Interface a Chessboard to Your KIM-1

Jeff Teeters
1720 Coolidge Ct
Eau Claire WI 54701

Chess is a fascinating game. Computer chess is especially fascinating because the complex analysis which determines each move is performed by a machine instead of a human. Computer chess offers an excellent way to demonstrate the power and versatility of personal computers.

Most computer chess systems are unable to "see" a chessboard. A human playing against a computer will usually set up a chessboard beside the computer, and the moves will be communicated to and from the machine through the use of a keyboard and a display in some type of abstract notation.

Keyboard entry of moves is undesirable. It is inconvenient, error prone, and inelegant. The abstract notation promotes errors and makes play difficult for people who do not know the notation system. Furthermore, errors may not be detected until many intervening moves have occurred.

An ideal chess-playing system would contain a digital television camera to observe the board and a mechanical arm to move the pieces. *[A mechanical arm designed for exactly this application was described in the article "A Hobbyist Robot Arm," by Keith Baxter and Timothy Daly in the February 1979 BYTE, page 84...***RSS***]* A less costly alternative is to construct a chessboard which can electronically communicate with the computer. The computer may then "look" at the board position through its I/O(input/output) ports. A means of indicating the computer's moves on the chessboard itself may also be provided.

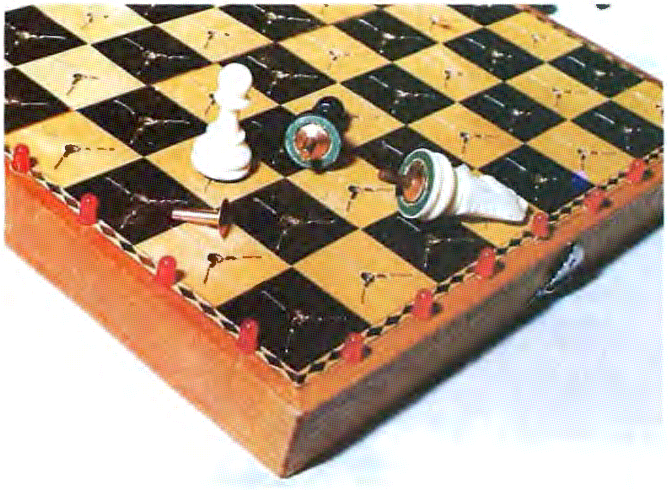


Photo 1: Two pawns, a White Knight, and a loose rivet are shown on top of the electronic chessboard. One row of 8 light emitting diodes (LEDs) is placed along the left side of the board, and another row is placed along the bottom of the board as seen by the human player. Two LEDs are lit to indicate a single square, using an X,Y axis system. A single large hole is drilled in the center of each square to accept entrance of the rivet which is glued to the bottom of each chessman. The rivet completes an electrical circuit between 2 pieces of wire that run from smaller holes through the large central hole. This switching arrangement allows the computer to detect the presence or absence of a piece at each square of the board. In this prototype, an additional set of 3 wires is seen in each square; these wires remain from an earlier, unsuccessful switching attempt.

In the system that I have constructed, the user makes his move on the electronic chessboard, instead of typing each move on a keyboard. The computer's moves are displayed on the chessboard through the use of discrete light emitting diodes (LEDs), arranged in an X,Y coordinate system. The LEDs show the user exactly which chessman the computer wants to move, and to which square. In addition to being aesthetically pleasing, this system makes it impossible to enter your move in-

About the Author

Jeff Teeters is an undergraduate student at the University of Wisconsin at River Falls where he majors in mathematics.

correctly, and easy to interpret the computer's move. The board is continuously scanned so that even if the user moves the computer's piece incorrectly, the mistake is detected immediately. A speaker is connected to the computer to let unwary users know (by a buzz) when they misinterpret a computer move. This speaker also emits a brief sound when the chess program has decided on a move and when it has been recorded into the computer's internal board representation.

This project is designed for specific use with Peter Jenning's Microchess, running on a KIM-1 with about 0.5 K bytes of extra memory. Implementation on other 6502 based computer systems should be relatively easy since only a few minor software modifications would be needed. The required hardware consists of a chess set, a package of cheap switching diodes, 2 integrated circuits, 16 discrete LEDs and 32 copper rivets.

The chessboard should have a thin, nonconductive surface that is easy to drill holes through. This surface must be supported by side panels so there is a hollow space of about 2 cm under the board for wiring. I used a cheap plywood chess set that is designed to fold into a storage box for the chessmen. The copper rivets should be small in diameter, about 12 mm long, and have a flat top. The ones that I used were size 9 rivets manufactured by the Tower Corporation of Madison IN.

System Concepts

KIM-1 Microchess uses an internal board-status table to keep track of the whereabouts of the chessmen. This table contains 32 square numbers which indicate the position of the 32 pieces. It is important to realize that Microchess generates moves solely on the basis of what is in that table, and not how it was placed there. My plan of attack was simple. I had only to wire a chessboard to the computer and write an interface program that would translate moves on the chessboard into changes in the table. Since this program will be needed only when moves are physically being made, it can be called from Microchess and used in place of the Microchess keyboard I/O (input/output) routines. After the user has finished moving, control can be transferred back to Microchess to compute the machine's next move.

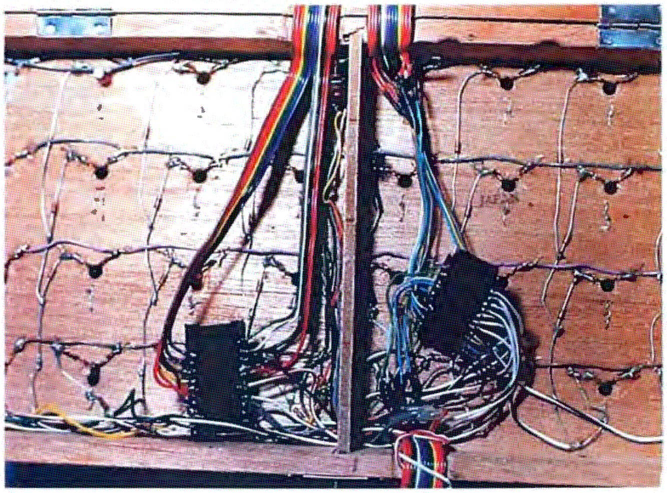


Photo 2: The bottom of the chessboard. The switching diodes and connecting wires are soldered directly to the wire contacts in the central holes. The 2 integrated circuits are type SN74154 decoder/demultiplexers. Note the tips of rivets protruding through some of the holes.

The Microchess to chessboard interface program is logically straightforward. If no move is being made, the table should be an accurate representation of the board. A move is detected when the table does not correctly represent the current board position. If an empty square appears on the board where the table indicates that a chessman resides, then the user has just picked up that man. If the table shows an unoccupied square which the board indicates is occupied, a chessman has just been set down in that square. A move is constituted by the user picking up a man and setting it down in some other location. A capture is completed by picking up 2 men and setting 1 down in the space formerly occupied by the other. Because the Microchess table is updated each time a simple move or capture is made, the table always gives an accurate representation of the current board position.

Photo 3: The complete chessplaying system. The completed electronic chessboard stands in the foreground. The chessboard and the sound-effect speaker are connected to the KIM-1 computer residing in the suitcase in the background.

Hardware Details

Note that the chessboard interface program can keep track of the moves that are made simply by knowing if individual squares are occupied by a piece or are empty. The circuit which provides this information to the computer is illustrated in figure 1. For purposes of square identification, the chessboard is conceptually cut in half. The 2 pieces are placed logically end to end, forming an arrangement

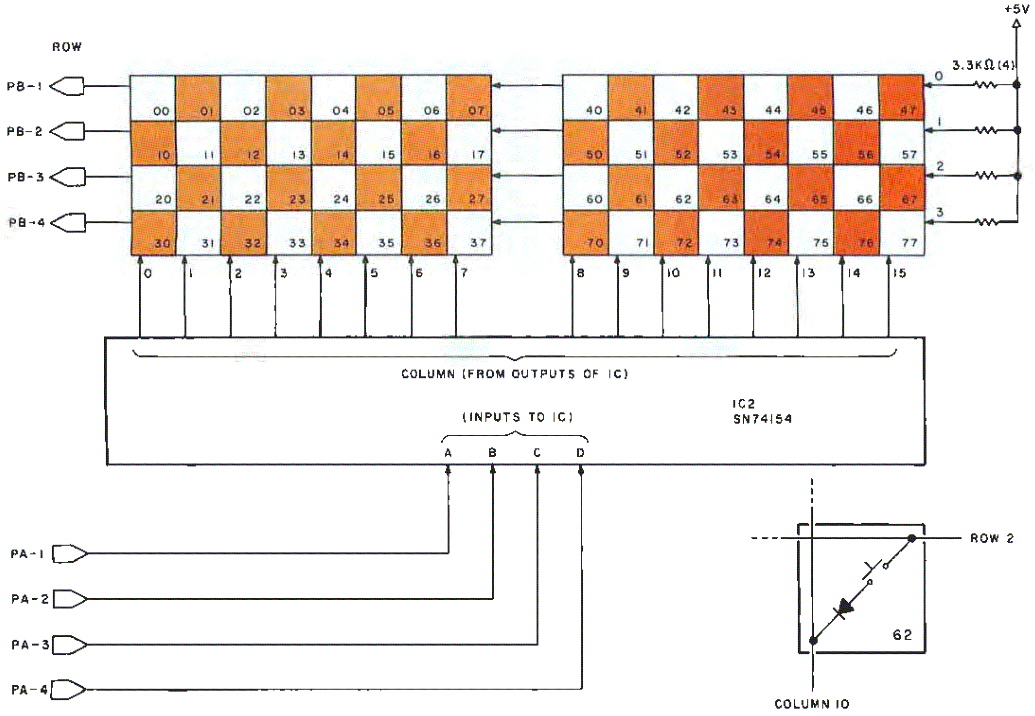


Figure 1: Circuit which determines whether or not a given square is occupied. The chessboard is conceptually cut in half. It is placed so that the squares form a 4 by 16 matrix. For each square, a diode and a switch are wired in series between the appropriate row and column lines. A closed switch indicates an occupied square; an open switch indicates an empty square.

of 4 rows and 16 columns. A diode matrix allows the hardware to identify the individual squares.

The integrated circuit in figure 1 is a type SN74154 4 to 16 line decoder/demultiplexer. The 4 input lines to the device are connected to the KIM-1 I/O port A. Each of the 16 output lines is linked to a column in the matrix. This portion of the circuit allows the KIM-1 to select 4 squares out of the total of 64. The 4 rows of the matrix are connected to the I/O port B. Row and column addressing allows scanning of a single square. Each square of the chessboard has a switch. A closed switch indicates that the square has a piece on it; an open switch shows that the square is empty.

To determine whether or not a piece is on a particular square, the interface program first selects the column by sending the correct binary code to the 4 input lines on the SN74154. This brings 1 of the 16 output lines low, while the diodes keep the rest high. If the switch is closed (ie: a piece is on the square), then the corresponding row-line will be pulled low and the matching port-B data register bit will be a 0. Thus,by selecting the column through port A and testing the row bits in port B, it is possible to determine the status of every square on the board.

Switch Experimentation

Now for the hard part: what can be used as a switch? The actual mechanical operation remains the only unresolved detail. All that is needed is some means of closing the switch whenever a piece is set down, and opening it when one is picked up. There are several ways to accomplish this—some of which are better than others.

In my first attempt I put aluminum foil on the bottom of the pieces and used simple wire contacts on top of the board. I punched 6 holes into each square using a large needle to form the corners of 2 concentric, equilateral triangles. Three strands of wire were looped through the holes forming 3 symmetric contacts (see figure 2a). The third contact was used only to balance the pieces.

The concept is simple. The piece is set on top of the wire contacts and the aluminum foil makes the necessary connection. Unfortunately it didn't work. The contacts were not sufficiently stable, and the slightest vibration rocked the pieces, leading the program to believe that the user was trying to move 5 or 10 pieces at once.

That problem might have been solved by mounting magnets on the pieces and using a chessboard with a nonconductive magnetic surface.

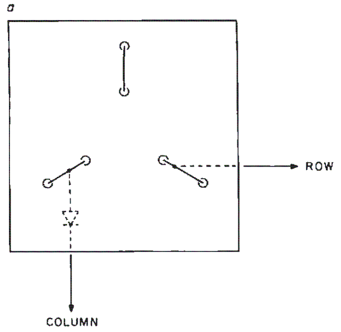


Figure 2a: The first attempt to form a switch for the squares. Three symmetric contacts on top of each square were made by looping bare wire through holes in the board. Two of the contacts were wired to the row and column lines on the back side of the board. (The third wire was simply to balance the piece upright.) The pieces had aluminum foil glued to their bottoms. When such a chessman was set down on the contacts, electrical continuity was achieved. Unfortunately, vibration caused intermittent contact and confused the computer.

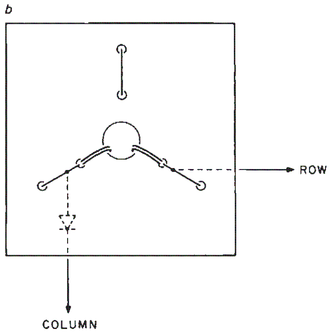


Figure 2b: The second attempt to form a square switch. This attempt was successful. Copper rivets were glued to the bottom of the chessmen. A large hole was drilled in the center of each square to receive the rivet. Two wires were looped through the large central hole from 2 smaller holes (left over from the first switch attempt). The rivet closes the electrical circuit.

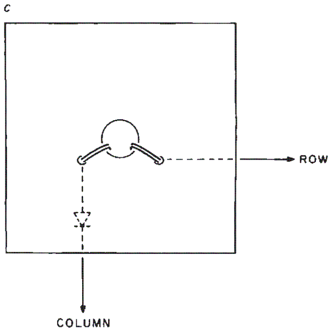


Figure 2c: Illustration of the appearance of a square which uses rivet switches, and which previously did not have other methods installed in it. The reader may do it correctly the first time.

Another possibility would be to eliminate wire contacts entirely and use reed switches or some type of photocell. Unfortunately, one such device must be mounted under each square, necessitating a total of 64 devices. Although they would have undoubtedly worked, 64 photocells or reed switches would have cost more than I was willing to spend on the project.

Switch Success

I eventually figured out a contact method that was both cheap and reliable. I drilled a small hole in the center of each square, just large enough to slide in a copper rivet. Two strands of bare copper wire from 2 of the inner contact holes used in my first attempt were looped *through* the larger central hole forming 2 contacts inside of the hole (see figure 2b). The felt on the bottom of the pieces was peeled off and the tapered copper rivets were glued onto the metal weight underneath the felt with an instant bonding adhesive.

I have found that these contacts work quite well. The tapered copper rivets slide easily in and out of the hole, while slight pressure from the sides of the hole forces the rivet to make good contact with the copper wire. The pieces remain intact and the electrical contacts remain solid, even when the chessboard is held upside down and shaken gently. Of course when you wire your chessboard, you should leave out the 3 symmetric wires that I tried on my first version. Only the 2 strands which were looped through the rivet hole need to be installed (see figure 2c).

Hardware for Computer Output

The LEDs are wired according to figure 3. The integrated circuit is another 4 to 16 line decoder whose 4 inputs are connected to the I/O ports. Note that decoder outputs 0 thru 7 are connected sequentially to the rank— indicating (Y axis) LEDs with the 0-bit output being connected to the uppermost LED. Likewise, the file—indicating (X axis) LEDs are connected left to right with outputs 8 thru 15. The chip-enable line is connected to I/O port pin PB0 so that the LEDs can be turned off while Microchess is computing a move.

Mounting of the LEDs on the sides of the chessboard is relatively straightforward. I used a large needle to punch the holes for the leads prior to insertion. Glue can be used to hold them in place. Be sure to orient the chessboard so that a white square is in the lower right-hand corner of the side facing the human player. This means that the 2 rows of LEDs installed on the left side and bottom of the board will meet at a corner containing a black square.

The speaker is connected to output port pin PA0 in the manner described in the *KIM-1 User's Manual* on page 57. See figure 4 for an illustration of the I/O port connections.

Software

The necessary modifications to Microchess are shown in listing 1. The Microchess to chessboard interface program with source and object listing is given in listing 2. Although I used a nonstandard meta-assembler, most of the mnemonics are similar to, if not the same as, the MOS Technology standard mnemonics. The listings are fairly well documented.

There are, however, some general concepts that may be difficult to deduce from the listings. The workhorse of the chessboard interface program is subroutine GET-MOVE. GET-MOVE calls the KIM monitor routine GETKEY before doing anything else, in order to see if the user has pressed the **DA** key (which is used when setting up a new position) or the **PC** key (which clears

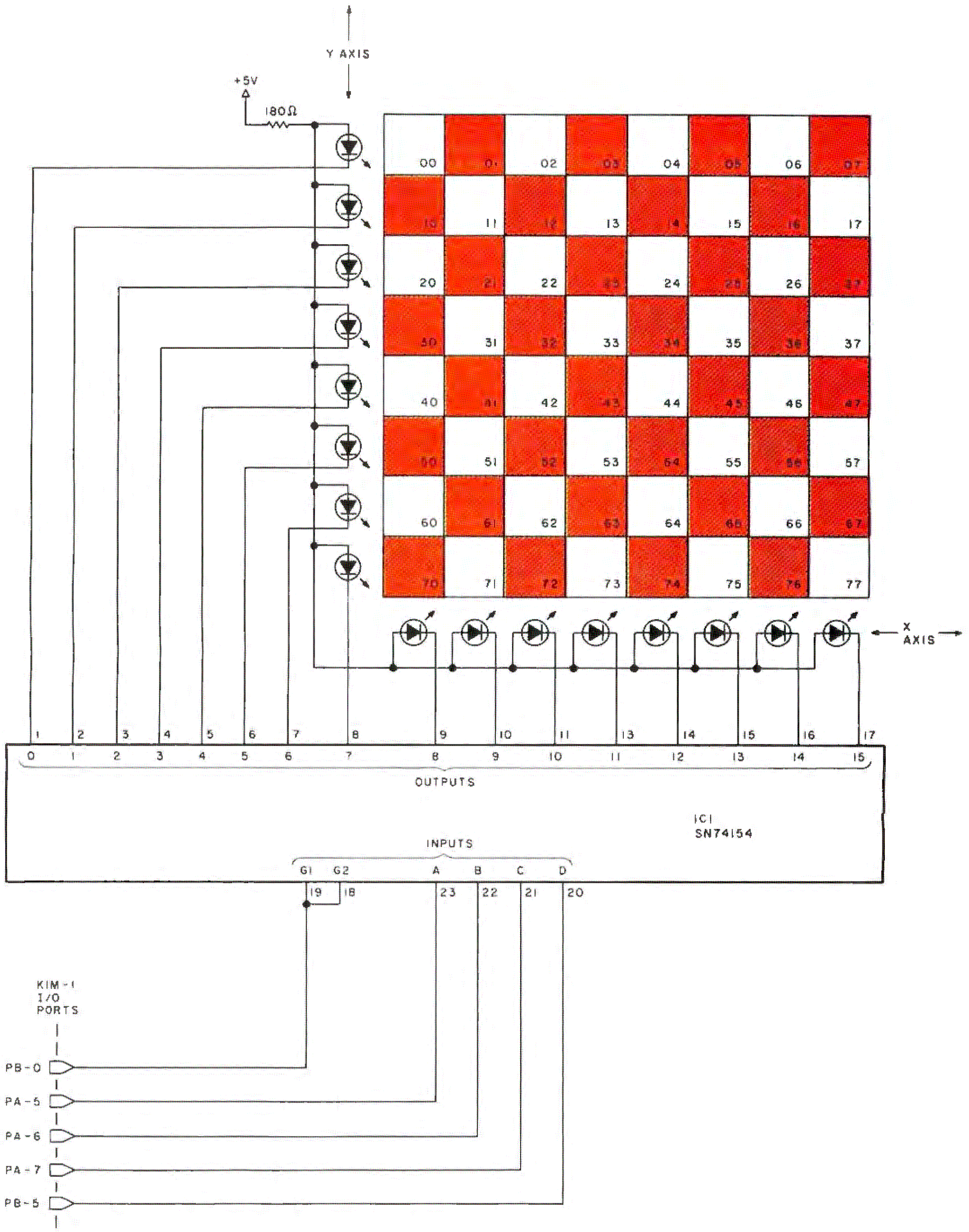


Figure 3: Circuit for lighting the light emitting diodes (LEDs) that indicate the computer's move. The computer moves as follows. The program lights the X and Y axis LEDs which together indicate the single square on which the piece to be moved resides. The person picks up the indicated piece. After the user picks up the piece, different LEDs light up that point to the square to which the piece is to be moved. The person then places the chessman as indicated. A mistake causes the computer to emit a characteristic sound. The chip-enable line of IC1 is connected to I/O (input/output) port pin PB 0 so that the LEDs may be turned off while the chess program is computing its next move.

the board for a new game). If neither the DA nor PC key is depressed, GET-MOVE scans the chessboard, square by square, searching for pieces that were recently picked up or set down. This is done by comparing the Microchess board-status table to the current board position, as previously described. There is one important exception. When the user picks up a piece to make a move, SHOULDBEUP-FLAG is made nonzero, and the square where the piece used to be is stored in hexadecimal addresses FA and F9. A nonzero SHOULDBEUP-FLAG tells subroutine GET-MOVE that the 2 squares in FA and F9 should not be occupied, even if they are shown in the table. This is done to prevent GET-MOVE from continuously reporting that the same piece was picked up.

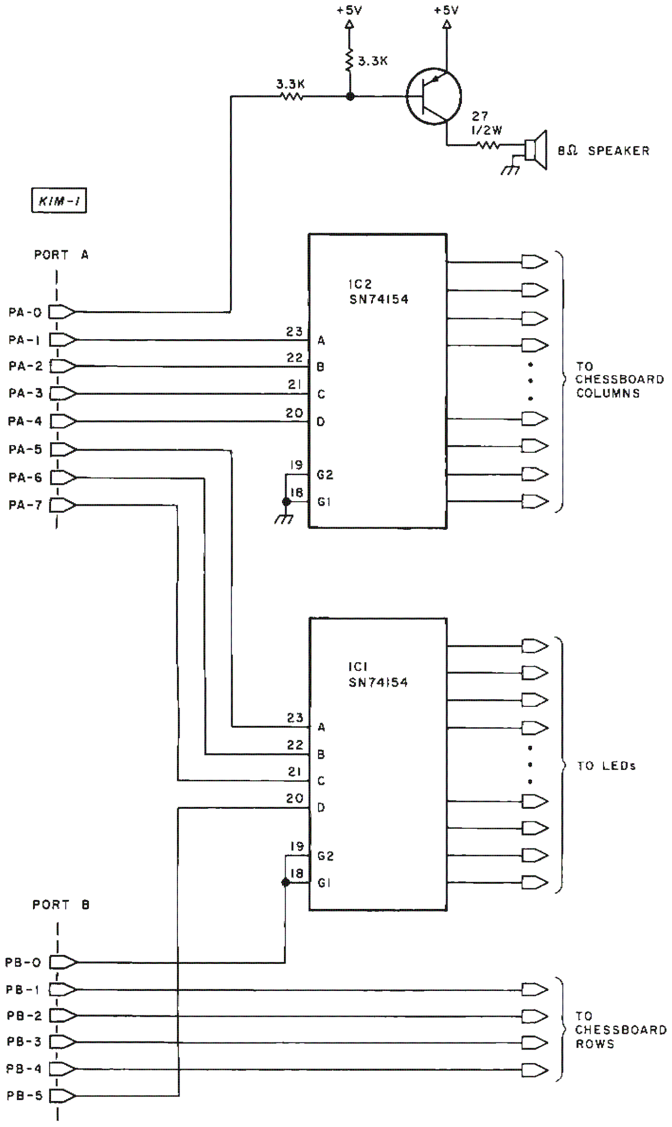


Figure 4: Schematic diagram of chessboard input connections for the KIM-1. If the speaker is built into the chessboard, a 16 conductor cable is required to connect the board to the KIM-1 application connector. Thirteen conductors control the chessboard and light emitting diodes; 3 are needed for speaker, ground, and +5 V supply. The cable should be of sufficient length that the chessboard may be set in a convenient position for game playing.

Upon exit from the subroutine, the result of the search is stored in the accumulator and in location UP-CLEAR-DOWN. A +1 is returned if a piece has been picked up, a 0 if there is no change, and a −1 if a piece was set down. If a piece *was* picked up or set down, then CHANGING-SQUARE will contain the number of the square where the pickup or set-down occurred. Likewise, if a piece was picked up, then CHANGING-PIECE will contain the hexadecimal designation of that piece as outlined on page 3 of the Microchess player's manual.

While GET-MOVE is scanning the chessboard, it also lights up the X and Y axis LEDs that point to the square in LIGHT-SQUARE. If SPEAKER-FLAG is nonzero, the speaker is rapidly toggled to produce a hum.

Subroutine CLEAR-STACK resets the Microchess and the machine stack pointers back to their initial values. The subroutine is called from various parts of the interface program to prevent the stacks from overflowing into Microchess code.

After Microchess has computed each move, control is transferred to the start of the interface program at hexadecimal address 2000. The user must physically move the pieces for the computer. The piece designation and the from and to squares of the calculated move are stored in the KIM display at hexadecimal addresses FB, FA, and F9 respectively. Because of the no-operation instructions inserted at address 03E1, the move has not been recorded in the board-status table. Addresses 2000 through 2040 of listing 2 contain code

Text continued on page 46

Listing 1: Modifications which were made to Peter Jennings' KIM-1 Microchess program to allow for the use of the electronic chessboard. Change the specified locations in memory with the KIM monitor.

Address (Hexadecimal)	New Code (Hexadecimal)			Comments
0008	A9	FF		Set up Port A-DDR
000A	8D	01	17	
000D	A9	21		Set up Port B-DDR
000F	8D	03	17	
0012	4C	00	20	Jump to interface program
0033	00			Toggle, must be −1 or zero
003F	60			Return from CLDSP
00B7	02	04		MASK-TABLE (used
00B9	08	10		to read row)
01AC	60			Return from DISP
03A7	B1			Use. SQUARE for flag
03E1	EA	EA	EA	Don't record move
03E9	20	39	00	Show all FFs
03EC	4C	00	00	(Concede defeat)

Listing 2: The Microchess to chessboard interface routine, a sort of chessboard device handler program. This listing is the output of an assembly with both source and hexadecimal object code shown. It is written in a nonstandard assembly language of the author's own design, although most of the mnemonics are similar to the MOS Technology standard mnemonics.

```
0000              !SET BIGC;ORG 2000;
2000              !  COMMENT  *** KIM-1 MICROCHESS TO CHESSBOARD INTERFACE ***
2000              !                PROGRAM.  WRITTEN BY JEFF TEETERS. 9/8/78
2000              !;
2000              !DEFINE  .BOARD=50,              % ADDRESS OF PIECE TABLE
2000              !        .BD-1=4F,               % .BOARD LESS ONE
2000              !        .BK=60,                 % ADDRESS OF USERS PIECES
2000              !        .SP2=B2,                % MICROCHESS STACK POINTER
2000              !        .SQUARE=B1,             % TO SQUARE USED BY MOVE
2000              !        CHANGING-SQUARE=27,     % RETURNED BY GET-MOVE
2000              !        CHANGING-PIECE=28,      % PIECE PICKED UP AT CH-SQR
2000              !        CLDSP=3900,             % CLEAR DISPLAY
2000              !        CLEAR-BOARD=1800,       % SET UP NEW GAME
2000              !        COUNT-FLAG=29,          % SET WHEN COUNTING DOWN
2000              !        DISP=9D01,              % DISPLAY PIECE NAME IN FB
2000              !        FLASH-DISPLAY=1F1F,     % KIM MONITOR ROUTINE
2000              !        FROM-SQUARE=2A,         % USED WHEN UNMOVING CAPTURE
2000              !        GETKEY=6A1F,            % KIM MONITOR ROUTINE
2000              !        GO=A203,                % ADDRESS OF CHESS PROGRAM
2000              !        LIGHT-SQUARE=2B,        % SQUARE LIGHTED BY LEDS
2000              !        MASK-TABLE=B7,          % USED TO READ ROW
2000              !        MOVE=4B03,              % ROUTINE TO UPDATE .BOARD
2000              !        PORT-LIGHT=2C,          % USED TO BUILD IO PORT
2000              !        PORT-SQUARE=2D,         %   "                "
2000              !        PORT-A=0017,            % KIM-1 I/O PORT
2000              !        PORT-B=0217,            %  "            "
2000              !        RANDOMN=0417,           % KIM-1 INTERVAL TIMER
2000              !        REVERSE=B202,           % ROUTINE TO EXCHANGE SIDES
2000              !        SPEAKER-FLAG=2E,        % =1,GET-MOVE GENERATES TONE
2000              !        SHOULDBEUP-FLAG=2F,     % =1,SQUARES IN FA & F9 UP
2000              !        SWITCH-FLAG=30,         % SET IN EXCHANGE
2000              !        TCHANGING-PIECE=31,     % TEMPORARY CHANGING PIECE
2000              !        TCHANGING-SQUARE=32,    % TEMPORARY CHANGING-SQUARE
2000              !        TEMP=F3,                % TEMPORARY STORAGE LOCATION
2000              !        TOGGLE=33,              % ALTERNATLEY LIGHTS FA &  F9
2000              !        TUP-CLEAR-DOWN=34,      % TEMPORARY UP-CLEAR-DOWN
2000              !        UMOVE=3103,             % ROUTINE TO UNMAKE MOVE
2000              !        UP-CLEAR-DOWN=35,       % STATUS OF CHANGING-SQUARE
2000              !        "+"=12,                 % RETURN VALUE OF PLUS KEY
2000              !;
2000              !        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2000              !        %%%%%%%%%%    USER MOVE COMPUTER PIECE    %%%%%%%%%%
2000              !        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2000              !          %*********      PICK THE PIECE UP      *********%
2000   A9 00      !               LDA# 00                   %RESET FLAG
2002   85 2F      !               STA  SHOULDBEUP-FLAG
2004   A5 FA      !               LDA  FA                   %LIGHT "FROM SQUARE"
2006   85 2B      !               STA  LIGHT-SQUARE
2008   E6 2E      !PICK-IT-UP:    INC  SPEAKER-FLAG         %SOUND OFF
200A              !               LOOP
```

```
200A  20 6B 21  !                   JSR  GET-MOVE        %WAIT FOR PLAYER TO
200D  F0        !                   BEQ                  %%PICKUP PIECE.
200E  FB        !                ENDLOOP
200F  30 F7     !                BMI  PICK-IT UP         %ERROR, PIECE SET DOWN
2011  A5 27     !                LDA  CHANGING-SQUARE    %IS PIECE PICKED UP
2013  C5 FA     !                CMP  FA                 %%CORRECT ONE?
2015  D0 F1     !                BNE  PICK-IT UP
2017  A6 FB     !                LDX  FB                 %YES, SET TABLE ENTRY
2019  A9 CC     !                LDA# CC                 %%TO "CC"
201B  95 50     !                STAX .BOARD
201D            !          %*********     SET THE PIECE DOWN     *********%
201D  A5 F9     !                LDA  F9                 %LIGHT TO SQUARE
201F  85 2B     !                STA  LIGHT-SQUARE
2021  E6 2E     !SET-IT-DOWN:    INC  SPEAKER-FLAG       %MAKE NOISE
2023            !                LOOP
2023  20 6B 21  !                   JSR  GET-MOVE        %WAIT FOR CHANGES
2026  F0        !                   BEQ
2027  FB        !                ENDLOOP
2028  A5 F9     !                LDA  F9                 %IS USER MOVEING
202A  C5 27     !                CMP  CHANGING-SQUARE    %%CORRECT PIECE?
202C  D0 F3     !                BNE  SET-IT-DOWN
202E  A5 35     !                LDA  UP-CLEAR-DOWN      %YES
2030  10 08     !                IF NEGATIVE THEN
2032  A6 FB     !                   LDX  FB              %UPDATE TABLE.(PIECE
2034  A5 27     !                   LDA  CHANGING-SQUARE %%SET DOWN, MOVE HAS
2036  95 50     !                   STAX .BOARD          %%BEEN COMPLETED.)
2038  10        !                   BPL
2039  08        !                ELSE
203A  A6 28     !                   LDX  CHANGING-PIECE  %CAPTURED PIECE HAS
203C  A9 CC     !                   LDA# CC              %%BEEN PICKED UP...
203E  95 50     !                   STAX .BOARD          %%UPDATE TABLE AND
2040  30 DF     !                   BMI  SET-IT-DOWN     %%WAIT FOR SET DOWN.
2042            !                ENDELSE
2042            !        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2042            !        %%%%%%%%%%%    USER MOVE USER'S PIECE     %%%%%%%%%%
2042            !        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2042  E6 2E     !                INC  SPEAKER-FLAG       %MAKE NOISE
2044  A9 00     !WAIT-FOR-MOVE: LDA# 00                  %CLEAR UP FLAG
2046  85 2F     !                STA  SHOULDBEUP-FLAG
2048  85 29     !SET-COUNT:      STA  COUNT-FLAG         %SET COUNT FLAG
204A  20 39 00  !SET-DISPLAY:    JSR  CLDSP              %CLEAR DISPLAY
204D  20 6B 21  !GET-MOVE1:      JSR  GET-MOVE
2050  D0 41     !                IF ZERO THEN
2052            !            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2052            !            %%%%%%%%      NO CHANGE IN BOARD     %%%%%%%%
2052            !            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2052            !              %*********  CHECK FOR "GO" KEY  *********%
2052  20 6A 1F  !                   JSR  GETKEY
2055  C9 13     !                   CMP# 13              %13=VALUE OF "GO" KEY
2057  F0 17     !                   BEQ  GO-COUNTDOWN
2059            !              %*********  CHECK FOR "E" KEY   *********%
2059  C9 0E     !                   CMP#  0E
205B  D0 16     !                   IF ZERO THEN         %USER WANTS TO
205D            !                      LOOP              %%SWITCH SIDES
205D  20 6A 1F  !                         JSR GETKEY
2060  C9 0E     !                         CMP# 0E        %DEBOUNCE KEYBOARD
2062  F0        !                         BEQ
2063  F9        !                      ENDLOOP
2064  A5 30     !                      LDA  SWITCH-FLAG  %TOGGLE FLAG
2066  49 FF     !                      EOR# FF
2068  85 30     !                      STA  SWITCH-FLAG
206A  20 B2 02  !SWITCH-SIDES:         JSR  REVERSE      %PERFORM EXCHANGE
206D  20 5D 21  !                      JSR  CLEAR-STACK
2070  4C 4C 21  !GO-COUNTDOWN:         JMP  START-COUNTING
2073            !                   ENDIF
2073            !              %*********  "GO" OR "E" NOT FOUND *********%
2073  A5 29     !                   LDA  COUNT-FLAG      %COUNTING DOWN?
2075  F0 12     !                   IF NOTZERO THEN
2077  C6 FB     !                      DEC  FB           %YES.
2079  A5 FB     !                      LDA  FB
207B  D0 06     !                      IF ZERO THEN
207D  20 5D 21  !                         JSR CLEAR-STACK %PLAY CHESS
2080  4C A2 03  !                         JMP GO
2083            !                      ENDIF
2083  A9 0F     !                      LDA# 0F           %STILL COUNTING DOWN,
2085  65 2B     !                      ADC  LIGHT-SQUARE %%LIGHT NEXT SQUARE.
2087  D0        !                      BNE
2088  03        !                   ELSE
```

Listing 2 continued on page 46

Listing 2 continued from page 45:

```
2089  AD 04 17  !          LDA@ RANDOM#      %WAITING FOR MOVE...
208C            !       ENDELSE              %%LIGHT RANDOM SQUARE
208C  85 2B     !       STA  LIGHT-SQUARE
208E  A5 FB     !       LDA  FB
2090  4C 4A 20  !       JMP  SET-DISPLAY
2093            !     ENDIF
2093  10 49     !     IF NEGATIVE THEN
2095            !  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2095            !  %%%%%%%%      NEW PIECE SET DOWN    %%%%%%%%
2095            !  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2095            !    %********* TAKING BACK CAPTURE? *********%
2095  A5 B2     !       LDA  .SP2
2097  C9 C8     !       CMP# C8             %UNDOING PREVIOUS
2099  F0 0C     !       IF NOTZERO THEN     %%  CAPTURE?
209B  A5 2A     !          LDA  FROM-SQUARE
209D  C5 27     !          CMP  CHANGING-SQUARE
209F  D0 06     !          IF ZERO THEN
20A1  20 31 03  !             JSR  UMOVE    %     YES.
20A4  4C 44 20  !             JMP  WAIT-FOR-MOVE
20A7            !          ENDIF
20A7            !       ENDIF
20A7            !    %*********USER ADDING NEW PIECE**********%
20A7  20 6A 1F  !       JSR  GETKEY         %WAIT FOR KEY ENTRY
20AA  C9 15     !       CMP# 15             %%OF HEX NAME OR "+"
20AC  F0 1C     !       IF NOTZERO THEN
20AE  C9 12     !          CMP# "+"
20B0  D0 0C     !          IF ZERO THEN
20B2  A5 FA     !             LDA  FA       %FOUND "+", ENTER NEW
20B4  10 97     !             BPL  GET-MOVE1 %%PIECE INTO TABLE IF
20B6  A6 FB     !             LDX  FB       %%NOT IN ALREADY
20B8  A5 27     !             LDA  CHANGING-SQUARE
20BA  95 50     !             STAX .BOARD
20BC  10 B6     !             BPL  WAIT-FOR-MOVE
20BE            !          ENDIF
20BE  85 F3     !          STA  TEMP        %FOUND HEX DIGIT
20C0  A5 FB     !          LDA  FB          %%"OR" IT INTO
20C2  0A 0A     !          ASL  ASL         %%PIECE NAME.
20C4  0A 0A     !          ASL  ASL
20C6  05 F3     !          ORA  TEMP
20C8  85 FB     !          STA  FB
20CA            !       ENDIF
20CA  A5 27     !       LDA  CHANGING-SQUARE  %BUILD DISPLAY
20CC  85 2B     !       STA  LIGHT-SQUARE
20CE  85 F9     !       STA  F9             %%%F9=SQUARE ON BOARD
20D0  A5 FB     !       LDA  FB             %PUT PIECE NAME IN
20D2  29 1F     !       AND# 1F             %%RANGE
20D4  85 FB     !       STA  FB             %%%FB=PIECE NAME
20D6  AA        !       TAX
20D7  B5 50     !       LDAX .BOARD
20D9  85 FA     !       STA  FA             %%%FA=TABLE ENTRY
20DB  4C 4D 20  !       JMP  GET-MOVE1
20DE            !     ENDIF
20DE            !  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20DE            !  %%%%%%%%       PIECE PICKED UP      %%%%%%%%
20DE            !  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20DE            !    %**********  USER PLAY WHITE?  **********%
20DE  A5 B1     !     LDA  .SQUARE          %SEE IF USER MAKING
20E0  C9 CC     !     CMP# CC               %%THE FIRST MOVE.
20E2  D0 05     !     IF ZERO THEN
20E4  E6 B1     !        INC  .SQUARE       %YES, CHANGE .SQUARE
20E6  4C 6A 20  !        JMP  SWITCH-SIDES  %%AND EXCHANGE
20E9            !     ENDIF
20E9            !    %**********   WAIT FOR CHANGE   **********%
20E9  20 5D 21  !     JSR  CLEAR-STACK      %CLEAR POSSIBLE JUNK
20EC  A5 27     !     LDA  CHANGING-SQUARE
20EE  85 FA     !     STA  FA               %DISPLAY SQUARE NUM
20F0  85 F9     !     STA  F9
20F2  85 2B     !     STA  LIGHT-SQUARE
20F4  20 9D 01  !     JSR  DISP             %DISPLAY PIECE NAME
20F7  A9 01     !     LDA# 01
20F9  85 2F     !     STA  SHOULDBEUP-FLAG  %SET FLAG
20FB            !     LOOP
20FB  20 6B 21  !        JSR  GET-MOVE      %AWAIT MOVE.
20FE  F0        !        BEQ
20FF  FB        !     ENDLOOP
2100            !    %********* PIECE SET BACK DOWN *********%
2100  10 11     !     IF NEGATIVE THEN
```

Listing 2 continued on page 48

Text continued from page 42:

to light the correct LEDs and modify the board-status table as the user completes the computer's move. The speaker sounds briefly after each correct step is completed. If a wrong piece is moved or a piece is set down on a wrong square, the speaker will hum continuously to signal an error.

The logic for interpreting the user's move starts at location 2042. If COUNT-FLAG is 0, the user has not yet moved. Subroutine GET-MOVE is repeatedly called from location 204D in anticipation of the user's move.

If the accumulator is 0 upon return from GET-MOVE, then the board position remains unchanged and the user has not made a move. GETKEY is called to see if the user has depressed either the GO or E key. If the E key is depressed, the Microchess routine REVERSE is called to swap the user and computer entries of the board-status table. After the exchange is completed or if the GO key is depressed, a branch is made to START-COUNTING at hexadecimal address 214C.

Three provisions are made for a delayed return back to Microchess. COUNT-FLAG is made nonzero, a countdown is initiated by setting the display to 0F, and control is then transferred back to address 204D where GET-MOVE is repeatedly called as before.

After each return from GET-MOVE the display is decremented by 1 until it equals 0. This provides an approximate 10 second delay during which the user can make a new move or retract an old one. At the end of the countdown, a branch is made to the Microchess routine GO which calculates the computer's next move.

If the GET-MOVE call at 204D returns a negative value then the user has set down a *new* piece, and control is transferred to address 2095. In an ordinary game of chess, putting a new piece on the board would be considered cheating. I have allowed it here to prevent 2 possible problems.

The first problem is caused by indecisive players who change their minds while in the middle of a move. Suppose such a player picks up 2 chessmen, as if to capture, and then decides to set both down again. When the first man is set down the program will think that the user has completed a capture, modify the board-status

Listing 2 continued from page 46

```
2102   A5 27       !                  LDA  CHANGING-SQUARE %FROM SQUARE =
2104   C5 FA       !                  CMP  FA              %%TO SQUARE?
2106   D0 03       !                  IF ZERO THEN
2108   4C 44 20    !                     JMP  WAIT-FOR-MOVE %YES, NO MOVE MADE.
210B               !                  ENDIF
210B   85 B1       !                  STA  .SQUARE         %NO,
210D   20 4B 03    !                  JSR  MOVE            %%RECORD MOVE,
2110   4C 4C 21    !                  JMP  START-COUNTING  %%AND COUNT DOWN.
2113               !               ENDIF
2113               !            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2113               !            %%%%%%%%     2'ND PIECE PICKED UP    %%%%%%%%
2113               !            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2113               !              %*********   WAIT FOR SET DOWN  *********%
2113   A5 27       !               LDA  CHANGING-SQUARE
2115   85 F9       !               STA  F9                 %F9=TO SQUARE
2117   85 B1       !               STA  .SQUARE
2119               !2-UP:          LOOP
2119   A5 33       !                  LDA  TOGGLE          %FLASH LIGHTS FOR TO
211B   10 06       !                  IF NEGATIVE THEN     %%AND FROM SQUARES.
211D   A5 FA       !                     LDA  FA
211F   E6 33       !                     INC  TOGGLE
2121   F0          !                     BEQ
2122   04          !                  ELSE
2123   A5 F9       !                     LDA  F9
2125   C6 33       !                     DEC  TOGGLE
2127               !                  ENDELSE
2127   85 2B       !                  STA  LIGHT-SQUARE
2129   20 6B 21    !                  JSR  GET-MOVE        %WAIT FOR SET DOWN
212C   F0          !                  BEQ
212D   EB          !               ENDLOOP
212E   10 29       !               IF NEGATIVE THEN
2130               !              %*********    PIECE SET DOWN    *********%
2130   A5 27       !                  LDA  CHANGING-SQUARE
2132   C5 F9       !                  CMP  F9              %C-5 = TO SQUARE?
2134   F0 0C       !                  IF NOTZERO THEN      %%NOPE,
2136   C5 FA       !                     CMP  FA           %= FROM SQUARE?
2138   D0 1F       !                     BNE  ERROR-3      %%NOPE, ERROR
213A   A6 F9       !                     LDX  F9
213C   85 F9       !                     STA  F9           %TO & FROM SQUARES
213E   85 B1       !                     STA  .SQUARE      %REVERSED, SWITCH BACK
2140   86 FA       !                     STX  FA
2142               !                  ENDIF
2142   A5 FA       !                  LDA  FA              %SAVE FROM SQUARE FOR
2144   85 2A       !                  STA  FROM SQUARE     %%USE IF UNDOING MOVE
2146   20 9D 01    !                  JSR  DISP            %SET .PIECE
2149   20 4B 03    !                  JSR  MOVE            %RECORD MOVE
214C               !              %********* INITIALIZE COUNT DOWN ********%
214C   A9 00       !START-COUNTING:   LDA# 00              %CLEAR FLAG
214E   85 2F       !                  STA  SHOULDBEUP-FLAG
2150   85 2B       !                  STA  LIGHT-SQUARE
2152   A9 0F       !                  LDA# 0F              %LOAD DELAY
2154   E6 2E       !                  INC  SPEAKER-FLAG    %SOUND OFF
2156   4C 48 20    !                  JMP  SET-COUNT
2159               !               ENDIF
2159   E6 2E       !ERROR-3:       INC  SPEAKER-FLAG       %SOME TYPE OF ERROR,
215B   D0 BC       !               BNE  2-UP               %%WAIT UNTIL CORRECTED
215D               !
215D               !
215D               !
215D               !SUBROUTINE  CLEAR-STACK:          %RESETS BOTH STACK MARKERS
215D   68 A8 68    !               PLA  TAY  PLA           %STORE RETURN ADDRESS
2160   A2 FF 9A    !               LDX# FF   TXS           %RESET MACHINE STACK
2163   A2 C8       !               LDX# C8                 %RESET CHESS STACK
2165   86 B2       !               STX  .SP2
2167   48 98 48    !               PHA  TYA  PHA           %SET UP RETURN
216A   60          !               RTS
216B               !
216B               !
216B               !
216B               !SUBROUTINE  GET-MOVE:            %SCANS CHESSBOARD
216B               !        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
216B               !        %%%%%%%%%%%     CHECK FOR "DA" OR "PC"    %%%%%%%%%%
216B               !        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
216B   20 6A 1F    !               JSR  GETKEY
216E   C9 11       !               CMP# 11                 %11="DA" KEY
2170   D0 0B       !               IF ZERO THEN
2172   A2 1F       !                  LDX# 1F              %FOUND "DELETE ALL"
```

Listing 2 continued on page 50

table accordingly, and proceed to countdown. During the countdown the second man would pop in and there would be no way to know what it was.

In order to prevent this, each capturing move is saved in the Microchess stack. When a new piece is set down, the stack pointer is checked to see if the previous move was a capture. If it was, and if the location of the new piece corresponds to the square where the capturing piece used to be, then the Microchess routine UMOVE is called to restore the board-status table.

The second problem arises when the user wants to add new pieces to the current board, or set up an entirely new board position. Previously the only way to add new pieces was to stop the chess program and enter the square numbers manually into the board-status table using the KIM-1 monitor. This method is both inconvenient and error prone. The control logic for the "new improved" method occupies hexadecimal addresses 20A7 through 20DE.

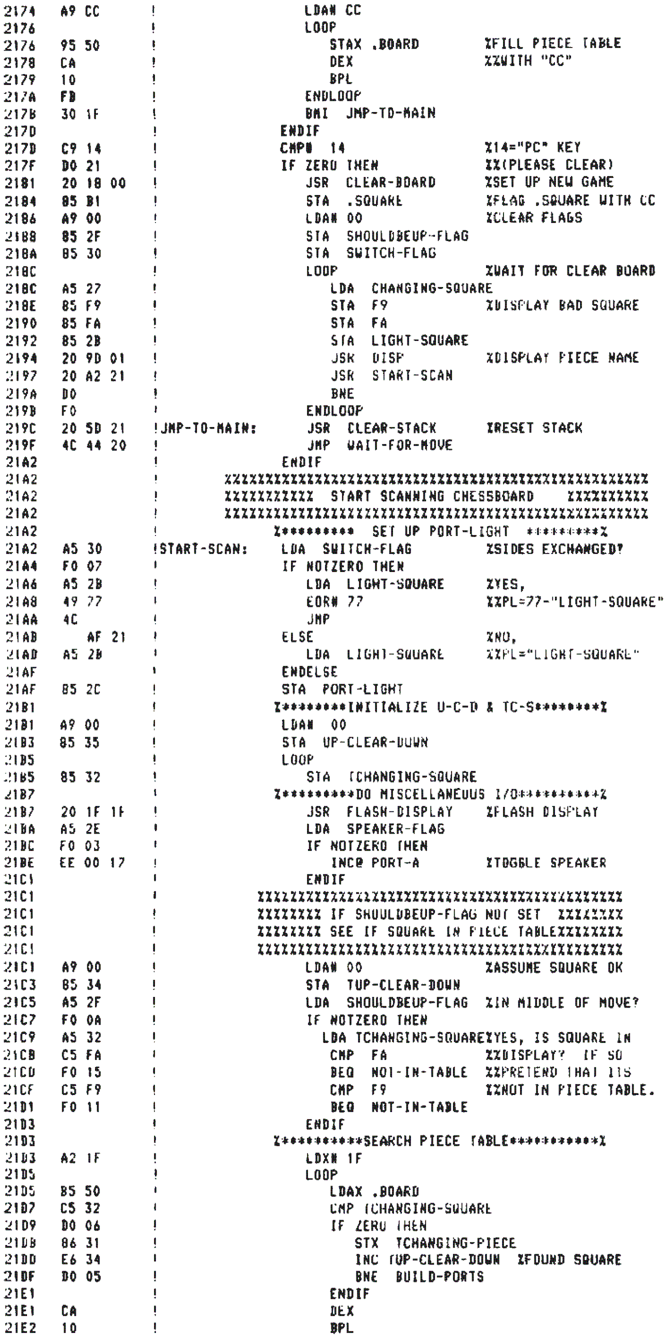
After setting the new pieces down, the user simply types the piece name (its numeric designation) into the hexadecimal keyboard. The designation is displayed in FB, the current board-status table entry in FA, and the square where the new piece was set down is stored in F9. If the current table entry is "CC" (indicating that the piece is not currently on the board), the user may enter the piece into the table by pressing the + key.

Interpreting the User's Move

If the original call to GET-MOVE at hexadecimal address 204D returns a positive value, it means that the user has picked up a piece, and control will transfer to address 20DE. If .SQUARE contains "CC", the Microchess board-status table has just been initialized, and the user is making the first move of a new game. The board-status table has been initialized assuming that the user would play Black. A branch may be made to address 206A where the user and computer table entries are exchanged.

After checking to see whether or not the user is playing White, GET-MOVE is again called at hexadecimal address 20FB. If the piece is set down at a new square, the move has been completed and a countdown is started. If, after picking up a piece a

Listing 2 continued from page 48:



```
2174  A9 CC     !                   LDA# CC
2176            !                   LOOP
2176  95 50     !                      STAX .BOARD          %FILL PIECE TABLE
2178  CA        !                      DEX                  %%WITH "CC"
2179  10        !                      BPL
217A  FB        !                   ENDLOOP
217B  30 1F     !                   BMI  JMP-TO-MAIN
217D            !                ENDIF
217D  C9 14     !                CMP#  14                   %14="PC" KEY
217F  D0 21     !                IF ZERO THEN               %%(PLEASE CLEAR)
2181  20 18 00  !                   JSR  CLEAR-BOARD        %SET UP NEW GAME
2184  85 B1     !                   STA  .SQUARE            %FLAG .SQUARE WITH CC
2186  A9 00     !                   LDA# 00                 %CLEAR FLAGS
2188  85 2F     !                   STA  SHOULDBEUP-FLAG
218A  85 30     !                   STA  SWITCH-FLAG
218C            !                   LOOP                    %WAIT FOR CLEAR BOARD
218C  A5 27     !                      LDA  CHANGING-SQUARE
218E  85 F9     !                      STA  F9              %DISPLAY BAD SQUARE
2190  85 FA     !                      STA  FA
2192  85 2B     !                      STA  LIGHT-SQUARE
2194  20 9D 01  !                      JSR  DISP            %DISPLAY PIECE NAME
2197  20 A2 21  !                      JSR  START-SCAN
219A  D0        !                      BNE
219B  F0        !                   ENDLOOP
219C  20 5D 21  !JMP-TO-MAIN:       JSR  CLEAR-STACK        %RESET STACK
219F  4C 44 20  !                   JMP  WAIT-FOR-MOVE
21A2            !                ENDIF
21A2            !          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21A2            !          %%%%%%%%%%%  START SCANNING CHESSBOARD    %%%%%%%%%%
21A2            !          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21A2            !               %*********  SET UP PORT-LIGHT  *********%
21A2  A5 30     !START-SCAN:    LDA  SWITCH-FLAG           %SIDES EXCHANGED?
21A4  F0 07     !                IF NOTZERO THEN
21A6  A5 2B     !                   LDA  LIGHT-SQUARE       %YES,
21A8  49 77     !                   EOR# 77                 %%PL=77-"LIGHT-SQUARE"
21AA  4C        !                   JMP
21AB     AF 21  !                ELSE                       %NO,
21AD  A5 2B     !                   LDA  LIGHT-SQUARE       %%PL="LIGHT-SQUARE"
21AF            !                ENDELSE
21AF  85 2C     !                STA  PORT-LIGHT
21B1            !               %********INITIALIZE U-C-D & TC-S********%
21B1  A9 00     !                LDA#  00
21B3  85 35     !                STA  UP-CLEAR-DOWN
21B5            !                LOOP
21B5  85 32     !                   STA  TCHANGING-SQUARE
21B7            !               %*********DO MISCELLANEOUS I/O**********%
21B7  20 1F 1F  !                   JSR  FLASH-DISPLAY      %FLASH DISPLAY
21BA  A5 2E     !                   LDA  SPEAKER-FLAG
21BC  F0 03     !                   IF NOTZERO THEN
21BE  EE 00 17  !                      INC@ PORT-A          %TOGGLE SPEAKER
21C1            !                   ENDIF
21C1            !              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21C1            !              %%%%%%%% IF SHOULDBEUP-FLAG NOT SET  %%%%%%%
21C1            !              %%%%%%%% SEE IF SQUARE IN PIECE TABLE%%%%%%%%
21C1            !              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21C1  A9 00     !                   LDA# 00                 %ASSUME SQUARE OK
21C3  85 34     !                   STA  TUP-CLEAR-DOWN
21C5  A5 2F     !                   LDA  SHOULDBEUP-FLAG  %IN MIDDLE OF MOVE?
21C7  F0 0A     !                   IF NOTZERO THEN
21C9  A5 32     !                     LDA TCHANGING-SQUARE%YES, IS SQUARE IN
21CB  C5 FA     !                      CMP  FA              %%DISPLAY?  IF SO
21CD  F0 15     !                      BEQ  NOT-IN-TABLE  %%PRETEND THAT ITS
21CF  C5 F9     !                      CMP  F9              %%NOT IN PIECE TABLE.
21D1  F0 11     !                      BEQ  NOT-IN-TABLE
21D3            !                   ENDIF
21D3            !               %**********SEARCH PIECE TABLE***********%
21D3  A2 1F     !                   LDX# 1F
21D5            !                   LOOP
21D5  B5 50     !                      LDAX .BOARD
21D7  C5 32     !                      CMP TCHANGING-SQUARE
21D9  D0 06     !                      IF ZERO THEN
21DB  86 31     !                         STX  TCHANGING-PIECE
21DD  E6 34     !                         INC TUP-CLEAR-DOWN  %FOUND SQUARE
21DF  D0 05     !                         BNE  BUILD-PORTS
21E1            !                      ENDIF
21E1  CA        !                      DEX
21E2  10        !                      BPL
```

Listing 2 continued on page 52

player decides to set it down on the same square, the move is ignored.

If the GET-MOVE call at location 20FB reports that a second piece has been picked up, a capture is in progress and control branches to location 2113. FROM-SQUARE is defined as the square from which the first chessman is picked up. Similarly, TO-SQUARE is associated with the chessman that is picked up second. GET-MOVE is again called at hexadecimal address 2129.

If a piece is set down on either the TO or FROM squares then the program assumes that a capture has been made. The Microchess routine MOVE is called to modify the board-status table, and a countdown is initiated.

If a piece is set down on a square other than the FROM or TO square, or if a third piece is picked up, a branch will be made to hexadecimal address 2159, and the speaker will hum to indicate an error.

Using the System

Playing the chessboard-interfaced version of Microchess is easy. Moves are made by physically picking up the pieces and setting them down on a new square, as in a normal game of chess with a human opponent. The only difference is that the opponent (the KIM-1) is unable to pick up a chessman, so you have to move the pieces to the location indicated by the LEDs.

The KIM display will be all 0s and the LEDs will blink from square to square in a semirandom fashion when it is your turn to move. After you move, the KIM display will countdown from 0F, and the Y axis LEDs will blink sequentially from the top to the bottom of the board. During this countdown you have the option to change your move. When the display reaches 0, the machine will begin computing a response, and no moves can be made until it is your turn again.

Operating the System

The interfaced version of Microchess is started at address 0000, just as the unmodified Microchess. The speaker will probably hum. To start a new game, press the **PC** key. The speaker's sound will cease. Choose the White or Black pieces, and set up the board with your choice

Listing 2 continued from page 50.

```
21E3   F1          !                ENDLOOP
21E4   C6 34       !NOT-IN-TABLE:   DEC  TOP-CLEAR-DOWN  %SQUARE NOT IN .BOARD
21E6               !          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21E6               !          %%%%%%%%%       BUILD I/O PORTS     %%%%%%%%
21E6               !          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21E6               !            %**********SET UP PORT-SQUARE***********%
21E6   A5 30       !BUILD-PORTS:    LDA  SWITCH-FLAG     %SIDES EXCHANGED?
21E8   F0 07       !                IF NOTZERO THEN
21EA   A5 32       !                   LDA  TCHANGING-SQUARE  %YES, SET PS TO
21EC   49 77       !                   EORN 77                %%77-"TCH-SQUARE"
21EE   4C          !                   JMP
21EF      F3 21    !                ELSE                      %NO,
21F1   A5 32       !                   LDA  TCHANGING-SQUARE  %%PS="TCH-SQUARE"
21F3               !                ENDELSE
21F3   85 2D       !                STA  PORT-SQUARE
21F5               !            %****************PORT-A*****************%
21F5   A5 2C       !                LDA  PORT-LIGHT
21F7   30 04       !                IF POSITIVE THEN
21F9   0A 0A       !                    ASL  ASL         %NEGATIVE=X AXIS,
21FB   0A 0A       !                    ASL  ASL         %POSITIVE=Y AXIS.
21FD               !                ENDIF
21FD   29 70       !                ANDN 70
21FF   EE 02 17    !                INC@ PORT-B          %DISABLE LEDS
2202   0A          !                ASL                  %DON T TOGGLE SPEAKER
2203   85 F3       !                STA  TEMP
2205   A9 01       !                LDAN 01
2207   2D 00 17    !                AND@ PORT-A
220A   05 F3       !                ORA  TEMP
220C   8D 00 17    !                STA@ PORT-A          %STORE ALL BUT COLUMN
220F   A5 2D       !                LDA  PORT-SQUARE
2211   29 40       !                ANDN 40
2213   4A 4A 4A    !                LSR  LSR  LSR
2216   05 2D       !                ORA  PORT-SQUARE
2218   29 0F       !                AND# 0F
221A   0A          !                ASL
221B   0D 00 17    !                ORA@ PORT-A
```

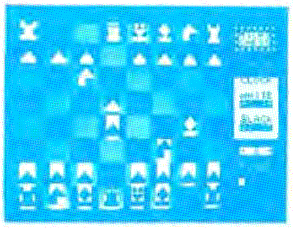
Listing 2 continued on page 54

of color placed closest to the bottom X axis LEDs. After the chessmen are in place, the display will show all 0s. If you are playing White, make your opening move. If the computer is playing White, press the GO key.

To set up the pieces in a new configuration, or to continue a game that was halted earlier, set the chessboard up with the chessman in their desired position. Start the chess program as described above, but instead of pressing the PC key, press the DA key. Type in the name of each piece using the hexadecimal keyboard as you would when adding a new piece. Start the play by either making a move or by pressing the GO key.

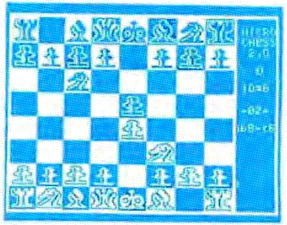
To add a new piece to the board, set the piece on the desired square. The KIM-1 display will show 3 bytes of information. The first byte will be a random piece designation (as described on page 3 of the Microchess player's manual). The second byte is the square that the piece is on, according to the Microchess board-status table. If the piece has been captured, "CC" will be displayed. The third byte is the number of the square

PET / TRS-80 / APPLE: Personal Software brings you the finest!

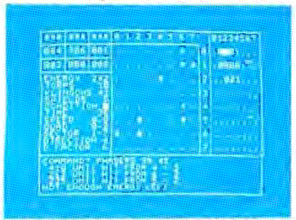


MICRO CHESS

The Industry's First Gold Cassette Over 50,000 Sold



MICROCHESS is the industry's best selling computer game. And no wonder—because MICROCHESS gives you more than just a chessplaying program: A convenient, foolproof set of commands and error checks ... complete instructions in a 5½" by 8½" booklet ... a cassette that's guaranteed to load, with disk versions coming soon ... and several levels of difficulty to challenge you not just once, but time after time. It's available through well over three hundred computer stores and many mail order sources ... always originating from Personal Software. What's more, *every* Personal Software product is selected to give you these same benefits of easy availability, reliable cassettes, readable documentation, a carefully thought out user interface ... and most important, continuing challenge and enjoyment, not just once but time after time. If you haven't already, order your own gold cassette: MICROCHESS, by **Peter Jennings**, for 8K **PET**s, 16K **APPLE**s, and 4K Level I and II **TRS-80s** **$19.95**



TIME TREK

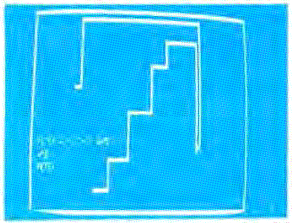
A Tour De Force In Real Time Action Strategy Games

ENTERPRISE HIT

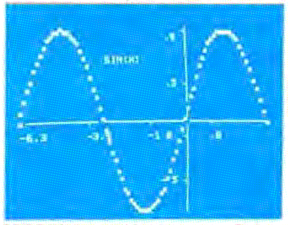
THE ENTERPRISE HAS BEEN DESTROYED
THE FEDERATION WILL BE CONQUERED   YOUR SCORE IS 0

CARE TO PLAY AGAIN?

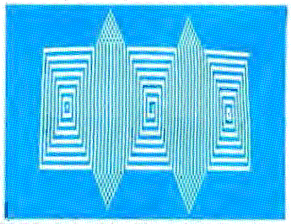
TIME TREK by **Brad Templeton** for 8K **PET**s and **Joshua Lavinsky** for 4K Level I and II **TRS-80s** adds a dramatic new dimension to the classic Star Trek type strategy game: REAL TIME ACTION! You'll need fast reflexes as well as sharp wits to win in this constantly changing game. Be prepared—the Klingons will fire at you as you move, and will move themselves at the same time, even from quadrant to quadrant—but with practice you can change course and speed, aim and fire in one smooth motion, as fast as you can press the keys. Steer under power around obstacles—evade enemy shots as they come towards you—lower your shields just long enough to fire your phasers, betting that you can get them back up in time! With nine levels of difficulty, this challenging game is easy to learn, yet takes most users months of play to master. ADD SOUND EFFECTS with a simple two-wire hookup to any audio amplifier; the TRS-80 also produces sound effects directly through the keyboard case, to accompany spectacular graphics explosions! You won't want to miss this memorable version of a favorite computer game **$14.95**



BLOCKADE by **Ken Anderson** for 4K Level I and II **TRS-80s** is a real time action game for two players, with high speed graphics in machine language. Each player uses four keys to control the direction of a moving wall. Try to force your opponent into a collision without running into a wall yourself! A strategy game at lower speeds, BLOCKADE turns into a tense game of reflexes and coordination at faster rates. Play on a flat or spherical course at any of ten different speeds. You can hear SOUND EFFECTS through a nearby AM radio—expect some razzing if you lose! **14.95**



GRAPHICS PACKAGE by **Dan Fylstra** for 8K **PET**s includes programs for the most common 'practical' graphics applications: PLOTTER graphs both functions and data to a resolution of 80 by 50 points, with automatic scaling and labeling of the axes; BARPLOT produces horizontal and vertical, segmented and labeled bar graphs; LETTER displays messages in large block letters, using any alphanumeric or special character on the PET keyboard; and DOODLER can be used to create arbitrary screen patterns and save them on cassette or in a BASIC program **$14.95**



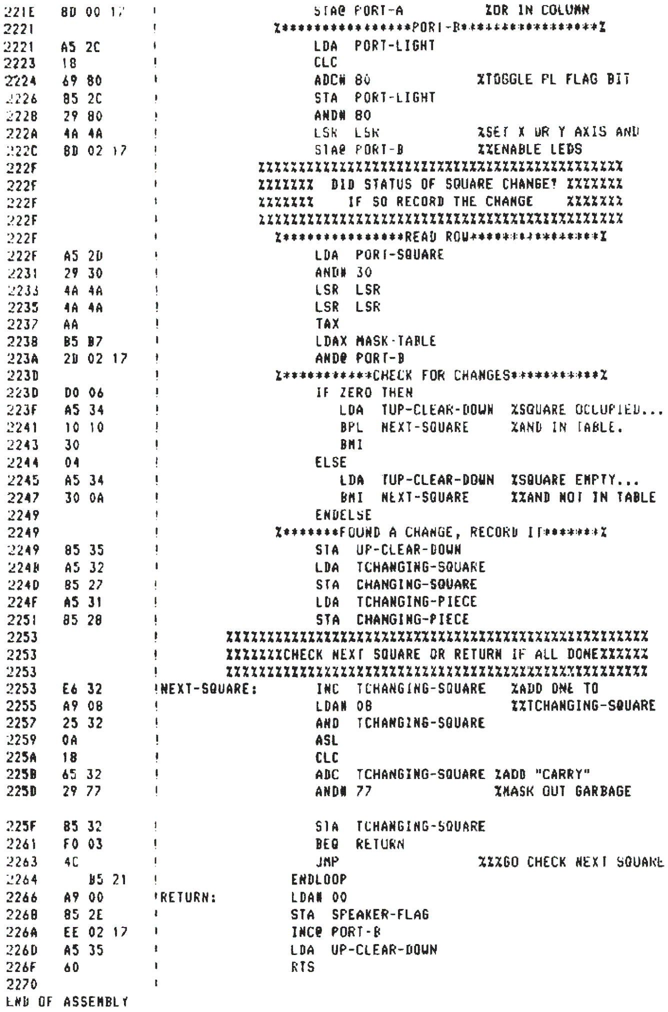
ELECTRIC PAINTBRUSH by **Ken Anderson** for 4K Level I and II **TRS-80s**: Create dazzling real time graphics displays at speeds far beyond BASIC, by writing 'programs' consisting of simple graphics commands for a machine language interpreter. Commands let you draw lines, turn corners, change white to black, repeat previous steps, or call other programs. The ELECTRIC PAINTBRUSH manual shows you how to create a variety of fascinating artistic patterns including the one pictured. Show your friends some special effects they've never seen on a TV screen! **$14.95**

WHERE TO GET IT: *Look for the* ***Personal Software***™ *display rack at your local computer store. For the name and address of the dealer nearest you, call Personal Software at (408) 745-7841. If you don't have a dealer nearby, you can call or mail us your order with your check, money order or VISA/Master Charge card number. For a free catalog, ask your dealer or use the reader service card at the back of this magazine.*

PERSONAL SOFTWARE INC.
592 WEDDELL DR. • SUNNYVALE, CA 94086

Circle 302 on inquiry card.

Listing 2 continued from page 52:



```
221E    8D 00 17   !                 STA@ PORT-A          %OR IN COLUMN
2221               !             %****************PORT-B*****************%
2221    A5 2C      !                 LDA  PORT-LIGHT
2223    18         !                 CLC
2224    69 80      !                 ADC# 80              %TOGGLE PL FLAG BIT
2226    85 2C      !                 STA  PORT-LIGHT
2228    29 80      !                 AND# 80
222A    4A 4A      !                 LSR  LSR             %SET X OR Y AXIS AND
222C    8D 02 17   !                 STA@ PORT-B          %%ENABLE LEDS
222F               !           %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
222F               !           %%%%%%%  DID STATUS OF SQUARE CHANGE? %%%%%%%
222F               !           %%%%%%%    IF SO RECORD THE CHANGE    %%%%%%%
222F               !           %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
222F               !             %****************READ ROW*****************%
222F    A5 2D      !                 LDA  PORT-SQUARE
2231    29 30      !                 AND# 30
2233    4A 4A      !                 LSR  LSR
2235    4A 4A      !                 LSR  LSR
2237    AA         !                 TAX
2238    B5 B7      !                 LDAX MASK-TABLE
223A    2D 02 17   !                 AND@ PORT-B
223D               !             %***********CHECK FOR CHANGES***********%
223D    D0 06      !                 IF ZERO THEN
223F    A5 34      !                    LDA  TUP-CLEAR-DOWN  %SQUARE OCCUPIED...
2241    10 10      !                    BPL  NEXT-SQUARE     %AND IN TABLE.
2243    30         !                    BMI
2244    04         !                 ELSE
2245    A5 34      !                    LDA  TUP-CLEAR-DOWN  %SQUARE EMPTY...
2247    30 0A      !                    BMI  NEXT-SQUARE     %%AND NOT IN TABLE
2249               !                 ENDELSE
2249               !             %*******FOUND A CHANGE, RECORD IT*******%
2249    85 35      !                 STA  UP-CLEAR-DOWN
224B    A5 32      !                 LDA  TCHANGING-SQUARE
224D    85 27      !                 STA  CHANGING-SQUARE
224F    A5 31      !                 LDA  TCHANGING-PIECE
2251    85 28      !                 STA  CHANGING-PIECE
2253               !        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2253               !        %%%%%%%CHECK NEXT SQUARE OR RETURN IF ALL DONE%%%%%%
2253               !        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2253    E6 32      !NEXT-SQUARE:     INC  TCHANGING-SQUARE   %ADD ONE TO
2255    A9 08      !                 LDA# 08                 %%TCHANGING-SQUARE
2257    25 32      !                 AND  TCHANGING-SQUARE
2259    0A         !                 ASL
225A    18         !                 CLC
225B    65 32      !                 ADC  TCHANGING-SQUARE %ADD "CARRY"
225D    29 77      !                 AND# 77                 %MASK OUT GARBAGE

225F    85 32      !                 STA  TCHANGING-SQUARE
2261    F0 03      !                 BEQ  RETURN
2263    4C         !                 JMP                   %%%GO CHECK NEXT SQUARE
2264       B5 21   !              ENDLOOP
2266    A9 00      !RETURN:       LDA# 00
2268    85 2E      !              STA  SPEAKER-FLAG
226A    EE 02 17   !              INC@ PORT-B
226D    A5 35      !              LDA  UP-CLEAR-DOWN
226F    60         !              RTS
2270               !
END OF ASSEMBLY
```

upon which the new piece was set down. Modify the first byte by typing in the correct name of the new piece. If the piece has been previously captured, it may be added to the piece table by typing the + key.

To change sides (Black to White, or vice versa), type the E key. A countdown will be initiated. Do not change sides before the opening move of the game; the King, Queen, and other pieces could become incorrectly reversed.

Conclusion

Although it may require a lot of solder, building the hardware is neither hard nor exacting work. As with most projects, if it doesn't work the first time the problem can usually be traced to an incorrect program, faulty wiring, or bad integrated circuits. In this particular project, the program is already written, the wiring is easy to check, and there are only 2 integrated circuits.

The electronic chessboard can, of course, be used for activities other than chess. Almost any game that is played with an X,Y type grid can be played by the computer, among these: checkers, tic-tac-toe, and nim.

I have found that the chessboard interface makes playing chess with the KIM-1 much more enjoyable. Even if you lose the chess game, the method of playing is sure to be impressive. ■

Editor's Note

The program described in this article was designed to be "foolproof" for the beginning chess player. The countdown period for changing a move will greatly ease the frustration often experienced by players of computer games, the sinking feeling of "Oh no, I didn't mean that, and there's no way to take back the move!" More programmers should pay such attention to the user interface of their systems.

More experienced chess players generally abide by the following rule: a piece once touched by the player must *be moved, and an opponent's piece once touched* must *be captured. Such users would probably wish to delete the countdown period to speed the progress of the game.*

*An electronic chessboard operating in a similar fashion appeared in the article "Chess 4.7 versus David Levy" by J R Douglas (December 1978 BYTE, page 84). That board, constructed by Dr David Cahlander of Control Data Corp, uses 1 light emitting diode (LED) in each square of the chessboard to indicate the computer's move, and uses magnetic switches placed under the squares which are activated by the metal weights in the pieces. Controlled by a 6800 microprocessor, Cahlander's board transmits and receives moves to and from a remote computer on which the Chess 4.7 program runs...*RSS