0_3	1_3	2_3	6_4	7_4	0_4
7_3	4_3	3_3	5_4	4_4	1_4
6_3	5_3	2_5	4_5	3_4	2_4
2_2	3_2	0_5	6_5	5_1	6_1
1_2	4_2	5_2	3_1	4_1	7_1
0_2	7_2	6_2	2_1	1_1	0_1

Figure 3: Bit assignments of five bytes to a 6 x 6 game board to facilitate 90° rotations by whole byte manipulations, and reflections about a diagonal axis by consistent rules for each byte plus the swapping of two bytes.

0_3	1_3	2_3	5_4	0_4
5_3	4_3	3_3	4_4	1_4
2_2	3_2	6_3 6_4 6_2 6_1	3_4	2_4
1_2	4_2	3_1	4_1	5_1
0_2	5_2	2_1	1_1	0_1

Figure 4: Bit assignments of four bytes to a 5 x 5 game board. 90° rotations by whole byte manipulations are facilitated, but reflection rules are complex. A better 5 x 5 compromise is found in figure 5.

0_2	1_2	6_3	7_2	4_2
3_2	2_2	7_3	6_2	5_2
4_3	5_3	0_4	1_3	0_3
5_1	6_1	3_3	2_1	3_1
4_1	7_1	2_3	1_1	0_1

Figure 5: The bits of four bytes can be assigned to a 5 x 5 game board in this pattern. 90° rotations are accomplished as in the 4 x 4 matrix of figure 4 for bytes 1 and 2 which are in the corner positions; the 90° rotation component of byte 3 is a 2 position circular shift; byte 4 is unchanged under 90° rotation due to its central position. Reflections about a horizontal axis are accomplished without bit exchanges between bytes.

that by four successive rotations about the center of the board covers the whole board. Since bits 0 through 3 were chosen arbitrarily, many such patterns can be constructed.

I have not been able to find such a bit assignment scheme that facilitates both the reflecting and rotation. Accordingly, since there are many more rotations than reflections, I have optimized the arrangement for rotation. For simplifying the reflecting, I have selected the arrangement of figure 2 in which one byte represents each half of the board. (In figure 4 and subsequent figures, bits are denoted by numbers 0 to 7, and the byte is designated by a subscript.)

Thus, a single subroutine can be used twice to reflect each byte about a vertical axis. Any axis would serve for reflection, but in this case the vertical axis avoids the need for transferring bits from one byte to the other. Table 1 shows the bit correspondence under this reflection. This is a fairly complicated transformation compared to the simple shift operation needed for 90° rotation. Should the utmost in program speed be needed, a hardware approach along the lines suggested by James Luscher in "Taking Memory Address Space," page 60, January 1976 BYTE, could be used.

The general principles described above for TACTIX 4 x 4 can be applied to other board sizes as well, but as more bytes become necessary to represent the larger boards, rotation representation by whole byte shifting becomes attractive. Figure 3 shows an arrangement for the bits of the five bytes needed for a 6 x 6 board.

Four bytes represent all the squares except the central four, which are represented by four of the bits of byte 5. For a 90° rotation, the first four bytes are shifted, 1 replacing 2, 2 to 3, 3 to 4, and 4 to 1, while the fifth byte is shifted two places. For this case, reflecting about a diagonal, such as the one running from lower left to upper right, is

Table 2: Bit interchanges for the reflection of a 6 x 6 game board about the lower left to upper right diagonal, using bit assignments of figure 3. After swapping full bytes 1 and 3, the same internal reorganization is applied to all bytes.

Each bit in the old byte:	7	6	5	4	3	2	1	0
is moved to	↓	↓	↓	↓	↓	↓	↓	↓
A bit in the new byte:	1	2	3	4	5	6	7	0