


- | | | |
|--|------------|--|
| <i>David Koh, M.D.</i> | 26 | Raster Scan Graphics for the 6800 . . . Part 1: the hardware |
| <i>Robert L. Kurtz</i> | 34 | World of the Brass Pounders . . . receive Morse code the easy way |
| <i>Peter A. Stark</i> | 38 | Kilobaud Classroom . . . No 14: parallel and serial I/O ports |
|  <i>Bruce A. Barker</i> | 46 | Point of Sale/Inventory Control System for 8080s |
| <i>D. Borland</i> | 50 | Financier/Mortgage with Prepayment . . . from Instant Software |
| <i>Stephen Gibson</i> | 52 | Hey, Kids! It's "Mickey Modem"! . . . not a toy |
| <i>Rick Grossman</i> | 60 | Do It with a Kimsi! . . . KIM expansion the easy way |
| <i>Robert J. Bishop</i> | 62 | The Remarkable Apple-II . . . a fun machine, plus |
| <i>Easton Beymer</i> | 67 | Universal Number Converter . . . touches all bases |
| <i>Cal R. Rasmussen</i> | 68 | Hexadecimal Memory Dump . . . painless |
| <i>Douglas A. Schwab</i> | 70 | IC Testing . . . a complete system |
| <i>Rod Hallen</i> | 82 | Music, Music and More Music . . . it hath charms |
| <i>F. R. Ruckdeschel</i> | 84 | Build a Serial-to-Parallel Converter . . . a snap |
| <i>Stuart Mitchell, Phil Poole</i> | 90 | Expansion Time! . . . S-100 memory for SWTP system |
| <i>Erich A. Pfeiffer</i> | 94 | What Is a Bus? . . . climb aboard for the answer |
| <i>William L. Colsher</i> | 98 | Lucas' Puzzle . . . solution can be tricky |
| <i>Jay Kuca</i> | 100 | Home Brewing in Style . . . with solderless breadboarding |
| <i>Thomas E. Hutchinson</i> | 108 | Modify Your COSMAC Elf . . . for a little pelf |
| <i>James Gross</i> | 114 | Loop the Loop . . . iteration techniques explained |

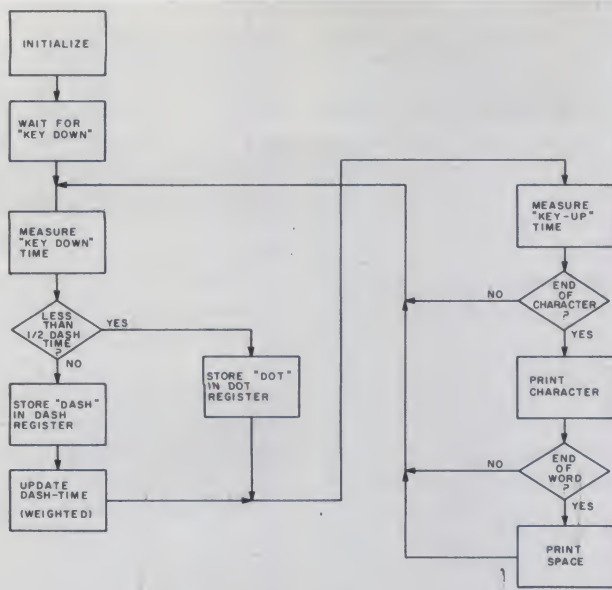


Fig. 1. Simplified flow diagram.

this must be changed to fit your particular system.

In addition, the program assumes that when a dot or a dash occurs, a logic 0 appears on the input port. This is in agreement with a "key down" shorting the Input port to ground, and also in agreement with the hardware interface circuit described later. If your hookup provides a logic 1 during a dot and a dash, then lines 11, 30, 70, 120 and 160 must be changed so that all IF A=1 statements should read IF A=0, and vice versa.

Program Description

Fig. 1 is a simplified flow diagram of the program. The initialization routine (lines 1 through 6) sets the lookup table that will permit the printout of the proper character. The program then waits for a key down to occur (lines 10 and 11). The first part of the operating program (lines 20 through 80) measures the length of time that the key is down and compares this with the stored value for the length of a dash.

If the key is raised in less than one-half of the stored dash time, the computer writes a dot into the dot register (line 30) and goes to the second part of the operating program.

If the key remains down longer than one-half of the dash time, a dash is stored in

the dash register (line 50), and the value of the dash time is updated with a one-to-four weighting (line 80). This is accomplished by multiplying the old value of the dash time by four, adding the new value, and then dividing by five. As a result, the stored value of the dash time cannot change drastically from character to character, and the copy is not susceptible to errors from erratic sending habits.

The second part of the program (lines 100 through 190) measures the length of time the key is up. If it's up less than one-half of the dash length, the program assumes that the character is not complete and no printout is provided (see lines 100 through 130). If the key is up longer, the program jumps to line 300, the print-character subroutine. If the key is up longer than twice the dash length, the program assumes that a word is complete and a

EXAMPLE: D = — ● ●

Initial conditions: Dot register = 0, Dash register = 0.

First period: Input a dash.

$$2 \text{ times dash register} = 2 \times 0 = 0.$$

$$2 \text{ times dot register} = 2 \times 0 = 0.$$

$$\text{Add 1 to dash register} = 1 + 0 = 1.$$

$$\text{Summary: dash register} = 1, \text{ dot register} = 0.$$

Second period: Input a dot.

$$2 \text{ times dash register} = 2 \times 1 = 2.$$

$$2 \text{ times dot register} = 2 \times 0 = 0.$$

$$\text{Add 1 to dot register} = 1 + 0 = 1.$$

$$\text{Summary: dash register} = 2, \text{ dot register} = 1$$

Third period: Input a dot.

$$2 \text{ times dash register} = 2 \times 2 = 4.$$

$$2 \text{ times dot register} = 2 \times 1 = 2.$$

$$\text{Add 1 to dot register} = 1 + 2 = 3.$$

$$\text{Summary: dash register} = 4, \text{ dot register} = 3.$$

End of Character—determine lookup number for D.

$$2 \text{ times dash register} = 2 \times 4 = 8.$$

$$\text{Add dot register and dash register} = 8 + 3 = 11.$$

Answer: D = 11.

Example 1.

1. If the input signal is a dash:

A. Double the values in the dot and dash registers.

B. Add 1 to the dash register (see line 50).

2. If the input signal is a dot:

A. Double the values in the dot and dash registers.

B. Add 1 to the dot register (see line 30).

3. If the character is complete:

A. Double the value in the dash register.

B. Add the dash and dot registers to obtain the lookup number.

C. Clear the dot and dash registers.

Table 1.

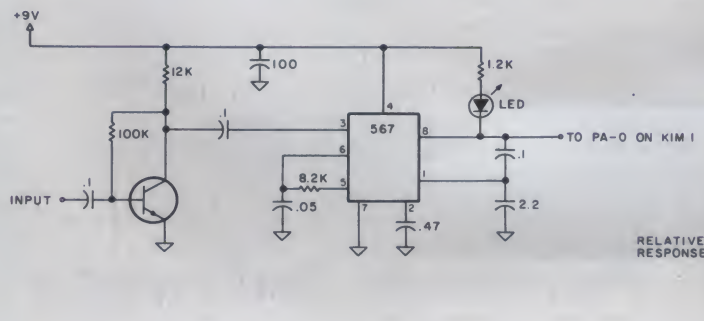
"space" is printed (lines 170 and 180).

Lookup Table

The heart of the program is the algorithm that counts the dots and dashes and develops a number used to look up the actual character to be printed. In other words, each combination of dots and dashes in Morse code has a discrete number that commands a given charac-

ter to be printed. This algorithm has three conditions as listed in Table 1.

Steps 1 and 2 keep repeating until the character is complete. When the program detects a key-up period longer than one-half of a dash length, it is assumed that the character is complete and step 3 is accomplished (lines 300 to 430). The manner in which the lookup number for the letter D is



RELATIVE RESPONSE

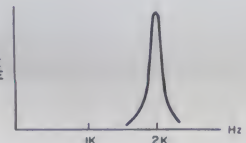


Fig. 2. Interface circuit.

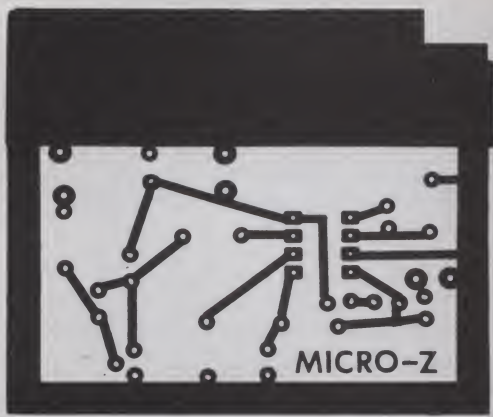


Fig. 3. Circuit board.

formed is shown in Example 1.

Interface Hardware

Fig. 2 shows a typical circuit for connecting your radio receiver to the computer. The NPN transistor is an R/C coupled audio amplifier connected to a type 567 phase-lock loop circuit. The free-running frequency of the phase-lock loop is set by the values of the

capacitor and resistor connected to pins 5 and 6 of the 567, and is approximately 2000 Hz. The capacitors on pins 1 and 2 of the PLL adjust the bandwidth to about 100 Hz, and the LED serves as a tuning indicator—that is, it will start blinking when the signal is in the center of this narrow band-pass.

This circuit is compatible

with the program, as written, in that the output signal goes to a logic 0 when a dot or a dash occurs. The circuit shown is not my original idea, but has appeared in numerous publications; you may have your own favorite circuit that you would like to use. As long as you maintain the same output logic (a 0 for a dot or dash), there will be no problem.

Incidentally, the circuit has another unique application. Since it is only activated by audio signals over a fairly narrow band, it can also be used to key an audio oscillator set to any frequency desired. When Morse CW comes in amidst a jumble of other signals, the phase-lock loop picks out the signal you want and keys the audio oscillator . . . and that is all you hear. A full-scale outline of the circuit board is shown in Fig. 3.

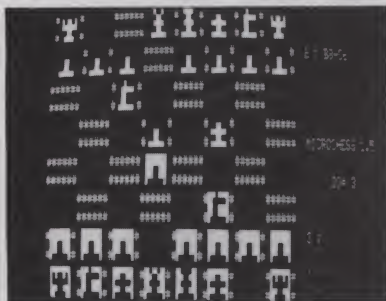
Adjusting the Program

One of the advantages of

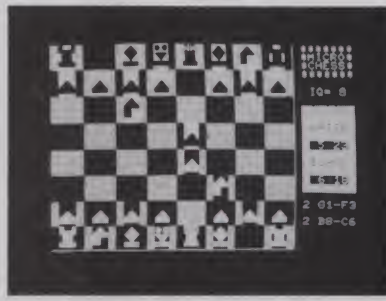
writing this program in BASIC is the ease with which the computation constants can be changed. For instance, you may wish to experiment with different algorithms to detect whether a key-down signal is a dot or a dash . . . to take care of "swing-fisters." This can be accomplished easily by changing the factor in line 40 from (.5°C) to (.25°C) or (.75°C). By the same token, the constants in lines 111 and 151 can be changed to provide more leeway for the formation of characters and spaces.

One final note for KIM users: The 9K Microsoft BASIC is excellent and may be obtained from Micro-Z Co., Box 2426, Rolling Hills CA, for \$100. In addition, a full kit of parts for the interface circuit, circuit board, Morse-code reader listings in both BASIC and machine language, and instructions on how to connect it to an audio oscillator are available from Micro-Z Co. for \$16.50 postpaid. ■

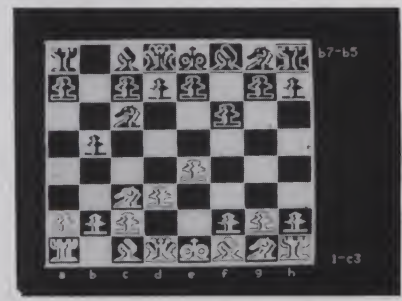
TRS-80



PET



APPLE



PLAY CHESS WITH YOUR COMPUTER!

MICROCHESS is the culmination of two years of chessplaying program development by **Peter Jennings**, author of the famous 1K byte chess program for the KIM-1. **MICROCHESS 2.0** for 8K PETs and 16K APPLES, in 6502 machine language, offers 8 levels of play to suit everyone from the beginner learning chess to the serious player. It examines positions as many as 6 moves ahead, and includes a chess clock for tournament play. **MICROCHESS 1.5** for **BRIDGE CHALLENGER** by **George Duisman** for 8K PETs, Level II 16K TRS-80s, and 16K APPLES: You and the dummy play 4 person Contract Bridge against the computer. The program will deal hands at random or according to your criterion for high card points. You can review tricks, swap sides or replay hands when the cards are known. No longer do you need 4 people to play! **\$14.95**
ORDERS: Check, money order or VISA/Master Charge accepted; programs and cassettes guaranteed. If you have questions, please call us at **617-783-0694**. If you know what you want and have your VISA/MC card ready, you can **DIAL TOLL FREE 1-800-325-6400**

4K TRS-80s, in Z-80 machine language, offers 3 levels of play (both Level I and Level II versions are included and can be loaded on any TRS-80 without TBUG). **MICROCHESS** checks every move for legality and displays the current position on a graphic chessboard. You can play White or Black, set up and play from special board positions, or even watch the computer play against itself! Available now at a special introductory price of only **\$19.95**
STIMULATING SIMULATIONS by **Dr. C.W. Engel** for 8K PETs, 4K Level I and II TRS-80s, and APPLES with Applesoft II: Ten original simulation games such as Forest Fire, Lost Treasure, Gone Fishing and Diamond Thief, progressing from elementary to quite complex with most suitable for schoolchildren. Includes a 64 page book giving flowcharts, listings and suggested modifications . . . **\$14.95** (24 hours, 7 days; in Missouri, dial 1-800-342-6600). Or you can mail your order to the address below. **Personal Software™ products are now AVAILABLE NATIONWIDE FROM COMPUTER STORES.** Look for the Personal Software™ display in your local store!

P.O. Box 136-K11 **Personal Software™** P39 Cambridge, MA 02138