

MICROCHESS

A CHESS PLAYING PROGRAM

FOR THE 6502 MICROCOMPUTER

BY PETER JENNINGS

MICROCHESS is available on KIM cassette for \$5.00.

M I C R O C H E S S

KIM Cassette Loading Instructions

- 1 Enter (RS) to reset KIM.
- 2 Enter (AD) 0 0 F 1 (DA) 0 0 to reset decimal flag.
- 3 Enter (AD) 1 7 F 9 (DA) C 1 to enter tape ID.
- 4 Enter (AD) 1 8 7 3 (GO) to begin read routine.
- 5 Start your cassette player.
- 6 When you see: 0000 D8 stop your cassette player.
- 7 Enter (RS) (AD) 1 8 7 3 (GO) to read block 2.
- 8 Start your cassette player.
- 9 When you see: 0000 D8 stop your cassette player.
- 10 Enter (RS) (GO) to start program execution.

If you wish KIM to play a specific opening, enter the ID in address 17F9 and load the opening data. Enter (RS) before and after each tape load.

Data for Openings

Microchess plays		Opening
white	black	
A0	A1	Four Knights
A2	A3	French Defence
A4	A5	Ruy Lopez
A6	A7	Queen's Indian
A8	A9	Guioco Piano

Remember to always press (RS) between each tape load. Otherwise, data at 0100 and 0101 may be overwritten by the stack. Verify these locations against the program listing if you have trouble executing the program.

A second copy of the two main programs can be found after the openings.

M I C R O C H E S S

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:

**Micro-Ware Ltd.
27 Firstbrooke Road,
Toronto Ontario,
Canada. M4E 2L2**

TABLE OF CONTENTS

PLAYER'S MANUAL

LOADING THE PROGRAMS	1
MICROCHESS NOTATION	1
MICROCHESS COMMAND KEYS	2
THE COMPUTER'S MOVE	3
TABLE OF PIECE CODES	3
ENTERING YOUR MOVE	4
SPECIAL MOVES	5
LEVEL OF PLAY	6
POSITION VERIFICATION	6
MEMORY LOCATIONS FOR THE PIECES	7
NOTES	8

PROGRAMMER'S MANUAL

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

SOURCE LISTING

MICROCHESS	1
SYMBOL TABLE AND CROSS REFERENCES	13
BLOCK DATA	15

M I C R O C H E S S

P L A Y E R ' S M A N U A L

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 17E6. 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.

COMPUTER

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77

PLAYER

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.

[E] 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. EEEEEEE is displayed immediately after the [E] key is pressed to verify operation.

[F] 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.

[PC] 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:

0 - KING	4 - King Bishop	8 - K R Pawn	C - K B Pawn
1 - Queen	5 - Queen Bishop	9 - Q R Pawn	D - Q B Pawn
2 - King Rook	6 - King Knight	A - K N Pawn	E - Q Pawn
3 - Queen Rook	7 - Queen Knight	B - Q N Pawn	F - K Pawn

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

For example the display [OF 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.

LEVEL	LOCATION 02F2	LOCATION 018B	AVGE TIME PER MOVE
SUPER BLITZ	00	FF	3 seconds
BLITZ	00	FB	10 seconds
NORMAL	08	FB	100 second

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

COMPUTER PIECES		YOUR PIECES
0050	King	0060
0051	Queen	0061
0052	King Rook	0062
0053	Queen Rook	0063
0054	King Bishop	0064
0055	Queen Bishop	0065
0056	King Knight	0066
0057	Queen Knight	0067
0058	K R Pawn	0068
0059	Q R Pawn	0069
006A	K N Pawn	006A
005B	Q N Pawn	006B
005C	K B Pawn	006C
005D	Q B Pawn	006D
005E	Q Pawn	006E
005F	K Pawn	006F

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.

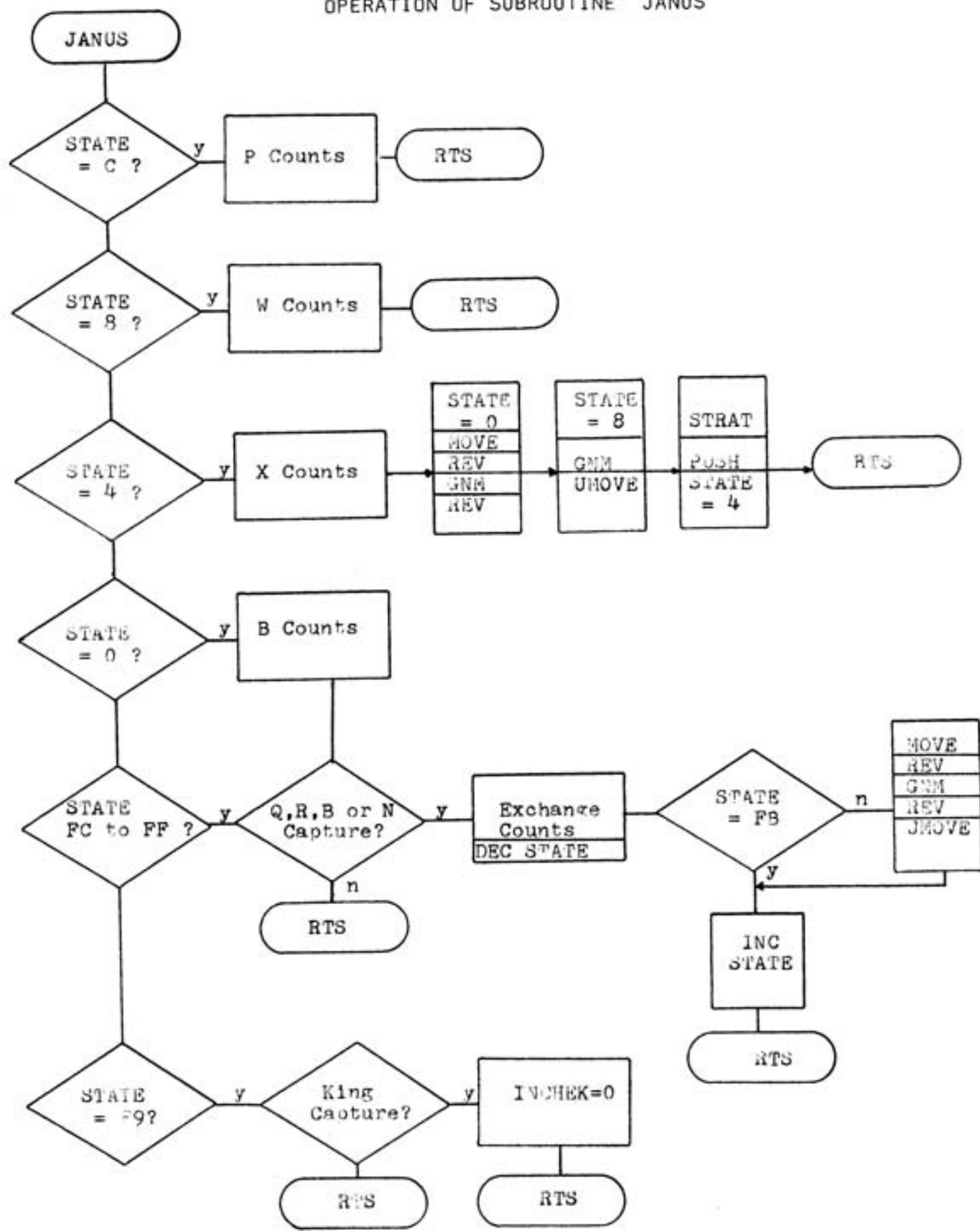
- I Control and Input/Output. This section comprises the initialization routines, the input and output routines, and the main entry into the move generation and evaluation routines.
- II Move Generation and Data Collection. This program group generates the moves available to the computer, one at a time. 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

STATE	SET BY	FUNCTION
C	GO	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.
4	GO	Generate all available moves, evaluating each one and assigning a value to it as a possible selection.
8	JANUS	Having made one trial move, generate the possible second moves for analysis.
0	JANUS	Having made one trial move, generate the possible replies for analysis.
FF	JANUS	Since a reply move was a capture, reverse the board and evaluate the exchange that could result.
FE	JANUS	Stage two of the exchange evaluation started by STATE FF.
FD	JANUS	Stage three of the exchange evaluation.
FC	JANUS	Last stage of the exchange evaluation.
F9	CHKCHK	Look for a capture of the king which signifies that the move being calculated is illegal.

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 (.WCAPO, .WCAP1, .WCAP2, .BCAPO, .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 STRATGY algorithm.

All information available is combined by the algorithm in the subprogram STRATGY 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 * WCAPO
          + 1.25 * WCAP1
          + 0.75 * (WMAXC + WCC)
          + 0.25 * (WMOB + WCAP2)
          - 2.50 * BMAXC
          - 2.00 * BCC
          - 1.25 * BCAP1
          - 0.50 * BMAXC
          - 0.25 * (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 WCAPO 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 00C0 through 00DB. 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 0008, 000B, 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 6 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 EEEEEEE 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.

WR	WN	WB	WK	WQ	WB		WR
WP	WP	WP		WP	WP	WP	WP
	**		**		WN		**

SPECIAL MOVES

Several types of moves are not included in the basic MICROCHESS program in order to reduce the memory requirements. These moves, castling, 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 castling, 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- Xray 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.

The data below enables the computer to play the opening specified from memory. The data is in a block from 00C0 to 00DB. W specifies that the computer will play white, B specifies that the computer is black.

[illegible]

EXPLANATION OF SYMBOLS

ADDR	SYMBOL	EXPLANATION
0050	.BOARD	: LOCATION OF PIECES
0060	.BK	: OPPONENT'S PIECES
0070	.SETW	: INITIAL PIECE LOCATIONS
008F	.MOVEX	: TABLE OF MOVE DIRECTIONS
00A0	.POINTS	: TABLE OF PIECE VALUES
00B0	.PIECE	: CURRENT PIECE UNDER ANALYSIS
00B1	.SQUARE	: TO SQUARE OF PIECE
00B2	.SP2	: STACK POINTER FOR STACK 2
00B3	.SP1	: STACK POINTER FOR STACK 1
00B4	.INCHEK	: MOVE INTO CHECK FLAG
00B5	.STATE	: STATE OF ANALYSIS
00B6	.MOVEN	: MOVE TABLE POINTER
00DC	.OMOVE	: OPENING POINTER
00DC	.OPNING	: OPENING MOVE TABLE
00DD	.WCAPO	: COMPUTER CAPTURE 0
00DE	.COUNT	: START OF COUNT TABLE
00DE	.BCAP2	: OPPONENT CAPTURE 2
00DF	.WCAP2	: COMPUTER CAPTURE 2
00E0	.BCAP1	: OPPONENT CAPTURE 1
00E1	.WCAP1	: COMPUTER CAPTURE 1
00E2	.BCAPO	: OPPONENT CAPTURE 0
00E3	.MOB	: MOBILITY
00E4	.MAXC	: MAXIMUM CAPTURE
00E5	.CC	: CAPTURE COUNT
00E6	.PCAP	: PIECE ID OF MAXC
00E3	.BMOB	: OPPONENT MOBILITY
00E4	.BMAXC	: OPPONENT MAXIMUM CAPTURE
00E5	.BCC	: OPPONENT CAPTURE COUNT
00E6	.BMAXP	: OPPONENT MAXP
00E8	.XMAXC	: CURRENT MAXIMUM CAPTURE
00EB	.WMOB	: COMPUTER MOBILITY
00EC	.WMAXC	: COMPUTER MAXIMUM CAPTURE
00ED	.WCC	: COMPUTER CAPTURE COUNT
00EE	.WMAXP	: COMPUTER MAXP
00EF	.PMOB	: PREVIOUS COMPUTER MOB
00F0	.PMAXC	: PREVIOUS COMPUTER MAXC
00F1	.PCC	: PREVIOUS COMPUTER CC
00F2	.PCP	: PREVIOUS COMPUTER MAXP
00F3	.OLDKY	: KEY INPUT TEMPORARY
00FB	.BESTP	: PIECE OF BEST MOVE FOUND
00FA	.BESTV	: VALUE OF BEST MOVE FOUND
00F9	.BESTM	: TO SQUARE OF BEST MOVE
00FB	.DIS1	: DISPLAY POINT 1
00FA	.DIS2	: DISPLAY POINT 2
00F9	.DIS3	: DISPLAY POINT 3

0000:	D8	A2	FF	9A	A2	C8	86	B2	20	1F	1F	20	6A	1F	C5	F3
0010:	F0	F6	85	F3	C9	0C	D0	0F	A2	1F	B5	70	95	50	CA	10
0020:	F9	86	DC	A9	CC	D0	12	C9	0E	D0	07	20	B2	02	A9	EE
0030:	D0	07	C9	14	D0	0B	20	A2	03	85	FB	85	FA	85	F9	D0
0040:	BF	C9	0F	D0	06	20	4B	03	4C	9D	01	4C	96	01	10	00
0070:	03	04	00	07	02	05	01	06	10	17	11	16	12	15	14	13
0080:	73	74	70	77	72	75	71	76	60	67	61	66	62	65	64	63
0090:	F0	FF	01	10	11	0F	EF	F1	DF	E1	EE	F2	12	0E	1F	21
00A0:	0B	0A	06	06	04	04	04	04	02	02	02	02	02	02	02	02
0100:	A6	B5	30	5C	A5	B0	F0	08	E0	08	D0	04	C5	E6	F0	2E
0110:	F6	E3	C9	01	D0	02	F6	E3	50	1E	A0	0F	A5	B1	D9	60
0120:	00	F0	03	88	10	F8	B9	A0	00	D5	E4	90	04	94	E6	95
0130:	E4	18	08	75	E5	95	E5	28	E0	04	F0	03	30	31	60	A5
0140:	E8	85	DD	A5	00	85	B5	20	4B	03	20	B2	02	20	00	02
0150:	20	B2	02	A9	08	85	B5	20	09	02	20	31	03	4C	80	17
0160:	E0	F9	D0	0B	A5	60	C5	B1	D0	04	A9	00	85	B4	60	50
0170:	FD	A0	07	A5	B1	D9	60	00	F0	05	88	F0	F1	10	F6	B9
0180:	A0	00	D5	E2	90	02	95	E2	C6	B5	A9	FB	C5	B5	F0	03
0190:	20	25	03	E6	B5	60	C9	08	B0	12	20	EA	03	A2	1F	B5
01A0:	50	C5	FA	F0	03	CA	10	F7	86	FB	86	B0	4C	00	00	C0
0200:	A2	10	A9	00	95	DE	CA	10	FB	A9	10	85	B0	C6	B0	10
0210:	01	60	20	1E	03	A4	B0	A2	08	86	B6	C0	08	10	41	C0
0220:	06	10	2E	C0	04	10	1F	C0	01	F0	09	10	0E	20	8E	02
0230:	D0	FB	F0	D9	20	9C	02	D0	FB	F0	D2	A2	04	86	B6	20
0240:	9C	02	D0	FB	F0	C7	20	9C	02	A5	B6	C9	04	D0	F7	F0
0250:	BC	A2	10	86	B6	20	8E	02	A5	B6	C9	08	D0	F7	F0	AD
0260:	A2	06	86	B6	20	CA	02	50	05	30	03	20	00	01	20	1E
0270:	03	C6	B6	A5	B6	C9	05	F0	EB	20	CA	02	70	8F	30	8D
0280:	20	00	01	A5	B1	29	F0	C9	20	F0	EE	4C	0D	02	20	CA
0290:	02	30	03	20	00	01	20	1E	03	C6	B6	60	20	CA	02	90
02A0:	02	50	F9	30	07	08	20	00	01	28	50	F0	20	1E	03	C6
02B0:	B6	60	A2	0F	38	B4	60	A9	77	F5	50	95	60	94	50	38
02C0:	A9	77	F5	50	95	50	CA	10	EB	60	A5	B1	A6	B6	18	75
02D0:	8F	85	B1	29	88	D0	42	A5	B1	A2	20	CA	30	0E	D5	50
02E0:	D0	F9	E0	10	30	33	A9	7F	69	01	70	01	B8	A5	B5	30
02F0:	24	C9	08	10	20	48	08	A9	F9	85	B5	85	B4	20	4B	03
0300:	20	B2	02	20	09	02	20	2E	03	28	68	85	B5	A5	B4	30
0310:	04	38	A9	FF	60	18	A9	00	60	A9	FF	18	B8	60	A6	B0
0320:	B5	50	85	B1	60	20	4B	03	20	B2	02	20	09	02	20	B2
0330:	02	BA	86	B3	A6	B2	9A	68	85	B6	68	85	B0	AA	68	95
0340:	50	68	AA	68	85	B1	95	50	4C	70	03	BA	86	B3	A6	B2
0350:	9A	A5	B1	48	A8	A2	1F	D5	50	F0	03	CA	10	F9	A9	CC
0360:	95	50	8A	48	A6	B0	B5	50	94	50	48	8A	48	A5	B6	48
0370:	BA	86	B2	A6	B3	9A	60	A6	E4	E4	A0	D0	04	A9	00	F0
0380:	0A	A6	E3	D0	06	A6	EE	D0	02	A9	FF	A2	04	86	B5	C5
0390:	FA	90	0C	F0	0A	85	FA	A5	B0	85	FB	A5	B1	85	F9	4C
03A0:	1F	1F	A6	DC	10	17	A5	F9	D5	DC	D0	0F	CA	B5	DC	85
03B0:	FB	CA	B5	DC	85	F9	CA	86	DC	D0	1A	85	DC	A2	0C	86
03C0:	B5	86	FA	A2	14	20	02	02	A2	04	86	B5	20	00	02	A6
03D0:	FA	E0	0F	90	12	A6	FB	B5	50	85	FA	86	B0	A5	F9	85
03E0:	B1	20	4B	03	4C	00	00	A9	FF	60	A2	04	06	F9	26	FA
03F0:	CA	D0	F9	05	F9	85	F9	85	B1	60	00	00	00	00	00	00
1780:	18	A9	80	65	EB	65	EC	65	ED	65	E1	65	DF	38	E5	F0
1790:	E5	F1	E5	E2	E5	E0	E5	DE	E5	EF	E5	E3	B0	02	A9	00
17A0:	4A	18	69	40	65	EC	65	ED	38	E5	E4	4A	18	69	90	65
17B0:	DD	65	DD	65	DD	65	DD	65	E1	38	E5	E4	E5	E4	E5	E5
17C0:	E5	E5	E5	E0	A6	B1	E0	33	F0	16	E0	34	F0	12	E0	22
17D0:	F0	0E	E0	25	F0	0A	A6	B0	F0	09	B4	50	C0	10	10	03
17E0:	18	69	02	4C	77	03										

© COPYRIGHT 1976, ALL RIGHTS RESERVED.

MICROCHESS

MICROCHESS

© COPYRIGHT 1976, PETER JENNINGS, MICROCHESS,
 27 Firstbrooke Road, TORONTO, CANADA.
 ALL RIGHTS RESERVED. REPRODUCTION BY ANY
 MEANS, IN WHOLE OR IN PART, IS PROHIBITED.

2		;	EXECUTION BEGINS AT ADDRESS 0000		
3		;			
4			+++		
5	0000 D8	CHES	CLD		INITIALIZE
6	0001 A2 FF		LDXIM	FF	TWO STACKS
7	0003 9A		TXS		
8	0004 A2 C8		LDXIM	C8	
9	0006 86 B2		STXZ	.SP2	
10		;			
11		;			
12		;			
13		;			
14		;			
15	0008 20 1F 1F	OUT	JSR	*OUT	DISPLAY AND
16	000B 20 6A 1F		JSR	*GETKEY	GET INPUT
17	000E C5 F3		CMPZ	.OLDKY	KEY IN ACC
18	0010 F0 F6		BEQ	OUT	(DEBOUNCE)
19	0012 85 F3		STAZ	.OLDKY	
20		;			
21	0014 C9 0C		CMPIM	0C	[C]
22	0016 D0 0F		BNE	NOSET	SET UP
23	0018 A2 1F		LDXIM	1F	BOARD
24	001A B5 70	WHSET	LDAZX	.SETW	FROM
25	001C 95 50		STAZX	.BOARD	SETW
26	001E CA		DEX		
27	001F 10 F9		BPL	WHSET	
28	0021 86 DC		STXZ	.OMOVE	
29	0023 A9 CC		LDAIM	CC	
30	0025 D0 12		BNE	CLDSP	
31		;			
32	0027 C9 0E	NOSET	CMPIM	0E	[E]
33	0029 D0 07		BNE	NOREV	REVERSE
34	002B 20 B2 02		JSR	REVERSE	BOARD AS
35	002E A9 EE		LDAIM	EE	IS
36	0030 D0 07		BNE	CLDSP	
37		;			
38	0032 C9 14	NOREV	CMPIM	14	[PC]
39	0034 D0 0B		BNE	NOGO	PLAY CHESS
40	0036 20 A2 03		JSR	GO	
41		;			
42	0039 85 FB	CLDSP	STA	.DIS1	DISPLAY
43	003B 85 FA		STAZ	.DIS2	ACROSS
44	003D 85 F9		STAZ	.DIS3	DISPLAY
45	003F D0 BF		BNE	CHESS	
46		;			
47	0041 C9 0F	NOGO	CMPIM	0F	[F]
48	0043 D0 06		BNE	NOMV	MOVE MAN
49	0045 20 4B 03		JSR	MOVE	AS ENTERED
50	0048 4C 9D 01		JMP	DISP	

```

51 004B 4C 96 01      NOMV      JMP      INPUT
52                      ;
53                      ;      THE ROUTINE JANUS DIRECTS THE
54                      ;      ANALYSIS BY DETERMINING WHAT
55                      ;      SHOULD OCCUR AFTER EACH MOVE
56                      ;      GENERATED BY GNM
57                      ;
58                      ;
59                      ;      +++
60 0100 A6 B5      JANUS      LDXZ      .STATE
61 0102 30 5C      BMI      NOCOUNT
62                      ;
63                      ;      THIS ROUTINE COUNTS OCCURRENCES
64                      ;      IT DEPENDS UPON STATE TO INDEX
65                      ;      THE CORRECT COUNTERS
66                      ;
67 0104 A5 B0      COUNTS      LDAZ      .PIECE
68 0106 F0 08      BEQ      OVER      IF STATE=8
69 0108 E0 08      CPXIM      08      DO NOT COUNT
70 010A D0 04      BNE      OVER      BLK MAX CAP
71 010C C5 E6      CMPZ      .BMAXP    MOVES FOR
72 010E F0 2E      BEQ      XRT      WHITE
73                      ;
74 0110 F6 E3      OVER      INCZX      .MOB      MOBILITY
75 0112 C9 01      CMPIM      01      + QUEEN
76 0114 D0 02      BNE      NOQ      FOR TWO
77 0116 F6 E3      INCZX      .MOB
78                      ;
79 0118 50 1E      NOQ      BVC      NOCAP
80 011A A0 0F      LDYIM      OF
81 011C A5 B1      LDAZ      .SQUARE    CALCULATE
82 011E D9 60 00      ELOOP      CMPAY      .BK      POINTS
83 0121 F0 03      BEQ      FOUN      CAPTURED
84 0123 88      DEY      BY THIS
85 0124 10 F8      BPL      ELOOP      MOVE
86 0126 B9 A0 00      FOUN      LDAAY      .POINTS
87 0129 D5 E4      CMPZX      .MAXC
88 012B 90 04      BCC      LESS
89 012D 94 E6      STYZX      .PCAP
90 012F 95 E4      STAZX      .MAXC      SAVE IF
91                      ;      BEST THIS
92 0131 18      LESS      CLC      STATE
93 0132 08      PHP
94 0133 75 E5      ADCZX      .CC      ADD TO
95 0135 95 E5      STAZX      .CC      CAPTURE
96 0137 28      PLP      COUNTS
97                      ;
98 0138 E0 04      NOCAP      CPXIM      04
99 013A F0 03      BEQ      ON4
100 013C 30 31      BMI      TREE      (=00 ONLY)

```

```

101 013E 60      XRT      RTS
102              ;
103              ;      GENERATE FURTHER MOVES FOR COUNT
104              ;      AND ANALYSIS
105              ;
106 013F A5 E8    ON4      LDAZ      .XMAXC      SAVE ACTUAL
107 0141 85 DD              STAZ      .WCAPO      CAPTURE
108 0143 A9 00              LDAIM     00          STATE=0
109 0145 85 B5              STAZ      .STATE
110 0147 20 4B 03          JSR      MOVE      GENERATE
111 014A 20 B2 02          JSR      REVERSE    IMMEDIATE
112 014D 20 00 02          JSR      GNMZ      REPLY MOVES
113 0150 20 B2 02          JSR      REVERSE
114              ;
115 0153 A9 08              LDAIM     08          STATE=8
116 0155 85 B5              STAZ      .STATE    GENERATE
117 0157 20 09 02          JSR      GNM        CONTINUATION
118 015A 20 31 03          JSR      UMOVE     MOVES
119              ;
120 015D 4C 80 17          JMP      STRATGY    FINAL EVALUATION
121 0160 E0 F9      NOCOUNT CPXIM     F9
122 0162 D0 0B              BNE      TREE
123              ;
124              ;      DETERMINE IF THE KING CAN BE
125              ;      TAKEN, USED BY CHKCHK
126              ;
127 0164 A5 60              LDAZ      .BK        IS KING
128 0166 C5 B1              CMPZ      .SQUARE    IN CHECK?
129 0168 D0 04              BNE      RETJ      SET INCHEK=0
130 016A A9 00              LDAIM     00        IF IT IS
131 016C 85 B4              STAZ      .INCHEK
132 016E 60      RETJ      RTS
133              ;
134              ;      IF A PIECE HAS BEEN CAPTURED BY
135              ;      A TRIAL MOVE, GENERATE REPLIES &
136              ;      EVALUATE THE EXCHANGE GAIN/LOSS
137              ;
138 016F 50 FD      TREE    BVC      RETJ      NO CAP
139 0171 A0 07              LDYIM     07        (PIECES)
140 0173 A5 B1              LDAZ      .SQUARE
141 0175 D9 60 00      LOOPX CMPAY     .BK
142 0178 F0 05              BEQ      FOUNX
143 017A 88              DEY
144 017B F0 F1              BEQ      RETJ      (KING)
145 017D 10 F6              BPL      LOOPX    SAVE
146 017F B9 A0 00      FOUNX LDAAY     .POINTS  BEST CAP
147 0182 D5 E2              CMPZX     .BCAPO   AT THIS
148 0184 90 02              BCC      NOMAX    LEVEL
149 0186 95 E2              STAZX     .BCAPO
150 0188 C6 B5      NOMAX  DEC      .STATE

```

```

151 018A A9 FB          LDAIM      FB          IF STATE=FB
152 018C C5 B5          CMPZ       .STATE     TIME TO TURN
153 018E F0 03          BEQ        UPTREE     AROUND
154 0190 20 25 03      JSR        GENRM     GENERATE FURTHER
155 0193 E6 B5          INC        .STATE     CAPTURES
156 0195 60
157                      ;
158                      ;      THE PLAYER'S MOVE IS INPUT
159                      ;
160 0196 C9 08          INPUT      CMPIM      08          NOT A LEGAL
161 0198 B0 12          BCS        ERROR     SQUARE #
162 019A 20 EA 03      JSR        DISMV
163 019D A2 1F          DISP      LDXIM      1F
164 019F B5 50          SEARCH    LDAZX     .BOARD
165 01A1 C5 FA          CMPZ       .DIS2
166 01A3 F0 03          BEQ        HERE      DISPLAY
167 01A5 CA            DEX              PIECE AT
168 01A6 10 F7          BPL        SEARCH   FROM
169 01A8 86 FB          HERE      STXZ      .DIS1  SQUARE
170 01AA 86 B0          STXZ      .PIECE
171 01AC 4C 00 00      ERROR     JMP        CHESS
172                      ;
173                      ;      GENERATE ALL MOVES FOR ONE
174                      ;      SIDE, CALL JANUS AFTER EACH
175                      ;      ONE FOR NEXT STEP
176                      ;
177                      +++
178 0200 A2 10          GNMZ      LDXIM      10          CLEAR
179 0202 A9 00          GNMX      LDAIM      00          COUNTERS
180 0204 95 DE          CLEAR    STAZX     .COUNT
181 0206 CA            DEX
182 0207 10 FB          BPL        CLEAR
183                      ;
184 0209 A9 10          GNM      LDAIM      10          SET UP
185 020B 85 B0          STAZ      .PIECE     PIECE
186 020D C6 B0          NEWP     DECZ      .PIECE  NEW PIECE
187 020F 10 01          BPL      NEX        ALL DONE?
188 0211 60            RTS              -YES
189                      ;
190 0212 20 1E 03      NEX      JSR        RESET   READY
191 0215 A4 B0          LDYZ      .PIECE     GET PIECE
192 0217 A2 08          LDXIM     08
193 0219 86 B6          STXZ      .MOVEN    COMMON START
194 021B C0 08          CPYIM     08        WHAT IS IT?
195 021D 10 41          BPL      PAWN      PAWN
196 021F C0 06          CPYIM     06
197 0221 10 2E          BPL      KNIGHT    KNIGHT
198 0223 C0 04          CPYIM     04
199 0225 10 1F          BPL      BISHOP    BISHOP
200 0227 C0 01          CPYIM     01

```

201	0229	F0	09		BEQ	QUEEN	QUEEN
202	022B	10	0E		BPL	ROOK	ROOK
203							
204	022D	20	8E	02	; KING	JSR	SNGMV
205	0230	D0	FB			BNE	KING
206	0232	F0	D9			BEQ	NEWP
207	0234	20	9C	02	QUEEN	JSR	LINE
208	0237	D0	FB			BNE	QUEEN
209	0239	F0	D2			BEQ	NEWP
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
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

MUST BE KING!

MOVES

8 TO 1

MOVES

8 TO 1

MOVES

4 TO 1

MOVES

8 TO 5

MOVES

16 TO 9

RIGHT CAP?

YES

LEFT CAP?

AHEAD

ILLEGAL

GETS TO

3RD RANK?

DO DOUBLE

```

251      ;
252      ;      CALCULATE SINGLE STEP MOVES
253      ;      FOR K, N
254      ;
255      028E 20 CA 02      SNGMV      JSR      CMOVE      CALC MOVE
256      0291 30 03      BMI      ILL1      -IF LEGAL
257      0293 20 00 01      JSR      JANUS      -EVALUATE
258      0296 20 1E 03      ILL1      JSR      RESET
259      0299 C6 B6      DECZ      .MOVEN
260      029B 60      RTS
261      ;
262      ;      CALCULATE ALL MOVES DOWN A
263      ;      STRAIGHT LINE FOR Q,B,R
264      ;
265      029C 20 CA 02      LINE      JSR      CMOVE      CALC MOVE
266      029F 90 02      BCC      OVL      NO CHK
267      02A1 50 F9      BVC      LINE      CH, NOCAP
268      02A3 30 07      OVL      BMI      ILL      RETURN
269      02A5 08      PHP
270      02A6 20 00 01      JSR      JANUS      EVALUATE POSN
271      02A9 28      PLP
272      02AA 50 F0      BVC      LINE      NOT A CAP
273      02AC 20 1E 03      ILL      JSR      RESET      LINE STOPPED
274      02AF C6 B6      DECZ      .MOVEN      NEXT DIR
275      02B1 60      RTS
276      ;
277      ;      EXCHANGE SIDES FOR REPLY
278      ;      ANALYSIS
279      ;
280      02B2 A2 0F      REVERSE      LDXIM      OF
281      02B4 38      ETC      SEC
282      02B5 B4 60      LDYZX      .BK      SUBTRACT
283      02B7 A9 77      LDAIM      77      POSITION
284      02B9 F5 50      SBCZX      .BOARD      FROM 77
285      02BB 95 60      STAZX      .BK
286      02BD 94 50      STYZX      .BOARD      AND
287      02BF 38      SEC
288      02C0 A9 77      LDAIM      77      EXCHANGE
289      02C2 F5 50      SBCZX      .BOARD      PIECES
290      02C4 95 50      STAZX      .BOARD
291      02C6 CA      DEX
292      02C7 10 EB      BPL      ETC
293      02C9 60      RTS
294      ;
295      ;
296      ;
297      ;
298      ;
299      ;
300      ;

```

```

301      ;      CMOVE CALCULATES THE TO SQUARE
302      ;      USING .SQUARE AND THE MOVE
303      ;      TABLE.  FLAGS SET AS FOLLOWS:
304      ;      N - ILLEGAL MOVE
305      ;      V - CAPTURE (LEGAL UNLESS IN CH)
306      ;      C - ILLEGAL BECAUSE OF CHECK
307      ;      [MY THANKS TO JIM BUTTERFIELD
308      ;      WHO WROTE THIS MORE EFFICIENT
309      ;      VERSION OF CMOVE]
310      ;
311      02CA A5 B1      CMOVE      LDAZ      .SQUARE      GET SQUARE
312      02CC A6 B6      LDXZ      .MOVEN      MOVE POINTER
313      02CE 18      CLC
314      02CF 75 8F      ADCZX      .MOVEX      MOVE LIST
315      02D1 85 B1      STAZ      .SQUARE      NEW POS'N
316      02D3 29 88      ANDIM      88
317      02D5 D0 42      BNE      ILLEGAL      OFF BOARD
318      02D7 A5 B1      LDAZ      .SQUARE
319      ;
320      02D9 A2 20      LDXIM      20
321      02DB CA      LOOP      DEX      IS TO
322      02DC 30 0E      BMI      NO      SQUARE
323      02DE D5 50      CMPZX      .BOARD      OCCUPIED?
324      02E0 D0 F9      BNE      LOOP
325      ;
326      02E2 E0 10      CPXIM      10      BY SELF?
327      02E4 30 33      BMI      ILLEGAL
328      ;
329      02E6 A9 7F      LDAIM      7F      MUST BE CAP!
330      02E8 69 01      ADCIM      01      SET V FLAG
331      02EA 70 01      BVS      SPX      (JMP)
332      ;
333      02EC B8      NO      CLV      NO CAPTURE
334      ;
335      02ED A5 B5      SPX      LDAZ      .STATE      SHOULD WE
336      02EF 30 24      BMI      RETL      DO THE
337      02F1 C9 08      CMPIM      08      CHECK CHECK?
338      02F3 10 20      BPL      RETL
339      ;
340      ;      CHKCHK REVERSES SIDES
341      ;      AND LOOKS FOR A KING
342      ;      CAPTURE TO INDICATE
343      ;      ILLEGAL MOVE BECAUSE OF
344      ;      CHECK.  SINCE THIS IS
345      ;      TIME CONSUMING, IT IS NOT
346      ;      ALWAYS DONE.
347      ;
348      02F5 48      CHKCHK      PHA      STATE
349      02F6 08      PHP
350      02F7 A9 F9      LDAIM      F9

```

351	02F9	85	B5		STAZ	.STATE	GENERATE
352	02FB	85	B4		STAZ	.INCHEK	ALL REPLY
353	02FD	20	4B	03	JSR	MOVE	MOVES TO
354	0300	20	B2	02	JSR	REVERSE	SEE IF KING
355	0303	20	09	02	JSR	GNM	IS IN
356	0306	20	2E	03	JSR	RUM	CHECK
357	0309	28			PLP		
358	030A	68			PLA		
359	030B	85	B5		STAZ	.STATE	
360	030D	A5	B4		LDAZ	.INCHEK	
361	030F	30	04		BMI	RETL	NO - SAFE
362	0311	38			SEC		YES - IN CHK
363	0312	A9	FF		LDAIM	FF	
364	0314	60			RTS		
365							
366	0315	18			RETL	CLC	LEGAL
367	0316	A9	00		LDAIM	00	RETURN
368	0318	60			RTS		
369							
370	0319	A9	FF		ILLEGAL	LDAIM	FF
371	031B	18			CLC		ILLEGAL
372	031C	B8			CLV		RETURN
373	031D	60			RTS		
374							
375							
376							
377	031E	A6	B0		RESET	LDXZ	.PIECE
378	0320	B5	50			LDAXX	.BOARD
379	0322	85	B1			STAZ	.SQUARE
380	0324	60			RTS		FROM BOARD
381							
382							
383							
384	0325	20	4B	03	GENRM	JSR	MOVE
385	0328	20	B2	02	GENR2	JSR	REVERSE
386	032B	20	09	02		JSR	GNM
387	032E	20	B2	02	RUM	JSR	REVERSE
388							
389							
390							
391							
392	0331	BA			UMOVE	TSX	UNMAKE MOVE
393	0332	86	B3			STXZ	.SP1
394	0334	A6	B2			LDXZ	.SP2
395	0336	9A				TXS	EXCHANGE
396	0337	68				PLA	STACKS
397	0338	85	B6			STAZ	MOVEN
398	033A	68				PLA	
399	033B	85	B0			STAZ	.PIECE
400	033D	AA				TAX	PIECE

401	033E	68		PLA		FROM SQUARE
402	033F	95	50	STAZX	.BOARD	
403	0341	68		PLA		PIECE
404	0342	AA		TAX		
405	0343	68		PLA		TO SQUARE
406	0344	85	B1	STAZ	.SQUARE	
407	0346	95	50	STAZX	.BOARD	
408	0348	4C	70 03	JMP	STRV	
409						
410						
411						
412						
413						
414						
415	034B	BA		TSX		
416	034C	86	B3	STXZ	.SP1	SWITCH
417	034E	A6	B2	LDXZ	.SP2	STACKS
418	0350	9A		TXS		
419	0351	A5	B1	LDAZ	.SQUARE	
420	0353	48		PHA		TO SQUARE
421	0354	A8		TAY		
422	0355	A2	1F	LDXIM	1F	
423	0357	D5	50	CMPZX	.BOARD	CHECK FOR
424	0359	F0	03	BEQ	TAKE	CAPTURE
425	035B	CA		DEX		
426	035C	10	F9	BPL	CHECK	
427	035E	A9	CC	LDAIM	CC	
428	0360	95	50	STAZX	.BOARD	
429	0362	8A		TXA		CAPTURED
430	0363	48		PHA		PIECE
431	0364	A6	B0	LDXZ	.PIECE	
432	0366	B5	50	LDAZX	.BOARD	
433	0368	94	50	STYZX	.BOARD	FROM
434	036A	48		PHA		SQUARE
435	036B	8A		TXA		
436	036C	48		PHA		PIECE
437	036D	A5	B6	LDAZ	.MOVEN	
438	036F	48		PHA		MOVEN
439	0370	BA		TSX		
440	0371	86	B2	STXZ	.SP2	SWITCH
441	0373	A6	B3	LDXZ	.SP1	STACKS
442	0375	9A		TXS		BACK
443	0376	60		RTS		
444						
445						
446						
447						
448						
449	0377	A6	E4	LDXZ	.BMAXC	CAN BLK CAP
450	0379	E4	A0	CPXZ	.POINTS	MY KING?

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

MOVE

CHECK

TAKE

STRV

CKMATE

```

451 037B D0 04      BNE      NOCHEK
452 037D A9 00      LDAIM     00      GULP!
453 037F F0 0A      BEQ      RETV      DUMB MOVE!
454
455 0381 A6 E3      ; NOCHEK      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      ; THE VALUE OF THE MOVE (IN ACC)
466      ; IS COMPARED TO THE BEST MOVE AND
467      ; REPLACES IT IF IT IS BETTER
468 038F C5 FA      PUSH      CMPZ      .BESTV     IS THIS BEST
469 0391 90 0C      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      ; MAIN PROGRAM TO PLAY CHESS
480      ; PLAY FROM OPENING OR THINK
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     .OPNING     OPPONENT'S
485 03AA D0 0F      BNE      END      MOVE OK?
486 03AC CA      DEX
487 03AD B5 DC      LDAZX     .OPNING     GET NEXT
488 03AF 85 FB      STAZ      .DIS1      CANNED
489 03B1 CA      DEX      OPENING MOVE
490 03B2 B5 DC      LDAZX     .OPNING
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 85 DC      ; END      STAZ      .OMOVE     FLAG OPENING
497 03BD A2 0C      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

```

501	03C5 20 02 02		JSR	GNMX	MOVES
502		;			
503	03C8 A2 04		LDXIM	04	STATE=4
504	03CA 86 B5		STXZ	.STATE	GENERATE AND
505	03CC 20 00 02		JSR	GNMZ	TEST AVAILABLE
506		;			MOVES
507		;			
508	03CF A6 FA		LDXZ	.BESTV	GET BEST MOVE
509	03D1 E0 0F		CPXIM	0F	IF NONE
510	03D3 90 12		BCC	MATE	OH OH!
511		;			
512	03D5 A6 FB	MV2	LDXZ	.BESTP	MOVE
513	03D7 B5 50		LDAZX	.BOARD	THE
514	03D9 85 FA		STAZ	.BESTV	BEST
515	03DB 86 B0		STXZ	.PIECE	MOVE
516	03DD A5 F9		LDAZ	.BESTM	
517	03DF 85 B1		STAZ	.SQUARE	AND DISPLAY
518	03E1 20 4B 03		JSR	MOVE	IT
519	03E4 4C 00 00		JMP	CHESS	
520		;			
521	03E7 A9 FF	MATE	LDAIM	FF	RESIGN
522	03E9 60		RTS		OR STALEMATE
523		;			
524		;			
525		;			
526		;			
527	03EA A2 04	DISMV	LDXIM	04	ROTATE
528	03EC 06 F9	ROL	ASLZ	.DIS3	KEY
529	03EE 26 FA		ROLZ	.DIS2	INTO
530	03F0 CA		DEX		DISPLAY
531	03F1 D0 F9		BNE	ROL	
532	03F3 05 F9		ORAZ	.DIS3	
533	03F5 85 F9		STAZ	.DIS3	
534	03F7 85 B1		STAZ	.SQUARE	
535	03F9 60		RTS		
536		;			
537		;			
538		;			
539		;			
540		;			
541		;			
542		;			
543	1780 18	STRATGY	+++		
544	1781 A9 80		CLC		
545	1783 65 EB		LDAIM	80	
546	1785 65 EC		ADCZ	.WMOB	PARAMETERS
547	1787 65 ED		ADCZ	.WMAXC	WITH WEIGHT
548	1789 65 E1		ADCZ	.WCC	OF 0.25
549	178B 65 DF		ADCZ	.WCAP1	
550	178D 38		ADCZ	.WCAP2	
			SEC		

551	178E	E5	F0		SBCZ	.PMAXC	
552	1790	E5	F1		SBCZ	.PCC	
553	1792	E5	E2		SBCZ	.BCAP0	
554	1794	E5	E0		SBCZ	.BCAP1	
555	1796	E5	DE		SBCZ	.BCAP2	
556	1798	E5	EF		SBCZ	.PMOB	
557	179A	E5	E3		SBCZ	.BMOB	
558	179C	B0	02		BCS	POS	UNDERFLOW
559	179E	A9	00		LDAIM	00	PREVENTION
560	17A0	4A		POS	LSRA		
561	17A1	18			CLC		*****
562	17A2	69	40		ADCIM	40	
563	17A4	65	EC		ADCZ	.WMAXC	PARAMETERS
564	17A6	65	ED		ADCZ	.WCC	WITH WEIGHT
565	17A8	38			SEC		OF 0.5
566	17A9	E5	E4		SBCZ	.BMAXC	
567	17AB	4A			LSRA		*****
568	17AC	18			CLC		
569	17AD	69	90		ADCIM	90	
570	17AF	65	DD		ADCZ	.WCAPO	PARAMETERS
571	17B1	65	DD		ADCZ	.WCAPO	WITH WEIGHT
572	17B3	65	DD		ADCZ	.WCAPO	OF 1.0
573	17B5	65	DD		ADCZ	.WCAPO	
574	17B7	65	E1		ADCZ	.WCAP1	
575	17B9	38			SEC		[UNDER OR OVER-
576	17BA	E5	E4		SBCZ	.BMAXC	FLOW MAY OCCUR
577	17BC	E5	E4		SBCZ	.BMAXC	FROM THIS
578	17BE	E5	E5		SBCZ	.BCC	SECTION]
579	17C0	E5	E5		SBCZ	.BCC	
580	17C2	E5	E0		SBCZ	.BCAP1	
581	17C4	A6	B1		LDXZ	.SQUARE	*****
582	17C6	E0	33		CPXIM	33	
583	17C8	F0	16		BEQ	POSN	POSITION
584	17CA	E0	34		CPXIM	34	BONUS FOR
585	17CC	F0	12		BEQ	POSN	MOVE TO
586	17CE	E0	22		CPXIM	22	CENTRE
587	17D0	F0	0E		BEQ	POSN	OR
588	17D2	E0	25		CPXIM	25	OUT OF
589	17D4	F0	0A		BEQ	POSN	BACK RANK
590	17D6	A6	B0		LDXZ	.PIECE	
591	17D8	F0	09		BEQ	NOPOSN	
592	17DA	B4	50		LDYZX	.BOARD	
593	17DC	C0	10		CPYIM	10	
594	17DE	10	03		BPL	NOPOSN	
595	17E0	18		POSN	CLC		
596	17E1	69	02		ADCIM	02	
597	17E3	4C	77 03	NOPOSN	JMP	CKMATE	CONTINUE
598							
599							
600							

CHESS	0000	5	1	45	171	519
OUT	0008	15	18			
WHSET	001A	24	27			
NOSET	0027	32	22			
NOREV	0032	38	33			
CLDSP	0039	42	30	36		
NOGO	0041	47	39			
NOMV	004B	51	48			
JANUS	0100	60	236	245	257	270
COUNTS	0104	67				
OVER	0110	74	68	70		
NOQ	0118	79	76			
ELOOP	011E	82	85			
FOUN	0126	86	83			
LESS	0131	92	88			
NOCAP	0138	98	79			
XRT	013E	101	72			
ON4	013F	106	99			
NOCOUNT	0160	121	61			
RETJ	016E	132	129	138	144	
TREE	016F	138	100	122		
LOOPX	0175	141	145			
FOUNX	017F	146	142			
NOMAX	0188	150	148			
UPTREE	0193	155	153			
INPUT	0196	160	51			
DISP	019D	163	50			
SEARCH	019F	164	168			
HERE	01A8	169	166			
ERROR	01AC	171	161			
GNMZ	0200	178	112	505		
GNMX	0202	179	501			
CLEAR	0204	180	182			
GNM	0209	184	117	355	386	
NEWP	020D	186	206	209	215	221 229 243 244 250
NEX	0212	190	187			
KING	022D	204	205			
QUEEN	0234	207	201	208		
ROOK	023B	211	202			
AGNR	023F	213	214			
BISHOP	0246	217	199	220		
KNIGHT	0251	223	197			
AGNN	0255	225	228			
PAWN	0260	231	195			
P1	0264	233	241			
P2	026E	237	234	235		
P3	0279	242	249			
SNGMV	028E	255	204	225		
ILL1	0296	258	256			
LINE	029C	265	207	213	217	267 272
OVL	02A3	268	266			
ILL	02AC	273	268			
REVERSE	02B2	280	34	111	113	354 385 387
ETC	02B4	281	292			
CMOVE	02CA	311	233	242	255	265
LOOP	02DB	321	324			
NO	02EC	333	322			

SYMBOL	ADDR	DEF	CROSS REFERENCES
SPX	02ED	335	331
CHKCHK	02F5	348	
RETL	0315	366	336 338 361
ILLEGAL	0319	370	317 327 343
RESET	031E	377	190 237 258 273
GENRM	0325	384	154
GENR2	0328	385	
RUM	032E	387	356
UMOVE	0331	392	118
MOVE	034B	415	49 110 353 384 518
CHECK	0357	423	426
TAKE	035E	427	424
STRV	0370	439	408
CKMATE	0377	449	597
NOCHEK	0381	455	451
RETV	038B	461	453 456 458
PUSH	038F	468	
RETP	039F	476	469 470
GO	03A2	481	40
END	03BB	496	485
NOOPEN	03BD	497	482
MV2	03D5	512	494
MATE	03E7	521	510
DISMV	03EA	527	162
ROL	03EC	528	531
STRATGY	1780	543	120
POS	17A0	560	558
POSN	17E0	595	583 585 587 589
NOPOSN	17E3	597	591 594
.BOARD	0050	602	25 164 284 286 289 290 323 378 402 407 423 428 432 433 513 592
.BK	0060	603	82 127 141 282 285
.SETW	0070	604	24
.MOVEX	008F	605	314
.POINTS	00A0	606	86 146 450
.PIECE	00B0	607	67 170 185 186 191 377 399 431 472 515 590
.SQUARE	00B1	608	81 128 140 246 311 315 318 379 406 419 474 517 534 581
.SP2	00B2	609	9 394 417 440
.SP1	00B3	610	393 416 441
.INCHEK	00B4	611	131 352 360
.STATE	00B5	612	60 109 116 150 152 155 335 351 359 462 498 504
.MOVEN	00B6	613	193 212 218 224 226 232 238 239 259 274 312 397 437
.OMOVE	00DC	614	28 481 493 496
.OPNING	00DC	615	484 487 490
.WCAPO	00DD	616	107 570 571 572 573
.COUNT	00DE	617	180
.BCAP2	00DE	618	555
.WCAP2	00DF	619	549
.BCAP1	00E0	620	554 580
.WCAP1	00E1	621	548 574
.BCAPO	00E2	622	147 149 553
.MOB	00E3	623	74 77
.MAXC	00E4	624	87 90
.CC	00E5	625	94 95

SYMBOL	ADDR	DEF	CROSS REFERENCES
.PCAP	00E6	626	89
.BMOB	00E3	627	455 557
.BMAXC	00E4	628	449 566 576 577
.BCC	00E5	629	578 579
.BMAXP	00E6	630	71
.XMAXC	00E8	631	106
.WMOB	00EB	632	545
.WMAXC	00EC	633	546 563
.WCC	00ED	634	547 564
.WMAXP	00EE	635	457
.PMOB	00EF	636	556
.PMAXC	00F0	637	551
.PCC	00F1	638	552
.PCP	00F2	639	
.OLDKY	00F3	640	17 19
.BESTP	00FB	641	473 512
.BESTV	00FA	642	468 471 499 508 514
.BESTM	00F9	643	475 516
.DIS1	00FB	644	42 169 488
.DIS2	00FA	645	43 165 529
.DIS3	00F9	646	44 483 491 528 532 533
*OUT	1F1F	647	15 476
*GETKEY	1F6A	648	16

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 04 02 02 02 02 02 02 02 02
.OPNING	00C0	99 25 0B 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.

Micro-ADE 6502

ASSEMBLER

This flexible two pass assembler can be used to assemble small programs directly in memory, or with up to two computer controlled cassettes for easy handling of large programs. The allocation of memory to the source, object, and symbol table is user defined. The symbol table may be listed at any time in alphabetical or address order. Efficient packed ASCII coding reduces the memory required by the symbol table. Error messages warn you of mistakes before the program crashes the system.

DISASSEMBLER

The disassembler translates object code into assembler source language. Symbolic arguments and labels are defined from the symbol table. The assembler symbol table can be saved at assembly time for use with the disassembler for easy debugging. Relocation of undocumented programs becomes a snap. Use this disassembler once, and you'll never look at a hex dump again!

EDITOR

Quick edit features include the FIX, INSERT, MOVE, and DELETE commands. Lines are automatically numbered. Cassette commands: GET, SAVE, and REPRODUCE simplify the editing of multiple file source programs on cassettes. A page mode formats the output for CRT terminals to allow easy viewing of long listings.

Micro-ADE is a well documented package of programs which may be used with any 6502 microcomputer system. The comprehensive 56 page user manual includes the full source listing for all input/output and KIM cassette I/O routines enabling you to interface your own peripheral devices with ease. All programs and utility routines coreside in 4K. Schematics are included for automatic control of two cassette recorders.

Full documentation is available from Micro-Ware Limited. The User Manual, hex dump, and object program on paper tape or KIM cassette costs only \$25.00. The complete annotated source listing is also available for an additional \$25.00. This is the program development tool that you have been waiting for. Send today to:

Micro-Ware Ltd 27 FIRSTBROOKE ROAD, TORONTO, ONTARIO, CANADA. M4E 2L2.

Micro-Ware Limited
27 Firstbrooke Road
Toronto Ontario
Canada M4E 2L2