Contents

1 DEB Capabilities

- Introduction
- 16-Color Graphics
- Look-Up Table (LUT)
- Overlay Modes

INTRODUCTION

The Display Enhancement Board option (DEB) adds improved color and graphics functionality to your AT&T PC 6300. When you use the DEB with the PC 6300 color monitor, you can display graphics in up to 16 colors simultaneously or display text-on-graphics or graphics-on-graphics overlays. When you use the DEB with the PC 6300 monochrome monitor, you have the same capabilities as you do with the color monitor, except that colors are displayed as "shades of green."

The DEB is compatible with existing software, so that all the programs you have already can be used now as if the DEB were not installed. Of course, these programs do not have access to any of the new capabilities.

The purpose of this supplement to the GWBASIC Programmer's Guide is to give you the information you need to take complete advantage of the DEB's capabilities. It assumes that you are familiar with video programming in GWBASIC. If you are not, read the chapter on Graphics, and the portions of the Command Reference that discuss graphics statements, in the GWBASIC Programmer's Guide.

Before you begin writing programs for the DEB, follow the procedures in the DEB Installation Manual for installing the DEB hardware and device driver software.

The DEB is an optional hardware component for the AT&T PC 6300 that works in conjunction with the PC 6300's built-in Video Display Controller (VDC) to provide improved color and graphics functionality.

The built-in VDC contains circuitry and memory that supports either 4 color medium resolution (320 × 200 pixels) graphics, 1 color high resolution (640 × 200 pixels) graphics, or 1 color super resolution (640 × 400 pixels) graphics.

The DEB contains additional circuitry and memory that can be combined with the capabilities of the built-in VDC to produce up to 16 colors in either high or super resolution. You can also program the VDC and DEB separately, treating them as two separate images which are combined on one screen to produce text-on-graphics or graphics-on-graphics overlays. These overlay modes let you use up to 8 colors.

16-COLOR GRAPHICS

This feature lets you display 16 colors in either high resolution (640 × 200) or super resolution (640 × 400). Not only can you use the standard 16 colors, you can also combine colors to form new colors and cause pixels to blink from one color to another.

The DEB provides 5 palettes for you to use when programming in color. At any point in your program, you select one of the palettes as the "active" palette. The color combinations contained in that palette determine what colors and effects show on the screen.

Each of the first 4 palettes contains a default set of 16 color combinations, but to suit the needs of your program you can change the contents of the palette to any one of the following:

- any of the 16 standard colors with which you are already familiar from the standard applications. The standard colors are:

0 = black	8 = gray
1 = blue	9 = light blue
2 = green	10 = light green
3 = cyan	11 = light cyan
4 = red	12 = light red
5 = magenta	13 = light magenta
6 = brown	14 = yellow
7 = white	15 = high-intensity white

- a mixture, or "dithering," of any 2 of the 16 standard colors
- an alternation, or blinking, between any 2 of the standard 16 colors

The fifth palette contains no default combinations. You program the fifth palette by loading color values into a 256-element array of integers. GWBASIC uses this special palette to program the DEB's color look-up table (LUT).

LOOK-UP TABLE (LUT)

The LUT resides in RAM on the DEB board. The LUT contains 256 values that determine the colors, blinking, and dithering that appear on the screen. Whether you need to learn about the use and layout of the LUT depends on the application you are writing.

If you use the standard palettes, you need not be concerned with the LUT. GWBASIC automatically programs the LUT to correspond to the way you set up the palettes.

If you program a custom LUT, you greatly increase the color combinations and blinking effects available to you.

OVERLAY MODES

These modes let you display text-on-graphics or graphics-on-graphics images by treating the VDC and DEB as separate entities that write to the same screen. In the overlay modes, the output of the VDC takes precedence over the output of the DEB. If you program the VDC and DEB to display different attributes at the same pixel, the attributes selected by the VDC are displayed.

You can use either of two text-on-graphics modes. In one, you can program the DEB to display high resolution graphics in up to 8 colors; in the other, you can program the DEB to display super resolution graphics in up to 8 colors. In both, the VDC displays 25 lines of 80 characters each.

You can select either of two graphics-on-graphics modes. One mode uses the VDC to display high resolution graphics in one color while the DEB displays high resolution graphics in up to 8 colors. The other mode uses the VDC for super high resolution graphics in one color and the DEB for super high resolution graphics in 8 colors.

The overlay modes offer 5 palettes. Each of the first 4 palettes has 8 positions. These four palettes have default colors that you can change to suit your needs. You can choose 8 color combinations from any of the 16 standard colors, or blink between 2 of the standard colors. The dithering combinations of the 16-color graphics modes are not available. You can also use the fifth palette to custom program the LUT.

2 How to Program the DEB

- Programming Steps

PROGRAMMING STEPS

There are three steps for video programming in GWBASIC, which apply whether or not you are using the DEB capability:

1 Set the video mode by using the SCREEN statement.

2 Select the color combinations and effects you want to use.

3 Construct the graphics images you want to display.

This chapter describes each of these steps in detail. This chapter does **not** describe how to use the fifth palette to program the LUT directly. (See **Chapter 4, Programming the LUT.**)

Setting Mode and Page

As in standard GWBASIC, you use the SCREEN statement to select an operating mode. If you are using one of the overlay modes, the SCREEN statement also selects the active page, which determines whether the VDC or the DEB receives the output of PRINT or graphics display statements. The VDC is page 0 and the DEB is page 128. In the text-on-graphics modes, all text output statements default to page 0 and all graphics display statements default to page 128. If you want text to appear on the DEB graphics screen, you must issue a SCREEN statement that sets the active page to 128 before you display the text.

Setting Colors and Effects

Colors and effects are controlled by two statements: COLOR and PALETTE. The COLOR statement syntax extends the standard GWBASIC COLOR statement, allowing you to select background and foreground default colors and to select the active palette. The PALETTE statement is new. You use PALETTE to program color combinations into the active palette or to reset the active palette to its default assignments. A form of the statement, PALETTE USING, allows you to reprogram the entire active palette at once by specifying an integer array that contains the new values. Tables of the available color combinations and the default values for each palette are in the next chapter on DEB Statements.

Displaying Graphics Images

You use the same statements for DEB graphics as you do for normal GWBASIC graphics. However, in normal GWBASIC statements, you specify the color number to be used in drawing a line or circle. For DEB graphics, you specify the *palette position* in the active palette that contains the color combination or effect you want to use. For example, you could select 16-color super resolution mode, select palette 1 as the active palette, and draw a red circle, with the following code fragment:

```
10 SCREEN 102                    'select 640 × 400
20 REM                           '16-color mode
30 COLOR ,,1                     'set active
40 REM                           'palette to 1
50 CIRCLE (320,200),100,2        'the default color in
60 REM                           'position 2 is red
```

3 DEB Statements

- Overview
- SCREEN Statement
- COLOR Statement
- PALETTE and PALETTE USING Statements
- Default Palettes
- Blinking Color Effects for DEB Palettes 0-3
- Dither Combinations for DEB Palettes 0-3
- Remarks
- Examples

OVERVIEW

This chapter gives detailed descriptions of the GWBASIC statements that you can use for DEB graphics programming.

If you plan to use Palette 4, the LUT palette, carefully read **Chapter 4** before you begin using the statements in this chapter to program the LUT.

SCREEN STATEMENT

SCREEN The SCREEN statement establishes the mode for the display and lets you select the active display page. SCREEN also selects and initializes Palette 0 as the active palette when you enter a new mode.

Syntax **SCREEN [mode][,dummy1][,apage][,dummy2]**

mode is an integer expression which evaluates to one of the following:

101	16-color graphics with a resolution of 640 × 200.
102	16-color graphics with a resolution of 640 × 400.
103	an overlay mode. The DEB image is 8-color graphics with 640 × 200 resolution. The VDC image is 80 character by 25 line text.
104	an overlay mode. The DEB image is 8-color graphics with 640 × 400 resolution. The VDC image is 80 character by 25 line text.
105	an overlay mode. The DEB image is 8- color graphics with 640 × 200 resolution. The VDC image is 1-color graphics with 640 × 200 resolution.
106	an overlay mode. The DEB image is 8-color graphics with 640 × 400 resolution. The VDC image is 1-color graphics with 640 × 400 resolution.

dummy1 is ignored, but is allowed for compatibility with non-DEB syntax.

apage selects the active page, i.e., the page to be written to by output statements to the screen. Apage is an integer expression that results in a value of 0 or 128. Page 0 is the VDC page and page 128 is the DEB page.

In the two 16-color graphics modes (101 and 102), the active page is always zero.

dummy2 is ignored, but is allowed for compatibility.

Examples

```
SCREEN 105,,128     'Selects a graphics-on-graph-
                    ics overlay mode, with all sub-
                    sequent output sent to the
                    DEB page.

SCREEN   ,,0        'Do not change modes,
                    but send subsequent
                    output to the VDC page.
```

COLOR STATEMENT

COLOR

The COLOR statement sets the background and foreground colors and selects the active palette. The syntax for the COLOR statement varies according to the mode you select with the SCREEN statement.

Syntax 1
(Modes 101,102)

COLOR [DEBfg][,DEBbg][,palette]

Syntax 2
(Modes 103,104)

COLOR [DEBfg][,DEBbg][,VDCfg]
[,VDC bg][,palette]

Syntax 3
(Modes 105,106)

COLOR [DEBfg][,DEBbg][,VDCfg][,palette]

DEBfg
(foreground)

is an integer expression in the range 1-7 for overlay modes and 1-15 for 16-color graphics modes. DEBfg identifies the position in the active palette which controls the color combination or effect of subsequent output to the screen. The color combination or effect in the DEBfg position will be used for writing text to the screen, and also for the output of graphics statements unless some other position is specified in the graphics statement itself.

When you enter a DEB mode, DEBfg is set to a default of 7. If you do not enter a value for DEBfg, it does not change from the value set by the last COLOR statement.

DEBbg (background) is an integer expression in the range 0-255 which defines the color combination or effect to be used for palette position 0. This is the background, or color displayed when the value of the DEB image for a particular pixel is 0. (See tables of combinations in next section on PALETTE statement.) When you enter a DEB mode, DEBbg defaults to 0 (black).

VDCfg is an integer expression in the range of 0-15 for graphics and 0-31 for text that specifies the color for the VDC foreground. When you enter an overlay mode, VDCfg defaults to 7 (white).

VDCbg is an integer expression in the range 0-15 that specifies the VDC background when displaying characters in text mode. VDCbg defaults to 0 (black) when you enter an overlay mode.

palette is an integer expression that sets the active palette. Valid ranges are 0-3 for the standard palettes and 4 for the LUT palette. If you omit palette from the COLOR statement, the active palette does not change.

Remarks

The values you specify in DEB COLOR statements fall into three categories:

- a color selection for the VDC from the same ranges as you use in the standard text mode. These selections produce the same effect on the screen as they do in the standard (non-DEB) text mode.

- a color selection for the DEB foreground. Here you specify a palette position instead of a color number. GWBASIC then looks up the color combination or effect in the palette position you've specified, and uses it in the PRINT statements and some of the graphics statements that follow the COLOR statement. If the syntax of a particular graphics statement includes a parameter for specifying a palette position, that value overrides the position specified in the COLOR statement.

- specification of the DEB background based on a color combination from the tables following the PALETTE statement in this chapter. You can also set the DEB background by using the PALETTE statement to change Palette position 0.

PALETTE AND PALETTE USING STATEMENTS

PALETTE Use this statement to set values in palettes and reset palettes to their default values.

Syntax 1 **PALETTE**

Syntax 2 **PALETTE [position][,value]**

Syntax 3 **PALETTE USING array (array index)**

Remarks The PALETTE and PALETTE USING statements work on the active graphics page and on the active palette.

Syntax 1 sets the active palette to its default values. (See the following tables.)

Syntax 2 lets you change the values in the active palette, one palette position at a time.

position is an integer expression which identifies the position to be changed. If the active palette is 0-3, then the valid range for position is 0-15 for 16-color graphics modes and 0-7 for overlay modes. For Palette 4, the valid range for position is 0-255.

value is an integer expression which identifies the color combination or effect to be programmed into the selected position in the active palette. For Palettes 0-3, valid values range from 0-255. For Palette 4, valid values range from 0-15 and values greater than 15 are treated modulo 16.

Syntax 3 lets you set all the values in the active palette with one statement.

array is an integer array of at least 256 elements.

array index is an integer expression which defines the element within the specified array at which palette programming begins. At least 256 elements must follow this element.

Standard Palettes (0-3)

The first 8 or 16 elements of the array are loaded into the active palette. The entire active palette is reprogrammed based on the values in the array. The array values range from −1 to 255. Values greater than 255 are treated modulo 256. A value of −1 specifies that the value in the corresponding palette position not be changed. The values from 0 to 255 come from the tables at the end of the chapter.

NOTE: Dimension the array to have 256 elements even though only 8 or 16 are used for the standard palettes.

The LUT Palette (Palette 4)

All 256 elements are used to program the LUT directly. Valid values are in the range −1 to 15. Values greater than 15 are treated modulo 16. A value of −1 specifies that the value in the corresponding position in the LUT not be changed, and values 0-15 represent the standard 16 colors.

In Syntax 2 and Syntax 3, if you specify a palette position greater than the value allowed for the mode in which you are working, the value you specify will be put in that palette's highest position. For example, if you attempted to set palette position 13 to red when working in overlay mode, which has 8-position palettes, the **8th** palette position would be set to red.

DEFAULT PALETTES

The defaults for each of the four palettes are:

Palette Number 0

Position	Color
0	0 = black
1	2 = green
2	4 = red
3	6 = brown
4	1 = blue
5	3 = cyan
6	5 = magenta
7	7 = white
8	8 = gray
9	9 = light blue
10	10 = light green
11	11 = light cyan
12	12 = light red
13	13 = light magenta
14	14 = yellow
15	15 = high-intensity white

Palette Number 1

Position	Color
0	0 = black
1	3 = cyan
2	5 = magenta
3	7 = white
4	1 = blue
5	2 = green
6	4 = red
7	6 = brown
8	8 = gray
9	9 = light blue
10	10 = light green
11	11 = light cyan
12	12 = light red
13	13 = light magenta
14	14 = yellow
15	15 = high-intensity white

Palettes 2 and 3 are the same, and they contain the standard colors in numerical order.

Palette Number 2 and Palette Number 3

Position	Color
0	0 = black
1	1 = blue
2	2 = green
3	3 = cyan
4	4 = red
5	5 = magenta
6	6 = brown
7	7 = white
8	8 = gray
9	9 = light blue
10	10 = light green
11	11 = light cyan
12	12 = light red
13	13 = light magenta
14	14 = yellow
15	15 = high-intensity white

BLINKING COLOR EFFECTS FOR DEB PALETTES 0-3

Color combinations 16-135 have been pre-assigned to allow you easy access to blinking effects while using the standard palettes. The following table describes the available combinations.

A ↓ / B →	blue	green	cyan	red	magenta	brown	white	gray	light blue	light green	light cyan	light red	light magenta	yellow	high-intensity white
black	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
blue		31	32	33	34	35	36	37	38	39	40	41	42	43	44
green			45	46	47	48	49	50	51	52	53	54	55	56	57
cyan				58	59	60	61	62	63	64	65	66	67	68	69
red					70	71	72	73	74	75	76	77	78	79	80
magenta						81	82	83	84	85	86	87	88	89	90
brown							91	92	93	94	95	96	97	98	99
white								100	101	102	103	104	105	106	107
gray									108	109	110	111	112	113	114
light blue										115	116	117	118	119	120
light green											121	122	123	124	125
light cyan												126	127	128	129
light red													130	131	132
light magenta														133	134
yellow															135

NOTE: To select a value that will cause blinking between colors A and B, find the number at the intersection of row A and column B.

DITHER COMBINATIONS FOR DEB PALETTES 0-3

Color combinations 136-255 have been pre-assigned to allow you easy access to dithering effects while using the standard palettes. The following table describes the available combinations.

A ↓ / B →	black	blue	green	cyan	red	magenta	brown	white	gray	light blue	light green	light cyan	light red	light magenta	yellow
black															
blue	136														
green	137	138													
cyan	139	140	141												
red	142	143	144	145											
magenta	146	147	148	149	150										
brown	151	152	153	154	155	156									
white	157	158	159	160	161	162	163								
gray	164	165	166	167	168	169	170	171							
light blue	172	173	174	175	176	177	178	179	180						
light green	181	182	183	184	185	186	187	188	189	190					
light cyan	191	192	193	194	195	196	197	198	199	200	201				
light red	202	203	204	205	206	207	208	209	210	211	212	213			
light magenta	214	215	216	217	218	219	220	221	222	223	224	225	226		
yellow	227	228	229	230	231	232	233	234	235	236	237	238	239	240	
high-intensity white	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

NOTE: To select a value that combines colors A and B to create a new color, find the number at the intersection of row A and column B.

REMARKS

In the text-on-graphics overlay modes, all graphics statements except GET and PUT use page 128 (the DEB page). GET and PUT use the active page only. There is no way to GET or PUT an entire overlayed screen; you can only work with the active page.

In the graphics-on-graphics overlay modes, all graphics statements including GET and PUT use the active page only.

In all DEB modes, tiling with the PAINT command requires a 4-byte string rather than the 1-byte used in standard modes.

EXAMPLES

The following program demonstrates the PALETTE USING statement to change the color combinations so that each color and its high intensity version are in consecutive positions in the palette.

```
 50 SCREEN 102                    '16 color graphics
 60 CLS:KEY OFF                   'clear screen
 70 PALETTE                       'use default palette
 80 DIM A%(256)                   'array for PALETTE
 85 REM                             USING statement
 90 J=0
100 FOR I=0 TO 7                  'load up the array
110 A%(J)=I:A%(J+1)=I+8
120 J=J+2
130 NEXT I
140 LOCATE 2,2
150 FOR I = 97 TO 112             'print 15 characters in
155 REM                             15 colors
160 COLOR I-96,0:PRINT CHR$(I);
170 NEXT I
180 LOCATE 22,2
190 INPUT "Hit <CR> to change the colors",A$
195 REM          Reprogram the entire palette
200 PALETTE USING A%(0)
210 LOCATE 22,2
220 INPUT "Hit <CR> to change the colors" ,A$
230 PALETTE                       'use default palette
240 GOTO 180
```

The following example draws 3 interlocking circles in 16-color graphics mode and fills each separate section with various colors.

```
 10  SCREEN 102                 'set 16 color graphics
 15  REM                          mode
 20  CLS:KEY OFF                'clear screen and turn
 25  REM                          functions keys off
 30  COLOR ,,1                  'use palette 1
 35  REM
 40  CIRCLE(320,200),100,15      'draw circle 1
 50  CIRCLE(270,150),100,15      'draw circle 2
 60  CIRCLE (370,150),100,15     'draw circle 3
 70  PAINT (320,200),13,15       'fill with palette
 75  REM                           position 13
 80  PAINT (269,150),12,15       'fill with palette
 85  REM                           position 12
 90  PAINT (371,150),11,15       'fill with palette
 95  REM                           position 11
100  PAINT (320,250),10,15       'fill with palette
105  REM                           position 10
110  PAINT (320,100),9,15        'fill with palette
115  REM                           position 9
120  PAINT (220,150),8,15        'fill with palette
125  REM                           position 8
130  PAINT (420,150),7,15        'fill with palette
135  REM                           position 7
140  FOR I = 7 TO 13             'loop thru the used
145  REM                           palette positions
150  PALETTE I,135+RND*120       'use a random
155  REM                           dithered color for
157  REM                           palette position
160  FOR A=1 TO 100:NEXT A       'wait awhile
170  NEXT I
180  IF LEN(INKEY$) = 0 THEN GOTO 140
185  REM                         'check for keypress
190  SCREEN 0,0,0                'return to normal
200  END
```

The following program uses a tiling pattern to fill in a circle.

```
 30  SCREEN 102               'set 16 color graphics
 40  CLS                      'clear screen
 50  KEY OFF                  'turn function keys off
 60  CIRCLE(320,200), 100,1   'draw a circle
 70  REM      do the tiling to fill the circle
 80  PAINT(320,200), CHR$(&HCC) + CHR$
     (&H3C) + CHR$(&HC) + CHR$(&H3),1
 90  IF(LEN (INKEY$)) = 0
     THEN 90                  'check for keypress
100  SCREEN 0,0,0             'return to normal
110  END
```

This program draws a small circle and cycles through all the available color combinations for the standard palette.

```
 30  SCREEN 101               '16 color 640 x 200
 35  REM                        graphics
 40  CLS                      'clear screen
 50  CIRCLE (320,100),100,1   'draw a circle
 60  PAINT (320,100),1,1      'fill the circle with
 65  REM                        palette position 1
 70  FOR J = 0 TO 255         'use all color
 75  REM                        combinations
 80  PALETTE 1,J              'change the palette
 85  REM                        position color
 90  FOR A = 1 TO 500:NEXT A 'wait a bit
100  IF(LEN (INKEY$))<>0
     THEN 120                 'check for
105  REM                        keypress
110  NEXT J
120  SCREEN 0,0,0             'return to normal
130  END
```

This program shows 3 ways in which a box can be drawn with palette position 2 and filled with palette position 14.

```
 40 SCREEN 102              '16 color graphics
 50 CLS:KEY OFF             'clear screen
 60 DRAW "c2r50u50l50
    d50br2bu2p14,2"         'draw a box
 70 REM                       and fill it in
 75 REM
 80 LINE (270,100)-
    (320,150),2,B           'draw a box
 90 LINE (271,101)-
    (321,151),14,BF         'fill it in
100 REM
110 LINE (220,150)-
    (270,200),2,B           'draw a box
120 PAINT (221,151),14,2    'fill it in
130 IF LEN(INKEY$) = 0 THEN 130
140 SCREEN 0,0,0
150 END
```

The following example draws a wheel with the number of spokes you specify, using random colors. Then it uses the PALETTE statement to cycle through the standard colors.

```
 10 SCREEN 102 : CLS :
    KEY OFF                   'set 16 color
 15 REM                         graphics
 20 INPUT "Number of spokes on wheel - ";N
 30 ANGLE = 360 / N           'calculate # of angles
 40 RADIANS = ANGLE /57.29578
 50 CLS                       'clear screen
 60 FOR X = 1 TO N            'do the real work
 70 FOR Y = X TO N
 80 SX = SIN(X * RADIANS) * 195 + 320
 90 SY = SIN(Y * RADIANS) * 195 + 320
100 CX = COS(X * RADIANS) * 150 + 200
110 CY = COS(Y * RADIANS) * 150 + 200
120 LINE (SY,CY)- (SX,CX),
    INT(RND*(15) + 1)         'draw line with
125 REM                         random color
130 NEXT Y,X
140 FOR I = 1 TO 1000
150 FOR J = 1 TO 15
160 FOR K = 1 TO 15
170 PALETTE K,J               'change palette
180 IF (LEN(INKEY$)) <> 0
    THEN 220                  'check for
185 REM                         keypress
190 NEXT K
200 NEXT J
210 NEXT I
220 SCREEN 0,0,0              'return to normal
230 END
```

This program demonstrates overlay mode by drawing a box on the DEB screen and a circle on the VDC screen. It then cycles through the blinking color combinations on the DEB and the standard colors on the VDC.

```
 30  SCREEN 106                '8 color graphics on
 35  REM                         graphics overlay
 40  CLS:KEY OFF               'clear screen
 50  CIRCLE (320,200),100,1    'draw a circle
 55  REM                         on the VDC screen
 60  PAINT (320,200),
     CHR$(1)+CHR$(1),1         'fill the circle with
 65  REM                         palette position 1
 70  LOCATE 23,2;
 75  PRINT "The circle is on the VDC screen";
 80  SCREEN ,,128              'set the active page
 85  REM                         to the DEB screen
 90  LOCATE 24,2;
 95  PRINT "The box is on the DEB screen";
100  LINE (250,50)-
     (390,350), 5, BF          'draws a box on
105  REM                         the DEB screen
110  FOR J=0 TO 135            'use all color
115  REM                         combinations
120  SCREEN ,,0:PALETTE
     0,J-1 MOD 15              'change the palette
125  REM                         position color on VDC
130  SCREEN ,,128:PALETTE
     5,J                       'change the palette
135  REM                         position color on DEB
140  FOR A=1 TO 500:NEXT A 'wait a bit
150  IF (LEN(INKEY$)) <>0
     THEN 170                  'check for
155  REM                         keypress
160  NEXT J
170  SCREEN 0,0,0              'return to normal
180  END
```

The following program takes two color numbers as input and finds their position in the dither and blinking tables and makes colored boxes in each of the color effects.

```
40  SCREEN 101                    '16 Color 640 x 200
45  REM                             graphics mode
50  CLS:KEY OFF                   'clear screen
60  REM   Input the two colors and do range checking
70  LOCATE 2,2:INPUT "Enter Color 1 (0-15) " ,C1
80  IF C1 > 15 OR C1 < 0 THEN GOTO 70
90  LOCATE 3,2:INPUT "Enter Color 2 (0-15) " ,C2
100 IF C2 > 15 OR C2 < 0 THEN GOTO 90
110 IF C1 = C2 THEN INPUT "Colors must be different
       hit <CR> ",A$:CLS:GOTO 70
120 REM Set one color to high and one to low to
125 REM determine the position in the respective
130 REM tables
140 IF C1 < C2 THEN LOW = C1:HIGH = C2
                ELSE LOW = C2:HIGH = C1
150 REM Blinking is the sum of 16-I as I ranges
155 REM from 0 to the lower of the two colors
160 REM then adding the higher of the two colors
170 ROWMIN = 0
180 FOR I = 0 TO LOW
190 ROWMIN = ROWMIN + (16-I)
200 NEXT I
210 BLINKCOL = ROWMIN + (HIGH-LOW-1)
220 LOCATE 22,1
230 PRINT "Blinking Number is ";BLINKCOL;
240 REM Dithering is 136 plus the sum of I + 1
245 REM as I ranges from 1 to the higher of the
250 REM two colors plus the lower color.
260 ROWMIN = 0
270 FOR I = 1 TO HIGH
280 ROWMIN = ROWMIN + (I-1)
290 NEXT I
295 REM example continued on next page
```

```
300 DITHERCOL= ROWMIN + 136 + LOW
310 LOCATE 22,42
320 PRINT "Dithered Color Number is ";DITHERCOL
330 REM Set palette position 1 equal to the
335 REM result of the blinking color
340 REM and palette position 2 equal to the
345 REM result of the dithering color
350 PALETTE 1,BLINKCOL
360 PALETTE 2,DITHERCOL
370 REM draw a box with the blinking and
375 REM dithered color effects.
380 LINE (100,50)-(210,150),1,BF
390 LINE (420,50)-(530,150),2,BF
400 GOTO 70
```

The following program shows a box containing a circle and how the GET statement and the PUT statement work with the DEB. The GET array takes four times as much storage as it does in non-DEB graphics.

```
 40 DIM PIC%(3000)              'GET array
 50 KEY OFF                     'turn off function keys
 60 SCREEN 102                  'set 16 color graphics
 70 FOR X = 1 TO 15
 80 CLS                         'clear screen
 90 CIRCLE (100,100),50,1       'draw circle
100 LINE (49,50)-(151,150), 15-X,B
105 REM draw a box around the circle
110 PAINT (100,100),X,1         'fill the circle
120 GET (49,50)-(151,150),
    PIC%                        'get the graphics
125 REM                           image
130 FOR J = 1 TO 200 STEP 50
140 FOR I = 0 TO 50 STEP 10
150 PUT (RND*537+1,RND*297+1), PIC%,PSET
155 REM                         'put it randomly on the
157 REM                           screen
160 IF LEN(INKEY$) <> 0
    THEN 210                    'see if key
165 REM                           pressed
170 NEXT I
180 NEXT J
190 NEXT X
200 GOTO 70
210 SCREEN 0,0,0                'return to normal
220 END
```

The following program shows the use of a variety of DEB features. It includes a setup procedure to help you adjust your monitor for best viewing of DEB effects.

```
1100 REM Display Enhancement Board
1200 REM Monitor Setup Program
1300 REM
1400 SCREEN 0,0,0
1500 KEY OFF:CLS
1600 REM
1700 REM The following is a way to easily center
1800 REM the title text
1900 A$="AT&T PC-6300"
1910 LOCATE 1,(80-LEN(A$))/2;
1920 PRINT A$                        'Center text
2000 A$="DISPLAY ENHANCEMENT BOARD"
2010 LOCATE 2,(80-LEN(A$))/2:PRINT A$
2100 A$="MONITOR SETUP PROGRAM"
2110 LOCATE 3,(80-LEN(A$))/2:PRINT A$
2200 LOCATE 10,1:INPUT "Enter Monitor type
             ('MONO' or 'COLOR')";M$
2300 IF LEFT$(M$,1)="M" OR LEFT$(M$,1)="m"
          THEN GOTO 2900
2400 IF LEFT$(M$,1)="C" OR LEFT$(M$,1)="c"
          THEN GOTO 5000
2500 PRINT
2510 PRINT CHR$(7);"Can not use "";M$;"" as a monitor
          type"
2600 FOR A=1 TO 3000:NEXT A
2700 GOTO 2200
2800 REM
2900 REM Monochrome Monitor Setup
3000 REM
3100 DIM PAL(16)
3200 SCREEN 102: CLS
3300 FOR A=0 TO 15
3310 READ PAL(A):PALETTE A,PAL(A)
```

```
3320 NEXT A                        'setup gray levels
3400 FOR A=0 TO 15
3500 LINE (A*40,40)-(40+A*40,140), A,BF
3510 REM                           'draw shaded areas
3600 LINE (A*40,240)- (40+A*40,340), 15-A,BF
3610 REM draw inverted shaded areas
3700 NEXT A
3800 COLOR 15                      'use high intensity white
3810 REM                             for text
3900 LOCATE 1,20;
3910 PRINT "Adjust to get a complete shade scale"
4000 LOCATE 11,26;
4010 PRINT "Dark <-------------------> Light"
4100 LOCATE 14,25;
4110 PRINT "Light <-------------------> Dark"
4200 LOCATE 25,30;
4210 PRINT "(Hit any key to exit)";
4300 A$=INKEY$:IF LEN(A$)=0 THEN 4300
4310 REM 'wait for any key to be pressed
4400 SCREEN 0
4500 REM
4600 REM The data below is the palette for
4700 REM shades of green
4800 DATA 0,8,1,9,4,12,5,13, 2,10,3,11,6,14,7,15
4900 END
5000 REM
5100 REM Color Monitor Setup
5200 REM
5300 SCREEN 102:CLS
5400 COLOR ,,2                     'select standard color
5410 REM                             palette
5500 FOR A=0 TO 7
5600 LINE (A*40,0)-(40+A*40,199), A,BF
5610 REM draw colored filled boxes
5700 LINE (A*40,202)-(40+A*40,400), A+8,BF
5800 NEXT A
5900 COLOR 15                      'use high intensity white
5910 REM                             for text
6000 LOCATE 6,45: PRINT "Low intensity Colors"
6100 LOCATE 20,45: PRINT "High Intensity Colors"
```

```
6200 LOCATE 12,45;
6210 PRINT "Adjust Contrast and Brightness"
6300 LOCATE 13,45: PRINT "Controls to display 16"
6400 LOCATE 14,45: PRINT "different colors"
6500 LOCATE 25,50: PRINT "(Hit any key to exit)";
6600 A$ = INKEY$:IF LEN(A$) = 0 THEN 6600
6610 REM wait for a key to be pressed
6700 SCREEN 0                    'reset the screen mode
6800 END
```

The following program shows a text screen scrolling on top of a graphics screen.

```
 20 SCREEN 104               'set text on graphics
 22 REM                        mode
 25 CLS : KEY OFF
 30 N=15:ANGLE = 360 / N   'calculate # of angles
 40 RADIANS = ANGLE /57.29578
 50 CLS                      'clear screen
 60 FOR X = 1 TO N           'do the real work
 70 FOR Y = X TO N
 80 SX = SIN(X * RADIANS) * 195 + 320
 90 SY = SIN(Y * RADIANS) * 195 + 320
100 CX = COS(X * RADIANS) * 150 + 200
110 CY = COS(Y * RADIANS) * 150 + 200
120 LINE (SY,CY)-(SX,CX), INT(RND*(7)+1)
125 REM draw line with random color
130 NEXT Y,X
140 FOR I = 1 TO 1000
150 X=RND*14+1
155 Y=RND*50+1
157 COLOR ,,(RND*30),(RND*15)
159 GOSUB 270                'print text on VDC
160 X=RND*17+1
161 Y=RND*50+1
163 COLOR ,,0,(RND*31+1)
165 GOSUB 270                'print text on VDC
167 COLOR ,,,0               'change palette
170 LOCATE 24,1

180 FOR K = 1 TO 7
190 PALETTE K,RND*135+1  'change palette
200 PRINT                    'scroll text
210 IF(LEN(INKEY$)) <> 0 THEN 240
215 REM                      check for keypress
220 NEXT K
230 NEXT I
240 SCREEN 0,0,0             'return to normal
250 END
```

```
260 REM sub to display a box of text
270 LOCATE X,Y : PRINT CHR$(201);
280 FOR I = 1 TO 29:PRINT CHR$(205); :NEXT I
290 PRINT CHR$(187);
300 LOCATE X+1,Y;
305 PRINT CHR$(186)+"This box is on the VDC
                   screen" +CHR$(186);
310 LOCATE X+2,Y;
315 PRINT CHR$(186)+"This is more text"
                   +CHR$(186);
320 LOCATE X+3,Y;
325 PRINT CHR$(186)+"This is the last line of text"
                   +CHR$(186);
330 LOCATE X+4,Y: PRINT CHR$(200);
340 FOR I = 1 TO 29: PRINT CHR$(205);:NEXT I
350 PRINT CHR$(188);
360 RETURN
```

4 Programming the LUT

- Overview
- 16-Color Graphics LUT Programming
- Overlay Modes LUT Programming

OVERVIEW

This chapter describes programming the DEB look-up table (LUT). By programming the LUT yourself, you can create color patterns that are not available when you use standard palettes.

You need not read this chapter if you do not want to use this extended functionality.

The hardware uses the LUT to translate the contents of video memory into graphics effects. In the standard palettes, GWBASIC programs the LUT for you and thereby provides the pre-assigned color combinations and effects described in previous chapters.

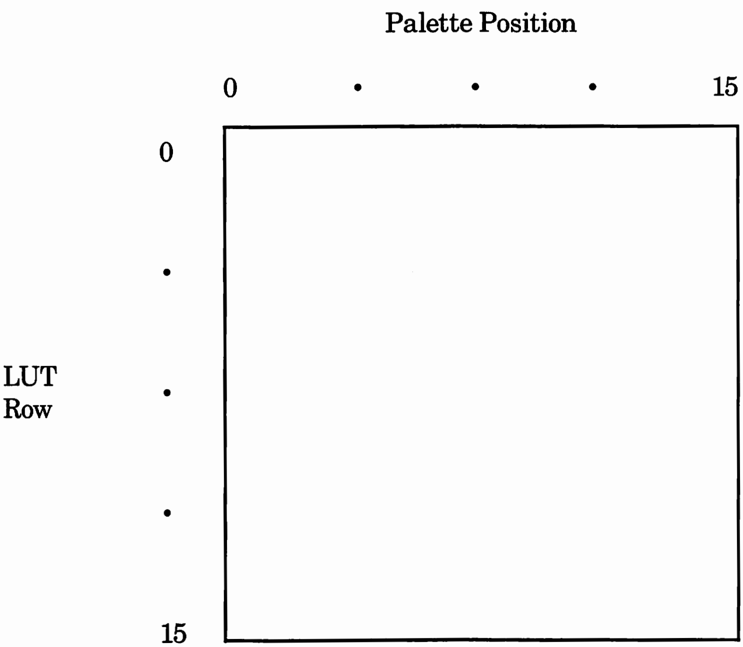
To program the LUT directly, you select Palette 4 in the COLOR statement. Palette 4, also called the "LUT palette," has a minimum of 256 positions. The contents of each palette position is an integer value between 0 and 15. These values map into the LUT locations on the DEB. The 256 locations on the DEB collectively determine the color and special effects displayed when you specify a particular palette position in a graphics statement. The color and special effect for each pixel on the screen are determined by:

- the palette position you specify
- the values in the LUT
- the active mode

There are some differences in the way the LUT is structured for 16-color graphics modes and overlay modes. This chapter describes LUT operation for 16-color graphics modes and overlay modes separately.

16-COLOR GRAPHICS LUT PROGRAMMING

In these modes the LUT can be viewed as a two-dimensional array (16 × 16). Each location contains one of the standard 16 colors.



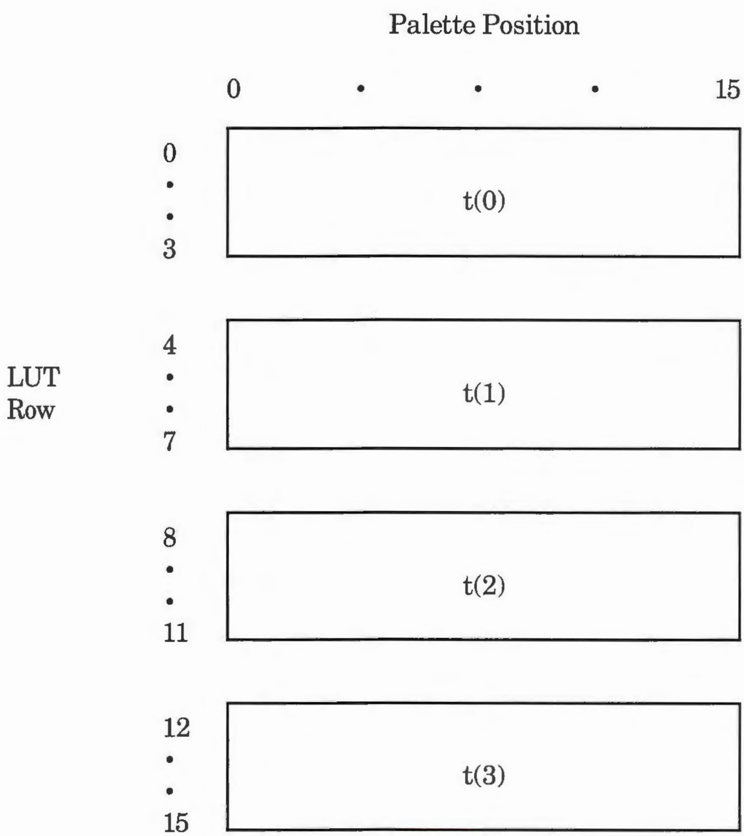
The locations in the LUT are numbered consecutively from left to right and top to bottom. Thus, location 17 corresponds to Row 1, palette position 1. This correspondence is used with both the PALETTE and PALETTE USING statements. To set location 17 to color 1 (blue) you would either use:

```
PALETTE 17,1
```

or

```
INTARRAY (17) = 1
PALETTE USING INTARRAY (0)
```

In the 16-color graphics mode, the LUT is divided into four "time states." At any one time, only one quarter of the LUT determines the display on the screen.



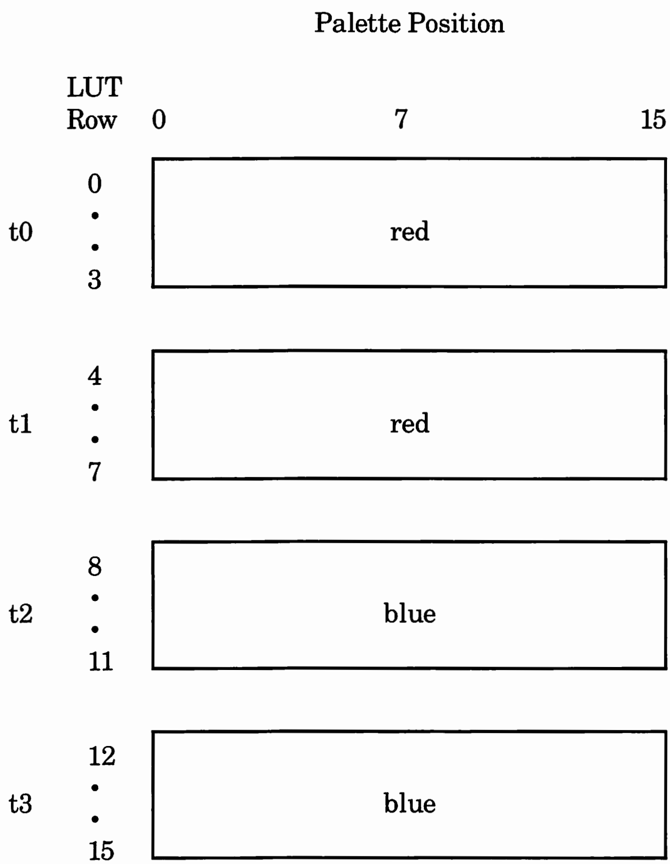
The hardware cycles through the LUT every second, so each quarter of the LUT is active for ¼ of each second. The cycling mechanism produces blinking. The following examples show the details of how you can produce several different blinking effects by setting different values in the LUT.

In this example, the graphics statements specify palette position 7 and the LUT is set up as shown. Pixels are displayed as a solid red color. In the first ¼ second, the DEB displays the color in the first quarter of the LUT, which in this case is red. In the second, third, and fourth ¼ seconds, the DEB displays the color in the second, third, and fourth quarters of the LUT, respectively. In this example, the DEB keeps finding the color value for red, so what you see on the screen is a solid (non-blinking) red color.

	LUT Row	Palette Position 0 … 7 … 15
t0	0 • • 3	red
t1	4 • • 7	red
t2	8 • • 11	red
t3	12 • • 15	red

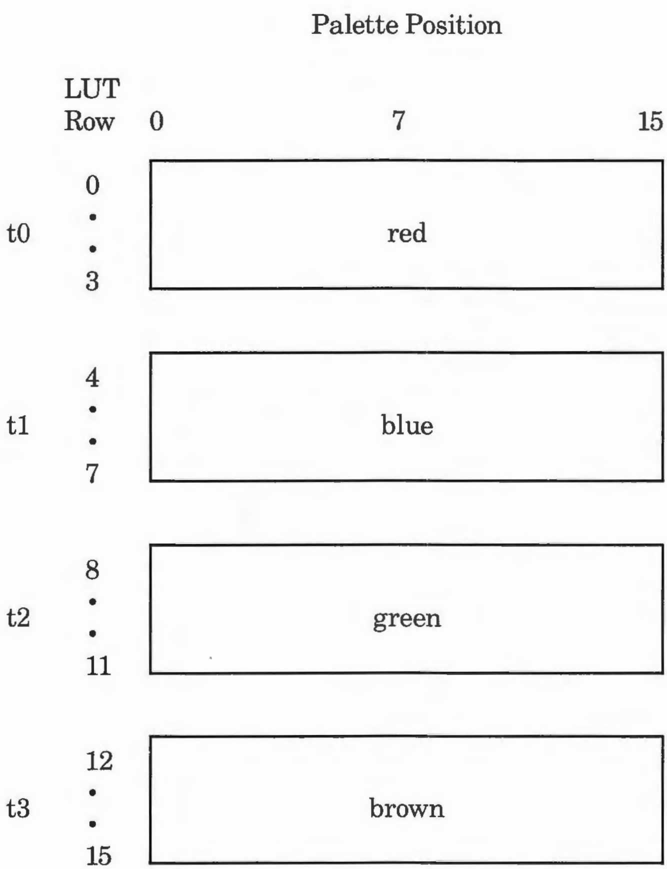
Non-Blinking Color

In this example, any item displayed on the screen with palette position 7 blinks between red and blue. For the first two ¼ seconds, the DEB picks up the color value for red from the first and second quarters of the LUT. For the second two ¼ seconds, the DEB obtains the color value of blue from the LUT. The net effect is a slow blink between red and blue.



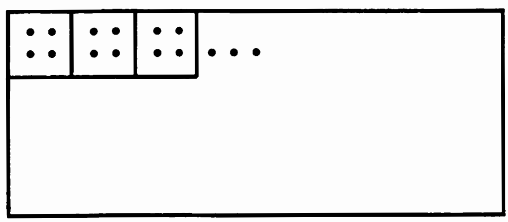
Slow Blink

In this example, any item displayed using palette position 7 blinks rapidly between red, blue, green, and brown.

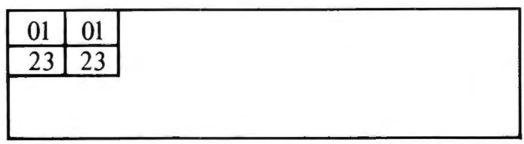


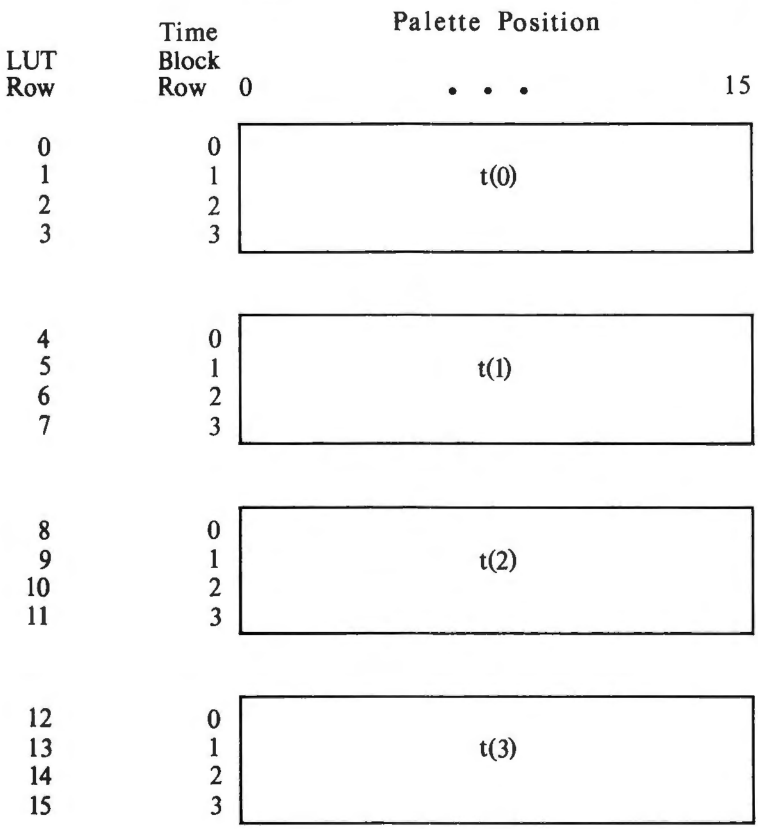
4-Color Fast Blink

For dithering colors, the DEB uses a scheme similar to the blinking scheme. Dithering is accomplished by manipulating groups of 4 adjacent pixels. The screen is divided into blocks of 4 pixels.



Each of the 4 time states is divided into four dither states that determine the dithering effect. The rows of the time state blocks correspond to the 4-pixel blocks on the screen in the following way:





The pixels in the pixel blocks are so close together that our eyes cannot perceive them as separate. If each of the pixels in a pixel block is a different color, our eyes perceive the pixel block as one color — a combination of the color of the individual pixels. If the adjacent pixels are the same color, our eyes see just that one color.

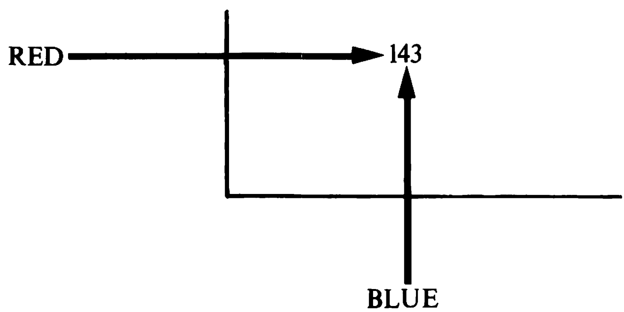
red	red
red	red

Palette Position

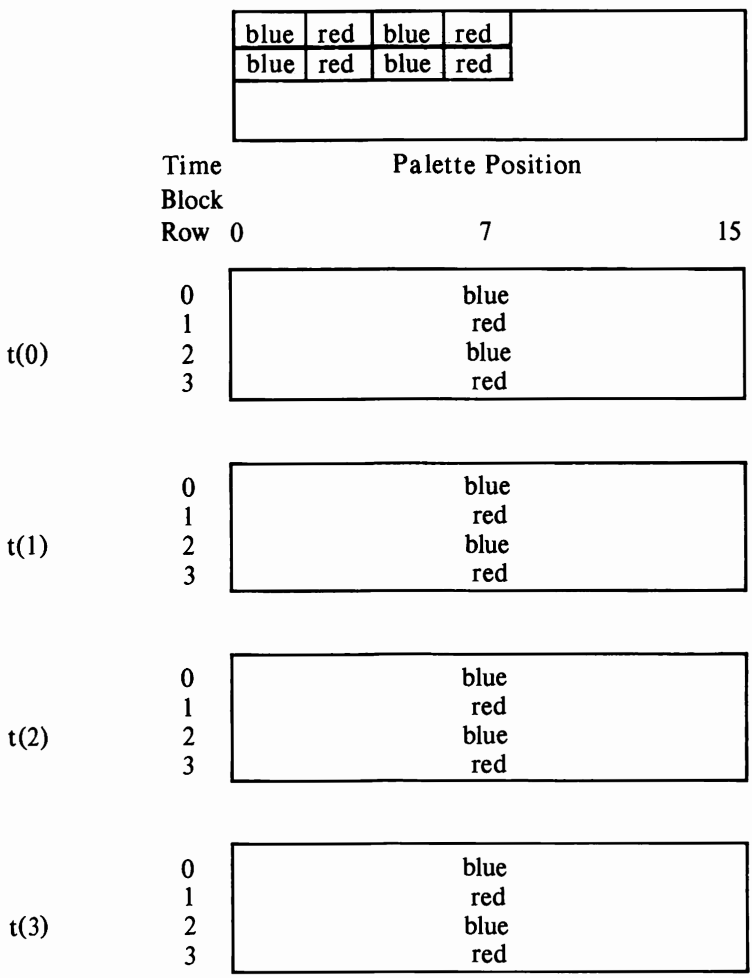
Time Block	Row	0	7	15
t(0)	0		red	
	1		red	
	2		red	
	3		red	
t(1)	0		red	
	1		red	
	2		red	
	3		red	
t(2)	0		red	
	1		red	
	2		red	
	3		red	
t(3)	0		red	
	1		red	
	2		red	
	3		red	

"Solid" Dither showing correspondence between pixel positions in a pixel block and time state rows

Remember the table of "pre-assigned" dithered colors in Chapter 3. To combine colors, you check the table for the color number for a particular dither effect. For example, you would choose this number to produce a dither between red and blue.



If you want to program the LUT to dither red and blue together, the LUT would look like this:



2-Color Dither

You can set up the LUT to dither two, three, or four colors together.

red	blue	red	blue
grn	brn	grn	brn

Palette Position

Time	Block Row	0 — 7 — 15
t(0)	0	red
	1	blue
	2	green
	3	brown
t(1)	0	red
	1	blue
	2	green
	3	brown
t(2)	0	red
	1	blue
	2	green
	3	brown
t(3)	0	red
	1	blue
	2	green
	3	brown

4-Color Dither

The following examples show the actual LUT values for each of the previous cases of blinking and dithering.

Palette Position

	LUT Row	0	7	15
	0		4 (red)	
	1		4	
t(0)	2		4	
	3		4	
	4		4	
	5		4	
t(1)	6		4	
	7		4	
	8		4	
	9		4	
t(2)	10		4	
	11		4	
	12		4	
	13		4	
t(3)	14		4	
	15		4	

Palette Position 7 programmed for Non-Blinking Red

Palette Position

	LUT Row	0	7	15
	0		4 (red)	
	1		4	
t(0)	2		4	
	3		4	
	4		4	
	5		4	
t(1)	6		4	
	7		4	
	8		1 (blue)	
	9		1	
t(2)	10		1	
	11		1	
	12		1	
	13		1	
t(3)	14		1	
	15		1	

Palette Position 7 programmed to blink slowly between red and blue.

Palette Position

	LUT Row	0	7	15
	0		4 (red)	
	1		4	
t(0)	2		4	
	3		4	
	4		1 (blue)	
	5		1	
t(1)	6		1	
	7		1	
	8		2 (green)	
	9		2	
t(2)	10		2	
	11		2	
	12		6 (brown)	
	13		6	
t(3)	14		6	
	15		6	

4-Color Fast Blink

Palette Position

	LUT Row	0	7	15
	0		4 (red)	
	1		4	
t(0)	2		4	
	3		4	
	4		4	
	5		4	
t(1)	6		4	
	7		4	
	8		4	
	9		4	
t(2)	10		4	
	11		4	
	12		4	
	13		4	
t(3)	14		4	
	15		4	

Solid Red Dither

Palette Position

	LUT Row	0	7	15
	0		1 (blue)	
	1		4 (red)	
t(0)	2		1 (blue)	
	3		4 (red)	
	4		1	
	5		4	
t(1)	6		1	
	7		4	
	8		1	
	9		4	
t(2)	10		1	
	11		4	
	12		1	
	13		4	
t(3)	14		1	
	15		4	

2-Color Dither: Red and Blue

Palette Position

	LUT Row	0	7	15
	0		4 (red)	
	1		2 (green)	
t(0)	2		1 (blue)	
	3		6 (brown)	
	4		4	
	5		2	
t(1	6		1	
	7		6	
	8		4	
	9		2	
t(2	10		1	
	11		6	
	12		4	
	13		2	
t(3)	14		1	
	15		6	

4-Color Dither Between Red, Green, Blue, and Brown

The following is an example that combines blinking and dithering:

Palette Position

	LUT Row	0	7	15
	0		1 (blue)	
	1		4 (red)	
t(0)	2		1	
	3		4	
	4		1	
	5		4	
t(1)	6		1	
	7		4	
	8		2 (green)	
	9		6 (brown)	
t(2)	10		2	
	11		6	
	12		2	
	13		6	
t(3)	14		2	
	15		6	

The following table of values can be used to program the LUT for normal 16-color graphics.

	LUT Row	Palette Position															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	1	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
t(o)	2	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	3	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	4	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	5	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
t(1)	6	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	7	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	8	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	9	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
t(2)	10	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	11	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	12	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	13	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
t(3)	14	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,
	15	0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	14,	15,

Non-Blinking Standard Colors

Note that palette position 7 in the first two time states has been programmed to show white and in the second two time states to show red.

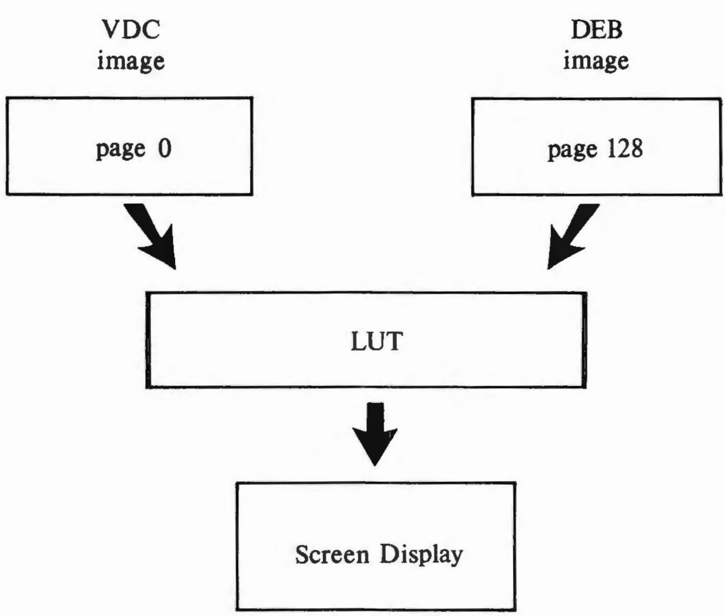
		Palette Position															
	LUT Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0,	1,	2,	3,	4,	5,	6,	**7,**	8,	9,	10,	11,	12,	13,	14,	15,
	1	0,	1,	2,	3,	4,	5,	6,	**7,**	8,	9,	10,	11,	12,	13,	14,	15,
t(o)	2	0,	1,	2,	3,	4,	5,	6,	**7,**	8,	9,	10,	11,	12,	13,	14,	15,
	3	0,	1,	2,	3,	4,	5,	6,	**7,**	8,	9,	10,	11,	12,	13,	14,	15,
	4	0,	1,	2,	3,	4,	5,	6,	**7,**	8,	9,	10,	11,	12,	13,	14,	15,
	5	0,	1,	2,	3,	4,	5,	6,	**7,**	8,	9,	10,	11,	12,	13,	14,	15,
t(1)	6	0,	1,	2,	3,	4,	5,	6,	**7,**	8,	9,	10,	11,	12,	13,	14,	15,
	7	0,	1,	2,	3,	4,	5,	6,	**7,**	8,	9,	10,	11,	12,	13,	14,	15,
	8	0,	1,	2,	3,	4,	5,	6,	**4,**	8,	9,	10,	11,	12,	13,	14,	15,
	9	0,	1,	2,	3,	4,	5,	6,	**4,**	8,	9,	10,	11,	12,	13,	14,	15,
t(2)	10	0,	1,	2,	3,	4,	5,	6,	**4,**	8,	9,	10,	11,	12,	13,	14,	15,
	11	0,	1,	2,	3,	4,	5,	6,	**4,**	8,	9,	10,	11,	12,	13,	14,	15,
	12	0,	1,	2,	3,	4,	5,	6,	**4,**	8,	9,	10,	11,	12,	13,	14,	15,
	13	0,	1,	2,	3,	4,	5,	6,	**4,**	8,	9,	10,	11,	12,	13,	14,	15,
t(3)	14	0,	1,	2,	3,	4,	5,	6,	**4,**	8,	9,	10,	11,	12,	13,	14,	15,
	15	0,	1,	2,	3,	4,	5,	6,	**4,**	8,	9,	10,	11,	12,	13,	14,	15,

LUT for Blinking Between White and Red in Palette Position 7

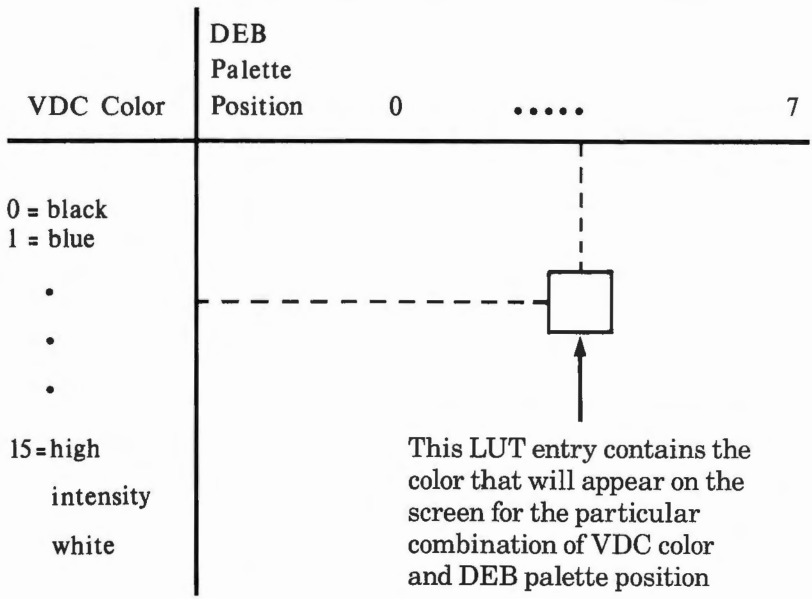
OVERLAY MODES LUT PROGRAMMING

When the LUT is used in the overlay modes it can be viewed as a two-dimensional array with 8 columns and 32 rows. The column values are DEB palette positions. The row values are VDC color values.

In overlay modes, there are 2 separately controlled images: the VDC image and the DEB image. The 2 images are combined on the display screen. Each pixel on the screen has 2 values associated with it: the VDC color and the DEB palette position. The LUT is used to resolve contention between the 2 values associated with each pixel.



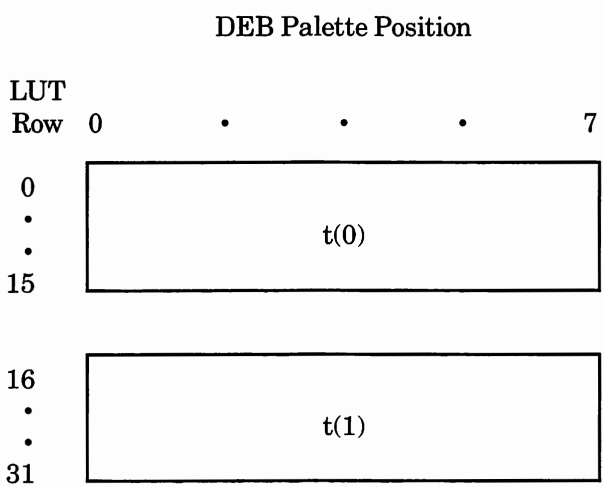
The LUT for overlay modes looks like this:



As in the 16-color graphics modes, the locations in the LUT are numbered consecutively from left to right and top to bottom. For example, location 17 corresponds to Row 2, Palette Position 0.

In the overlay modes, as in the 16-color graphics mode, the LUT is divided into time states that control blinking effects. However, in the overlay modes, the LUT is only divided into two time states. Half of the LUT determines what is being displayed at any time. The top half is used for the first ½ of each second and the bottom half is used for the second ½ of each second.

Using the overlay modes, you create blinking by making the values in the top half of the table different from the corresponding values in the bottom half of the table.



The following example shows the LUT values for standard Palette 2 of an overlay mode. The LUT is programmed so that the DEB image is displayed only if the VDC color is 0 (black). If the VDC requests any other color, then that color is displayed no matter what the DEB requests. This has the effect of overlaying the VDC image "on top" of the DEB image.

	VDC Color Values	DEB Palette Position 0	1	2	3	4	5	6	7
	0	**0,**	**1,**	**2,**	**3,**	**4,**	**5,**	**6,**	**7,**
	1	1,	1,	1,	1,	1,	1,	1,	1,
	2	2,	2,	2,	2,	2,	2,	2,	2,
	3	3,	3,	3,	3,	3,	3,	3,	3,
t(0)	4	4,	4,	4,	4,	4,	4,	4,	4,
	5	5,	5,	5,	5,	5,	5,	5,	5,
	6	6,	6,	6,	6,	6,	6,	6,	6,
	7	7,	7,	7,	7,	7,	7,	7,	7,
	8	8,	8,	8,	8,	8,	8,	8,	8,
	9	9,	9,	9,	9,	9,	9,	9,	9,
	10	10,	10,	10,	10,	10,	10,	10,	10,
	11	11,	11,	11,	11,	11,	11,	11,	11,
	12	12,	12,	12,	12,	12,	12,	12,	12,
	13	13,	13,	13,	13,	13,	13,	13,	13,
	14	14,	14,	14,	14,	14,	14,	14,	14,
	15	15,	15,	15,	15,	15,	15,	15,	15,

	VDC Color Values	0	1	2	3	4	5	6	7
		DEB Palette Position							
	0	**0,**	**1,**	**2,**	**3,**	**4,**	**5,**	**6,**	**7,**
	1	1,	1,	1,	1,	1,	1,	1,	1,
	2	2,	2,	2,	2,	2,	2,	2,	2,
	3	3,	3,	3,	3,	3,	3,	3,	3,
t(1)	4	4,	4,	4,	4,	4,	4,	4,	4,
	5	5,	5,	5,	5,	5,	5,	5,	5,
	6	6,	6,	6,	6,	6,	6,	6,	6,
	7	7,	7,	7,	7,	7,	7,	7,	7,
	8	8,	8,	8,	8,	8,	8,	8,	8,
	9	9,	9,	9,	9,	9,	9,	9,	9,
	10	10,	10,	10,	10,	10,	10,	10,	10,
	11	11,	11,	11,	11,	11,	11,	11,	11,
	12	12,	12,	12,	12,	12,	12,	12,	12,
	13	13,	13,	13,	13,	13,	13,	13,	13,
	14	14,	14,	14,	14,	14,	14,	14,	14,
	15	15,	15,	15,	15,	15,	15,	15,	15,

In this example, the standard Palette 2 is modified so that position 2 is a blinking between blue (color 1) and red (color 4).

DEB Palette Position

	VDC Color Values	0	1	2	3	4	5	6	7
	0	0,	1,	**1,**	3,	4,	5,	6,	7,
	1	1,	1,	1,	1,	1,	1,	1,	1,
	2	2,	2,	2,	2,	2,	2,	2,	2,
	3	3,	3,	3,	3,	3,	3,	3,	3,
t(0)	4	4,	4,	4,	4,	4,	4,	4,	4,
	5	5,	5,	5,	5,	5,	5,	5,	5,
	6	6,	6,	6,	6,	6,	6,	6,	6,
	7	7,	7,	7,	7,	7,	7,	7,	7,
	8	8,	8,	8,	8,	8,	8,	8,	8,
	9	9,	9,	9,	9,	9,	9,	9,	9,
	10	10,	10,	10,	10,	10,	10,	10,	10,
	11	11,	11,	11,	11,	11,	11,	11,	11,
	12	12,	12,	12,	12,	12,	12,	12,	12,
	13	13,	13,	13,	13,	13,	13,	13,	13,
	14	14,	14,	14,	14,	14,	14,	14,	14,
	15	15,	15,	15,	15,	15,	15,	15,	15,

VDC Color Values	DEB Palette Position 0	1	2	3	4	5	6	7
0	0,	1,	4,	3,	4,	5,	6,	7,
1	1,	1,	1,	1,	1,	1,	1,	1,
2	2,	2,	2,	2,	2,	2,	2,	2,
3	3,	3,	3,	3,	3,	3,	3,	3,
t(1) 4	4,	4,	4,	4,	4,	4,	4,	4,
5	5,	5,	5,	5,	5,	5,	5,	5,
6	6,	6,	6,	6,	6,	6,	6,	6,
7	7,	7,	7,	7,	7,	7,	7,	7,
8	8,	8,	8,	8,	8,	8,	8,	8,
9	9,	9,	9,	9,	9,	9,	9,	9,
10	10,	10,	10,	10,	10,	10,	10,	10,
11	11,	11,	11,	11,	11,	11,	11,	11,
12	12,	12,	12,	12,	12,	12,	12,	12,
13	13,	13,	13,	13,	13,	13,	13,	13,
14	14,	14,	14,	14,	14,	14,	14,	14,
15	15,	15,	15,	15,	15,	15,	15,	15,

In this example, values in the LUT cause the DEB's output to take precedence over the VDC's output. The VDC's output is only displayed when you specify DEB palette position 0 in a graphics statement.

		DEB Palette Positions							
	VDC Color Values	0	1	2	3	4	5	6	7
	0	**0,**	1,	2,	3,	4,	5,	6,	7,
	1	**1,**	1,	2,	3,	4,	5,	6,	7,
	2	**2,**	1,	2,	3,	4,	5,	6,	7,
	3	**3,**	1,	2,	3,	4,	5,	6,	7,
	4	**4,**	1,	2,	3,	4,	5,	6,	7,
	5	**5,**	1,	2,	3,	4,	5,	6,	7,
	6	**6,**	1,	2,	3,	4,	5,	6,	7,
t(o)	7	**7,**	1,	2,	3,	4,	5,	6,	7,
	8	**8,**	1,	2,	3,	4,	5,	6,	7,
	9	**9,**	1,	2,	3,	4,	5,	6,	7,
	10	**10,**	1,	2,	3,	4,	5,	6,	7,
	11	**11,**	1,	2,	3,	4,	5,	6,	7,
	12	**12,**	1,	2,	3,	4,	5,	6,	7,
	13	**13,**	1,	2,	3,	4,	5,	6,	7,
	14	**14,**	1,	2,	3,	4,	5,	6,	7,
	15	**15,**	1,	2,	3,	4,	5,	6,	7,

DEB Palette Positions

	VDC Color Values	0 1 2 3 4 5 6 7
	1	**0,** 1, 2, 3, 4, 5, 6, 7,
	1	**0,** 1, 2, 3, 4, 5, 6, 7,
	2	**2,** 1, 2, 3, 4, 5, 6, 7,
	3	**3,** 1, 2, 3, 4, 5, 6, 7,
	4	**4,** 1, 2, 3, 4, 5, 6, 7,
t(1)	5	**5,** 1, 2, 3, 4, 5, 6, 7,
	6	**6,** 1, 2, 3, 4, 5, 6, 7,
	7	**7,** 1, 2, 3, 4, 5, 6, 7,
	8	**8,** 1, 2, 3, 4, 5, 6, 7,
	9	**9,** 1, 2, 3, 4, 5, 6, 7,
	10	**10,** 1, 2, 3, 4, 5, 6, 7,
	11	**11,** 1, 2, 3, 4, 5, 6, 7,
	12	**12,** 1, 2, 3, 4, 5, 6, 7,
	13	**13,** 1, 2, 3, 4, 5, 6, 7,
	14	**14,** 1, 2, 3, 4, 5, 6, 7,
	15	**15,** 1, 2, 3, 4, 5, 6, 7,

The following LUT entirely blocks out VDC output:

DEB Palette Positions

	VDC Color Values	0 1 2 3 4 5 6 7
	0	0, 1, 2, 3, 4, 5, 6, 7,
	1	0, 1, 2, 3, 4, 5, 6, 7,
	2	0, 1, 2, 3, 4, 5, 6, 7,
	3	0, 1, 2, 3, 4, 5, 6, 7,
	4	0, 1, 2, 3, 4, 5, 6, 7,
t(0)	5	0, 1, 2, 3, 4, 5, 6, 7,
	6	0, 1, 2, 3, 4, 5, 6, 7,
	7	0, 1, 2, 3, 4, 5, 6, 7,
	8	0, 1, 2, 3, 4, 5, 6, 7,
	9	0, 1, 2, 3, 4, 5, 6, 7,
	10	0, 1, 2, 3, 4, 5, 6, 7,
	11	0, 1, 2, 3, 4, 5, 6, 7,
	12	0, 1, 2, 3, 4, 5, 6, 7,
	13	0, 1, 2, 3, 4, 5, 6, 7,
	14	0, 1, 2, 3, 4, 5, 6, 7,
	15	0, 1, 2, 3, 4, 5, 6, 7,

DEB Palette Positions

	VDC Color Values	0 1 2 3 4 5 6 7
	0	0, 1, 2, 3, 4, 5, 6, 7,
	1	0, 1, 2, 3, 4, 5, 6, 7,
	2	0, 1, 2, 3, 4, 5, 6, 7,
	3	0, 1, 2, 3, 4, 5, 6, 7,
	4	0, 1, 2, 3, 4, 5, 6, 7,
t(1)	5	0, 1, 2, 3, 4, 5, 6, 7,
	6	0, 1, 2, 3, 4, 5, 6, 7,
	7	0, 1, 2, 3, 4, 5, 6, 7,
	8	0, 1, 2, 3, 4, 5, 6, 7,
	9	0, 1, 2, 3, 4, 5, 6, 7,
	10	0, 1, 2, 3, 4, 5, 6, 7,
	11	0, 1, 2, 3, 4, 5, 6, 7,
	12	0, 1, 2, 3, 4, 5, 6, 7,
	13	0, 1, 2, 3, 4, 5, 6, 7,
	14	0, 1, 2, 3, 4, 5, 6, 7,
	15	0, 1, 2, 3, 4, 5, 6, 7,