An Upgrade for KIM MICROCHESS 1.0

Garold R. Stone
P.O. Box 153
Annapolis Junction, MD. 20701

If you have Peter Jennings' MICROCHESS program for the KIM-1 microcomputer you can teach it to play a significantly better game of chess without adding a single byte of expansion memory. This article describes a "patch" I have written for MICROCHESS which gives the computer a more flexible opening game and two new strategies for the middle and end game. Just load your copy of MICROCHESS, enter my code from the accompanying program listing along with the chess opening sample from table one, and play chess. There are no changes in the way you run the program. (For a description of the MICROCHESS program see KB, August 1978, page 74). For clarity I will use the term MICROCHESS only to refer to the original program as written by Peter Jennings. I will say "patch" to refer to the changes I am describing here.

Off the Shelf

The MICROCHESS I bought from Micro-ware Ltd. opens the game by playing from a pre-selected list of moves for a user chosen chess opening (Roy Lopez, French Defence, etc.). That opening list also contains one anticipated opponent move for each computer move. Things go well as long as the opponent makes the anticipated replies. But a human opponent seldom does that -- at least I don't. As soon as I make a novel move MICROCHESS permanently abandons the opening list. Whenever MICROCHESS is forced to quit the opening list too early, coherent development of pieces stops, the queen usually comes out too early, an ill-prepared attack is launched, and the computer loses its ability to castle (because castling is only possible from the opening list).

Compromises in 1.1K

Mr. Jennings points to these problems in his excellent documentation manual:

> "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."

The single strategy used by MICROCHESS is best suited for the middle game, where the capture of pieces dominates. In order to add a dynamic opening strategy which would emphasize the development and positioning of pieces, I had to settle for my own set of compromises, as you'll see. I should point out that Mr. Jennings seems to have surmounted this problem in the other versions of MICROCHESS he has written for microcomputers with more memory, such as the PET, TRS-80, and the APPLE.

The Opening

Table 1 shows my data format for eight opening development moves. Unlike in MICROCHESS, anticipated opponent replies are **not** listed. On each turn the **patched** program evaluates all of the computer's available moves. The available move which comes out with the highest evaluation is compared with the evaluation for the next legal move in my opening list and the higher of the two is selected as the computer's move for that turn. The development move is usually selected because its evaluation is always boosted by a threshold factor. I set the threshold factor high enough so that only moves with a significantly higher evaluation can override the development move. The higher the threshold, the more likely it is that the development move will be selected for that turn. Thus, the computer follows an opening game plan, responds to significant attack threats or capture opportunities, and then continues to carry out the opening game plan on the next turn by consulting the opening list again.

Books on chess openings and opening game strategy can serve as guides in writing new lists of development moves. Choose openings which are general in nature and do not depend on specific moves by the opponent. Specify each development move by giving the piece (variable DEVP), the square of origin (FROM), and the destination (TO), using the same notation as in MICROCHESS (see tables 2 and 3). Openings for white and black will require separate notation. Fill all unused locations in the opening list with the magic number 1F (hexadecimal), which causes those locations to be skipped because they are off the board.

Castling

As in MICROCHESS the computer's castling move must be completed for it by moving its rook after the computer signals castling by moving its King the necessary two squares. My added programming will prevent castling if the computer's King is off its starting square or if it would end up in check. The other rules for castling are not checked, however. If the computer castles illegally, then the move must be refereed. The simplest way is to use the "touch-move" rule -- once a player touches a piece it

Table 1
Opening Move Data

ADDR	VARIABLE	MOVE	WHITE	BLACK	COMMENT
00C3	.FACTOR		05	05	THRESHOLD FACTOR
00C4	.DEVP-1	N-KB3	06	06	PIECE
00C5	.FROM		01	06	ORIGIN
00C6	.TO		22	25	DESTINATION
00C7	.DEVP-2	P-KN3	0A	0A	PIECE
00C8	.FROM		11	16	ORIGIN
00C9	.TO		21	26	DESTINATION
00CA	.DEVP-3	B-KN2	04	04	PIECE
00CB	.FROM		02	05	ORIGIN
00CC	.TO		11	16	DESTINATION
00CD	.DEVP-4	P-K3	0F	0F	PIECE
00CE	.FROM		13	14	ORIGIN
00CF	.TO		23	24	DESTINATION
00D0	.DEVP-5	0-0	00	00	PIECE (KING SIDE CASTLE)
00D1	.FROM		03	04	ORIGIN
00D2	.TO		01	06	DESTINATION
00D3	.DEVP-6	K-QB3	07	07	PIECE
00D4	.FROM		06	01	ORIGIN
00D5	.TO		25	22	DESTINATION
00D6	.DEVP-7	P-Q4	0E	0E	PIECE
00D7	.FROM		14	13	ORIGIN
00D8	.TO		34	33	DESTINATION
00D9	.DEVP-8	(NO	1F	1F	
00DA	.FROM	(MOVE)	1F	1F	
00DB	.TO		1F	1F	

See Tables 2 and 3 for coding of Pieces and Squares

must be moved. Thus, the computer would have to move its King somewhere else, and you would enter that move for it. If there are no legal moves left for the King, then the computer must resign. This situation seldom comes up because I write openings which castle early enough to avoid the risk and annoyance of an illegal attempt.

Program Flow

What follows is a description of how the patched program works. MICROCHESS subroutines which are not defined in my accompanying program listing are in bold letters.

Whenever it is the computer's turn to move, MICROCHESS command loop **CHESS** calls my version of subroutine GO (see 03A2 in the program listing). MICROCHESS uses the value of a variable called STATE to keep track of what it's doing. State 4 guides the generation and evaluation of the computer's available moves. There are other states for generating potential opponent replies, etc. MICROCHESS subroutine **GNMX** (see 03AA) initializes some variables called "counts" for evaluating moves and then generates all moves available to the computer on that turn. **GNMX** calls MICROCHESS subroutine **JANUS** to calculate and evaluate the counts for each trial move. Based on the value in STATE, **JANUS** decides what to do next -- generate potential opponent replies for evaluation, calculate exchanges of pieces, etc. **JANUS** changes the value in STATE as it goes.

Table 2
Microchess Piece Notation and Storage

		MEMORY LOCATION	
CODE	PIECE	COMPUTER	OPPONENT
00	KING	0050	0060
01	QUEEN	0051	0061
02	KING ROOK	0052	0062
03	QUEEN ROOK	0053	0063
04	KING BISHOP	0054	0064
05	QUEEN BISHOP	0055	0065
06	KING KNIGHT	0056	0066
07	QUEEN KNIGHT	0057	0067
08	KR PAWN	0058	0068
09	QR PAWN	0059	0069
0A	KN PAWN	005A	006A
0B	QN PAWN	005B	006B
0C	KB PAWN	005C	006C
0D	QB PAWN	005D	006D
0E	Q PAWN	005E	006E
0F	K PAWN	005F	006F

Table 3
Board 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

OPPONENT

Note: Whether playing White or Black, the Computer's starting squares arc always 00 through 17. Be sure to orient the playing board so that the lower left corner is black. The White Queen should be on a white square and the Black Queen should be on a black square.

Table 4
New Variables Used

ADDR	VARIABLE	COMMENT
00C3	.FACTOR	Threshold factor for opening moves
00DC	.OMOVE	MICROCHESS opening move flag
00DC	.OMOVE	Base for opening move array
00EF	.BKMOB	Number of legal moves for Opponent King
00F0	.BIAS	Receives threshold factor for legal list move

JANUS and portions of **GNMX** call each other recursively, again and again, until all of the computer's available moves have been evaluated in the light of all possible opponent replies. By the time program control returns from that very first call to subroutine **GNMX**, one move has emerged with an evaluation higher than all the others.

Then my patch searches the opening move list from the beginning to find the first piece (variable DEVP) which is still where it is supposed to be (FROM) (see 03B1). The move by this piece to its destination (TO) is checked for legality by a call into the middle of MICROCHESS subroutine **CMOVE.**

If the list move is legal, then the threshold factor (FACTOR) is stored in the variable BIAS for later use (see 03D8). MICROCHESS subroutine **JANUS** is called to do the counts for this list move and for the opponent's potential replies.

To evaluate these counts **JANUS** calls up my version of subroutine STRATEGY (see 1780-17C1). This is where the evaluation of the list move is boosted by adding the threshold factor which was stored earlier in the variable BIAS. Actually, this same subroutine STRATEGY is used by **JANUS** to evaluate any trial move but BIAS is always zero except for legal list moves. If the selected list move is not legal, then **JANUS** is not called to evaluate it, and no more list moves will be tried for that turn. This ensures that moves from the opening list are made in the order you wrote them. After the last list move has actually been moved, the variable OMOVE is set to zero and the opening list is ignored for the rest of the game (see 03AF).

As you exit subroutine STRATEGY you enter that portion of MICROCHESS which compares the evaluation of the current trial move with that of the best move so far, saving the better of the two as the new best move so far. This is also where MICROCHESS tests for check or checkmate before returning to **JANUS.** Control then passes to the MICROCHESS subroutine which takes the best trial move and actually moves it (see 03E3). The computer's move is flashed on the KIM display and the program returns to the MICROCHESS command loop, ready for the opponent to enter his move.

Middle and End Game

MICROCHESS sees only one and a half moves ahead. With this limited horizon it has trouble finding and closing in on the opposing King. To compensate for this I give a bonus of two points for moves inside a zone which surrounds the opposing King and moves along with it. The computer's Pawns and King do not get the bonus (see 179D).

Another strategy encourages moves which hem in the opposing King, in preparation for checkmate. The value of any trial move is decreased by the number of safe moves it leaves for the opposing King. This is the same as adding a point for each square denied to the opposing King. Since MICROCHESS calls subroutine **JANUS** to evaluate only legal moves, it was easy enough to put a subroutine call inside **JANUS** which would increment a mobility count (BKMOB) for each legal move found for the opponent King when the computer is checking for opponent reply moves during state zero (see 0112, 17D9,179A).

Both strategies come into play only after the opening list has been emptied, so as not to interfere with the development of pieces during the opening game (see 1796).

Evaluation

I approached move evaluation in much the same way as in MICROCHESS -- adding and subtracting weighted counts representing captures, position, and mobility for both sides. I did not use some of the counts generated by MICROCHESS and I created the new ones I described above. Given the severe memory restrictions, my goal was an evaluation formula which emphasizes immediate and tangible factors, such as position and the values of pieces captureable during the current turn. Less immediate factors, such as overall attack strengths, are given fractional weighting. These become influential only after more significant factors have cancelled each other out.

For now I've had to be satisfied with just breaking MICROCHESS of its habit of throwing away its pieces by occasionally making bad decisions about captures where pieces are exchanged. In my patch any piece the computer wants to capture must be greater than or equal to the most valuable piece the computer would lose by making that move (variable BMAXC). Only trial moves which pass this admittedly simplistic test are given an extra 20 hex points (see 17B1). There is more that could be done, like making better use of the MICROCHESS counts for exchanges involving up to three captures per side.

I hope I've made my point. All you need is a shoe horn and you can slip just about any changes you want into the 1.1K KIM MICROCHESS. You may pinch a few toes in the process, but the result is a KIM that plays better chess. By trying to "upgrade" MICROCHESS I really learned to appreciate what an excellent piece of work it is.

MICROCHESS is available on KIM cassette with documentation manual from Micro-Ware Ltd., 496 Albert St., Suite 7, Waterloo, Ontario, Canada, N2L 3V4

Abbreviated Instructions for Loading and Running MICROCHESS 1.0 UPGRADE

Load:

Enter (RS) to reset KIM
Enter (AD) 00F1 (DA) 00 to reset decimal flag
Enter (AD) 17F9 (DA) C1 to enter tape ID for program segment
Enter (AD) 1873 (GO) to start read routine of KIM
Press "Play" on cassette player
STOP recorder when display shows: 0000
Enter (RS) (AD) 1873 (GO) to read second program segment (same label "C1")
STOP recorder when display shows: 0000
Enter (RS) (GO) to start program execution

Playing:

Enter (C) on KIM hexpad keyboard to reset program for new game
Enter (PC) (for "play chess") because KIM plays first
After KIM gives its move, enter your move as FROM-TO according to the board notations in table 3 of the article. Keep typing until your move shows correctly, then enter (F) (PC).

```
                    0110            .BA $3A2
03A2- A2 04         0120 GO         LDX #$04      ; RESET BEST EVALUATION
03A4- 86 FA         0130            STX *BESTV    ;   SO FAR
03A6- 86 B5         0140            STX *STATE    ; STATE = 4; TRAIL MOVES
03A8- A2 12         0150            LDX #$12      ; ZERO COUNTERS & BIAS
03AA- 20 02 02      0160            JSR GNMX      ; GENERATE TRAIL MOVES
03AD- A4 DC         0170            LDY *OMOVE    ; OPENING LIST DONE?
03AF- 10 32         0180            BPL NODEVP    ;   - YES, MID-GAME
03B1- A0 E6         0190            LDY #$E6      ;   - NO, NEXT DEVP
03B3- C8            0200 NEXT       INY
03B4- C8            0210            INY   ;         INDEX OF DEVP
03B5- 84 DC         0220            STY *OMOVE    ; OPENING LIST EMPTY?
03B7- 10 2A         0230            BPL NODEVP    ;   - YES, MID-GAME
03B9- B6 DC         0240            LDX *DEVP,Y   ;   -NO, NEXT DEVP
03BB- 86 B0         0250            STX *PIECE
03BD- B5 50         0260            LDA *BOARD,X       ; DEVP LOCATION
03BF- C8            0270            INY   ;         INDEX OF  FROM
03C0- 48            0280            PHA   ;         (SAVE DEVP LOCATION)
03C1- 98            0290            TYA   ;         TRANSFER INDEX OF
03C2- AA            0300            TAX   ;           FROM INTO X
03C3- 68            0310            PLA   ;         DEVP LOCATION IN ACCUM
03C4- D5 DC         0320            CMP *FROM,X   ; DEVP AT ORIGIN?
03C6- D0 EB         0330            BNE NEXT      ;   - NO, GET NEW DEVP
03C8- E8            0340            INX   ;         INDEX OF  TO
03C9- B5 DC         0350            LDA *TO,X     ; CHECK LEGALLITY OF DEVP
03CB- 20 D1 02      0360            JSR CMOVE     ; FROM .FROM TO .TO
03CE- 30 13         0370            BMI NODEVP    ; NEQ = ILLEGAL MOVE
03D0- A6 B0         0380            LDX *PIECE    ;   - LEGAL MOVE
03D2- E0 08         0390            CPX #$08      ; IS PIECE A PAWN
03D4- 30 02         0400            BMI LEGAL     ;   NEG = NOT PAWN
03D6- 70 0B         0410            BVS NODEVP    ; SET = ILLEGAL PAWN CAPTURE
03D8- A6 C3         0420 LEGAL      LDX *FACTOR   ; LEGAL OPENING MOVE!!
03DA- 86 F0         0430            STX *BIAS     ; SET BIAS TO FACTOR
03DC- A2 04         0440            LDX #$04      ; EVALUATE OPENING MOVE
03DE- 86 B5         0450            STX *STATE    ;   AND PUT IT IN BESTV
03E0- 20 00 01      0460            JSR JANUS     ;   IF ITS THE BEST MOVE
03E3- A6 FA         0470 NODEVP     LDX *BESTV    ;   SO FAR
03E5- E0 0F         0480            CPX #$0F      ; RESIGN OR STALEMATE IF
03E7- 4C C2 17      0490            JMP CONT      ;   BESTV TOO LOW
                    0500 ;
                    0510            .BA $17C2
17C2- 90 12         0520 CONT       BCC MATE      ; (ORIGINAL MICROCHESS
17C4- A6 FB         0530 MV2        LDX *BESTP    ;   CODING)
17C6- B5 50         0540            LDA *BOARD,X       ; MOVE AND DISPLAY THE
17C8- 85 FA         0550            STA *BESTV    ;   BEST MOVE
17CA- 86 B0         0560            STX *PIECE
17CC- A5 F9         0570            LDA *BESTM
17CE- 85 B1         0580            STA *SQUARE
17D0- 20 4B 03      0590            JSR MOVE
17D3- 4C 00 00      0600            JMP CHESS     ; END COMPUTER'S TURN
17D6- A9 FF         0610 MATE       LDA #$FF      ; RESIGN OR
17D8- 60            0620            RTS
                    0630 ;
                    0640            .BA $1780
1780- A9 80         0650 STRATEGY   LDA #$80      ; EVALUATION = 80 + OR - SCORE
1782- 18            0660            CLC
1783- 65 EB         0670            ADC *WMOB     ; COMPUTERS'S MOBILITY
1785- 4A            0680            LSR A
1786- 18            0690            CLC
1787- 69 40         0700            ADC #$40      ; RESET EVAL TO 80 +OR- SCORE
1789- 65 ED         0710            ADC *WCC      ; COMPUTER'S ATTACK STRENGTH
```

```
178B- 38        Ø72Ø          SEC
178C- E5 E5     Ø73Ø          SBC *BCC     ; OPPONENT'S ATTACK STRENGTH
178E- 4A        Ø74Ø          LSR A
178F- 4A        Ø75Ø          LSR A        ; MOBILITY X 1/16
179Ø- 4A        Ø76Ø          LSR A        ; ATTACK STRENGTH X 1/8
1791- 18        Ø77Ø          CLC
1792- 69 7Ø     Ø78Ø          ADC #$7Ø     ; RESET EVAL TO 8Ø +OR- SCORE
1794- 65 FØ     Ø79Ø          ADC *BIAS    ; ZERO UNLESS DEVP MOVE
1796- A4 DC     Ø8ØØ          LDY *OMOVE   ; NEGATIVE IF STILL DEVP
1798- 3Ø 17     Ø81Ø          BMI CAPTEST  ; MID-GAME IF POSITIVE
179A- 38        Ø82Ø          SEC  ;         DEDUCT MOBILITY OF THE
179B- E5 EF     Ø83Ø          SBC *BKMOB   ;   OPPONENT'S KING
179D- A6 BØ     Ø84Ø          LDX *PIECE   ; BONUS FOR MOVE INTO
179F- CA        Ø85Ø          DEX  ;          OPPONENT'S KING ZONE
17AØ- EØ Ø7     Ø86Ø          CPX #$Ø7     ; NOT FOR COMPUTER'S KING
17A2- BØ ØD     Ø87Ø          BCS CAPTEST  ;   OR PAWNS
17A4- 48        Ø88Ø          PHA  ;         (SAVE EVALUATION)
17A5- A5 6Ø     Ø89Ø          LDA *BK      ; LOCATION OF OPPONENT'S KING
17A7- 38        Ø9ØØ          SEC
17A8- E9 38     Ø91Ø          SBC #$38     ; CALCULATE KING ZONE
17AA- C5 B1     Ø92Ø          CMP *SQUARE  ; MOVE INTO ZONE?
17AC- 68        Ø93Ø          PLA  ;         (RESTORE EVALUATION)
17AD- BØ Ø2     Ø94Ø          BCS CAPTEST  ; CARRY CLEAR IS IN ZONE
17AF- 69 Ø2     Ø95Ø          ADC #$Ø2     ;   ADD BONUS, NEAR KING
17B1- A6 DD     Ø96Ø CAPTEST  LDX *WCAPØ   ; IF COMPUTER'S CAPTURE
17B3- E4 E4     Ø97Ø          CPX *BMAXC   ;   IS NOT GREATER THAN
17B5- 9Ø Ø3     Ø98Ø          BCC QUIT     ;   OR EQUAL OPP, QUIT
17B7- 18        Ø99Ø MOVEOK   CLC  ;         PASSES CAPTURE TEST
17B8- 69 2Ø     1ØØØ          ADC #$2Ø     ; POINTS FOR GOOD MOVE
17BA- 65 DD     1Ø1Ø QUIT     ADC *WCAPØ   ; POINTS FOR CAPTURE
17BC- 38        1Ø2Ø          SEC  ;         POINTS FOR OPPONENT'S
17BD- E5 E4     1Ø3Ø          SBC *BMAXC   ;   MAX CAPTURE IN REPLY
17BF- 4C 77 Ø3  1Ø4Ø          JMP CKMATE   ; TEST FOR CHECKMATE
                1Ø5Ø ;
                1Ø6Ø          .BA $17D9
17D9- DØ Ø6     1Ø7Ø BKMOVE   BNE OUTBK    ; RTS IF STATE NOT ZERO
17DB- C9 ØØ     1Ø8Ø          CMP #$ØØ     ; RTS IF NOT OPP KING'S
17DD- DØ Ø2     1Ø9Ø          BNE OUTBK    ;   MOVE
17DF- E6 EF     11ØØ          INC *BKMOB   ; COUNT LEGAL OPP KING
17E1- 6Ø        111Ø OUTBK    RTS  ;         MOVES
                112Ø ;
                113Ø          .BA $Ø112
Ø112- EØ ØØ     114Ø          CPX #$ØØ     ;COUNT LEGAL REPLY MOVES
Ø114- 2Ø D9 17  115Ø          JSR BKMOVE   ;   FOR OPPONENT'S KING
Ø117- EA        116Ø          NOP
                117Ø ;
                118Ø          .BA $2ØØ
Ø2ØØ- A2 11     119Ø          LDX #$11     ; CLEAR COUNTERS, NOT BIAS
                12ØØ          .EN
```

©