

1 segmented into three sections
2 corresponding to the three
3 sections of the display. When
4 addressing these tables, the high
5 order byte (D) of the two-byte
6 START_INDEX value is a "segment
7 specifier" ($0 \leq D \leq 2$), while
8 the low order byte (E) specifies
9 the index of the entry in that
10 segment.

11
12 In the case of the sprite
13 generator table, please note that
14 COUNT refers to 8-byte shape for
15 entries whether one is using size
16 0 or size 1 sprites.

17
18 DATA

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

19
20
21 COUNT

Number of entries to be read from
the VRAM table.

22
23
24
25
26

1
2
3 3.2.1.3 PUT_VRAM
4

5 Calling Sequence:

6
7 LD A, TABLE_CODE
8 LD DE, START_INDEX
9 LD HL, DATA
10 LD IY, COUNT
11 CALL PUT_VRAM
12

13 Description:

14
15 PUT_VRAM writes from the buffer DATA, COUNT entries to
16 the table specified by TABLE_CODE, which starts at the
17 table entry number START_INDEX.

18
19 PUT_VRAM uses the VDP_MODE_WORD and VRAM_ADDR_TABLE to
20 calculate VRAM address and byte counts. It is
21 imperative that the graphics mode be set up using
22 WRITE_REGISTER and the table being accessed be ini-
23 tialized using INIT_TABLE before PUT_VRAM is called.
24
25
26

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

10
11 Parameters:

12
13 TABLE_CODE VRAM table code (Refer to Table
14 3-1) to be written.

15
16 START_INDEX START_INDEX is a two-byte number
17 which indicates the starting
18 entry number of the table. For
19 other considerations, refer to
20 the START_INDEX parameter of
21 GET_VRAM in Section 3.2.1.2.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

1	DATA	Starting address of a data buffer
2		where data to be written to VRAM
3		resides.
4		
5	COUNT	Number of entries to be put to the
6		VRAM table.
7		
8		The restrictions on COUNT are
9		again table dependent. In other
10		words, it should always be the
11		case that $START_INDEX + COUNT \leq$
12		Table Size.
13		
14	Side Effects:	
15		
16		- Destroys AF, BC, DE, HL, IX and IY.
17		- Uses local storage locations, SAVE_TEMP and
18		SAVED_COUNT.
19		
20	Calls to other OS routines:	
21		
22		- WRITE_VRAM
23		
24		
25		
26		

1
2
3 3.2.2 Table-Oriented Graphics Routines
4

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

16
17 With one exception, the routines in this package always
18 process generators one at a time, and write them to the
19 destination block in the same order in which they are
20 extracted from the source block. This has important
21 implications for their use with size 1 sprites.

22
23 When the sprite size is 1, the hardware accesses four
24 generators at the index found in a sprite's attribute
25
26

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

3
4 Sprite Screen Location

5 first generator	6 third generator
7 second generator	8 fourth generator

9 Figure 3-3
10 Sprite Size 1 Orientation

11
12 Thus, OS routines operating on the individual generators
13 for a size 1 sprite will not be sufficient to orient the
14 entire object. The four generators that make up the
15 sprite will have to be permuted as well. The
16 applications program will have to include a small
17 routine that performs the required permutation in tandem
18 with the OS call.

19
20 The following operations are available in the table-
21 oriented graphics package:
22
23
24
25
26

-
- 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

1
2
3 3.2.2.1 REFLECT_VERTICAL
4

5 Calling Sequence:

6
7 LD A, TABLE_CODE
8 LD DE, SOURCE
9 LD HL, DESTINATION
10 LD BC, COUNT
11 CALL REFLECT_VERTICAL
12

13 Description:

14
15 REFLECT_VERTICAL takes each generator in a block of
16 COUNT generators following SOURCE in the table indicated
17 by TABLE_CODE and modifies it in such a way that the new
18 generator thus created will appear to be a reflection
19 about the vertical screen axis of the old. The created
20 generators are put back into a block of COUNT generators
21 following DESTINATION in the same table.
22

23 The user must provide the permutation for size 1 sprite
24 generators as diagrammed in Figure 3-4 below:
25
26

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

Block indicated by sprite name:

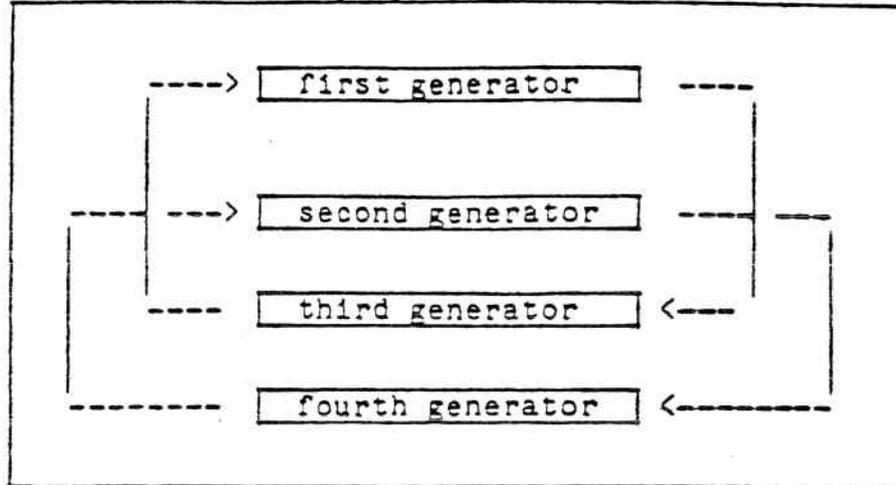


Figure 3-4
REFLECT_VERTICAL Size 1 Sprite Permutation

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.

1 Parameters:

2
3 TABLE_CODE VRAM table code (Ref. Table 3-1)
4 to be operated upon.

5
6 SOURCE SOURCE is the two-byte index of
7 the first entry in the specified
8 table to be operated on.

9
10 For table operations of sprite
11 generator or pattern generator in
12 graphics mode 1, SOURCE should be
13 in the range $0 \leq \text{SOURCE} \leq 255$.
14 For pattern generators in mode 2,
15 it should be in the range $0 \leq$
16 $\text{SOURCE} \leq 767$. In either case, if
17 a value of SOURCE supplied is
18 outside the table's range but
19 still is a legal VRAM address, the
20 specified number of "entries" will
21 be read and modified from the VRAM
22 location (table location) + 8 *

1 SOURCE. For the proper table
2 entries and table boundary, refer
3 to Table 3-2.
4

5 Sprite size has no effect on the
6 range of SOURCE.
7

8
9
10 DESTINATION (HL) DESTINATION indexes the place where
11 REFLECT_VERTICAL will start putting
12 generators back into VRAM after
13 modifying them.
14

15 The same restrictions apply to the
16 value of DESTINATION as to the value of
17 SOURCE. They are both intended to be
18 indices into the same generator table.
19

20 COUNT (BC) A two-bytes count of the number of
21 entries to be processed sequentially
22 after SOURCE.
23
24
25
26

1 The legal value for COUNT is dependent
2 on the size of the table being operated
3 on and the values of SOURCE and
4 DESTINATION. In general, both of the
5 following statements should be true:
6

7 $COUNT + SOURCE \leq (\text{table size})$

8 $COUNT + DESTINATION \leq (\text{table size})$

9
10 Side Effects:

- 11
12 - Destroys AF, AF', BC, DE, DE', HL, HL', IX and IY.
13 - Uses the first 16 bytes of the data area pointed to by
14 WORK_BUFFER.

15
16 Calls to other OS routines:

- 17
18 - GET_VRAM
19 - PUT_VRAM
20
21
22
23
24
25
26

1
2
3 3.2.2.2 REFLECT_HORIZONTAL
4

5 Calling Sequence:

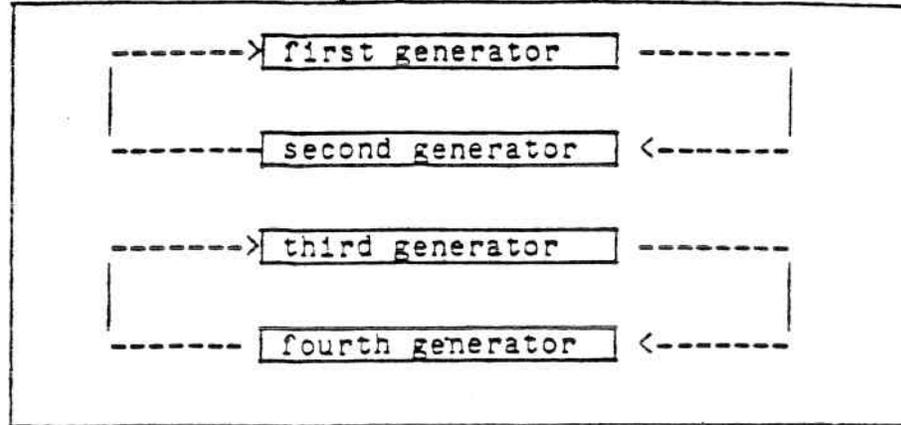
6 LD A, TABLE_CODE
7 LD DE, SOURCE
8 LD HL, DESTINATION
9 LD BC, COUNT
10 CALL REFLECT_HORIZONTAL
11

12 Description:

13
14 REFLECT_HORIZONTAL takes each generator in a block of
15 COUNT generators following SOURCE in the table indicated
16 by TABLE_CODE and modifies it in such a way that the new
17 generator created will appear to be a reflection about
18 the horizontal screen axis of the old. The created
19 generators are placed back into a block of COUNT
20 generators following DESTINATION in the same table.
21

22 The user has to provide the permutation for size 1
23 sprite generators as diagrammed in Figure 3-5.
24
25
26

1
2 Block indicated by sprite name:



9
10 Figure 3-5
REFLECT_HORIZONTAL Size 1 Sprite Permutation

11
12
13 If TABLE_CODE is 3 (indicating the pattern generator
14 table) and the graphics mode is 2, REFLECT_HORIZONTAL
15 also performs the identical reflection on the
16 corresponding color table entry for each generator it
17 processes. This means that the reflected generators
18 will be colored in a way that is consistent with their
19 unreflected counterparts. When in mode 1, the color
20 table is untouched.

21
22
23
24
25
26

1
2 Parameters:

3
4 TABLE_CODE VRAM table code (Ref. Table 3-1)
5 to be operated upon.

6
7 SOURCE SOURCE is the two-byte index of
8 the first entry in the specified
9 table to be operated on.

10
11 For table operations on sprite
12 generator or pattern generator in
13 graphics mode 1, SOURCE should be
14 in the range $0 \leq \text{SOURCE} \leq 255$.
15 For pattern generators in mode 2,
16 it should be in the range $0 \leq$
17 $\text{SOURCE} \leq 767$. In either case, if
18 a value of SOURCE is supplied and
19 is outside the table's range but
20 still a legal VRAM address, the
21 specified number of "entries" will
22 be read and modified from the VRAM
23 location $(\text{table location}) + 8 *$
24 SOURCE. For the proper table
25 entries and table boundary, refer
26 to Table 3-2.

1		Sprite size has no effect on the
2		range of SOURCE.
3		
4	DESTINATION	DESTINATION indexes the place
5		where REFLECT_VERTICAL will start
6		putting generators back into VRAM
7		after modification.
8		
9		The same restrictions apply to the
10		value of DESTINATION as to the
11		value of SOURCE. They are both
12		intended to be indices into the
13		same generator table.
14		
15	COUNT	A two-byte count of the number of
16		entries to be processed
17		sequentially after SOURCE.
18		
19		A legal value for count depends on
20		the size of the table being
21		operated on and the values of
22		SOURCE and DESTINATION. In
23		
24		
25		
26		

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

1
2
3 3.2.2.3 ROTATE_90
4

5 Calling Sequence:

6 LD A, TABLE_CODE
7 LD DE, SOURCE
8 LD HL, DESTINATION
9 LD BC, COUNT
10 CALL ROTATE_90
11

12 Description:

13
14 ROTATE_90 takes each generator in a block of COUNT
15 generators following SOURCE in the table indicated by
16 TABLE_CODE and modifies it in such a way that the new
17 generator thus created will appear to be a 90-degree
18 clockwise rotation of the old. The created generators
19 are put back into a block of COUNT generators following
20 DESTINATION in the same table.
21

22
23 The user must provide the permutation for size 1 sprite
24 generators as diagrammed in Figure 3-6 below:
25
26

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

Block indicated by sprite name:

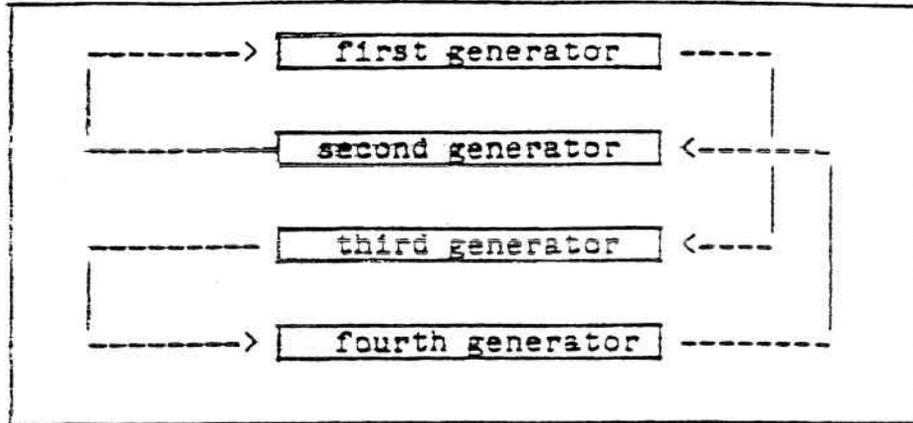


Figure 3-6
ROTATE_90 Size 1 Sprite Permutation

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

1 generator table) and the mode is 2, ROTATE_90 will copy
2 the corresponding color table entries indexed by SOURCE
3 to the block indexed by DESTINATION.
4

5
6 Parameters:

7
8 TABLE_CODE VRAM table code (Ref. Table 3-1)
9 to be operated upon.

10
11 SOURCE SOURCE is the two-byte index of
12 the first entry in the specified
13 table to be operated on.

14
15 For table operations of sprite
16 generator or pattern generator in
17 graphics mode 1, SOURCE should be
18 in the range $0 \leq \text{SOURCE} \leq 255$.
19 For pattern generators in mode 2,
20 it should be in the range $0 \leq$
21 SOURCE ≤ 767 . In either case, if
22 a value of SOURCE is supplied and
23 is outside the table's range but
24
25
26

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

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.

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

4
5 The legal value for count is
6 dependent on the size of the table
7 being operated on and the values
8 of SOURCE and DESTINATION. In
9 general, both of the following
10 statements should be true:

11
12 COUNT + SOURCE <= (table size)
13 COUNT + DESTINATION <= (table
14 size)

15
16 Side Effects:

- 17
18 - Destroys AF, AF', BC, DE, DE', HL, HL' IX and IY.
19 - Uses the first 16 bytes of the data area pointed to by
20 WORK_BUFFER.

21
22
23
24
25
26

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

Calls to other OS routines:

- GET_VRAM

- PUT_VRAM