```
*********************************** *****************
* LOW LEVEL GRAPHICS INTRINSICS * * SECTION 2.1.4 *
*********************************** *****************
```

Version I.5 September 1978
Updated April 1979, TERAK Corporation
Pub no. 60-0024-101

PRELIMINARY INFORMATION

Graphics on the TERAK 8510/a are bit mapped raster scan graphics, refreshed directly from main memory. The display presented is a composite this 240 by 320 dot graphics display with the 24 by 80 character display. Two 8510/a registers, in the I/O memory page, control the display of graphics: the Graphics Address Register (GAR) contains the starting address of the memory to be displayed as graphics. The Video Control Register (VCR) controls the blanking/unblanking of the graphics and characters on the video monitor. Detailed descriptions of the operatons of these registers are contained in the VIDEO DISPLAY AND 24K MEMORY SYSTEM User Reference Guide, TERAK Publication Number 52-0002-001.

The GAR and VCR may be set from high level Pascal code by the UNITWRITE intrinsic operating on Unit #3. Before issuing a call to UNITWRITE, the Pascal program should have allocated memory for graphics by declaring a variable. For example, the following statements allocate one picture space:

```
TYPE
TERAKSCREEN = RECORD
    CASE INTEGER OF
        1:( BITS:PACKED ARRAY[0..239] OF PACKED ARRAY[0..319] OF BOOLEAN);
        2:( CHRS:PACKED ARRAY[0..9599] OF CHAR);
        3:( INTS:ARRAY[0..4799] OF INTEGER;);
        4:( SETS:ARRAY[0..4799] OF SET OF [0..15]
    END;(*CASE*)

VAR SCREEN :TERAKSCREEN;
```

These allocate one picture-full of memory the the variable SCREEN. The screen contents can be manipulated either by direct assignment:

```
                      SCREEN.BITS[10,100]:= TRUE
```

(which lights the dot at row 10, column 100) by I/O intrinsics:

```
          RESET(PIX,'PIX.FOTO); UNITREAD(PIX,SCREEN,19);
```

(which loads a .FOTO file into the memory) by high level operations:

```
    FOR I:=0 TO 4799 DO SCREEN.SETS[I]:=[0..15] - SCREEN.SETS[I];
```

(which reverses the entire picture) or by the graphics intrinsics supplied with the Terak release of the Pascal system, documented below. Note that a picture memory space need not be a full screen, and need not necessarily be displayed while being manipulated.

Typically, the picture memory space must be initialized to all blanks or all dots lit. These can be accomplished, respectively, by these two statements:

```
FILLCHAR(SCREEN,9600,CHR(0))                 for blanks, or
FILLCHAR(SCREEN,9600,CHR(255))               for all dots lit.
```

The generic call of UNITWRITE to volume #3 connects the graphics display hardware of the 8510/a with the allocated picture memory:

```
                    UNITWRITE(3,GARVAL,VCRVAL);
where GARVAL = <base of picture memory>,
and   VCRVAL = <integer zone blanking variable>
```

The GARVAL directs the location of the graphics display memory, and the VCRVAL parameter directs which of the character and graphics zones of the 8510/a are to be visible. When using UNITWRITE to volume #3 the address of the second parameter is loaded into the GAR, and the third parameter is loaded directly into the VCR. Thus, any of the bits in the VCR can be changed by placing the decimal representation of the bits into the third parameter of a UNITWRITE call to volume #3. Addresses loaded into the GAR must always be on even (integer) boundaries. The following illustrate different effects of the UNITWRITE parameters:

UNITWRITE(3,SCREEN,63);
Display 3 (all) zones of graphics from picture memory in SCREEN, and display 3 (all) zones of the character display.

UNITWRITE(3,SCREEN,56);
Display 3 (all) zones of graphics from picture memory in SCREEN, and blank all zones of the character display.

UNITWRITE(3,SCREEN,49);
Display upper two zones of graphics from picture memory in SCREEN, and lower one zone of the character display.

UNITWRITE(3,SCREEN.INTS[1600],19);
Display middle one zone of graphics from picture memory in SCREEN, starting at SCREEN[3200] thru SCREEN[4799], and lower one zone of the character display. The upper display zone is blanked. Note that the GAR must be directed to the virtual starting address of the upper zone, although it and other zones may be blanked.

UNITWRITE(3,I,263); UNITWRITE(3,I,63);
Blank all graphic display zones, unblank all character display zones, and sound a 'click' at the display by toggling the state of the Audio bit in the VCR. In this case, 'I' is a dummy second parameter.

GRAPHICS PROCEDURE CALLS

The Procedures DRAWLINE and DRAWBLOCK are provided by UCSD. The Procedures DRWBLK, GCHAR, GMARK, and THROTTLE are provide by TERAK. All procedures are contained in *SYSTEM.LIBRARY and must be declared EXTERNAL before use.

```
********************************************************************
****************************   WARNING   ***************************
** These graphics procedures do no range checking on parameters.  **
** If parameters passed to the procedures are 'out of bounds' the **
** procedures will produce unexpected results -- most likely,     **
** destruction of the user program, or operating system.          **
********************************************************************
```

DRAWLINE, DRAWBLOCK, and DRWBLK CONVENTIONS

The Coordinate System used by DRAWLINE, DRAWBLOCK, and DRWBLK fixes the point (Ø,Ø) in the upper left portion of the display. X and Y locations of the screen should be addressed using the following scheme.

```
(Ø,Ø)---------------------------(319,Ø)
  |                                |
  |   positive X direction RIGHT.  |
  |   positive Y direction DOWN.   |
  |                                |
  |                                |
  |                                |
(Ø,239)-------------------------(319,239)
```

DRAWLINE

This procedure draws lines in one of five modes, into memory. The screen address parameter must be on an even (integer) boundary. Note that the ROWWIDTH parameter indicates the width of the picture space, and is not necessarily restricted to the standard screen width. Picture space widths must be on integer boundaries; thus the parameter indicates the multiples of 16 bits of width required. Drawing into reduced width pictures is useful to prepare a sub-picture for transfer by DRAWBLK, which also has a width parameter. In all DRAWLINE calls, the starting bit is not affected by the line. RADAR mode will return the number of steps from the starting point to the nearest obstacle (bits set) along the line, into RANGE.

```
PROCEDURE DRAWLINE(
      VAR RANGE : INTEGER; {returns result of radar scan when PENSTATE=4}
      VAR SCREEN: TERAKSCREEN; {graphics memory}
          ROWWIDTH,          {#of 16 bit words per row, typically 20 }
          XSTART,            {beginning X point of line}
          YSTART,            {beginning Y point of line}
          DELTAX,            {distance to move in X}
          DELTAY,            {distance to move in Y}
          PENSTATE:INTEGER
             );  EXTERAL;
```

PENSTATE	ACTION	
Ø	PENUP	no change in picture
1	PENDOWN	force bits on
2	ERASE	force bits off
3	COMPLEMENT	reverse bits
4	RADAR	scan for obstacle, no change in picture

NOTE: A Pascal implementation of DRAWLINE is provided on page 159 of the UCSD PASCAL MANUAL.

DRAWBLOCK

This procedure will do a two-dimension oriented transfer of bits, from a source block into a target block. The source and destination block must be of the same width and height, but may be located at any bit location within the same or different picture memory spaces. Different picture memory spaces are allowed to have different widths. The effect which the source block has upon the target block is controlled by the mode parameter. Complement mode is useful to overlay a picture with a block image, and then erase it while restoring the original picture contents. Graphics animation typically makes use of Complement mode. NOTE: DRAWBLK calls which overlap the source and target blocks should be approached with caution. Note also that row widths are given in bits, not words (as in DRAWLINE), and must be a multiple of 8.

```
CONST
     SRCXSIZE = {# of bits in source x direction. Use ((multiple of 8)-1)}
     SRCYSIZE = {number of bits in source y direction}
     TGTXSIZE = 319; {320 bits in x when target is TERAKSCREEN}
     TGTYSIZE = 239; {240 bits in y when target is TERAKSCREEN}

TYPE
  TERAKSCREEN = PACKED ARRAY[0..TGTYSIZE] OF
                      PACKED ARRAY[0..TGTXSIZE] OF BOOLEAN;
  SRC         = ARRAY[0..SRCYSIZE] OF
                      PACKED ARRAY[0..SRCXSIZE] OF BOOLEAN;

PROCEDURE DRAWBLOCK(VAR SOURCE : SRC;      {source block}
                    SRCROW,                {#bits/row of block, multiple of 8}
                    SRCX,                  {x start location of source}
                    SRCY    :INTEGER;      {y start location of source}
                VAR DEST    :TERAKSCREEN;  {Destination block}
                    DSTROW,                {#bits/row of dst block, multple of 8}
                    DSTX,                  {x start location of destination}
                    DSTY,                  {y start location of destination}
                    CNTX,                  {number of bits to move x direction}
                    CNTY,                  {number of bits to move y direction}
                    MODE    :INTEGER       {see below}
                    ); EXTERNAL;
```

DRAWBLOCK MODE	ACTION	
0	tgt := src	{replace}
1	tgt := not (src)	{complement & overlay}
2	tgt := src XOR tgt	{eraseable overlay}
3	tgt := src OR tgt	{overlay}

NOTE1: The call interface and modes are different from the I.4 implementation of DRAWBLOCK. If you are converting programs from I.4 to I.5 either change mode parameters, or use the DRWBLK procedure provided below.

NOTE2: When using DRAWBLOCK or DRWBLK for animation the intrinsics UNITWAIT and UNITWRITE on volume #3 perform syncronization with vertical retrace of the video display (every 60th of a second). This is useful to pace the changes to the screen, maintaining uniform intensity of animated features.

DRWBLK

DRWBLK is provided for use in converting programs from I.4 to I.5. If you are beginning new development use DRAWBLOCK above as it performs the same function as DRWBLK in a more general fashion. In particular, note that DRWBLK requires that the source block be on an even (integer) boundary, while DRAWBLK is completely general. Also, the Mode parameter differs in values from the two procedures.

To convert I.4 programs to I.5 include the following external declaration for DRWBLK then change every occurance of DRAWBLOCK to DRWBLK in the program.

```
PROCEDURE DRWBLK( VAR SOURCE:SRC;  {source block}
                  VAR SCREEN:TERAKSCREEN; {target block}
                      ROWSIZE,            {always 20}
                      STARTX,             {start x for target}
                      STARTY,             {start y for target}
                      SIZEX,              {number of bits to move in x}
                      SIZEY,              {number of bits to move in y}
                      MODE : INTEGER      {see below}
                      ); EXTERNAL;
```

DRWBLK MODES	ACTION
0	tgt := tgt OR src
1	tgt := src
2	tgt := not (src)
3	tgt := tgt XOR src

GCHAR & GMARK

The following routines GMARK & GCHAR support graphics on the 8510/a by drawing characters and markers in the graphics space.

Both routines address the screen in absolute screen coordinates with (0,0) defined as the lower left corner of the screen. Note that this is a diferent addressing convention from that of DRAWLINE or DRAWBLK.

Both routines will support a picture memory height smaller, equal to, or larger than the actual display height (as controlled by the VCR zone blanking. The y dimension must, however, be a multiple of 80 (i.e 1/3 screen or the equivalent of a single screen zone. The parameter NZONE conveys the the picture memory height to the procedures.

The screen dimension in the x direction is always 0..319.

```
(0, (nzone*80)-1 )-----------------------------(319, (nzone*80)-1)
    |                                                   |
    |                                                   |
    |                                                   |
    |            ABSOLUTE                               |
    |         screen coordinates for                    |
    |                      GMARK AND GCHAR              |
    |                                                   |
    |                                                   |
  (0,0)---------------------------------------------(319,0)
```

Linestyle for both routines is 0 for white (set bits on), 1 for black (clear bits out--erase). Neither routine supports XOR or COMPLIMENT mode.

Character patterns for GCHAR are derived from an 8 dot wide by 10 dot high template, which is fetched from the 8510/a writeable character generator. The HEIGHT and WIDTH parameters to GCHAR define how many templates high and wide the target character block will be. Thus a call to GCHAR with the parmeter values h=3 and w=2 would create a character in the graphics space which is 30 dots high and 16 dots wide. The X, Y coordinates locate the lower left corner of the target block.

```
PROCEDURE GCHAR( VAR:
             SCREEN ARRAY:  POINTER TO ARRAY USED AS SCREEN,
              NZONE       :  INTEGER, {NUMBER OF ZONES TO DRAW ON}
              ORD(CHAR)   :  INTEGER, {Character to print}
              X           :  INTEGER,   {RANGE 0<=X<=319}
              Y           :  INTEGER,   {RANGE 0<=Y<=(NZONE*80-1)}
              HEIGHT      ;  INTEGER,
              WIDTH       ;  INTEGER,
              LINESYLE    :  INTEGER  );   EXTERNAL;
```

GMARK

This routine draws a 7 dot wide by 7 dot high marker, into the graphics picture memory. The pattern of the marker is controlled by the parameter MN. The marker will be centered on the screen location X,Y. If the marker would lie outside the clipping boundary defined by [XLEFT..XRIGHT] and [YBOT..YTOP] then the marker will be trimmed to fit the boundary.

The following conditions are expected to be true. Violation of these conditions will result in unpredictable results.

```
                        Ø<=X<=319
                     Ø<=Y<=NZONE*8Ø-1
                   XLEFT <= X <= XRIGHT
                    YBOT  <= Y <= YTOP
```

```
PROCEDURE GMARK(  SCREEN: ARRAY FOR SCREEN DISPLAY
               NZONE : INTEGER,  # OF 1/3 ZONES OF SCREEN
               X     : INTEGER,  X LOCATION OF MARKER
               Y     : INTEGER,  Y LOCATION OF MARKER
               MN    : INTEGER,  MARKER NUMBER O<=MN<=7
               AXL   : INTEGER,  XLEFT OF WINDOW TO CLIP MARKER
               AXR   : INTEGER,  XRIGHT OF WINDOW TO CLIP MARKER
               AYB   : INTEGER,  YBOTTOM TO CLIP MARKER
               AYT   : INTEGER,  YTOP TO CLIP MARKER
               LSTY  : INTEGER,  LINESTYLE FOR PEN: Ø IS WHITE, 1 BLACK
```

THROTTLE

This procedure provides rudimentarty time control. Control will return to the calling program when the indicated time, in ticks of the line frequency clock, has passed.

```
PROCEDURE THROTTLE(TICKS:INTEGER); EXTERNAL;
```