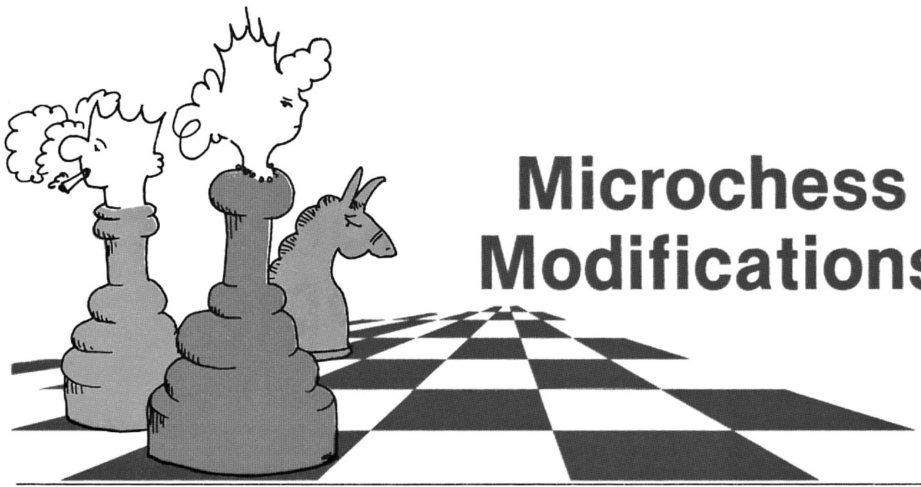

Microchess Modifications

Enhance this computer chess game with these modifications.

Chris McCormack
116 Milburn Lane
East Hills NY 11577

Microchess is a chess-playing program, written by Peter Jennings, designed to run on a 6502-based microcomputer system. Though it was configured for the Commodore KIM-1, the documentation contains suggestions on how it may be modified to run on other 6502 systems.

Remarkable about Microchess is that it can run on an unexpanded KIM, using only 1K of RAM. Though certain special moves (castling, en passant pawn captures, queening eighth rank pawns, etc.) cannot be made by the computer, it is possible for a human player to make these moves.

Poor Computer Play

Microchess was one of the first pieces of software that I was able to run on my KIM. I was elated to find that the program actually worked. For the next several weeks, I spent most of my free time sitting at the computer, playing one game of chess after another, never giving a moment's thought to how the program might operate.

After a while I became dissatisfied with the way the computer was playing. One of the main problems I noticed was that the computer would often move its queen time and time again, almost totally disregarding the other pieces at its disposal.

My second complaint was related to this. Because the computer was so preoccupied with its queen, it had a tendency to develop its minor pieces (knights and bishops) very slowly, if at all. Even if they were moved, they would often end up on a less than optimum square.

The final problem was that the computer was apathetic toward occupying the center of the board. Anyone who has tried to study chess has heard, time and time again, that the center is the most important area of the chessboard. In most cases, the player who has control of the four center squares will ultimately come out of the contest as the victor.

With these faults in mind, I decided to try to find out why the computer played the way it did. To begin my search, I read the programming manual supplied with the program. To my surprise, this approach yielded results almost immediately.

Changing Attitude

In the program, one of the criteria used by the computer to compare moves is mobility. The greater the mobility resulting from a move, the greater the appeal of the move to the computer. In order to reflect the unparalleled strength and mobility of the queen, a one-point bonus would be awarded whenever the computer moved the queen. This bonus was what convinced the computer to disregard the other pieces and constantly move the queen.

I felt that this bonus would have to be deleted or modified. Looking through the listing (also supplied), I soon found this section of the program. At address 0112, the piece being considered for a move is compared to the queen. If the piece is not the queen, a conditional branch is taken. If it is the queen, the instruction at address 0116 (F6) increments the mobility counter.

Instead of NOPing the six bytes involved (which would have resulted in a total one point less than the original coding), I decided to replace the increment instruction at address 0116 with a decrement instruction (D6). This gives a two point difference from the original. By doing this, the computer will now favor the other pieces on the board instead of the queen.

Openings

With the queen taken care of, there were only two more problems to contend with. The manual also supplied a partial solution to one of these problems: the computer's clumsy development of its pieces.

In an effort to counter this problem, the program includes provisions that allow the computer to play a stored opening of up to nine moves. This provides sufficient time to properly post the minor pieces. Besides this obvious advantage, there is a bigger bonus available to the computer when it plays a stored opening. That bonus is castling.

In normal play, the computer is unable to castle because it is considered an illegal move (moving two pieces at once, moving the king two squares, having one piece jump another). However, during the opening, the computer doesn't generate any of its moves. It simply plays whatever is stored in memory.

Using the opening has its limitations. Because of the limited memory available, only one opening may be stored in the computer to be used during a game. This leaves the computer with no flexibility in playing the opening. Once a move that contradicts the prestored sequence of moves is made, the computer will abandon the opening and generate its own moves for the remainder of the game.

In order for the computer to reap the greatest benefit from the opening, several factors must be kept in mind: (1) castling in the shortest and most efficient manner, (2) quick development of the minor pieces, (3) obtaining a position from which the computer may assert or contest control of the center.

After a little research, I realized that an opening similar to the Ruy Lopez given in the programming manual (see Table 1) seemed to be what I was after. I particularly liked its allowing the computer to castle in only four moves.

But it also contained some moves I really didn't care for. By playing bishop to N5 (move 3), the whole opening is jeopardized because the common reply is 3:... P-QR3, forcing the bishop to retreat and the computer to abandon the opening. Instead I prefer 3:B-B4. Though it is conservative, it improves the chances of continuing the opening.

I also didn't care for the moves following 4:... N × P. Here the computer is looking for moves that I have yet to see anyone play. White's fifth move is particularly questionable because it allows black to make any move it wants, which is rarely the expected 5:... B-K2. I feel that 5:P-Q3 is a stronger move. Besides forcing the knight to move, it also opens up the queen bishop. Many people, being content with the free pawn, will simply retreat to B3.

This allows white to continue with 6:N-B3. Black will often respond with B-B4 in order to prepare for castling. Now white can play 7:B-K3, attacking black's bishop. Black will usually respond P-QN3. (If black plays B × B, the computer will respond with P × B, opening an important file for the king rook, as well as strengthening the center.)

The computer can now play 8:P-Q4, expecting P × P and finishing up with 9:N × P. From here the computer should be able to finish the exchange, which results in a strong position for white. The whole opening is shown in Table 2. (The memory dump is Program A.)

The advantage of this opening is that it leaves white with a strong position, even if the

1:	P-K4	P-K4
2:	N-KB3	N-QB3
3:	B-N5	N-B3
4:	O-O	N × P
5:	P-Q4	B-K2
6:	Q-K2	N-Q3
7:	B × N	NP × B
8:	P × P	N-N2
9:	N-B3	. . .

Table 1.

1:	P-K4	P-K4
2:	N-KB3	N-QB3
3:	B-B4	N-B3
4:	O-O	N × P
5:	P-Q3	N-B3
6:	N-B3	B-B4
7:	B-K3	P-QN3
8:	P-Q4	P × P
9:	N × P	. . .

Table 2.

opening is not played all the way through. I haven't included a similar opening for black because such an opening has no real chance of succeeding. The reason for this is that white has the initiative in the opening, forcing black to simply make replies.

Program Modifications

With the opening out of the way, there was little left to glean from the programming manual. My next target was the program listing.

Near the end of the program (17C4 to 17E2) I found a bonus section similar to the one that

```
00C0  99 34 06 34 34 0E 56 23 05 45 25 07 53 24 0E 33
00D0  01 00 52 35 04 55 22 06 43 33 0F CC
```

Program A.

```
00A0  10 09 06 06 04 04 03 03 01 01 01 01 01 01 02 02
```

Program B.

involved the queen. In this case, the originating square and the destination square of a move are compared to prestored locations. If a match is found, a two point bonus is awarded.

The first part of the bonus checks to see if the piece is being moved from the back rank (squares 00 to 07). The second part compares the pieces' destination to four squares on the board. These squares are originally K4 (33), Q4 (34), KB3 (22) and QB3 (25).

These bonuses were originally included to help the computer through the opening stages of the game if the prestored opening were not played. The K4 and Q4 squares were intended for the king and queen pawns. The squares KB3 and QB3 were meant for the knights. Finally, the bonus for moving a back rank piece makes developing moves more attractive to the computer.

As far as the opening is concerned, these bonuses supply the computer with the guidelines it can follow. The problems begin to arise after the game has progressed a little further. These squares are no longer as important as they were in the beginning of the game.

I decided that different squares should receive the bonus. By changing the squares, I hoped to put the computer in a position where it could try to control the center. The simplest way to do this was to have the computer occupy the center.

The benefits of this are twofold. First, the center squares offer greater mobility for pieces posted there. Second, if a piece is in the center, it will probably be attacked. If this happens, it will have to be defended, thereby bringing more pieces to bear on the center.

To do this, I changed the bonus squares to K4, Q4, K5 and Q5 (33 at address 17C7, 34 at address 17CB, 43 at 17CF and 44 at 17D3).

I also changed the second bonus. Instead of awarding a move in the first rank, I decided to award moves in the first three ranks. This helps the computer post its pieces more offensively. To do this, simply replace the 10 at address 17DD with 30.

I have made two other changes in the program to increase the computer's awareness of the center. The first is quite simple. By increasing the value of the bonus section (at address 17E2) from 02 to 03, the availability of a bonus will have a greater influence on the move finally chosen.

The second change, a bit more involved, is to one of the lookup tables used by the computer when comparing moves. This table (from address 00A0 to 00AF) contains the "value" of the different pieces on the board. The original values are (decimal): king = 11, queen = 10, rook = 6, bishop = 4, knight = 4 and pawn = 2.

Because of the overriding importance of the king, I increased his value to 16. With the queen, I went in the opposite direction. Because the computer had a tendency to depend too heavily on this piece, I decreased its value to 9. I left the rooks and bishops unchanged. I didn't feel the knights should be worth as much as a bishop; therefore, I decreased their value to 3. I also devalued the pawns to 1.

There were two exceptions to this markdown. I chose to leave the king pawn and the queen pawn equal to two. This is another effort to focus the computer's attention on the center of the board. By making these pawns more valuable than the others, the computer will defend them more tenaciously, while attacking those of its opponent more fiercely (Program B).

That covers all of the changes I have made to date. By making these changes, you will be setting your computer to play a stronger level of chess. To guarantee that the computer is allowed to play its optimum game, don't try to play a serious game with the computer set to play at a blitz or superblitz level. ■