MICROCHESS

A CHESS PLAYING PROGRAM

FOR THE 6502 MICROCOMPUTER

BY PETER JENNINGS
MICROCHESS

MICROCHESS was originally conceived as a program which would play chess using only a minimum hobbyist microcomputer system. The program designed will run on a KIM-1, 6502 based system, using only 1.1 Kbytes of RAM. Elimination of some unnecessary features would even allow an implementation in less than 1K.

Although MICROCHESS does not play an expert level of chess, it will play a reasonable game in most instances. In addition, it can provide a useful opponent for practising checkmates, learning openings, and sharpening general playing skills.

The program has been carefully designed to allow the average user to expand or modify the basic package to suit the requirements of his particular system configuration, or to experiment with his own ideas for improvement of the playing strategy.

User documentation supplied with the MICROCHESS program consists of a Player's Manual, a complete source program listing, and a Programmer's Manual, which explains the operation of the program and includes suggestions for expansion and modifications.

© This copy of the MICROCHESS program and documentation is provided for the personal use and enjoyment of the purchaser. Reproduction by any means is prohibited. Use of the MICROCHESS programs, or any part thereof, for the purpose of promotion or sale of microcomputer hardware or software, without the express written permission of the author is prohibited. Address all communications to:

PETER JENNINGS,
MICROCHESS,
1612-43 THORNCLIFFE Pk., DR.,
TORONTO, ONTARIO,
M4H 1J4, CANADA.
# TABLE OF CONTENTS

**PLAYER'S MANUAL**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOADING THE PROGRAMS</td>
<td>1</td>
</tr>
<tr>
<td>MICROCHESS NOTATION</td>
<td>1</td>
</tr>
<tr>
<td>MICROCHESS COMMAND KEYS</td>
<td>2</td>
</tr>
<tr>
<td>THE COMPUTER'S MOVE</td>
<td>3</td>
</tr>
<tr>
<td>TABLE OF PIECE CODES</td>
<td>3</td>
</tr>
<tr>
<td>ENTERING YOUR MOVE</td>
<td>4</td>
</tr>
<tr>
<td>SPECIAL MOVES</td>
<td>5</td>
</tr>
<tr>
<td>LEVEL OF PLAY</td>
<td>6</td>
</tr>
<tr>
<td>POSITION VERIFICATION</td>
<td>6</td>
</tr>
<tr>
<td>MEMORY LOCATIONS FOR THE PIECES</td>
<td>7</td>
</tr>
<tr>
<td>NOTES</td>
<td>8</td>
</tr>
</tbody>
</table>

**PROGRAMMER'S MANUAL**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>SOURCE LISTING</td>
<td>1</td>
</tr>
<tr>
<td>SUBROUTINES GNM AND JANUS</td>
<td>1</td>
</tr>
<tr>
<td>OPERATION OF SUBROUTINE JANUS</td>
<td>2</td>
</tr>
<tr>
<td>PROGRAM FUNCTION FOR EACH VALUE OF STATE</td>
<td>3</td>
</tr>
<tr>
<td>STRATEGY OPERATION</td>
<td>4</td>
</tr>
<tr>
<td>OPENING PLAY</td>
<td>5</td>
</tr>
<tr>
<td>MODIFYING THE INPUT AND OUTPUT</td>
<td>6</td>
</tr>
<tr>
<td>EXPANDED INPUT AND OUTPUT ROUTINES</td>
<td>7</td>
</tr>
<tr>
<td>SPECIAL MOVES</td>
<td>8</td>
</tr>
<tr>
<td>STRATEGY IMPROVEMENTS</td>
<td>8</td>
</tr>
<tr>
<td>DATA FOR OPENINGS</td>
<td>10</td>
</tr>
<tr>
<td>EXPLANATION OF SYMBOLS</td>
<td>11</td>
</tr>
<tr>
<td>MICROCHESS HEX LISTING</td>
<td>12</td>
</tr>
</tbody>
</table>

**SOURCE LISTING**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICROCHESS</td>
<td>1</td>
</tr>
<tr>
<td>SYMBOL TABLE AND CROSS REFERENCES</td>
<td>13</td>
</tr>
<tr>
<td>BLOCK DATA</td>
<td>15</td>
</tr>
</tbody>
</table>
MICROCHESS

PLAYER'S MANUAL

MICROCHESS was designed to play a game of chess using the KIM-1 microcomputer system with no additional memory or peripherals. The human player's moves are entered on the self contained keyboard and the computer's responses are flashed on the LED display. Slight program alterations will permit the user to run the program using a teletype, CRT terminal, or another 6502 based system, (see the Programmer's Manual for details). All references in this manual assume that the KIM keyboard and display are being used.

LOADING THE PROGRAMS

Since the KIM-1 memory is divided into two non-contiguous segments, the program must be loaded in two sections. The first section will contain the program and data for the lower 1K of available memory between addresses 0000 and 03FF. The second section will contain the program segment between locations 1780 and 17B6. In addition, short program loaders may be used to enter the data necessary to use different "canned openings", which are stored between 00C0 and 00DB. Since sections of program reside in page one, which is normally reserved for the program stack, it is advisable to reset the stack pointer using the [RS] key before each load. In addition, it is prudent to check locations 0100 and 0101 before executing the program to ensure that they have not been inadvertently altered.

MICROCHESS NOTATION

In order to keep memory requirements to a minimum, (an absolute necessity when programming chess in the 1K environment of the KIM-1), it has been necessary to use a special octal chess notation. Each square on the chess board is uniquely identified by a two digit octal number as shown below. The first digit specifies the rank (0 to 7) from the computer's end of the board. The second digit specifies the file (0 to 7) from the player's left. Moves are specified uniquely by the FROM square and the TO square using this notation.
## MICROCHESS COMMAND KEYS

The following keys are used as commands while playing chess with the MICROCHESS program.

**[GO]** This key is depressed immediately after loading the tape in order to start the program execution, or to restart the program after a temporary exit. No change occurs in the display after the [GO] key has been depressed. After execution begins the key has no effect on the system at all.

**[ST]** This key is used to leave the MICROCHESS program and enter the KIM monitor in order to examine or change memory contents while playing a game. Under no circumstances should this key be pressed when the computer is contemplating its move. Only when the system is displaying a move is it permissible to press the [ST] key.

**[C]** This key CLEARS the internal chessboard and resets it to begin another game. The board is set up with the computer playing white. CCCCCC is displayed to indicate that the board has been reset.
This key EXCHANGES the computer's men with your men. The actual position of the board is unchanged. If [C] is pressed, followed immediately by [E], the board will be set up to begin a game with the computer playing black. By pressing [PC] followed by [E] followed by [PC] . . . the computer will play a game against itself, displaying the moves as it goes. EEEESEE is displayed immediately after the [E] key is pressed to verify operation.

This key is used to move the piece on the FROM square to the TO square to register the player's move, or to move one of the computer's men if desired.

This key instructs the computer to PLAY CHESS. The computer analyses the current position and formulates its optimum move. The display will darken and flash until the move has been decided. When it relights the move is displayed.

THE COMPUTER'S MOVE

The computer moves are displayed in the format shown below:

[piece|FROM square|TO square]

[piece] The piece which the computer is indicating that it wishes to move is encoded according to the table below:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>KING</td>
<td>1</td>
<td>Queen</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>King Bishop</td>
<td>5</td>
<td>Queen Bishop</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>K R Pawn</td>
<td>9</td>
<td>Q R Pawn</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>K B Pawn</td>
<td>D</td>
<td>Q B Pawn</td>
<td>E</td>
</tr>
<tr>
<td>B</td>
<td>Q N Pawn</td>
<td>F</td>
<td>K Pawn</td>
<td></td>
</tr>
</tbody>
</table>

[FROM square] The FROM and TO squares are indicated using the micronotation shown above.

For example the display [0F 13 33] indicates that the King Pawn is to be moved from King Pawn 2 to King Pawn 4. (This assumes that the computer is playing white.)
ENTERING YOUR MOVE

Your moves are described to the computer using the same octal notation described above. It is not necessary to enter the type of piece being moved, just the FROM square and TO square locations.

The computer verifies the input by indicating in the left two digits the piece located on the FROM square. The first digit will be 0, 1, or F. 0 indicates that the piece on the FROM square is one of the computer's men. 1 indicates that the piece is one of your men. F indicates that there is no piece on the FROM square.

The second digit indicates the type of piece located on the FROM square using the same hexadecimal code shown above.

If you have made an error in entering your move at this point just continue to press the appropriate keys. The numbers will scroll from right to left until the correct move is displayed.

For example, if you punch 6 3 4 3 and see the display [ 1F 63 43 ], the 1F indicates that the FROM square (63), contains the King Pawn and that you are preparing to move it to the square 43.

When you have entered and verified the move, depress the [F] key to register the move on the internal chess board. The first two digits of the display will be changed to FF to indicate that the FROM square is now unoccupied. If the TO square had been occupied, the previous occupant will have been captured automatically.

You may make as many moves in this manner as you wish, moving either your own men or the computer's. No verification of the legality of the moves is carried out. Illegal moves are accepted and executed as easily as legal moves, so care should be taken that you do not accidentally move in an illegal manner. Since the computer does not make a point of warning you if your king is in check, you must be careful not to leave this situation after your move. The computer will usually take off your king on its subsequent move if this is possible.
SPECIAL MOVES

CASTLING: You may make a castling move by making two moves in succession in the normal manner. First move the king to its new square, then move the rook. Remember to depress [F] after each move. The computer has no provision for castling during the middle game or end game, but may castle during the opening. If this occurs it will indicate a move of the king two squares over. You must complete the move for the computer by moving the rook for it. Just enter the appropriate TO and FROM square followed by [F] to make the move, then, go ahead and make your own move.

EN PASSANT: In order to capture en passant you must break the move into two separate components. First, move your pawn laterally to capture the computer's pawn. Then, move your pawn forward to its appropriate final square. Do not forget to depress [F] after each move to register it internally. Note that the computer cannot capture en passant itself and will not recognize the danger of your en passant captures in considering its double pawn moves.

QUEENING PAWNS: If you should succeed in pushing a pawn to the eighth rank (rank 7 in micronotation), it will be necessary for you to manually set up the queen on that square. Because of the internal representation of the position it is possible only to have one Queen per side at a time. Therefore, if you already have one, you will have to choose a rook, bishop, or knight instead. To replace the pawn with a Queen the following steps should be carried out.

1) Use the [ST] key to exit from the MICROCHESS program and return control to the KIM monitor.

2) Find the pawn using the table of piece locations below. Confirm by its position that it is the correct one. Remove it from the board by entering the data 'CC', which indicates a captured piece.

3) Enter the address of the queen (0061). This memory location should now contain 'CC', assuming the queen has been lost.
4) Press [DA] and enter the new location for the Queen, which is the square the pawn moved to. (e.g. 07)

5) Press [PC] followed by [GO] to reenter the MICROCHESS program. Continue in the normal manner from this point.

If the computer should push a pawn to the eighth rank, it will be necessary for you to replace the pawn with a Queen, or the highest piece available. Use the same procedure as above. The computer's Queen should be stored at address 0051.

LEVEL OF PLAY

There are several sections of the program which can be bypassed in order to reduce the computer's response time in a given situation. This will reduce the quality of play accordingly. The strategy levels and data changes are outlined below.

<table>
<thead>
<tr>
<th>LEVEL</th>
<th>LOCATION 02F2</th>
<th>LOCATION 018B</th>
<th>AVG TIME PER MOVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPER BLITZ</td>
<td>00</td>
<td>FF</td>
<td>3 seconds</td>
</tr>
<tr>
<td>BLITZ</td>
<td>00</td>
<td>FB</td>
<td>10 seconds</td>
</tr>
<tr>
<td>NORMAL</td>
<td>08</td>
<td>FB</td>
<td>100 seconds</td>
</tr>
</tbody>
</table>

POSITION VERIFICATION

Occasionally, while playing a game, you will come to the sudden realization that the computer is seeing a different board setup from the one you have. This results from your misinterpretation of one of its moves, from entering one of your moves incorrectly, or from forgetting to press [F] to register your move.

It is possible in this situation to sneak a peek at the location of each piece as it is internally stored in order to verify its location on the board. To do this, press [ST] to exit the MICROCHESS program and enter the KIM monitor. Then look at the addresses shown below to determine where the computer thinks each piece is. Afterwards, return to the chess program by pressing [PC] followed by [GO].
# MEMORY LOCATIONS FOR THE PIECES

<table>
<thead>
<tr>
<th>COMPUTER PIECES</th>
<th>YOUR PIECES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0050</td>
<td>King</td>
</tr>
<tr>
<td>0051</td>
<td>Queen</td>
</tr>
<tr>
<td>0052</td>
<td>King Rook</td>
</tr>
<tr>
<td>0053</td>
<td>Queen Rook</td>
</tr>
<tr>
<td>0054</td>
<td>King Bishop</td>
</tr>
<tr>
<td>0055</td>
<td>Queen Bishop</td>
</tr>
<tr>
<td>0056</td>
<td>King Knight</td>
</tr>
<tr>
<td>0057</td>
<td>Queen Knight</td>
</tr>
<tr>
<td>0058</td>
<td>K R Pawn</td>
</tr>
<tr>
<td>0059</td>
<td>Q R Pawn</td>
</tr>
<tr>
<td>006A</td>
<td>K N Pawn</td>
</tr>
<tr>
<td>005B</td>
<td>Q N Pawn</td>
</tr>
<tr>
<td>005C</td>
<td>K B Pawn</td>
</tr>
<tr>
<td>005D</td>
<td>Q B Pawn</td>
</tr>
<tr>
<td>005E</td>
<td>Q Pawn</td>
</tr>
<tr>
<td>005F</td>
<td>K Pawn</td>
</tr>
</tbody>
</table>

**IMPORTANT NOTE:**
Never depress the [ST] key while the computer is contemplating its move. Important parameters are stored in the same area of memory used by the KIM monitor programs. Reentry after these locations have been altered will probably destroy the board position.
NOTES

As mentioned above, there are three types of moves which the current version of MICROCHESS does not play. These are castling, en passant pawn captures, and queening of pawns. In order to make the game fair some players adopt one of the two following strategies. Recognizing that the computer cannot make these moves, some players choose not to make them themselves, thus both players suffer the same restrictions. On the other hand, other players have decided to help the computer by watching for appropriate castling or en passant situations and making the moves on the computer's behalf at that time. Of course, you may always play without regard to the computer's disadvantage, allowing it to fend for itself as best it can.

If you are an above average player, you may find that the MICROCHESS program is below your level of play and hence, always loses. You can add to the challenge of the game in the same way that you might against an inexperienced human player. Remove one or more of your pieces at the start of the game and see if you can come back from a position of disadvantage. The easiest way to remove a piece is to move one of the computer's men to the square of the piece you wish to remove, and then move it back to its original square.
M I C R O C H E S S

P R O G R A M M E R ' S M A N U A L

The program can be divided into three basic functional units.

I. Control and Input/Output. This section comprises the initialization routines, the input and output routines, and the main entry into the move generation and evaluation routines.

II. Move Generation and Data Collection. This program group generates the moves available to the computer, one at a time. For each of these moves, data are collected regarding available continuation moves, the threats of possible reply moves, and the gain or loss from subsequent piece exchanges.

III. Strategic Analysis. The data collected by the move generation routines are analysed by a mathematical algorithm which assigns a value to each available move. The move with the highest assigned value will be the move that the computer selects.

SOURCE LISTING

A complete listing of the program is included in source form. The average programmer should be able to use this document as a key to understanding the program's operation, and as a basis for further modifications. The complete cross reference table is included to assist in program relocation. As a convention in the listing, variables are preceded by a period to distinguish them from program labels, and external subroutines are preceded by an asterisk. Comment lines are preceded by a semicolon.

SUBROUTINES GNM AND JANUS

The key to the operation of the MICROCHESS program lies in the two subroutines GNM and JANUS. GNM calculates the available moves for one side with three nested loops: NEWP, which loops through the pieces from the pawns to the king; NEX, which loops through the four to eight directions through which each piece can move using the table MOVEX as pointed to by the move direction pointer MOVEN; and the individual loops for each piece which select the appropriate directions and distances to move.
After each move has been calculated by GNM, the subroutine JANUS is called. JANUS uses the value of STATE to determine which portion of the analysis the computer is working on and directs it to the appropriate continuation routines. As can be seen from the simplified flow chart of JANUS' operation, JANUS often alters the value of STATE and calls the subroutine GNM again. This series of recursive subroutine calls calculates approximately 20,000 moves per second--over 2 million moves in a 100 second analysis. Most of these moves are repetitions generated from a slightly different board position.

PROGRAM FUNCTION FOR EACH VALUE OF .STATE

<table>
<thead>
<tr>
<th>STATE</th>
<th>SET BY</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>GO</td>
<td>Generate all available moves from the current position and analyse as a benchmark with which to compare the real moves, which are generated by STATE 4.</td>
</tr>
<tr>
<td>4</td>
<td>GO</td>
<td>Generate all available moves, evaluating each one and assigning a value to it as a possible selection.</td>
</tr>
<tr>
<td>8</td>
<td>JANUS</td>
<td>Having made one trial move, generate the possible second moves for analysis.</td>
</tr>
<tr>
<td>0</td>
<td>JANUS</td>
<td>Having made one trial move, generate the possible replies for analysis.</td>
</tr>
<tr>
<td>FF</td>
<td>JANUS</td>
<td>Since a reply move was a capture, reverse the board and evaluate the exchange that could result.</td>
</tr>
<tr>
<td>FE</td>
<td>JANUS</td>
<td>Stage two of the exchange evaluation started by STATE FF.</td>
</tr>
<tr>
<td>FD</td>
<td>JANUS</td>
<td>Stage three of the exchange evaluation.</td>
</tr>
<tr>
<td>FC</td>
<td>JANUS</td>
<td>Last stage of the exchange evaluation.</td>
</tr>
<tr>
<td>F9</td>
<td>CHKCHK</td>
<td>Look for a capture of the king which signifies that the move being calculated is illegal.</td>
</tr>
</tbody>
</table>
STRATEGY OPERATION

After each real available move is generated and the various counts have been performed, the following information is available for decision making purposes.

MOB Mobility. The total number of moves available for a given side from a given position. Each queen move is counted as two moves.

MAXC Maximum Capture. The number of points to be gained by capturing the most valuable piece currently under attack.

CC Capture Count. The total points of all opposing pieces under attack.

MAXP Maximum Capturable Piece. Identification of the opponent's piece under attack which is worth the most points.

PRIOR COUNTS (.PMOB, .PMAXC, .PCC, .PMAXP) reflect the status of the position as it exists for the computer before any move is made. This is a benchmark, against which further moves are to be compared.

CONTINUATION COUNTS (.WMOB, .WMAXC, .WCC, .WMAXP) are obtained for each move tested to determine the potential of the new position that would result if the move were made.

REPLY COUNTS (.BMOB, .BMAXC, .BCC, .BMAXP) are obtained for each move tested to determine the potential danger of the opponent's available replies.

EXCHANGE COUNTS (.WCAP0, .WCAP1, .WCAP2, .BCAP0, .BCAP1, .BCAP2) are used to analyse the effect of the potential exchange combinations. Each count reflects the maximum number of points capturable at each level of an exchange combination. Capture chains are halted by pawn captures, king captures, or by reaching a limit of three captures per side.

In addition, information regarding the moving piece and its TO and FROM squares can also be used by the STRATGTY algorithm.

All information available is combined by the algorithm in the subprogram STRATGTY to calculate a single strategic value for the move under analysis. The algorithm, a weighted sum of the count information, is shown below:

\[
VALUE = + 4.00 \cdot WCAP0 \\
+ 1.25 \cdot WCAP1 \\
+ 0.75 \cdot (WMAXC + WCC) \\
+ 0.25 \cdot (WMOB + WCAP2) \\
- 2.50 \cdot BMAXC \\
- 2.00 \cdot BCC \\
- 1.25 \cdot BCAP1 \\
- 0.50 \cdot BMAXC \\
- 0.25 \cdot (PMAXC + PCC + PMOB + BCAPO + BCAP2 + BMOB)
\]
VALUE = VALUE + 02, A position bonus if the move is to the centre or out of the back rank.

VALUE = 00, If the move is illegal because the king is in check.

VALUE = FF, If the move results in a checkmate.

The move with the highest value is selected by the computer as the best move available. This algorithm can easily be modified by changing the weights assigned to the various parameters. For example, the program can be made to play more aggressively by increasing the importance of BMAXC and WCAPC in the equation above. On the other hand, it can be made to play more defensively by increasing the importance of BMAXC in the equation.

Note that the algorithm above has not yet been optimized. Therefore, it may be possible to significantly improve the play of the program by empirical testing to optimize the form and weights used for the equation.

An alternative form of algorithm to the weighted average type above, which also works well, assigns a fixed number of points to the occurrence of certain conditions. For example, the condition WMOB > PMOB may be considered to be worth 3 points regardless of the difference in value between the two variables. Similarly, conditions which are unfavourable would be assigned negative points. This type of strategy can be easily implemented by keeping a running total of the value in the accumulator and using CPX and CPY instructions to control branches around the addition and subtraction routines. In general, more memory is required to implement an equally complex strategy using this type of algorithm, but in the long run this strategy will be more flexible.

OPENING PLAY

The MICROCHESS program is designed in such a way that the opening can be played from memory, following established lines of play for up to nine moves per side. In order to conserve memory, only one opening is actually stored in the computer at a given time. The opening is stored in locations 00000 through 000DB. By storing each of the openings provided on cassette tape with a different ID for each, it is possible to load the desired opening before beginning play. More openings can be added to the repertoire by coding them in the format shown below.

Users with expanded memory can set up all the openings in a set of tables, allowing the program to select the appropriate opening as long as its opponent is following a standard procedure.
The ability to load an opening by name and play it with the computer also provides an excellent method of rehearsing openings for a chessplayer who is attempting to memorize the standard plays.

Each move and expected reply is stored in 3 bytes. The program first checks that the expected reply TO square is the same as the one in the stored opening. If it matches, the piece and the TO square for the computer's move are loaded into the display and moved. For example, the following illustrates the GIUOCO PIANO Opening. The computer is playing white.

Address Data Move
00DB CC Expected display when computer is making its first move.
00DA 0F King pawn.
00D9 33 To KP4.
00D8 43 Expected reply P-KP4.
00D7 06 Knight.
00D6 22 To KB3.
00D5 52 Expected reply: N-QB3.
00D4 04 Bishop.

The last line of the opening sequence must be 99, or any impossible position square, to cause the program to leave the opening routine and enter the normal strategy evaluation routines.

MODIFYING THE INPUT AND OUTPUT ROUTINES

In order to use the MICROCHESS program on 6502 microprocessor systems other than the KIM-1, the only modifications necessary are changes to the input and output subroutine calls. These subroutines appear in the program listing as *OUT and *GETKEY at locations 008, 00B, and 039F.

*OUT is a subroutine in the KIM ROM at location 1F1F which displays, in hexadecimal format, the contents of memory locations 00FB, 00FA, and 00F9 on the 5 digit LED display. 00FB contains the coded piece identification and locations 00FA and 00F9 contain the FROM and TO squares respectively. These three locations are also used to display CCCCCC and EEEEEE as verification of the keyboard input. At address 039F, *OUT is called by CKMATE at the end of the move analysis to flash the display. This call is not necessary for operation of the program and may be eliminated by replacing the JMP instruction at that location with an RTS (60). The MICROCHESS program has been designed so that neither the X and Y registers, nor the accumulator contents need be preserved by a replacement output subroutine.
*GETKEY is a KIM subroutine which returns the value of the depressed key in the accumulator. Hexadecimal values are returned right justified (e.g. 0A). The only non-hex key used is [PC] which returns the value 14. This key is used only once, at location 0033, so is easy to replace with any other value. Once again, the X and Y registers need not be preserved by a replacement input subroutine.

EXPANDED INPUT AND OUTPUT ROUTINES

Users with CRT or teletype terminals and additional memory will probably want to customize the input and output features of the program.

A format which can be used for move entry and move display is shown by the example: N(KN1) - KB3. This format completely expresses the move, and also provides a check value in the piece descriptor. Translation from this notation to the internal octal FROM and TO square notation is easily accomplished with a simple table lookup program which contains the file descriptors and subtracts 01 from the rank value.

The board can be displayed by providing a routine which prints a layout such as the one illustrated below. Before printing each square, the program could search the piece tables to determine if the square is occupied, and by which piece. The table descriptor is then obtained from the same tables used by the I/O routines above. Users with graphic terminals will want to set up even more elaborate board display routines.

<table>
<thead>
<tr>
<th>WR</th>
<th>WN</th>
<th>WB</th>
<th>WK</th>
<th>WQ</th>
<th>WB</th>
<th>WR</th>
</tr>
</thead>
<tbody>
<tr>
<td>WP</td>
<td>WP</td>
<td>WP</td>
<td>WP</td>
<td>WP</td>
<td>WP</td>
<td>WP</td>
</tr>
<tr>
<td>**</td>
<td>**</td>
<td>WN</td>
<td>**</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SPECIAL MOVES

Several types of moves are not included in the basic MICROCHESS program in order to reduce the memory requirements. These moves, casting, en passant capture, and queening of pawns, can be added by expanding and modifying some of the subroutines which generate and execute moves. GNM must be modified to spot the occurrence of situations in which the moves are available. The actual move calculations must be added to CMOVE, and a flag to indicate the nature of the move set to allow MOVE and UMOVE to properly interpret them. The flag could use the two spare bits in .SQUARE. Additional parameters would be required to indicate when casting, or en passant moves are legal during the game, because these moves depend upon previous play for their legality. Expansion of the piece and point tables would allow the program to keep track of more than one queen per side.

STRATEGY IMPROVEMENTS

As you will soon discover when playing against the MICROCHESS program, it has a tendency to make ridiculous moves from time to time. These moves usually result from unusual positions, which point out deficiencies in the way the move value is calculated. A major problem in the analysis is that there is only one strategy which is used for the opening, the middle game, and the end game. This involves a considerable compromise of three different types of play. Users with memory expansion may wish to write three algorithms which can be switched in and out of the analysis at various points during the game.

Similarly, allowing more than 1K of memory enables the user to add more specialized evaluation routines. For example, a separate subroutine could be used to evaluate each of the following situations from both an offensive and defensive viewpoint, enabling a much more sophisticated level of play:
1- King in check. A major flaw in the current program causes the computer to minimize attacks by placing the opponent's king in check, even at the expense of a minor piece— a very short term solution to the problem! 2- En prise capture availability for either side. 3- Pawn development value: isolated pawns, passed pawns, doubled pawns, etc. 4- X-ray analysis: the value of pins, discovered attack threats, etc. 5- Mating strategies: each of the major types of mates. 6- Positional development: utilization of open files, control of the centre, king position, pawn chains, etc.
With the exception of the capture tree, the MICROCHESS program analyses in full only one move for each side beyond the move it will make. It is possible to use the same recursive technique used by TREE to carry out a full analysis to a further depth. To do this would require a routine to analyse and evaluate each intermediate position arrived at. Sequences of possible positions with positive values for computer moves and negative values for opponent's moves can be summed to give the total long term value of each currently available move. In order to be time efficient, this analysis can be performed on a subset of the available continuations selected by a quick static analysis. In addition, a system of 'tree pruning' should be implemented to prevent long excursions down low valued branches. Programmers embarking on this type of program should bear in mind that from an average position with 50 available moves per side, a total of 15,625 billion sequences are generated in three moves per side.

As can be seen, MICROCHESS is only the beginning. However, it does demonstrate the capability of a small scale hobbyist microcomputer system to tackle the game of chess. It is hoped that this program will provide an inspiration and a stepping stone that chess playing programmers will expand and build upon. Let us know what you have done to improve the system. We will attempt to publish or distribute some of your ideas. It is hoped that a tournament of chess playing microcomputers can be arranged at a future microcomputer gathering. Expanded and modified versions of MICROCHESS will then have the opportunity to prove their playing ability against other programs in the same memory utilization class.
DATA FOR OPENINGS

The data below enables the computer to play the opening specified from memory. The data is in a block from 00CO to 00DB. W specifies that the computer will play white, B specifies that the computer is black.

<table>
<thead>
<tr>
<th>ADDR</th>
<th>W</th>
<th>FRENCH DEFENCE</th>
<th>B</th>
<th>W</th>
<th>GIUOCO PIANO</th>
<th>B</th>
<th>W</th>
<th>RUY LOPEZ</th>
<th>B</th>
<th>W</th>
<th>QUEEN'S INDIAN</th>
<th>B</th>
<th>W</th>
<th>FOUR KNIGHTS</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>CC</td>
<td>44</td>
<td>CC</td>
<td>44</td>
<td>CC</td>
<td>44</td>
<td>CC</td>
<td>P-K4</td>
<td>0F</td>
<td>0E</td>
<td>P-K4</td>
<td>43</td>
<td>CC</td>
<td>P-K4</td>
<td>0F</td>
</tr>
<tr>
<td>DA</td>
<td>0F</td>
<td>P-K4</td>
<td>0F</td>
<td>0F</td>
<td>P-K4</td>
<td>0F</td>
<td>0F</td>
<td>P-K4</td>
<td>34</td>
<td>34</td>
<td>P-K4</td>
<td>25</td>
<td>34</td>
<td>P-K4</td>
<td>0F</td>
</tr>
<tr>
<td>D9</td>
<td>33</td>
<td>P-K3</td>
<td>24</td>
<td>33</td>
<td>P-K4</td>
<td>34</td>
<td>33</td>
<td>P-K4</td>
<td>43</td>
<td>43</td>
<td>P-K3</td>
<td>52</td>
<td>43</td>
<td>P-K3</td>
<td>42</td>
</tr>
<tr>
<td>D8</td>
<td>53</td>
<td>43</td>
<td>43</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>N-KB3</td>
<td>0D</td>
<td>0D</td>
<td>P-QB4</td>
<td>0E</td>
<td>0D</td>
<td>P-QB4</td>
<td>0F</td>
</tr>
<tr>
<td>D7</td>
<td>0E</td>
<td>P-Q4</td>
<td>0E</td>
<td>06</td>
<td>N-KB3</td>
<td>07</td>
<td>06</td>
<td>N-KB3</td>
<td>22</td>
<td>22</td>
<td>N-KB3</td>
<td>25</td>
<td>22</td>
<td>N-KB3</td>
<td>0F</td>
</tr>
<tr>
<td>D6</td>
<td>34</td>
<td>P-Q4</td>
<td>33</td>
<td>22</td>
<td>N-QB3</td>
<td>22</td>
<td>22</td>
<td>N-QB3</td>
<td>55</td>
<td>55</td>
<td>N-QB3</td>
<td>22</td>
<td>55</td>
<td>N-QB3</td>
<td>55</td>
</tr>
<tr>
<td>D5</td>
<td>44</td>
<td>52</td>
<td>52</td>
<td>55</td>
<td>42</td>
<td>55</td>
<td>55</td>
<td>0D</td>
<td>25</td>
<td>25</td>
<td>0D</td>
<td>55</td>
<td>25</td>
<td>0D</td>
<td>55</td>
</tr>
<tr>
<td>D4</td>
<td>07</td>
<td>N-QB3</td>
<td>06</td>
<td>04</td>
<td>B-B4</td>
<td>04</td>
<td>04</td>
<td>B-N5</td>
<td>25</td>
<td>25</td>
<td>B-N5</td>
<td>06</td>
<td>25</td>
<td>B-N5</td>
<td>06</td>
</tr>
<tr>
<td>D3</td>
<td>25</td>
<td>N-KB3</td>
<td>25</td>
<td>35</td>
<td>B-B4</td>
<td>32</td>
<td>32</td>
<td>B-N3</td>
<td>75</td>
<td>75</td>
<td>B-N3</td>
<td>31</td>
<td>75</td>
<td>B-N3</td>
<td>31</td>
</tr>
<tr>
<td>D2</td>
<td>52</td>
<td>36</td>
<td>45</td>
<td>52</td>
<td>75</td>
<td>56</td>
<td>56</td>
<td>N-KB3</td>
<td>56</td>
<td>56</td>
<td>N-KB3</td>
<td>31</td>
<td>56</td>
<td>N-KB3</td>
<td>31</td>
</tr>
<tr>
<td>D1</td>
<td>05</td>
<td>B-N5</td>
<td>45</td>
<td>0D</td>
<td>P-B3</td>
<td>06</td>
<td>06</td>
<td>0A</td>
<td>25</td>
<td>25</td>
<td>0A</td>
<td>31</td>
<td>25</td>
<td>0A</td>
<td>31</td>
</tr>
<tr>
<td>D0</td>
<td>41</td>
<td>B-K2</td>
<td>14</td>
<td>25</td>
<td>N-B3</td>
<td>25</td>
<td>25</td>
<td>0A</td>
<td>44</td>
<td>44</td>
<td>0A</td>
<td>21</td>
<td>44</td>
<td>0A</td>
<td>21</td>
</tr>
<tr>
<td>CF</td>
<td>63</td>
<td>52</td>
<td>52</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>0F</td>
<td>25</td>
<td>25</td>
<td>0F</td>
<td>21</td>
<td>25</td>
<td>0F</td>
<td>21</td>
</tr>
<tr>
<td>CE</td>
<td>0E</td>
<td>P-K5</td>
<td>06</td>
<td>0E</td>
<td>P-Q4</td>
<td>0F</td>
<td>0F</td>
<td>P-Q4</td>
<td>66</td>
<td>66</td>
<td>P-Q4</td>
<td>66</td>
<td>66</td>
<td>P-Q4</td>
<td>66</td>
</tr>
<tr>
<td>CD</td>
<td>43</td>
<td>KN-Q2</td>
<td>13</td>
<td>34</td>
<td>PxP</td>
<td>43</td>
<td>43</td>
<td>11</td>
<td>46</td>
<td>46</td>
<td>11</td>
<td>46</td>
<td>46</td>
<td>11</td>
<td>46</td>
</tr>
<tr>
<td>CC</td>
<td>64</td>
<td>14</td>
<td>34</td>
<td>14</td>
<td>63</td>
<td>63</td>
<td>63</td>
<td>11</td>
<td>36</td>
<td>36</td>
<td>11</td>
<td>36</td>
<td>36</td>
<td>11</td>
<td>36</td>
</tr>
<tr>
<td>CB</td>
<td>05</td>
<td>BxP</td>
<td>01</td>
<td>01</td>
<td>P-QK2</td>
<td>06</td>
<td>06</td>
<td>0A</td>
<td>52</td>
<td>52</td>
<td>0A</td>
<td>52</td>
<td>52</td>
<td>0A</td>
<td>52</td>
</tr>
<tr>
<td>CA</td>
<td>63</td>
<td>QxP</td>
<td>14</td>
<td>43</td>
<td>B-N5</td>
<td>25</td>
<td>25</td>
<td>0A</td>
<td>52</td>
<td>52</td>
<td>0A</td>
<td>52</td>
<td>52</td>
<td>0A</td>
<td>52</td>
</tr>
<tr>
<td>C9</td>
<td>63</td>
<td>14</td>
<td>36</td>
<td>14</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>0A</td>
<td>44</td>
<td>44</td>
<td>0A</td>
<td>44</td>
<td>44</td>
<td>0A</td>
<td>44</td>
</tr>
<tr>
<td>C8</td>
<td>01</td>
<td>Q-Q2</td>
<td>00</td>
<td>01</td>
<td>BxP</td>
<td>06</td>
<td>06</td>
<td>0A</td>
<td>25</td>
<td>25</td>
<td>0A</td>
<td>25</td>
<td>25</td>
<td>0A</td>
<td>25</td>
</tr>
<tr>
<td>C7</td>
<td>14</td>
<td>O-O</td>
<td>06</td>
<td>14</td>
<td>N-QK3</td>
<td>44</td>
<td>44</td>
<td>25</td>
<td>0B</td>
<td>0B</td>
<td>25</td>
<td>0B</td>
<td>0B</td>
<td>25</td>
<td>0B</td>
</tr>
<tr>
<td>C6</td>
<td>72</td>
<td>45</td>
<td>33</td>
<td>75</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>N-KB3</td>
<td>52</td>
<td>52</td>
<td>N-KB3</td>
<td>52</td>
<td>52</td>
<td>N-KB3</td>
<td>52</td>
</tr>
<tr>
<td>C5</td>
<td>0E</td>
<td>P-B4</td>
<td>0D</td>
<td>00</td>
<td>04</td>
<td>06</td>
<td>06</td>
<td>0A</td>
<td>52</td>
<td>52</td>
<td>0A</td>
<td>52</td>
<td>52</td>
<td>0A</td>
<td>52</td>
</tr>
<tr>
<td>C4</td>
<td>32</td>
<td>0E</td>
<td>32</td>
<td>01</td>
<td>NxN</td>
<td>52</td>
<td>52</td>
<td>0A</td>
<td>25</td>
<td>25</td>
<td>0A</td>
<td>25</td>
<td>25</td>
<td>0A</td>
<td>25</td>
</tr>
<tr>
<td>C3</td>
<td>45</td>
<td>0E</td>
<td>06</td>
<td>08</td>
<td>BxP</td>
<td>04</td>
<td>04</td>
<td>25</td>
<td>0A</td>
<td>0A</td>
<td>25</td>
<td>0A</td>
<td>0A</td>
<td>25</td>
<td>0A</td>
</tr>
<tr>
<td>C2</td>
<td>0E</td>
<td>P-QB4</td>
<td>25</td>
<td>25</td>
<td>0E</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>0A</td>
<td>0A</td>
<td>25</td>
<td>0A</td>
<td>0A</td>
<td>25</td>
<td>0A</td>
</tr>
<tr>
<td>C1</td>
<td>22</td>
<td>0E</td>
<td>22</td>
<td>25</td>
<td>O-O</td>
<td>52</td>
<td>52</td>
<td>25</td>
<td>0A</td>
<td>0A</td>
<td>25</td>
<td>0A</td>
<td>0A</td>
<td>25</td>
<td>0A</td>
</tr>
</tbody>
</table>
EXPLANATION OF SYMBOLS

<table>
<thead>
<tr>
<th>ADDR</th>
<th>SYMBOL</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0050</td>
<td>.BOARD</td>
<td>LOCATION OF PIECES</td>
</tr>
<tr>
<td>0060</td>
<td>.BK</td>
<td>OPPONENT'S PIECES</td>
</tr>
<tr>
<td>0070</td>
<td>.SETW</td>
<td>INITIAL PIECE LOCATIONS</td>
</tr>
<tr>
<td>0080</td>
<td>.MOVEX</td>
<td>TABLE OF MOVE DIRECTIONS</td>
</tr>
<tr>
<td>0090</td>
<td>.POINTS</td>
<td>TABLE OF PIECE VALUES</td>
</tr>
<tr>
<td>00A0</td>
<td>.PIECE</td>
<td>CURRENT PIECE UNDER ANALYSIS</td>
</tr>
<tr>
<td>00B0</td>
<td>.SQUARE</td>
<td>TO SQUARE OF .PIECE</td>
</tr>
<tr>
<td>00B2</td>
<td>.SP2</td>
<td>STACK POINTER FOR STACK 2</td>
</tr>
<tr>
<td>00B3</td>
<td>.SP1</td>
<td>STACK POINTER FOR STACK 1</td>
</tr>
<tr>
<td>00B4</td>
<td>.INCHECK</td>
<td>MOVE INTO CHECK FLAG</td>
</tr>
<tr>
<td>00B5</td>
<td>.STATE</td>
<td>STATE OF ANALYSIS</td>
</tr>
<tr>
<td>00B6</td>
<td>.MOVEN</td>
<td>MOVE TABLE POINTER</td>
</tr>
<tr>
<td>00DC</td>
<td>.MOVE</td>
<td>OPENING POINTER</td>
</tr>
<tr>
<td>00DC</td>
<td>.OPNIG</td>
<td>OPENING MOVE TABLE</td>
</tr>
<tr>
<td>00DD</td>
<td>.WCAP0</td>
<td>COMPUTER CAPTURE 0</td>
</tr>
<tr>
<td>00DE</td>
<td>.COUNT</td>
<td>START OF COUNT TABLE</td>
</tr>
<tr>
<td>00DF</td>
<td>.BCAP2</td>
<td>OPPONENT CAPTURE 2</td>
</tr>
<tr>
<td>00E0</td>
<td>.BCAP1</td>
<td>OPPONENT CAPTURE 1</td>
</tr>
<tr>
<td>00E1</td>
<td>.WCAP1</td>
<td>COMPUTER CAPTURE 1</td>
</tr>
<tr>
<td>00E2</td>
<td>.BCAP0</td>
<td>OPPONENT CAPTURE 0</td>
</tr>
<tr>
<td>00E3</td>
<td>.MOB</td>
<td>MOBILITY</td>
</tr>
<tr>
<td>00E4</td>
<td>.MAXC</td>
<td>MAXIMUM CAPTURE</td>
</tr>
<tr>
<td>00E5</td>
<td>.CC</td>
<td>CAPTURE COUNT</td>
</tr>
<tr>
<td>00E6</td>
<td>.PCAP</td>
<td>PIECE ID OF MAXC</td>
</tr>
<tr>
<td>00E7</td>
<td>.BMOB</td>
<td>OPPONENT MOBILITY</td>
</tr>
<tr>
<td>00E8</td>
<td>.BMAXC</td>
<td>OPPONENT MAXIMUM CAPTURE</td>
</tr>
<tr>
<td>00E9</td>
<td>.BCC</td>
<td>OPPONENT CAPTURE COUNT</td>
</tr>
<tr>
<td>00EB</td>
<td>.BMAXP</td>
<td>OPPONENT MAXP</td>
</tr>
<tr>
<td>00E8</td>
<td>.XMAXC</td>
<td>CURRENT MAXIMUM CAPTURE</td>
</tr>
<tr>
<td>00EB</td>
<td>.WMOB</td>
<td>COMPUTER MOBILITY</td>
</tr>
<tr>
<td>00EC</td>
<td>.WMAXC</td>
<td>COMPUTER MAXIMUM CAPTURE</td>
</tr>
<tr>
<td>00ED</td>
<td>.WCC</td>
<td>COMPUTER CAPTURE COUNT</td>
</tr>
<tr>
<td>00EE</td>
<td>.WMAXP</td>
<td>COMPUTER MAXP</td>
</tr>
<tr>
<td>00EF</td>
<td>.PMOB</td>
<td>PREVIOUS COMPUTER MOB</td>
</tr>
<tr>
<td>00F0</td>
<td>.PMAXC</td>
<td>PREVIOUS COMPUTER MAXC</td>
</tr>
<tr>
<td>00F1</td>
<td>.PCC</td>
<td>PREVIOUS COMPUTER CC</td>
</tr>
<tr>
<td>00F2</td>
<td>.PCP</td>
<td>PREVIOUS COMPUTER MAXP</td>
</tr>
<tr>
<td>00F3</td>
<td>.OLDKY</td>
<td>KEY INPUT TEMPORARY</td>
</tr>
<tr>
<td>00F4</td>
<td>.BESTP</td>
<td>PIECE OF BEST MOVE FOUND</td>
</tr>
<tr>
<td>00F6</td>
<td>.BESTV</td>
<td>VALUE OF BEST MOVE FOUND</td>
</tr>
<tr>
<td>00F9</td>
<td>.BESTM</td>
<td>TO SQUARE OF BEST MOVE</td>
</tr>
<tr>
<td>00FA</td>
<td>.DIS1</td>
<td>DISPLAY POINT 1</td>
</tr>
<tr>
<td>00FA</td>
<td>.DIS2</td>
<td>DISPLAY POINT 2</td>
</tr>
<tr>
<td>00F9</td>
<td>.DIS3</td>
<td>DISPLAY POINT 3</td>
</tr>
</tbody>
</table>
MICROCHESS

COPYRIGHT 1976, PETER JENNINGS, MICROCHESS, 1612-43 THORNCLIFFE PK DR, TORONTO, CANADA. ALL RIGHTS RESERVED. REPRODUCTION BY ANY MEANS, IN WHOLE OR IN PART, IS PROHIBITED.

EXECUTION BEGINS AT ADDRESS 0000

; CLD
; LDXIM FF
; TXS
; LDXIM C8
; STXZ .SP2

; ROUTINES TO LIGHT LED
; DISPLAY AND GET KEY
; FROM KEYBOARD.

; OUT JSR *OUT
; JSR *GETKEY
; CMPZ .OLDKY
; BEQ OUT (DEBOUNC)
; STAZ .OLDKY

; CMPIM UC
; BNE NOSET
; LDXIM 1F
; LDAZX .SETW
; STAZX .BOARD
; DEX
; BPL WHSET
; STXZ .OMOVE
; LDAIM CC
; BNE CLDSP

; NOSET CMPIM OE
; BNE NOREV
; JSR REVERSE
; LDAIM EE
; BNE CLDSP

; NOREV CMPIM 14
; BNE NOGO
; JSR GO

; CLDSP STA .DIS1
; STAZ .DIS2
; STAZ .DIS3
; BNE CHESS

; NOGO CMPIM 0F
; BNE NOMV
; JSR MOVE
; JMP DISP


004B 4C 96 01 NOMV JMP INPUT

; THE ROUTINE JANUS DIRECTS THE
; ANALYSIS BY DETERMINING WHAT
; SHOULD OCCUR AFTER EACH MOVE
; GENERATED BY GNM

51 0100 A6 B5 JANUS +++++
52 0102 30 5C LDXZ .STATE
53  ; BMI NOCOUNT
54
55  ; THIS ROUTINE COUNTS OCCURRENCES
56  ; IT DEPENDS UPON STATE TO INDEX
57  ; THE CORRECT COUNTERS
58
61 0104 A5 B0 COUNTS LDAZ .PIECE
62 0106 F0 08 BEQ OVER IF STATE=8
63 0108 E0 08 CPXM 08 DO NOT COUNT
64 010A DO 04 BNE OVER BLK MAX CAP
65 1 010C C5 E6 CMPZ .BMAXP MOVES FOR
66 2 010E F0 2E BEQ XRT WHITE
67
68 74 0110 F6 E3 OVER INCZX .MOB MOBILITY
69 75 0112 C9 01 CMPIM 01 + QUEEN
70 76 0114 DO 02 BNE NOQ FOR TWO
71 77 0116 F6 E3 INCZX .MOB
72
73 79 0118 50 1E NOQ BVC NOCAP
74 80 011A A0 0F LDYIM OF CALCULATE
75 81 011C A5 81 LDAZ .SQUARE POINTS
76 82 011E D9 60 00 ELOOP CMPAY .BK CAPTURED
77 83 0121 F0 03 BEQ FOUN BY THIS
78 84 0123 88 DEY MOVE
79 85 0124 10 F8 BPL ELOOP
80 86 0126 B9 80 00 FOUN LDAAY .POINTS
81 87 0129 D5 E4 CMPZX .MAXC
82 88 012B 90 04 BCC LESS SAVE IF
83 89 012D 94 E6 STYX .PCAP BEST THIS
84 90 012F 95 E4 STAZX .MAXC STATE
85
86 91 0131 18 LESS CLC
87 92 0132 08 PHP ADD TO
88 93 0133 75 B5 ADCZX .CC CAPTURE
89 94 0135 95 B5 STAZX .CC COUNTS
90 95 0137 28 PLP
91
92 96 0138 E0 04 NOCAP CPXM 04
93 97 013A F0 03 BEQ ON4
94 98 013C 30 31 BMI TREE (=00 ONLY)
; GENERATE FURTHER MOVES FOR COUNT AND ANALYSIS
; ON4
LDAZ .XMAXC SAVE ACTUAL
STAZ .WCAPO CAPTURE
LDAIM 00 STATE=0
STAZ .STATE
JSR MOVE GENERATE
JSR REVERSE IMMEDIATE
JSR GNMZ MOVE
JSR REVERSE
;
LDAIM 08 STATE=8
STAZ .STATE GENERATE
JSR GNM CONTINUATION
JSR UMOVE MOVES
;
JMP STRATGY FINAL EVALUATION
NOCOUNT CPXIM F9
BNE TREE
;
DETERMINE IF THE KING CAN BE TAKEN, USED BY CHKCHK
;
LDAZ .BK IS KING
CMPZ .SQUARE IN CHECK?
BNE RETJ SET INCHEK=0
LDAIM 00 IF IT IS
STAZ .INCHEK
RTS
;
IF A PIECE HAS BEEN CAPTURED BY A TRIAL MOVE, GENERATE REPLIES & EVALUATE THE EXCHANGE GAIN/LOSS
;
TREE BVC RETJ NO CAP
LDVIM 07 (PIECES)
LDAZ .SQUARE
BNE .SQUARE
CMPAY .BK
BEQ FOUNX
DEY
BEQ FOUNX
BEQ RETJ (KING)
BPL RETJ (KING)
BPL LOOPX (KING)
BPL LOOPX SAVE
FOUNX LDAAY .POINTS BEST CAP
CMPZX .BCAPO AT THIS
BCC NOMAX LEVEL
STAZ .BCAPO
STAZ .BCAPO
NOMAX DEC .STATE
LDAIM FB
CMPZ .STATE
BEQ UPTREE .STATE
JSR GENRM CAPTURES
UPTREE INC .STATE
RTS

; THE PLAYER'S MOVE IS INPUT

; INPUT
CMPIM 08
BCS ERROR
JSR DISMY
DISP LDXIM 1F
SEARCH LDAZX .BOARD
CMPZ .DIS2
BEQ HERE DISPLAY
DEX PIECE AT
BPL SEARCH FROM
HERE STXZ .DIS1
STXZ .PIECE
ERROR JMP CHESS

; GENERATE ALL MOVES FOR ONE
; SIDE, CALL JANUS AFTER EACH
; ONE FOR NEXT STEP

+++ 1
178 U200 A2 1U
179 U202 A9 00
180 U204 95 DE
181 U206 CA
182 U207 10 FB
183 ;
184 U209 A9 1U
185 U20B 85 B0
186 U20D C6 B0
187 U20F 1U 01
188 U211 6U
189 ;
1 U212 2U 1E 03
191 U215 A4 B0
192 U217 A2 08
193 U219 85 B6
194 U21B C0 08
195 U21D 10 41
196 U21F C0 06
197 U221 10 2E
198 U223 C0 04
199 U225 1U 1F
200 U227 C0 01

LDXIM 10 CLEAR
LDAIM 00 COUNTERS
CLEAR STAZX .COUNT
DEX BPL CLEAR

; GNM
184 U209 A9 1U
185 U20B 85 B0
186 U20D C6 B0
187 U20F 1U 01
188 U211 6U
189 ;
1 U212 2U 1E 03
191 U215 A4 B0
192 U217 A2 08
193 U219 85 B6
194 U21B C0 08
195 U21D 10 41
196 U21F C0 06
197 U221 10 2E
198 U223 C0 04
199 U225 1U 1F
200 U227 C0 01

LDYXZ .PIECE
LDXIM 08
STXZ .MOVEN
CPYIM 08
BPL PAWN
CPYIM 06
BPL KNIGHT
CPYIM 04
BPL BISHOP
CPYIM 01

JSR RESET READY
LDYXZ .PIECE GET PIECE
LDXIM 08
STXZ .MOVEN COMMON START
CPYIM 08
BPL PAWN WHAT IS IT?
CPYIM 06
BPL KNIGHT KNIGHT
CPYIM 04
BPL BISHOP BISHOP
CPYIM 01
201 0229 F0 09          BEQ  QUEEN
201 0228 10 0E          BPL  ROOK

204 022D 20 8E 02        KING  JSR  SNGMV
205 0230 D0 FB          BNE  KING  MOVES
206 0232 F0 D9          BEQ  NEWP  8 TO 1
207 0234 20 9C 02        QUEEN  JSR  LINE
208 0237 D0 FB          BNE  QUEEN  MOVES
209 0239 F0 D2          BEQ  NEWP  8 TO 1

210

211 023B A2 04          ROOK  LDXIM  04
212 023D 86 B6          STXZ  .MOVEN
213 023F 20 9C 02        AGNR  JSR  LINE
214 0242 D0 FB          BNE  AGNR
215 0244 F0 C7          BEQ  NEWP

216

217 0246 20 9C 02        BISHOP  JSR  LINE
218 0249 A5 B6          LDAZ  .MOVEN
219 024B C9 04          CMPIM  04  MOVES
220 024D D0 F7          BNE  BISHOP
221 024F F0 BC          BEQ  NEWP

222

223 0251 A2 10          KNIGHT  LDXIM  10
224 0253 86 B6          STXZ  .MOVEN
225 0255 20 8E 02        AGNN  JSR  SNGMV
226 0258 A5 B6          LDAZ  .MOVEN
227 025A C9 08          CMPIM  08
228 025C D0 F7          BNE  AGNN
229 025E F0 AD          BEQ  NEWP

230

231 0260 A2 06          PAWN  LDXIM  06
232 0262 86 B6          STXZ  .MOVEN
233 0264 20 CA 02        P1  JSR  CMOVE
234 0267 50 05          BVC  P2
235 0269 30 03          BMI  P2
236 026B 20 00 01        JSR  JANUS
237 026E 20 1E 03        P2  JSR  RESET
238 0271 C6 B6          DECZ  .MOVEN
239 0273 A5 B6          LDAZ  .MOVEN
240 0275 C9 05          CMPIM  05
241 0277 F0 EB          BEQ  P1
242 0279 20 CA 02        P3  JSR  CMOVE
243 027C 70 8F          BVS  NEWP
244 027E 30 8D          BMI  NEWP
245 0280 20 00 01        JSR  JANUS
246 0283 A5 B1          LDAZ  .SQUARE
247 0285 29 F0          ANDIM  F0
248 0287 C9 20          CMPIM  20
249 0289 F0 EE          BEQ  P3
250 028B 4C 0D 02        JMP  NEWP
; CALCULATE SINGLE STEP MOVES
; FOR K, N

; SNGMV
JSR CMOVE
BRI ILL1
JSR JANUS
ILL1
JSR RESET
DECZ .MOVEN
RTS

; CALCULATE ALL MOVES DOWN A
; STRAIGHT LINE FOR Q,B,R

; LINE
JSR CMOVE
BCC OVL
BVC LINE
OVL
BMI ILL
PHP
JSR JANUS
PLP
BVC LINE
ILL
JSR RESET
DECZ .MOVEN
RTS

; EXCHANGE SIDES FOR REPLY
; ANALYSIS

; REVERSE
LDXIM GF
ETC
SEC
LDYZX .BK
LDAIM 77
SBCZX .BOARD
STAZX .BK
STYXZ .BOARD
SEC
LDAIM 77
SBCZX .BOARD
STAZX .BOARD
DEX
BPL ETC
RTS
CMOVE CALCULATES THE TO SQUARE USING .SQUARE AND THE MOVE TABLE. FLAGS SET AS FOLLOWS:

N - ILLEGAL MOVE
V - CAPTURE (LEGAL UNLESS IN CH)
C - ILLEGAL BECAUSE OF CHECK

[MY THANKS TO JIM BUTTERFIELD WHO WROTE THIS MORE EFFICIENT VERSION OF CMOVE]

CMOVE
LDAZ .SQUARE GET SQUARE
LDXZ .MOVEN MOVE POINTER
CLC
ADCZX .MOVEX MOVE LIST
STAZ .SQUARE NEW POS'N
ANDIM 88
BNE ILLEGAL OFF BOARD
LDAZ .SQUARE

LOOP
LDXIM 20
DEX IS TO
BMI NO SQUARE
CMPZX .BOARD OCCUPIED?
BNE LOOP

CPXIM 10 BY SELF?
BMI ILLEGAL

LDAIM 7F MUST BE CAP!
ADCIM 01 SET V FLAG
BSV SPX (JMP)

NO CLV NO CAPTURE

SPX LDAZ .STATE SHOULD WE
BMI RETL DO THE
CMPIM 08 CHECK CHECK?
BPL RETL

CHKCHK REVERSES SIDES
AND LOOKS FOR A KING
CAPTURE TO INDICATE
ILLEGAL MOVE BECAUSE OF CHECK. SINCE THIS IS TIME CONSUMING, IT IS NOT ALWAYS DONE.

CHKCHK
PHA STATE
PHP
LDAIM F9
STAZ .STATE GENERATE
STAZ .INCHEK ALL REPLY
JSR MOVE MOVES TO
JSR REVERSE SEE IF KING
JSR GNM IS IN
JSR RUM CHECK
PLP
PLA
STAZ .STATE
LDAZ .INCHEK
BMI RETL NO - SAFE
SEC YES - IN CHK
LDAIM FF
RTS

REPLCAL CTC LEGAL
LDAIM 00 RETURN
RTS

ILLEGAL LDAIM FF
CLC
CLV
RTS

REPLACE PIECE ON CORRECT SQUARE
RESET LDZ .PIECE GET LOCAT.
LDAZ .BOARD FOR PIECE
STAZ .SQUARE FROM BOARD
RTS

GENRM JSR MOVE MAKE MOVE
GENR2 JSR REVERSE REVERSE BOARD
JSR GNM GENERATE MOVES
JSR RUM REVERSE BACK

ROUTINE TO UNMAKE A MOVE MADE BY
MOVE

UMOVE TSX UNMAKE MOVE
STXZ .SP1
LDXZ .SP2 EXCHANGE
TXS STACKS
PLA MOVEN
STAZ CAPTURED
PLA PIECE
STAZ PIECE
TAX
PLA
STAZX .BOARD
PLA
TAX
PLA
STAZ .SQUARE
STAZX .BOARD
JMP STRV

; THIS ROUTINE MOVES .PIECE
; TO .SQUARE, PARAMETERS
; ARE SAVED IN A STACK TO UNMAKE
; THE MOVE LATER

MOVE TSX
STXZ .SP1
LDXZ .SP2
TXS
LDAZ .SQUARE
PHA
TAY
LDXIM 1F
CMPZX .BOARD
BEQ TAKE
DEX
BPL CHECK
LDAIM CC
STAZX .BOARD
TXA
PHA
CAPTURED
LDXZ .PIECE
LDAZ .BOARD
STYZX .BOARD
FROM
PHA
TXA
PHA
LDAZ .MOVEN
PHA
TSX
STXZ .SP2
LDXZ .SP1
TXS
BACK
RTS

; CONTINUATION OF SUB STRATEGY
; -CHECKS FOR CHECK OR CHECKMATE
; AND ASSIGNS VALUE TO MOVE

CKMATE LDXZ .BMAXC
CPXZ .POINTS
CAN BLK CAP
MY KING?
451 037B D0 04  BNE  NOCHECK
452 037D A9 00  LDAIM  00  GULP!
453 037F F0 0A  BEQ  RETV  DUMB MOVE!

454  
455 0381 A6 E3  NOCHECK  LDXZ  .BMOB  IS BLACK
456 0383 D0 06  BNE  RETV  UNABLE TO
457 0385 A6 EE  LDXZ  .WMAXP  MOVE AND
458 0387 D0 02  BNE  RETV  KING IN CH?
459 0389 A9 FF  LDAIM  FF  YES! MATE

460  
461 038B A2 04  RETV  LDXIM  04  RESTORE
462 038D 86 B5  STXZ  .STATE  STATE=4

463  
464  
465  
466  

467  
468 038F C5 FA  PUSH  CMPZ  .BESTV  IS THIS BEST
469 0391 90 UC  BCC  RETP  MOVE SO FAR?
470 0393 F0 0A  BEQ  RETP
471 0395 85 FA  STAZ  .BESTV  YES!
472 0397 A5 B0  LDAZ  .PIECE  SAVE IT
473 0399 85 FB  STAZ  .BESTP
474 039B A5 B1  LDAZ  .SQUARE
475 039D 85 F9  STAZ  .BESTM  FLASH DISPLAY
476 039F 4C 1F 1F  RETP  JMP  *OUT  AND RTS

477  
478  
479  
480  

481 03A2 A6 DC  GO  LDXZ  .OMOVE  OPENING?
482 03A4 10 17  BPL  NOOPEN  -NO
483 03A6 A5 F9  LDAZ  .DIS3  -YES WAS
484 03A8 D5 DC  CMPZX  .OPENING  OPPONENT'S
485 03AA D0 0F  BNE  END  MOVE OK?
486 03AC CA  DEX
487 03AD B5 DC  LDAZX  .OPENING  GET NEXT
488 03AF 85 FB  STAZ  .DIS1  CANNED
489 03B1 CA  DEX
490 03B2 B5 DC  LDAZX  .OPENING
491 03B4 85 F9  STAZ  .DIS3  DISPLAY IT
492 03B6 CA  DEX
493 03B7 86 DC  STXZ  .OMOVE  MOVE IT
494 03B9 D0 1A  BNE  MV2 (JMP)
495  
496 03BB B5 DC  END  STAZ  .OMOVE  FLAG OPENING
497 03BD A2 UC  NOOPEN  LDXIM  0C  FINISHED
498 03BF 86 B5  STXZ  .STATE  STATE=C
499 03C1 86 FA  STXZ  .BESTV  CLEAR BESTV
500 03C3 A2 14  LDXIM  14  GENERATE P
JSR GNMX

MOVES

LDXIM 04
STXZ .STATE
JSR GNMZ
STATE=4
GENERATE AND
TEST AVAILABLE
MOVES

CPXIM 0F
BCC MATE
GET BEST MOVE
IF NONE
OH OH!

LDXZ .BESTP
MOVE
LDAZX .BOARD
THE
STAZ .BESTV
BEST
STXZ .PIECE
MOVE
LDAZ .BESTM
AND DISPLAY
STAZ .SQUARE
IT
JSR MOVE
JMP CHESS

MATE
LDAIM FF
RESIGN
RTS
OR STALEMATE

SUBROUTINE TO ENTER THE
PLAYER'S MOVE

DISMV
LDXIM 04
ROTATE
ROL
ASLZ .DIS3
KEY
ROLZ .DIS2
INTO
DEX
DISPLAY
BNE ROL
ORAZ .DIS3
STAZ .DIS3
STAZ .SQUARE
RTS

THE FOLLOWING SUBROUTINE ASSIGNS
A VALUE TO THE MOVE UNDER
CONSIDERATION AND RETURNS IT IN
THE ACCUMULATOR

STRATGY
CLC
LDAIM 8U
PARAMETERS
ADDZ .WMOB
WITH WEIGHT
ADDZ .WMAXC
OF 0.25
ADDZ .WCC
ADDZ .WCAP1
ADDZ .WCAP2
SEC
SBCZ .PMAXC
SBCZ .PCC
SBCZ .BCAP0
SBCZ .BCAP1
SBCZ .BCAP2
SBCZ .PMOB
SBCZ .BMOB
BCS POS
LEAIM 00

***************

UNDERFLOW
PREVENTION

***************

-------------------

ADCIM 40
ADCZ .WMAXC
ADCZ .WCC
SEC .BMAXC
SBCZ .BMAXC
ADCZ .WMOB
ADZ .WPAPO
ADZ .WCAPO
ADZ .WPAPO
ADZ .WCAPO
ADZ .WCAPO
SBCZ .BMAXC
SBCZ .BMAXC
SBCZ .BCC
SBCZ .BCC
SBCZ .BCC
SEC .BMAXC
LDXZ .SQUARE
CPXIM 33
BEQ POSN
CPXIM 34
BEQ POSN
CPXIM 22
BEQ POSN
CPXIM 25
BEQ POSN
LDXZ .PIECE
BEQ NOPOSN
LDYZX .BOARD
CPYIM 10
BPL NOPOSN

***************

[UNDER OR OVER-
FLOW MAY OCCUR
FROM THIS
SECTION]

***************

POSITION
BONUS FOR
MOVE TO
CENTRE
OR
OUT OF
BACK RANK

***************

CONTINUE
<table>
<thead>
<tr>
<th>SYMBOL TABLE</th>
<th>SYMBOL</th>
<th>ADDR</th>
<th>DEF</th>
<th>CROSS REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHESS</td>
<td>0000</td>
<td>5</td>
<td>1</td>
<td>45 171 519</td>
</tr>
<tr>
<td>OUT</td>
<td>0008</td>
<td>15</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>HSET</td>
<td>001A</td>
<td>24</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>NOSET</td>
<td>0027</td>
<td>32</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>NOREV</td>
<td>0032</td>
<td>38</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>CLDSP</td>
<td>0039</td>
<td>42</td>
<td>30</td>
<td>36</td>
</tr>
<tr>
<td>NOCO</td>
<td>0041</td>
<td>47</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>NOMV</td>
<td>004B</td>
<td>51</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>JANUS</td>
<td>0100</td>
<td>60</td>
<td>236</td>
<td>245 257 270</td>
</tr>
<tr>
<td>COUNTS</td>
<td>0104</td>
<td>67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OVER</td>
<td>0110</td>
<td>74</td>
<td>68</td>
<td>70</td>
</tr>
<tr>
<td>NOQ</td>
<td>0118</td>
<td>79</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>ELOOP</td>
<td>011E</td>
<td>82</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>FOUN</td>
<td>0126</td>
<td>86</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>LESS</td>
<td>0131</td>
<td>92</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>NOCAP</td>
<td>0138</td>
<td>98</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>XRT</td>
<td>013E</td>
<td>101</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>ON4</td>
<td>013F</td>
<td>106</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>NOCOUNT</td>
<td>0160</td>
<td>121</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>RETJ</td>
<td>016E</td>
<td>132</td>
<td>129</td>
<td>138 144</td>
</tr>
<tr>
<td>TREE</td>
<td>016F</td>
<td>138</td>
<td>100</td>
<td>122</td>
</tr>
<tr>
<td>TOPX</td>
<td>0175</td>
<td>141</td>
<td>145</td>
<td></td>
</tr>
<tr>
<td>FOUNX</td>
<td>017F</td>
<td>146</td>
<td>142</td>
<td></td>
</tr>
<tr>
<td>NOMAX</td>
<td>0188</td>
<td>150</td>
<td>148</td>
<td></td>
</tr>
<tr>
<td>UPTREE</td>
<td>0193</td>
<td>155</td>
<td>153</td>
<td></td>
</tr>
<tr>
<td>INPUT</td>
<td>0196</td>
<td>160</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>DISP</td>
<td>019D</td>
<td>163</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>SEARCH</td>
<td>019F</td>
<td>164</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>HERE</td>
<td>01A8</td>
<td>169</td>
<td>166</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>01AC</td>
<td>171</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>GNMZ</td>
<td>0200</td>
<td>178</td>
<td>112</td>
<td>505</td>
</tr>
<tr>
<td>GNMX</td>
<td>0202</td>
<td>179</td>
<td>501</td>
<td></td>
</tr>
<tr>
<td>CLEAR</td>
<td>0204</td>
<td>180</td>
<td>182</td>
<td></td>
</tr>
<tr>
<td>GNJ</td>
<td>0209</td>
<td>184</td>
<td>117</td>
<td>355 386</td>
</tr>
<tr>
<td>NEWP</td>
<td>020D</td>
<td>186</td>
<td>206</td>
<td>209 215 221 229 243 244 250</td>
</tr>
<tr>
<td>NEX</td>
<td>0212</td>
<td>190</td>
<td>187</td>
<td></td>
</tr>
<tr>
<td>KING</td>
<td>022D</td>
<td>204</td>
<td>205</td>
<td></td>
</tr>
<tr>
<td>QUEEN</td>
<td>0234</td>
<td>207</td>
<td>201</td>
<td>208</td>
</tr>
<tr>
<td>ROOK</td>
<td>023B</td>
<td>211</td>
<td>202</td>
<td></td>
</tr>
<tr>
<td>AGNR</td>
<td>023F</td>
<td>213</td>
<td>214</td>
<td></td>
</tr>
<tr>
<td>SHOP</td>
<td>0246</td>
<td>217</td>
<td>199</td>
<td>220</td>
</tr>
<tr>
<td>KNIGHT</td>
<td>0251</td>
<td>223</td>
<td>197</td>
<td></td>
</tr>
<tr>
<td>AGNN</td>
<td>0255</td>
<td>225</td>
<td>228</td>
<td></td>
</tr>
<tr>
<td>PAWN</td>
<td>0260</td>
<td>231</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>0264</td>
<td>233</td>
<td>241</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>026E</td>
<td>237</td>
<td>234</td>
<td>235</td>
</tr>
<tr>
<td>P3</td>
<td>0279</td>
<td>242</td>
<td>249</td>
<td></td>
</tr>
<tr>
<td>SNGMV</td>
<td>028E</td>
<td>255</td>
<td>204</td>
<td>225</td>
</tr>
<tr>
<td>ILL1</td>
<td>0296</td>
<td>258</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>LINE</td>
<td>029C</td>
<td>265</td>
<td>207</td>
<td>213 217 267 272</td>
</tr>
<tr>
<td>OVL</td>
<td>02A3</td>
<td>258</td>
<td>266</td>
<td></td>
</tr>
<tr>
<td>ILL</td>
<td>02AC</td>
<td>273</td>
<td>268</td>
<td></td>
</tr>
<tr>
<td>REVERSE</td>
<td>02B2</td>
<td>280</td>
<td>34</td>
<td>111 113 354 385 387</td>
</tr>
<tr>
<td>ETC</td>
<td>02B4</td>
<td>281</td>
<td>292</td>
<td></td>
</tr>
<tr>
<td>CMOVE</td>
<td>02CA</td>
<td>311</td>
<td>233</td>
<td>242 255 265</td>
</tr>
<tr>
<td>LOOP</td>
<td>02DB</td>
<td>321</td>
<td>324</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td>02EC</td>
<td>333</td>
<td>322</td>
<td></td>
</tr>
<tr>
<td>SYMBOL</td>
<td>ADDR</td>
<td>DEF</td>
<td>CROSS REFERENCES</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-----</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>SPX</td>
<td>02ED</td>
<td>335</td>
<td>331</td>
<td></td>
</tr>
<tr>
<td>CHKCHK</td>
<td>02F5</td>
<td>348</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETL</td>
<td>0315</td>
<td>366</td>
<td>336 338 361</td>
<td></td>
</tr>
<tr>
<td>ILLEGAL</td>
<td>0319</td>
<td>370</td>
<td>317 327 343</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>031E</td>
<td>377</td>
<td>190 237 258 273</td>
<td></td>
</tr>
<tr>
<td>GENRM</td>
<td>0325</td>
<td>384</td>
<td>154</td>
<td></td>
</tr>
<tr>
<td>GENR2</td>
<td>0328</td>
<td>385</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUM</td>
<td>032E</td>
<td>387</td>
<td>356</td>
<td></td>
</tr>
<tr>
<td>UMVE</td>
<td>0331</td>
<td>392</td>
<td>118</td>
<td></td>
</tr>
<tr>
<td>MOVE</td>
<td>034B</td>
<td>415</td>
<td>49 110 353 384 518</td>
<td></td>
</tr>
<tr>
<td>CHECK</td>
<td>0357</td>
<td>423</td>
<td>426</td>
<td></td>
</tr>
<tr>
<td>TAKE</td>
<td>035E</td>
<td>427</td>
<td>424</td>
<td></td>
</tr>
<tr>
<td>STRV</td>
<td>0370</td>
<td>439</td>
<td>408</td>
<td></td>
</tr>
<tr>
<td>CKMATE</td>
<td>0377</td>
<td>449</td>
<td>597</td>
<td></td>
</tr>
<tr>
<td>NOCHEK</td>
<td>0381</td>
<td>455</td>
<td>451</td>
<td></td>
</tr>
<tr>
<td>RETV</td>
<td>038B</td>
<td>461</td>
<td>453 456 458</td>
<td></td>
</tr>
<tr>
<td>PUSH</td>
<td>038F</td>
<td>468</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETP</td>
<td>039F</td>
<td>476</td>
<td>469 470</td>
<td></td>
</tr>
<tr>
<td>GO</td>
<td>03A2</td>
<td>481</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>03BB</td>
<td>496</td>
<td>485</td>
<td></td>
</tr>
<tr>
<td>IOOPEN</td>
<td>03BD</td>
<td>497</td>
<td>482</td>
<td></td>
</tr>
<tr>
<td>MW2</td>
<td>03D5</td>
<td>512</td>
<td>494</td>
<td></td>
</tr>
<tr>
<td>MATE</td>
<td>03E7</td>
<td>521</td>
<td>510</td>
<td></td>
</tr>
<tr>
<td>DISMV</td>
<td>03EA</td>
<td>527</td>
<td>162</td>
<td></td>
</tr>
<tr>
<td>ROL</td>
<td>03EC</td>
<td>528</td>
<td>531</td>
<td></td>
</tr>
<tr>
<td>STRATGY</td>
<td>1780</td>
<td>543</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>POS</td>
<td>17A0</td>
<td>560</td>
<td>558</td>
<td></td>
</tr>
<tr>
<td>POSN</td>
<td>17E0</td>
<td>595</td>
<td>583 585 587 589</td>
<td></td>
</tr>
<tr>
<td>NOPOSN</td>
<td>17E3</td>
<td>597</td>
<td>591 594</td>
<td></td>
</tr>
<tr>
<td>.BOARD</td>
<td>0050</td>
<td>602</td>
<td>25 164 284 286 289 290 323 378 402 407</td>
<td></td>
</tr>
<tr>
<td>.BK</td>
<td>0060</td>
<td>603</td>
<td>82 127 141 282 285</td>
<td></td>
</tr>
<tr>
<td>.SETW</td>
<td>0070</td>
<td>604</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>.MOVEX</td>
<td>008F</td>
<td>605</td>
<td>314</td>
<td></td>
</tr>
<tr>
<td>.POINTS</td>
<td>00A0</td>
<td>606</td>
<td>86 146 450</td>
<td></td>
</tr>
<tr>
<td>.PIECE</td>
<td>00B0</td>
<td>607</td>
<td>67 170 185 186 191 377 399 431 472 515</td>
<td></td>
</tr>
<tr>
<td>.SQUARE</td>
<td>00B1</td>
<td>608</td>
<td>81 128 140 246 311 315 318 379 406 419</td>
<td></td>
</tr>
<tr>
<td>.SP2</td>
<td>00B2</td>
<td>609</td>
<td>9 394 417 440</td>
<td></td>
</tr>
<tr>
<td>.SP1</td>
<td>00B3</td>
<td>610</td>
<td>393 416 441</td>
<td></td>
</tr>
<tr>
<td>.INCHEK</td>
<td>00B4</td>
<td>611</td>
<td>131 352 360</td>
<td></td>
</tr>
<tr>
<td>.STATE</td>
<td>00B5</td>
<td>612</td>
<td>60 109 116 150 152 155 335 351 359 462</td>
<td></td>
</tr>
<tr>
<td>.MOVEN</td>
<td>00B6</td>
<td>613</td>
<td>193 212 218 224 226 232 238 239 259 274</td>
<td></td>
</tr>
<tr>
<td>.OMOVE</td>
<td>00DC</td>
<td>614</td>
<td>28 481 493 496</td>
<td></td>
</tr>
<tr>
<td>.OPENING</td>
<td>00DC</td>
<td>615</td>
<td>484 487 490</td>
<td></td>
</tr>
<tr>
<td>.WCAPU</td>
<td>00DD</td>
<td>616</td>
<td>107 570 571 572 573</td>
<td></td>
</tr>
<tr>
<td>.COUNT</td>
<td>00DE</td>
<td>617</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>.BCAP2</td>
<td>00DE</td>
<td>618</td>
<td>555</td>
<td></td>
</tr>
<tr>
<td>.WCAP2</td>
<td>00DF</td>
<td>619</td>
<td>549</td>
<td></td>
</tr>
<tr>
<td>.BCAP1</td>
<td>00E0</td>
<td>620</td>
<td>554 580</td>
<td></td>
</tr>
<tr>
<td>.WCAP1</td>
<td>00E1</td>
<td>621</td>
<td>548 574</td>
<td></td>
</tr>
<tr>
<td>.BCAP0</td>
<td>00E2</td>
<td>622</td>
<td>147 149 553</td>
<td></td>
</tr>
<tr>
<td>.MOB</td>
<td>00E3</td>
<td>623</td>
<td>74 77</td>
<td></td>
</tr>
<tr>
<td>.MAXC</td>
<td>00E4</td>
<td>624</td>
<td>87 90</td>
<td></td>
</tr>
<tr>
<td>.CC</td>
<td>00E5</td>
<td>625</td>
<td>94 95</td>
<td></td>
</tr>
<tr>
<td>SYMBOL</td>
<td>ADDR</td>
<td>DEF</td>
<td>CROSS REFERENCES</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-----</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>.PCAP</td>
<td>00E6</td>
<td>626</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>.BMOB</td>
<td>00E3</td>
<td>627</td>
<td>455 557</td>
<td></td>
</tr>
<tr>
<td>.BMAXC</td>
<td>00E4</td>
<td>628</td>
<td>449 566 576 577</td>
<td></td>
</tr>
<tr>
<td>.BCC</td>
<td>00E5</td>
<td>629</td>
<td>578 579</td>
<td></td>
</tr>
<tr>
<td>.BMAXP</td>
<td>00E6</td>
<td>630</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>.XMAXC</td>
<td>00E8</td>
<td>631</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>.WMOB</td>
<td>00EB</td>
<td>632</td>
<td>545</td>
<td></td>
</tr>
<tr>
<td>.WMAXC</td>
<td>00EC</td>
<td>633</td>
<td>546 563</td>
<td></td>
</tr>
<tr>
<td>.WCC</td>
<td>00ED</td>
<td>634</td>
<td>547 564</td>
<td></td>
</tr>
<tr>
<td>.WMAXP</td>
<td>00EE</td>
<td>635</td>
<td>457</td>
<td></td>
</tr>
<tr>
<td>.PMOB</td>
<td>00EF</td>
<td>636</td>
<td>556</td>
<td></td>
</tr>
<tr>
<td>.PMA XC</td>
<td>00F0</td>
<td>637</td>
<td>551</td>
<td></td>
</tr>
<tr>
<td>.PCC</td>
<td>00F1</td>
<td>638</td>
<td>552</td>
<td></td>
</tr>
<tr>
<td>.PCP</td>
<td>00F2</td>
<td>639</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.OLDKY</td>
<td>00F3</td>
<td>640</td>
<td>17 19</td>
<td></td>
</tr>
<tr>
<td>.BESTP</td>
<td>00FB</td>
<td>641</td>
<td>473 512</td>
<td></td>
</tr>
<tr>
<td>.BESTV</td>
<td>00FA</td>
<td>642</td>
<td>468 471 499 508 514</td>
<td></td>
</tr>
<tr>
<td>.BESTM</td>
<td>00F9</td>
<td>643</td>
<td>475 516</td>
<td></td>
</tr>
<tr>
<td>.DIS1</td>
<td>00FB</td>
<td>644</td>
<td>42 169 488</td>
<td></td>
</tr>
<tr>
<td>.DIS2</td>
<td>00FA</td>
<td>645</td>
<td>43 165 529</td>
<td></td>
</tr>
<tr>
<td>.DIS3</td>
<td>00F9</td>
<td>646</td>
<td>44 483 491 528 532 533</td>
<td></td>
</tr>
</tbody>
</table>

**BLOCK DATA**

| .SETW | 0070 | 03 04 00 07 02 05 01 06 10 17 11 16 12 15 14 13 |
|       |      | 73 74 70 77 72 75 71 76 60 67 61 66 62 65 64 63 |

| .MOVEX | 0090 | F0 FF 01 10 11 0F EF F1 DF E1 EE F2 12 0E 1F 21 |

| POINTS | 00A0 | 0B 0A 06 06 04 04 04 02 02 02 02 02 02 02 02 |

| .OPNNG | 00C0 | 99 25 05 25 01 00 33 25 07 36 34 0D 34 34 0E 52 |
|        |      | 25 0D 45 35 04 55 22 06 43 33 0F CC |

NOTE THAT 00B7 TO 00BF, 00F4 TO 00F8, AND 00FC TO 00FF ARE AVAILABLE FOR USER EXPANSION AND I/O ROUTINES.