segmented into three sections corresponding to the three sections of the display. When addressing these tables, the high order byte (D) of the two-byte START_INDEX value is a "segment specifier" (0 <= D <= 2), while the low order byte (E) specifies the index of the entry in that segment.

In the case of the sprite generator table, please note that COUNT refers to 8-byte shape for entries whether one is using size 0 or size 1 sprites.

DATA — Starting address of a CRAM data buffer to receive data from VRAM.

COUNT — Number of entries to be read from the VRAM table.

The restrictions on COUNT are again table dependent. In other words, it should always be the case that START_INDEX + COUNT <= Table Size.

Side Effects:

- Destroys AF, BC, DE, HL, IX and IY.
- This routine uses the local storage area SAVED_COUNT and is therefore not re-entrant.

Calls to other OS routines:

- READ_VRAM

3.2.1.3 PUT_VRAM

Calling Sequence:

```
LD    A, TABLE_CODE
LD    DE, START_INDEX
LD    HL, DATA
LD    IY, COUNT
CALL  PUT_VRAM
```

Description:

PUT_VRAM w ites from the buffer DATA, COUNT entries to the table specified by TABLE_CODE, which starts at the table entry number START_INDEX.

PUT_VRAM uses the VDP_MODE_WORD and VRAM_ADDR_TABLE to calculate VRAM address and byte counts. It is imperative that the graphics mode be set up using WRITE_ REGISTER and the table being accessed be initialized using INIT_TABLE before PUT_VRAM is called.

The table level of graphics software contains a sprite reordering feature where the major effect is in the operation of PUT_VRAM. When the MUX_SPRITES flag is set to TRUE (1), PUT_VRAM writes sprite entries to a CRAM copy of the sprite attribute table instead of writing them to VRAM. It locates this table through a pointer in low cartridge ROM called LOCAL_SPR_TBL. The sprite entries will then be re-ordered before being written to VRAM.

Parameters:

TABLE_CODE	VRAM table code (Refer to Table 3-1) to be written.
START_INDEX	START_INDEX is a two-byte number which indicates the starting entry number of the table. For other considerations, refer to the START_ INDEX parameter of GET_VRAM in Section 3.2.1.2.

DATA	Starting address of a data buffer where data to be written to VRAM resides.
COUNT	Number of entries to be put to the VRAM table. The restrictions on COUNT are again table dependent. In other words, it should always be the case that START_INDEX + COUNT <= Table Size.

Side Effects:

- Destroys AF, BC, DE, HL, IX and IY.
- Uses local storage locations, SAVE_TEMP and SAVED_COUNT.

Calls to other OS routines:

- WRITE_VRAM

3.2.2 Table-Oriented Graphics Routines

A number of routines are included in the table level graphics software that perform useful operations on generators. Each of these takes a table code, a source index from that table, a destination index in the same table, and the number of entries to be processed. The routines work in read-modify-write mode, that is, they pull the generators out of the table one at a time, process them and put them back. They use a CRAM buffer for their scratch area. This buffer is allocated by the applications programmer and accessable only through the pointer at WORK_BUFFER in cartridge ROM.

With one exception, the routines in this package always process generators one at a time, and write them to the destination block in the same order in which they are extracted from the source block. This has important implications for their use with size 1 sprites.

When the sprite size is 1, the hardware accesses four generators at the index found in a sprite's attribute

table entry and displays them so that they appear on the screen as shown in Figure 3-3.

Sprite Screen Location

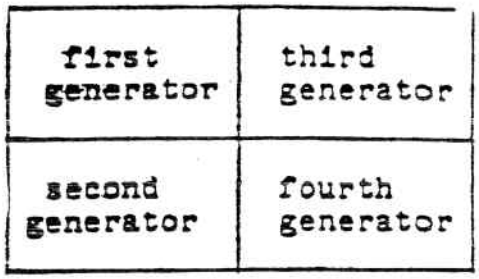

Figure 3-3
Sprite Size 1 Orientation

Thus, OS routines operating on the individual generators for a size 1 sprite will not be sufficient to orient the entire object. The four generators that make up the sprite will have to be permuted as well. The applications program will have to include a small routine that performs the required permutation in tandem with the OS call.

The following operations are available in the table-oriented graphics package:

1 - Reflection about the vertical axis

2 - Reflection about the horizontal axis

3 - 90-degree rotation

4 - Enlargement by a factor of two

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

3.2.2.1 REFLECT_VERTICAL

Calling Sequence:

```
LD    A, TABLE_CODE
LD    DE, SOURCE
LD    HL, DESTINATION
LD    BC, COUNT
CALL  REFLECT_VERTICAL
```

Description:

REFLECT_VERTICAL takes each generator in a block of COUNT generators following SOURCE in the table indicated by TABLE_CODE and modifies it in such a way that the new generator thus created will appear to be a reflection about the vertical screen axis of the old. The created generators are put back into a block of COUNT generators following DESTINATION in the same table.

The user must provide the permutation for size 1 sprite generators as diagrammed in Figure 3-4 below:

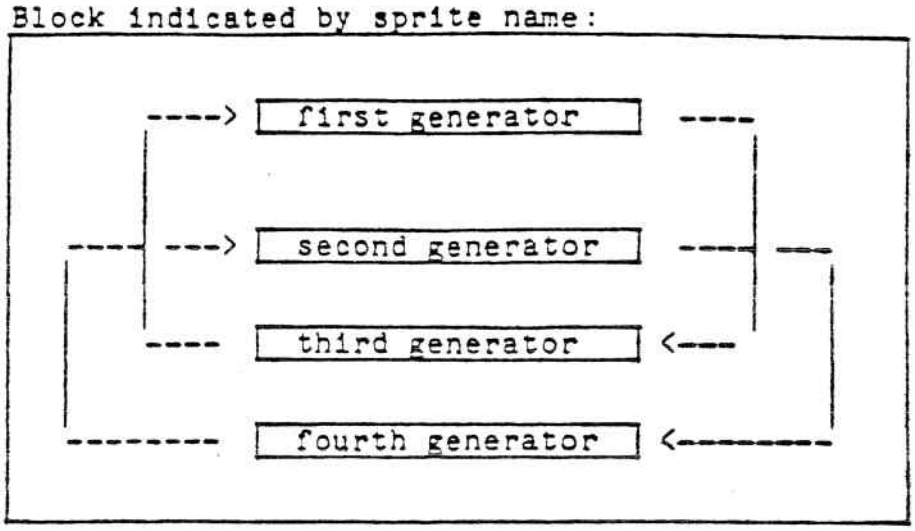

Figure 3-4
REFLECT_VERTICAL Size 1 Sprite Permutation

If TABLE_CODE is 3 (indicating the pattern generator table) and graphics mode 2 is used, REFLECT_VERTICAL also copies the color table entries for each generator it processes. Thus, when it is complete, the two-color table blocks indexed by SOURCE and DESTINATION will be identical. This means that the color scheme for the reflected generators will be the same as that for the originals.

Parameters:

TABLE_CODE	VRAM table code (Ref. Table 3-1) to be operated upon.
SOURCE	SOURCE is the two-byte index of the first entry in the specified table to be operated on.

For table operations of sprite generator or pattern generator in graphics mode 1, SOURCE should be in the range 0 <= SOURCE <= 255. For pattern generators in mode 2, it should be in the range 0 <= SOURCE <= 767. In either case, if a value of SOURCE supplied is outside the table's range but still is a legal VRAM address, the specified number of "entries" will be read and modified from the VRAM location (table location) + 8 *

SOURCE. For the proper table entries and table boundary, refer to Table 3-2.

Sprite size has no effect on the range of SOURCE.

DESTINATION (HL) — DESTINATION indexes the place where REFLECT_VERTICAL will start putting generators back into VRAM after modifying them.

The same restrictions apply to the value of DESTINATION as to the value of SOURCE. They are both intended to be indices into the same generator table.

COUNT (BC) — A two-bytes count of the number of entries to be processed sequentially after SOURCE.

The legal value for COUNT is dependent on the size of the table being operated on and the values of SOURCE and DESTINATION. In general, both of the following statements should be true:

COUNT + SOURCE <= (table size)
COUNT + DESTINATION <= (table size)

Side Effects:

- Destroys AF, AF', BC, DE, DE', HL, HL', IX and IY.
- Uses the first 16 bytes of the data area pointed to by WORK_BUFFER.

Calls to other OS routines:

- GET_VRAM
- PUT_VRAM

3.2.2.2 REFLECT_HORIZONTAL

Calling Sequence:

```
LD    A, TABLE_CODE
LD    DE, SOURCE
LD    HL, DESTINATION
LD    BC, COUNT
CALL  REFLECT_HORIZONTAL
```

Description:

REFLECT_HORIZONTAL takes each generator in a block of COUNT generators following SOURCE in the table indicated by TABLE_CODE and modifies it in such a way that the new generator created will appear to be a reflection about the horizontal screen axis of the old. The created generators are placed back into a block of COUNT generators following DESTINATION in the same table.

The user has to provide the permutation for size 1 sprite generators as diagrammed in Figure 3-5.

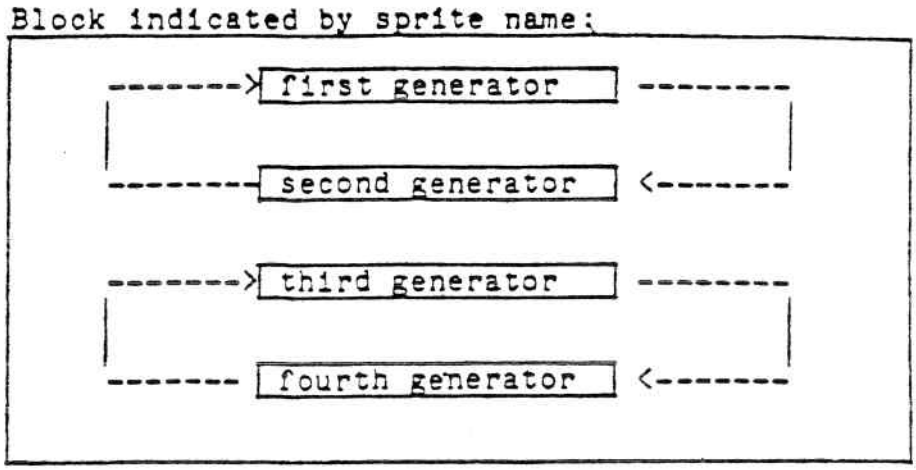

Figure 3-5
REFLECT_HORIZONTAL Size 1 Sprite Permutation

If TABLE_CODE is 3 (indicating the pattern generator table) and the graphics mode is 2, REFLECT_HORIZONTAL also performs the identical reflection on the corresponding color table entry for each generator it processes. This means that the reflected generators will be colored in a way that is consistent with their unreflected counterparts. When in mode 1, the color table is untouched.

Parameters:

TABLE_CODE	VRAM table code (Ref. Table 3-1) to be operated upon.
SOURCE	SOURCE is the two-byte index of the first entry in the specified table to be operated on.

For table operations on sprite generator or pattern generator in graphics mode 1, SOURCE should be in the range 0 <= SOURCE <= 255. For pattern generators in mode 2, it should be in the range 0 <= SOURCE <= 767. In either case, if a value of SOURCE is supplied and is outside the table's range but still a legal VRAM address, the specified number of "entries" will be read and modified from the VRAM location (table location) + 8 * SOURCE. For the proper table entries and table boundary, refer to Table 3-2.

Sprite size has no effect on the range of SOURCE.

DESTINATION

DESTINATION indexes the place where REFLECT_VERTICAL will start putting generators back into VRAM after modification.

The same restrictions apply to the value of DESTINATION as to the value of SOURCE. They are both intended to be indices into the same generator table.

COUNT

A two-byte count of the number of entries to be processed sequentially after SOURCE.

A legal value for count depends on the size of the table being operated on and the values of SOURCE and DESTINATION. In

general, both of the following statements should be true:

COUNT + SOURCE <= (table size)
COUNT + DESTINATION <= (table size)

Side Effects:

- Destroys AF, AF', BC, DE, DE', HL, HL', IX and IY.
- Uses the first 16 bytes of the data area pointed to by WORK_BUFFER.

Calls to other OS routines:

- GET_VRAM
- PUT_VRAM

3.2.2.3 ROTATE_90

Calling Sequence:

```
LD    A, TABLE_CODE
LD    DE, SOURCE
LD    HL, DESTINATION
LD    BC, COUNT
CALL  ROTATE_90
```

Description:

ROTATE_90 takes each generator in a block of COUNT generators following SOURCE in the table indicated by TABLE_CODE and modifies it in such a way that the new generator thus created will appear to be a 90-degree clockwise rotation of the old. The created generators are put back into a block of COUNT generators following DESTINATION in the same table.

The user must provide the permutation for size 1 sprite generators as diagrammed in Figure 3-6 below:

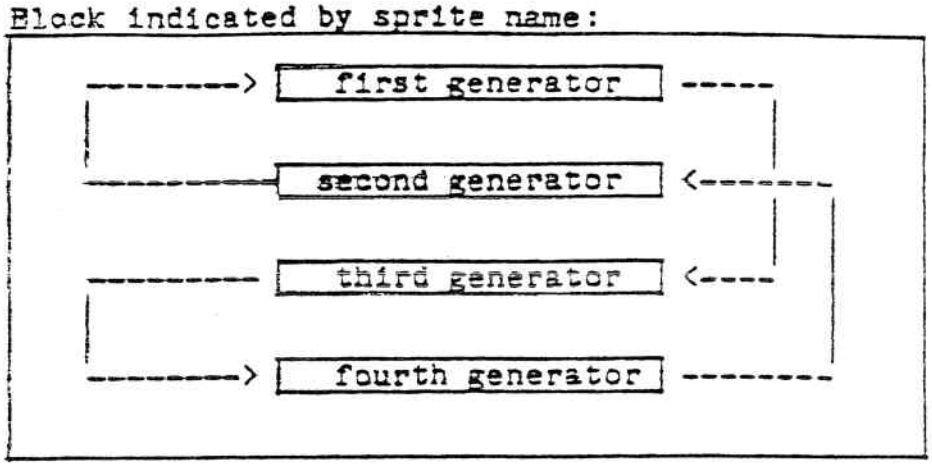

Figure 3-6
ROTATE_90 Size 1 Sprite Permutation

This routine should be used with great care when applied to pattern generators in mode 2. In this mode, the VDP allows arbitrary color combinations along vertical lines while it is still limited to two colors along a given 8-pixel horizontal line. The problem is that if the user attempts to rotate a figure that has more than two colors on a vertical line, ROTATE_90 will exhibit color problems after rotation. There is no way around this problem except to keep any generators that are intended for rotation simple. If the TABLE_CODE is 3 (pattern

generator table) and the mode is 2, ROTATE_90 will copy the corresponding color table entries indexed by SOURCE to the block indexed by DESTINATION.

Parameters:

TABLE_CODE	VRAM table code (Ref. Table 3-1) to be operated upon.
SOURCE	SOURCE is the two-byte index of the first entry in the specified table to be operated on. For table operations of sprite generator or pattern generator in graphics mode 1, SOURCE should be in the range 0 <= SOURCE <= 255. For pattern generators in mode 2, it should be in the range 0 <= SOURCE <= 767. In either case, if a value of SOURCE is supplied and is outside the table's range but

still is a legal VRAM address, the specified number of "entries" will be read and modified from the VRAM location (table location) + 8 * SOURCE. For the proper table entries and table boundary, refer to Table 3-2.

Sprite size has no effect on the range of SOURCE.

DESTINATION

DESTINATION indexes the place where REFLECT_VERTICAL will start putting generators back into VRAM after modifying them.

The same restrictions apply to the value of DESTINATION as to the value of SOURCE. They are both intended to be indices into the same generator table.

COUNT	A two-byte count of the number of entries to be processed sequentially after SOURCE.
	The legal value for count is dependent on the size of the table being operated on and the values of SOURCE and DESTINATION. In general, both of the following statements should be true:
	COUNT + SOURCE <= (table size)
	COUNT + DESTINATION <= (table size)

Side Effects:

- Destroys AF, AF', BC, DE, DE', HL, HL' IX and IY.
- Uses the first 16 bytes of the data area pointed to by WORK_BUFFER.

Calls to other OS routines:

- GET_VRAM
- PUT_VRAM