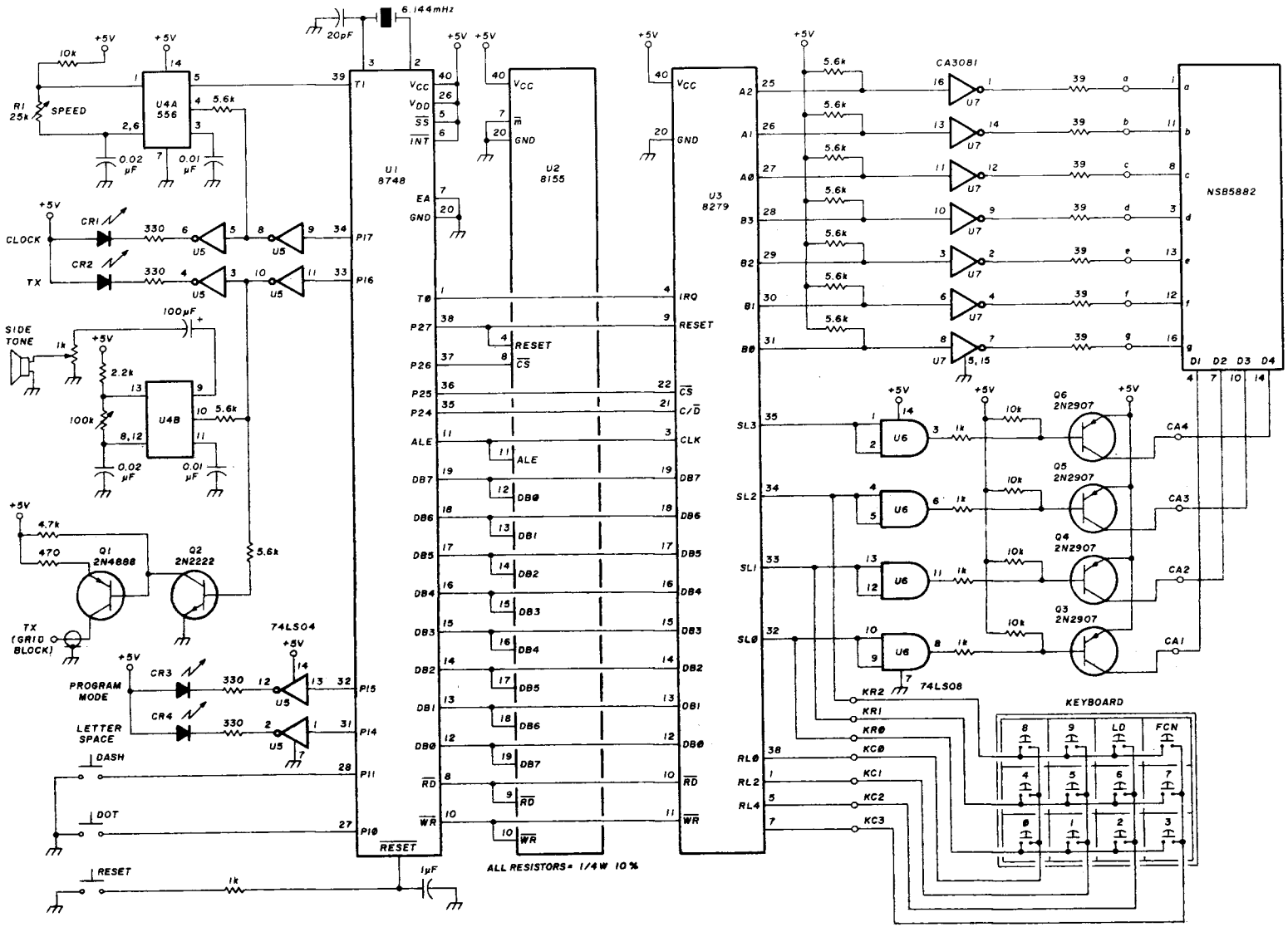


**fig. 2. The keyer schematic. The four-digit LED display module may be replaced with four individual seven-segment common-anode digits.**

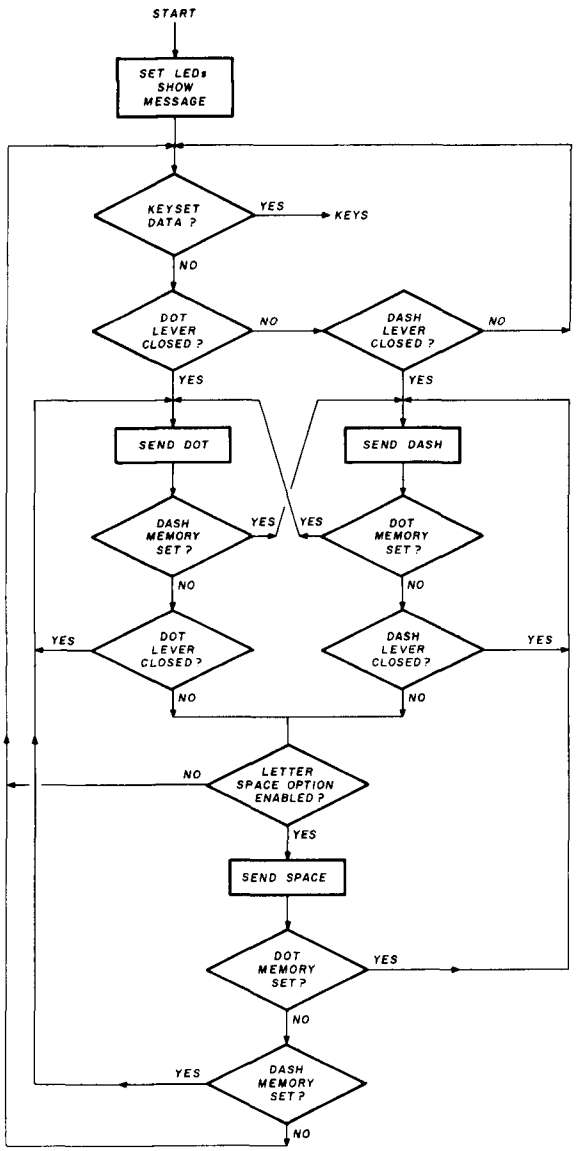


**fig. 3. A flow chart for the microcomputer keyer program. The keyer is fully iambic and has an optional forced letter space.**

pletion.* If no requests are pending, the program returns to the main polling loop and waits for the next input.

Pressing one of the keyboard switches calls a keypad processing routine. This routine decodes the keypad and executes one of several routines shown in the flow chart of **fig. 4.**

## function keys

To set various keyer options, the keypad F (FUNCTION) key is pressed, followed by 0 through 9 for the selected option. Pressing F will display -F- with the third digit blank to indicate an option number must be entered next. Entering the option will fill in the display and execute the function.

**Table 1** summarizes function-key options. The first four, F-0 through F-3, will display, increment, decrement, or load a four-digit QSO counter. If QSO count load is selected, the number keys are used to enter an initial count. Once this initial count is entered, pressing the F or L (load) key will load it into memory as the current QSO count.

**table 1. Function key options are activated by first entering F on the keypad, then the option number. Function F-9 may be terminated by pressing any key or key lever.**

| keypad entry | description |
|---|---|
| F-0 | display QSO total |
| F-1 | increment and display QSO total |
| F-2 | decrement and display QSO total |
| F-3 | load the QSO total |
| F-4 | toggle forced letter space option |
| F-5 | toggle 3-digit QSO number option |
| F-6 | toggle -HI-/QSO-total display option |
| F-7 | display code speed in wpm |
| F-8 | toggle code speed range |
| F-9 | tune transmitter (key-down hold) |

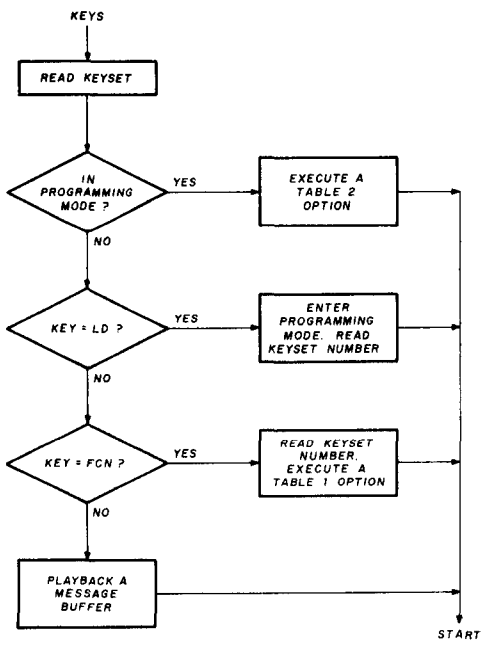


**fig. 4. The operation performed by each key depends on the current keyer mode. For example in the message programming mode, the table 2 options are selected.**

*The microcomputer is able to do several things seemingly at once due to execution speed and internal IC circuitry.

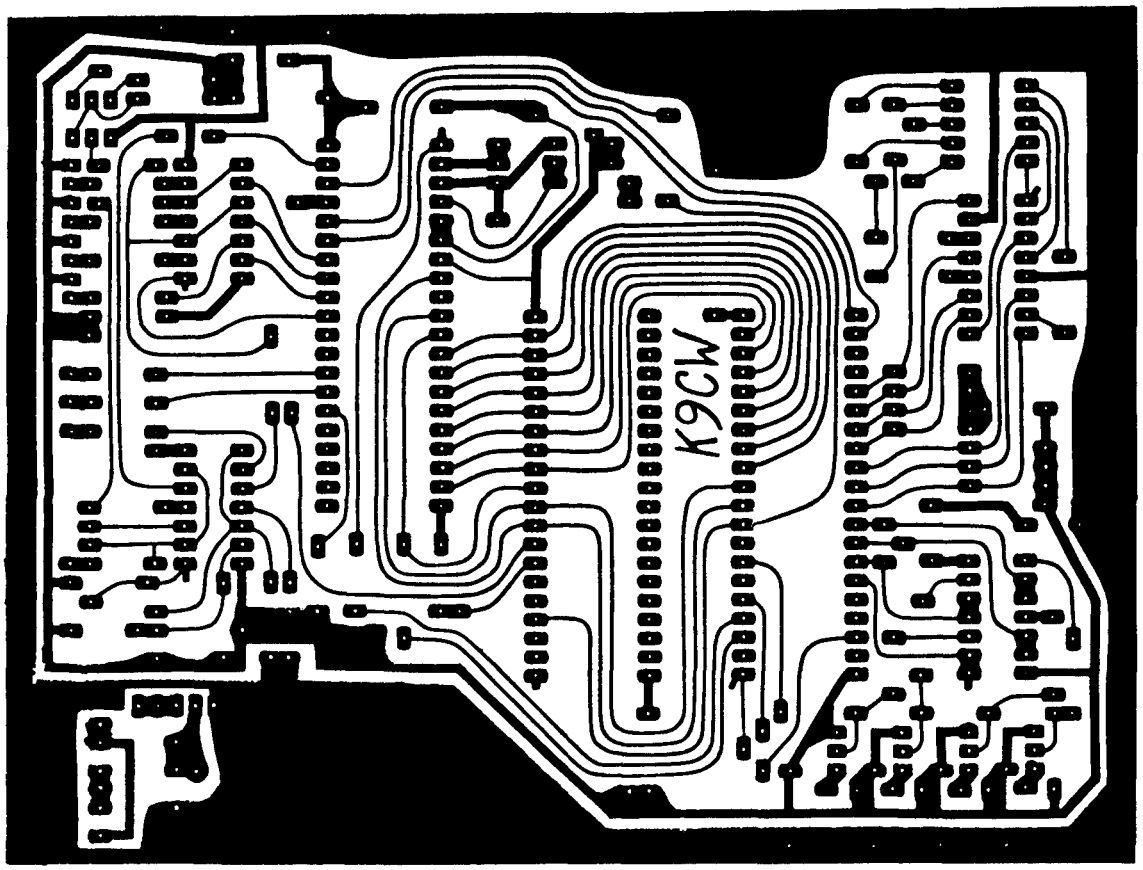


**fig. 5. Printed circuit board pattern for the keyer shown from the foil side.**

Options F-4, F-5, and F-6 select various keyer operating modes. Letter space option F-4 will force a letter space if neither dot nor dash is closed upon completion of a code element. Front panel indicator CR4 will light if this function is enabled. Option F-5 determines if QSO numbers should always be transmitted as three digits or if the leading zeros in the number should be suppressed. Option F-6 allows display of QSO total instead of -HI- during the input waiting period. Default values at turn-on are: Letter space enabled, suppress leading zeros, display -HI-.

Current code speed in words per minute is displayed by the F-7 option. Since the microcomputer is continuously measuring speed, speed control R1 can be adjusted through digital readout. Pressing any key or paddle lever will exit this function.

Option F-8 selects speed range, either 16-60 wpm or 5-30 wpm. Default value is 16-60 wpm. Option F-9 will key down the transmitter for tuning purposes; key up occurs on pressing any key or the paddle. Transmit indicator CR2 will be on and -F9- displayed during key down.

## loading messages

To load one of the message buffers, the L key is pressed, followed by a selected message number of 0 through 9. Entering only L will display -L- with a blank third digit prompting the user for a message number. Entering the number will fill in the display such as -L5- for message number five.

Once the load identification is complete, the manually sent code is stored in memory until either F or L is pressed to terminate the message. The number of memory bytes per message is displayed during message entry. Message buffers 0 through 9 are contiguous in U2 so that individual messages can be up to 25 bytes long without interfering with the next higher message number.

Twenty-five bytes can store 100 dots, dashes, or spaces, but this is not a limit. Because the keyer allows variable message length, one long message containing over 1000 dots, dashes, or spaces is possible; this is an average of 200 letters at three dots or dashes per letter.

In addition to manually entered code, several special functions can be loaded into a message. **Table 2** gives the function options and a specific display for user confirmation. CR3 will be on during message loading.

**table 2. Message loading options are activated by a single keypad entry. These options insert special functions into the message during message loading.**

| keypad | display | description |
|---|---|---|
| 0 | -C0- | send QSO total |
| 1 | -C1- | increment and send QSO total |
| 2 | -59- | send 5NN or 5 (keypad) 9 |
| 3 | -IP- | send keypad numbers |
| 4 | -SP- | insert a letter space |
| 5 - 8 | | ignored; reserved future use |
| 9 | -00- | restart message loading |
| L or F | | terminate a message load |

## special message options

During playback, option 0 transmits the current QSO total as two to four numbers depending on leading-zero suppression previously set by F-5. Option 1 first increments the QSO total then sends the number.

Option 2 transmits an RST in one of two formats. If a key has been typed during message playback but before this option is encountered, that keypad number is sent as the middle digit (signal strength) in the RST. If no keypad number has been pressed, 5NN is transmitted.

Option 3 allows transmitting a number directly from the keypad. The keypad FIFO buffer is checked when this option is encountered. If any keypad numbers are stored (maximum of 8), they are sent, the FIFO empties, and message playback resumes.

Option 4 simply inserts one letter space each time it is entered. Option 9 allows the user to restart loading in case of error. Options 5 through 8 are currently ignored, reserved for future additions.

These options allow almost any contest message to be loaded. For example, 599001 can be loaded by typing 2, then 1. Keypad 2 calls for the 5NN; keypad 1 calls for the QSO total (incremented from zero, no suppression of leading zeros). NR 100 IL is loaded by manually keying NR, pressing 1 on the keypad, pressing 4 twice for two letter spaces, then keying IL manually.

During message loading, a routine measures the time between manually keyed dots and dashes. It corrects most spacing errors. No more than one word space interval will be automatically stored in memory. The user can stop loading without filling the memory with spaces. No word space is automatically inserted after loading options. Inserting spaces (option 4) permits storage of perfect Morse code.

## message playback for transmission

A stored message is played back by simply pressing the appropriate keypad number. If message number seven is selected, the display shows -P7-. If one of the QSO number options is encountered in a message, the current or incremented QSO count is displayed. The keyer returns to the main polling loop when a message is complete and displays -HI- or the current QSO count (if F-6 is set).

Manually sent code can be inserted at any time during a message playback. When a paddle switch is closed, the keyer stops and waits for a word space interval. If the paddle switch is still closed, manual code transmission begins. When manual code ceases, the stored message playback continues. Message playback can be halted at any time by pressing the RESET button or briefly tapping the dot or dash levers.

A common method of storing Morse code in a digital memory is to use a length equivalent to each element. A Morse dot would appear in memory as binary 010. A Morse dash would be binary 01110. This keyer uses two bits for each element: A letter space is binary 00, a dot is binary 01, a dash is binary 10. Binary 11 is used as an option indicator for one of the message options. This assignment allows four code elements per memory byte.

## construction

The circuit may be built on the PC pattern of **figs. 5** and **6** or with wire-wrap techniques. Sockets are suggested for either case to avoid IC damage. Space has been provided on the PC layout for diode rectifiers and a filter capacitor. A three-terminal voltage regulator should be used with an output rating of +5 Vdc at 0.5 amperes. Mount it on the cabinet for heat sinking.

Several 0.1 $\mu$F disc capacitors are used on the PC board to bypass the +5 volt line at each IC and reduce logic switching noise. A wire-wrap version should have these capacitors installed first with minimum lead length.

The finished keyer can be mounted in a commercial case or a homemade one such as in the title photo. Some users may want to separate the keypad to move the main keyer circuitry off the desk. In either configuration, be certain to rf bypass all leads into or out of the keyer cabinet.

Locating components isn't difficult.* Most of the

*Drilled and etched PC boards and some parts are available from RADIO-KIT, Box 411, Greenville, N. H. 03048.

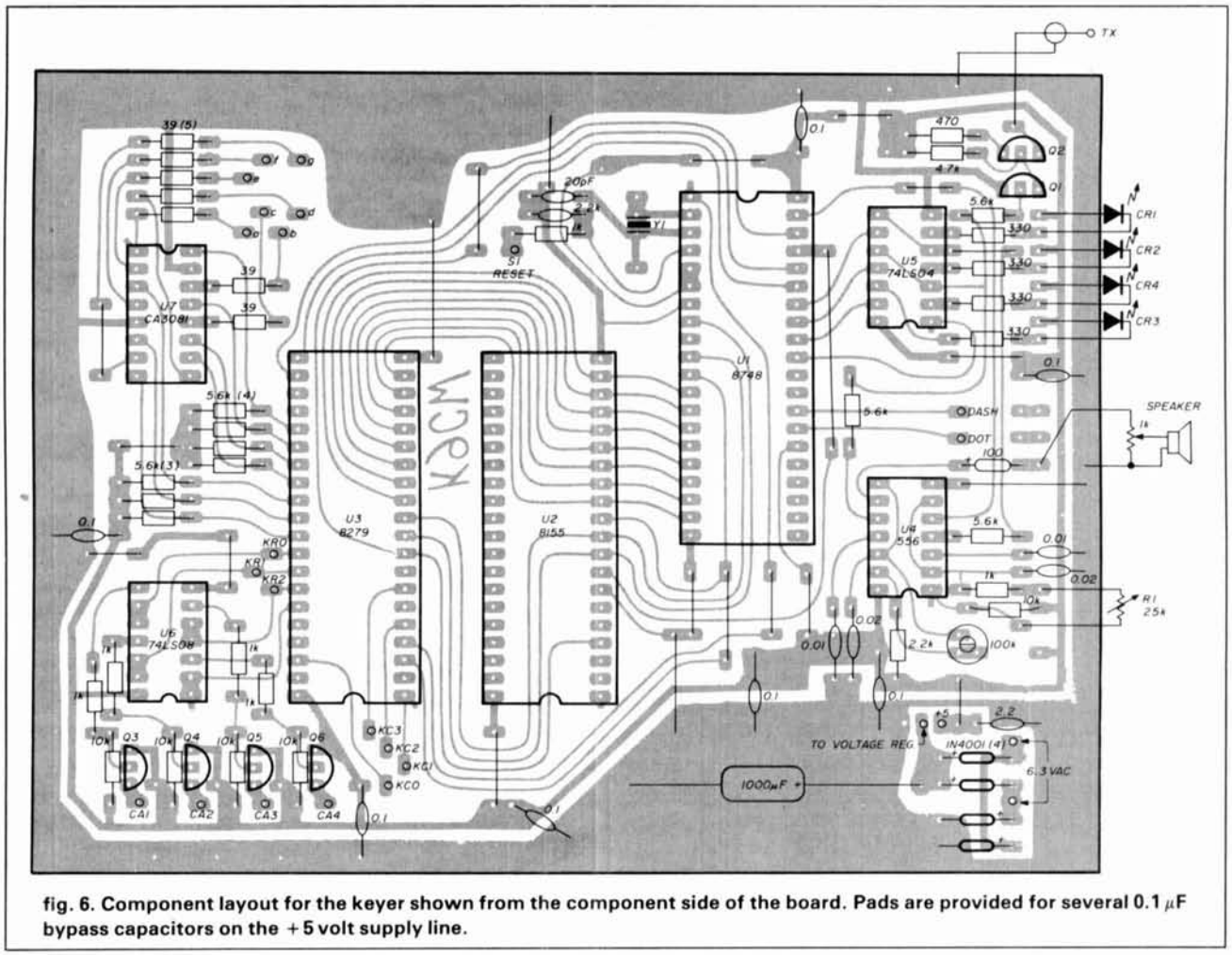


**fig. 6. Component layout for the keyer shown from the component side of the board. Pads are provided for several 0.1 μF bypass capacitors on the +5 volt supply line.**

computer hobbyist advertisers carry the Intel components. The four-digit LED display is a National Semiconductor NBS7882, but four separate seven-segment, common-anode displays may be substituted in place of that unit.

The 8748-8 is a lower-speed version that can use a 3.58 MHz color TV crystal; the program listing includes modification for this option. Either crystal should be series-resonant with series impedance less than 75 ohms for 6.144 MHz; less than 180 ohms for 3.58 MHz.

The 8748 is supplied unprogrammed. You must locate a nearby programming facility with an Intel compatible EPROM programmer. EPROM programming of the 8748 requires more care than a conventional hobby computer PROM burner. Intel supplies all information on EPROM programming in reference 4.

## conclusion

This keyer has been used in only a few contests so far. I find the automatic number generation and multiple message playback features to be useful operating aids. Automatic message transmission during each exchange is an advantage that allows filling in the dupe sheet and checking off section multipliers. In addition, errors are never made in transmitting the QSO number or other parts of the contest exchange.

My objective in writing this article is not only to describe a powerful, inexpensive contest keyer, but also to start people thinking about new applications for microcomputers in Amateur Radio. I'd be interested in your ideas for using this new technology. Microprocessors and computers are already appearing in common ham equipment. Several manufacturers sell combination RTTY and Morse keyboards and displays based on microprocessors.

Perhaps a future version of the contest keyer will be able to scan the band, copy a call, check the dupe list, and make a contest contact automatically. While that operation may be feasible, I don't believe it's desirable. After all, what would remain for the contest operator to do?

## references


1. A.B. White, K9CW, "Programmable Memory Accessory for Electronic Keyers," *ham radio*, August 1975, page 24.
2. J. Garret, WB4VVF, "The WB4VVF Accu-Keyer," *QST*, August 1973, page 19.
3. J. Beaston, N6TY, "CW Trainer/Keyer Using a Single Chip Microcomputer," *ham radio*, August 1979, page 16.
4. *MCS-48 MICROCOMPUTER USER'S MANUAL*, Intel Corporation, 1977.


**ham radio**