terak
CORPORATION

UCSD PASCAL OPERATING SYSTEM
VERSION 1.5E RELEASE NOTES

TERAK Publication Number 60-0023-001
REV 2

14405 NORTH SCOTTSDALE ROAD • SCOTTSDALE, ARIZONA 85254 • (602) 991-1580

****
********** UCSD PASCAL OPERATING SYSTEM **********
********** VERSION I.5E RELEASE NOTES **********
**********Pub no. 60-0023-001 Rev 2

The following documents and disks are contained in the Terak version of the UCSD Pascal operating system software. If any items are missing, contact the Terak Software Service Group.

DOCUMENTS:
1. UCSD PASCAL operating system manual, UCSD.
2. PASCAL USER MANUAL AND REPORT, 2nd Edition, Jensen & Wirth.
3. MICROCOMPUTER PROBLEM SOLVING USING PASCAL, Bowles.
4. A PRACTICAL INTRODUCTION TO PASCAL, Wilson & Addyman.
5. UCSD PASCAL O/S VERSION I.5 RELEASE NOTES, (this document)
6. DOCUMENTATION UPDATE: LOW LEVEL GRAPHICS INTRINSICS SECTION 2.1.4
7. 2 each 8510/a SYSTEM REFERENCE CARD, Terak # 60-0022-001.
8. Pascal Users Group All Purpose Coupon.
9. UCSD PASCAL SIGGRAPH CORE-79 GRAPHICS RELEASE GUIDE Terak # 60-0046-001.

DISKS:

1. PASCAL1: SYSTEM DISK.	{The advanced system disk}
2. PASCAL2: STUDENT SYSTEM DISK.	{The Student system disk, TURTLE graphics}
3. PASCAL3: UTILITY DISK 1.	{Utilities provided by UCSD}
4. PASCAL4: UTILITY DISK 2.	{Utilitied provided by TERAK}
5. PASCAL5: READY TO RUN DEMO PROGRAMS.	{Demonstration programs}
6. PASCAL6: DEMO SOURCES.	{Demonstration sources}

Upon receiving these disks, the manager of the software installation site should immediately:
   1) scan each disk for damaged data,
   2) verify the directory contents of each disk, and
   3) make a backup copy of each disk.
These procedures are documented in the UCSD Operating System Manual, and are summarized, for convenience, below.

Following this, the disks PASCAL4: or PASCAL5: may be run to demonstrate the system. For users just beginning with Pascal programming, a working copy of PASCAL3: should be made. The course in either BOWLES (elementary) or WILSON & ADDYMAN (intermediate) may be followed to gain knowledge in Pascal. Advanced programmers should make working disks from the contents of PASCAL1: and PASCAL2:. JENSEN & WIRTH is the language reference manual for PASCAL. The UCSD operating system manual supports all software outside of the language, and also includes documentation on extensions and exceptions to the JENSEN & WIRTH standard.

NOTE: In the following procedures, character sequences to be typed are enclosed in single quotes. The expression <RETURN> indicates that the RETURN key is to be typed.

INSPECTION OF DISKS FOR DAMAGED DATA

All disks provided in this kit may be bootstraped in the Terak 8510/a graphics computer system. To inspect a disk for damaged data, mount the selected disk into drive QX0:, and press the power switch upward. After about 7 seconds, the Operating System logo should appear. Type F to invoke the Filer, then 'B' to request a bad block scan, then '* <RETURN>' to direct the scanning command to the system disk. The disk contents will be scanned, requiring about a half minute. After the scan, the message: Ø BAD BLOCKS should appear. If the number in the message is not Ø, or the scanning does not complete, data may have been damaged on the disk. Contact the Terak Software Service Group.

INSPECTION OF DIRECTORY CONTENTS OF DISKS

All disks provided in this kit may be bootstraped in the Terak 8510/a graphics computer system. To inspect the directory contents of a disk, mount the selected disk into drive QX0:, and press the power switch upward. After about 7 seconds, the Operating System logo should appear. Type 'F' to invoke the Filer, then 'L' to request a listing, then '* <RETURN>' to direct the listing command to the system disk. The directories of each disk are provided later in this document and should be compared with the listings on the screen.

PROCEDURE TO COPY AN ENTIRE DISK

The copying (backup) of disks follows two different procedures, depending upon whether the 8510/a system has one (standard) or more than one (model 8512) disk drives.

COPYING WITH A SINGLE DISK DRIVE SYSTEM

All disks provided in this kit may be bootstraped in the Terak 8510/a graphics computer system. To copy the contents of a disk, mount the selected disk into drive QX0:, and press the power switch upward. After about 7 seconds, the Operating System logo should appear. Type F to invoke the Filer.

Remove the master (source) disk from drive QX0: and mount a blank (target) disk into QX0:. Now, type 'Z' to zero the disk. The prompt 'Zero what unit?' will be presented. Type '#4: <RETURN>'. The prompt Duplicate dir ? will be presented. Type 'N <RETURN>'. The prompt Number of Blocks? will be presented. Type 494 <RETURN> . The prompt 'New vol name ?' will be presented. Type 'X <RETURN>'. Lastly, the prompt 'X: correct ?' will be presented. Type 'Y'.

Remove the target disk from disk drive QX0: and mount the source disk into QX0:. Now, type T to request a file transfer. The prompt 'Transfer what file?' will be presented. At this time, type #4:,X: <RETURN>'. The prompt 'Put in X: type <space> to continue' will be presented. Remove the disk from drive QX0:, and mount the target disk (just zeroed) into drive QX0:. The prompt 'Possibly destroy directory of X:?', will be presented. Type 'Y'. The disk copying will now proceed, interrupted periodically by the message 'Put in XXXX: type <space> to continue , where XXXX is either X: or the name of the source disk. In response to this message, remove the present disk from drive QX0: and mount the requested disk into drive QX0:, and type <space>. When copying is completed, label the target disk, and inspect it for damaged data (see procedure above). Remove the original disk to a safe place.

COPYING WITH A DOUBLE DISK DRIVE SYSTEM

All disks provided in this kit may be bootstraped in the Terak 8510/a graphics computer system. To copy the contents of a disk, mount the selected disk into drive QX0:, and press the power switch upward. After about 7 seconds, the Operating System logo should appear. Type 'F' to invoke the Filer.

Type 'T' to request a file transfer. The prompt: 'Transfer what file? will be presented. At this time, mount a blank disk into drive QX1:. Now, type #4:,#5: <RETURN> . If the blank disk was previously zeroed using the Filer, then the prompt Possibly destroy directory of X:?, where X is the volume name, will be presented. If this occurs, type 'Y'. The disk copying will now proceed, and should require about a minute. Label the copy disk, and inspect it for damaged data (see procedure above). Remove the original disk to a safe place.

DISK DIRECTORY CONTENTS

```
PASCAL1:  27-Apr-79                    {title of system disk}
SYSTEM.FILER        32   5-Oct-78
SYSTEM.COMPILER     69  16-Nov-78
SYSTEM.LINKER       21  18-Dec-78
SYSTEM.EDITOR       45   6-Dec-78
SYSTEM.SYNTAX       14   1-Oct-78   {list of Compiler Syntax Errors}
SYSTEM.CHARSET       5   2-Feb-78
SYSTEM.PASCAL       38   3-Feb-79   {operating system kernal}
SYSTEM.8510/A       21  27-Apr-79   {interpreter}
SYSTEM.ASSMBLER     48   8-Jan-79
11.OPCODES           3  20-Dec-78
11.ERRORS            7  27-Sep-78
LIBRARY.CODE         7  17-Nov-78
LIBMAP.CODE          7  20-Jan-79
SYSTEM.LIBRARY      28  20-Nov-79
14 files, 336 blocks used, 148 unused
```

```
PASCAL2:  27-Apr-79                          {student system, Used with Bowles book}
SYSTEM.FILER       32   5-Oct-78
SYSTEM.COMPILER    69  16-Nov-78
SYSTEM.LINKER      21  18-Dec-78
SYSTEM.EDITOR      45   6-Dec-78
SYSTEM.SYNTAX      14   1-Oct-78
SYSTEM.CHARSET      5   2-Feb-78
SYSTEM.8510/A      21  27-Apr-79
SYSTEM.PASCAL      53   9-Apr-79
STRING1.TEXT        4   1-Apr-78
GRAPH1.TEXT         4   1-Apr-78
GRAPH1.CODE         2   9-Apr-79
SYSTEM.WRK.TEXT     4  27-Apr-79
SYSTEM.LIBRARY     28  20-Nov-79
13 files, 293 blocks used, 191 unused
```

```
PASCAL3:  27-Apr-79                    {UCSD system utilities}
SYSTEM.FILER       32   5-Oct-78
SYSTEM.EDITOR      45   6-Dec-78
SYSTEM.SYNTAX      14   1-Oct-78
SYSTEM.CHARSET      5   2-Feb-78
SYSTEM.PASCAL      38   3-Feb-79
SYSTEM.8510/A      21  27-Apr-79
BASIC.COMPILER     31  27-Sep-78
OPCODES.I5          5  24-Mar-78
YALOE.CODE         14  17-Nov-78   {line oriented editor}
PATCH.CODE         17  17-Nov-78   {file patch and dump utility}
COPYDUPDIR.CODE     3  28-Dec-78
MARKDUPDIR.CODE     4  28-Dec-78
L2.CODE            50  16-Jul-78   {screen oriented editor for large files}
DISASM.I5.CODE     41  28-Sep-78
RT11TOEDIT.CODE    10  23-Jan-79   {file transfer utility, RT-11 to Pascal}
GOTOXY.TEXT         4  17-Nov-78
XREF.CODE           8  23-Dec-77   {Cross reference generator for Pascal prc
CONCATTER.CODE      3  22-Feb-78   {Concattenator for TEXT files}
XREF.LISTING        2  27-Apr-79
CALC.CODE           8  26-Sep-78
BINDER.CODE         5   2-Oct-78
21 files, 360 blocks used, 124 unused
```

```
PASCAL4:  27-Apr-79
SYSTEM.FILER        32   5-Oct-78
SYSTEM.CHARSET       5   2-Feb-78
SYSTEM.PASCAL       38   3-Feb-79
SYSTEM.8510/A       21  27-Apr-79
GREDIT.TEXT         48  10-Mar-79
GREDIT.CODE         25  10-Mar-79   {Graphics Editor for FOTO files}
GREDIT.DOC.TEXT     56  10-Mar-79
RT2PAS.CODE          6  11-Mar-79
RT2PAS.TEXT         16  22-Apr-78
ANIMATE.CODE         5   3-Mar-79   {for Sequential display of FOTO files}
ANIMATE.TEXT        12  11-Apr-78
DIRECTORY.TEXT      14  12-Mar-79
DIRECTORY.CODE       5  12-Mar-79   {Generates/accesses Directories of Disks}
SERIAL.IC.TEXT      20  11-Nov-78   {Procedures to drive Serial Interface}
TERMINAL.TEXT       36  30-Mar-79   {terminal emulation program}
TERMINAL.CODE       10  30-Mar-79
CNTRL.DEMO.CODE      3  30-Mar-79   {Cursor addressing and Control Demo}
CNTRL.DEMO.TEXT      4  30-Mar-79
PRINTOUT.CODE        8  11-Apr-79   {prints TEXT files to Printer or...
PRINTOUT.TEXT       30  11-Apr-79   {... Serial Unit, with formating}
FATCAT.FOTO         19   7-Feb-78
GEOMETRY.FOTO       19  15-Apr-78   {foto files for GREDIT and ANIMATE}
TERAK.FOTO          19  28-Feb-78
IRWIN.FOTO          19  13-Apr-78
24 files, 470 blocks used, 14 unused, 14 in largest area
```

```
PASCAL5: 5-DEC-79
SYSTEM.FILER          32   5-Oct-78    10   512  Codefile
SYSTEM.CHARSET         5   2-Feb-78    42   512  Datafile
SYSTEM.PASCAL         38   3-Feb-79    47   512  Codefile
SYSTEM.8510/A         21  27-Apr-79    85   512  Datafile
SPHERE2.FOTO          22  15-Jul-78   106   488  Fotofile
MASK.FOTO              1   2-Jul-78   128   240  Fotofile
INVERSE.FOTO           1   2-Jul-78   129   240  Fotofile
LETTERS.CODE           4  12-Apr-79   130   512  Codefile
RATMAZE.CODE           6  25-Apr-79   134   512  Codefile
DERIVATIVE.CODE        3  25-Apr-79   140   512  Codefile
SURFACE.CODE           4  25-Apr-79   143   512  Codefile
WWII.CODE              7  25-Apr-79   147   512  Codefile
PATT1.CODE             3  25-Apr-79   154   512  Codefile
PATT2.CODE             3  25-Apr-79   157   512  Codefile
CYCLE.CODE             9  25-Apr-79   160   512  Codefile
LIFEDEMO.CODE          6  25-Apr-79   169   512  Codefile
CYCLE.PROMPT           6  14-Apr-78   175   512  Textfile
MATH.CHARSET           5  25-Sep-77   181   512  Datafile
APL.CHARSET            4   8-Oct-77   186   512  Infofile
OCRB.CHARSET           5  10-Nov-77   190   512  Datafile
SMUDGE.CHARSET         5   8-Oct-77   195   512  Datafile
LARGE.CHARSET          5   8-Oct-77   200   512  Datafile
INVERSE.CHARSET        5   2-Nov-77   205   512  Datafile
RUSSIAN.CHARSET        5   8-Oct-77   210   512  Datafile
FORMS.CHARSET          5  25-Sep-77   215   512  Datafile
HEBREW.CHARSET         5  11-Nov-77   220   512  Datafile
CURSIVE.CHARSET        5  17-May-78   225   512  Infofile
NEWSET.CHARSET         5  26-Feb-79   230   512  Datafile
CHEDIT.PROMPT         19   8-Oct-77   235   512  Datafile
CHEDIT.CODE            7   2-Feb-78   254   512  Codefile
BREAKOUT.CODE         13  11-Apr-79   261   512  Codefile
SYSTEM.LIBRARY        28  20-Nov-79   274   512  Datafile
PLANETS.CODE           4  30-Apr-79   302   512  Codefile
CHDEMO.CODE            6  27-Apr-79   306   512  Codefile
CHDEMO.PROMPT          4  10-Mar-79   312   512  Textfile
WESTERN.CHARSET        5  26-Feb-79   316   512  Datafile
AG1.CHARSET            4  12-Mar-79   321   512  Datafile
PANSINE.CODE           3  12-May-79   325   512  Codefile
OIL.CODE              12   5-Dec-79   328   512  Codefile
FIG1D2.CODE           11   5-Dec-79   340   512  Codefile
< UNUSED >           143                351
40 files,341 blocks used, 143 unused, 143 in largest area
```

```
PASCAL6: 5-DEC-79
SYSTEM.FILER          32   5-Oct-78    10   512  Codefile
SYSTEM.CHARSET         5   2-Feb-78    42   512  Datafile
SYSTEM.PASCAL         38   3-Feb-79    47   512  Codefile
SYSTEM.8510/A         21  27-Apr-79    85   512  Datafile
LETTERS.TEXT          12  12-Apr-79   106   512  Datafile
RATMAZE.TEXT          12  25-Apr-79   118   512  Datafile
DERIVATIVE.TEXT        4  25-Apr-79   130   512  Datafile
SURFACE.TEXT           8  25-Apr-79   134   512  Datafile
WWII.TEXT             16  25-Apr-79   142   512  Datafile
PATT1.TEXT             4  25-Apr-79   158   512  Datafile
PATT2.TEXT             6  25-Apr-79   162   512  Datafile
LIFEDEMO.TEXT         12  25-Apr-79   168   512  Datafile
CYCLE.TEXT            28  25-Apr-79   180   512  Datafile
BREAKOUT.TEXT         34  11-Apr-79   208   512  Datafile
PLANETS.TEXT           6  30-Apr-79   242   512  Textfile
CHDEMO.TEXT           18  27-Apr-79   248   512  Textfile
PANSINE.TEXT           4  15-Feb-79   266   512  Textfile
FIG1D2.TEXT            8   5-Dec-79   270   512  Textfile
OIL.TEXT               8   5-Dec-79   278   512  Textfile
< UNUSED >           208              286
19 Files, 276 blocks used, 208 unused, 208 in largest area
```

```
PASCALT:
SYSTEM.FILER          32   5-Oct-78     10    512  Codefile
SYSTEM.COMPILER       69  16-Nov-78     42    512  Codefile
SYSTEM.PASCAL         38  17-Nov-78    111    512  Datafile
SYSTEM.LIBRARY        14   7-Nov-78    149    512  Datafile
SYSTEM.EDITOR         45   6-Dec-78    163    512  Codefile
SYSTEM.PDP-11         21  14-Dec-78    208    512  Datafile
SYSTEM.CHARSET         5   2-Feb-78    229    512  Datafile
SYSTEM.LINKER         21   6-Dec-78    234    512  Codefile
SYSTEM.SYNTAX         14   1-Oct-78    255    512  Textfile
LIBRARY.CODE           7  17-Nov-78    269    512  Codefile
LIBMAP.CODE            7  18-Oct-78    276    512  Codefile
RT11TOEDIT.CODE        8  26-Jun-78    283    512  Codefile
BOOTER.CODE            3  27-Sep-78    291    512  Codefile
CALC.CODE              8  26-Sep-78    294    512  Codefile
SETUP.CODE            24  19-Sep-78    302    512  Codefile
BASIC.COMPILER        31  27-Sep-78    326    512  Codefile
BINDER.CODE            5   2-Oct-78    357    512  Codefile
OPCODES.I5             5  24-Mar-78    362     96  Datafile
DISASM.CODE           41  11-Oct-78    367    512  Codefile
YALOE.CODE            14  17-Nov-78    408    512  Codefile
QXBOOT                 2  20-Sep-78    422    512  Datafile
11.OPCODES             3  17-Sep-78    424    188  Datafile
11.ERRORS              7  27-Sep-78    427    498  Datafile
SAMPLEGOTO.TEXT        4  17-Nov-78    434    512  Textfile
PATCH.CODE            17  17-Nov-78    438    512  Codefile
< UNUSED >            39               455
25/25 files<listed/in-dir>, 445 blocks used, 39 unused, 39 in largest area
```

PASCALT: contains most of the code-files you will need to run. Other code files are scattered among the other disks. Specifically the code file for the assembler. It will be called ASM11.CODE. You must change the name of this file to SYSTEM.ASSEMBLER and put it on the system disk on which you wish to assemble.

```
11STUFF:
RT-11.DIR              8  30-Nov-78     10    512  Datafile
RT-11.FILES          340  14-Dec-78     18    512  Datafile
ASM11.CODE            46  14-Nov-78    358    512  Codefile
ASM11.TEXT            30  14-Nov-78    404    512  Textfile
CALC.TEXT             22  28-Oct-78    434    512  Textfile
BOOTER.TEXT            6  17-Nov-78    456    512  Textfile
READ.ME.TEXT           8  14-Dec-78    462    512  Textfile
RT11TOEDIT.TEXT       22  26-Jun-78    470    512  Textfile
< UNUSED >             2               492
8/8 files<listed/in-dir>, 482 blocks used, 2 unused, 2 in largest area
```

11STUFF: contains the RT-11 files necessary to assemble an interpreter, using READ.ME.TEXT as the instructions. This disk also contains various utility programs which are explained in the documentation.

```
COMP:
COMPGLBLS.TEXT     26  16-Nov-78     10    512  Textfile
COMPINIT.TEXT      26  16-Nov-78     36    512  Textfile
DECPART.A.TEXT     32  11-Sep-78     62    512  Textfile
DECPART.B.TEXT     20  25-Oct-78     94    512  Textfile
DECPART.C.TEXT     22  16-Nov-78    114    512  Textfile
BODYPART.A.TEXT    26  21-Sep-78    136    512  Textfile
BODYPART.B.TEXT    22  27-Sep-78    162    512  Textfile
BODYPART.C.TEXT    26  30-Sep-78    184    512  Textfile
BODYPART.D.TEXT    28   4-Sep-78    210    512  Textfile
BODYPART.E.TEXT    34  27-Sep-78    238    512  Textfile
UNITPART.TEXT      22  16-Nov-78    272    512  Textfile
PROCS.A.TEXT       30  28-Sep-78    294    512  Textfile
PROCS.B.TEXT       18  11-Sep-78    324    512  Textfile
BLOCK.TEXT         12  16-Nov-78    342    512  Textfile
COMPILER.TEXT       4  16-Nov-78    354    512  Textfile
L2.CODE            50  16-Jul-78    358    512  Codefile
BINDER.TEXT        14   2-Oct-78    408    512  Textfile
SMALL.PASCAL       42  18-Jul-78    422    512  Datafile
RADIX.TEXT         24  13-Sep-78    464    512  Textfile
< UNUSED >          6               488
19/19 files<listed/in-dir>, 478 blocks used, 6 unused, 6 in largest area
```

COMP contains the source for the pascal compiler and the GOTOXY binder. It also contains a small system.pascal which has many things stripped from it; when you need a stripped system, this is it. Herein is also contained the code file for the L2 large file editor.

```
SYS1:
LINK3B.TEXT        30  22-Sep-78      6    512  Textfile
LINK.TEXT           4  16-Nov-78     36    512  Textfile
LINK2.TEXT         20  17-Nov-78     40    512  Textfile
LINK3A.TEXT        34  15-Sep-78     60    512  Textfile
LINK0.TEXT         28  16-Nov-78     94    512  Textfile
LINK1.TEXT         18  14-Sep-78    122    512  Textfile
SYSSEGS.TEXT       38  29-Nov-78    140    512  Textfile
SYSTEM.TEXT         4  27-Sep-78    178    512  Textfile
SYSTEM.B.TEXT      44  26-Oct-78    182    512  Textfile
SYSTEM.C.TEXT      34  26-Oct-78    226    512  Textfile
PATCH1.TEXT        26   6-Jun-78    260    512  Textfile
PATCH2.TEXT        20   6-Jun-78    286    512  Textfile
FILER.TEXT          4   5-Oct-78    306    512  Textfile
FILER.A.TEXT       22   5-Oct-78    310    512  Textfile
FILER.B.TEXT       36   5-Oct-78    332    512  Textfile
FILER.C.TEXT       38   5-Oct-78    368    512  Textfile
FILER.D.TEXT       36   5-Oct-78    406    512  Textfile
LIBRARY.TEXT       26  16-Nov-78    442    512  Textfile
GLOBALS.TEXT       26  29-Nov-78    468    512  Textfile
19/19 files<listed/in-dir>, 488 blocks used, 0 unused, 0 in largest area
```

SYS1 contains the sources for the system linker, the filer, and the main portion of the operating system. It also contains sources for the patch/dump program, and the librarian. The system globals are on this disk as GLOBALS.TEXT.

```
SYS2:
SETUP.TEXT          58  18-Sep-78     10    512  Datafile
SGLOBALS.TEXT       24  24-Aug-78     68    512  Textfile
DISASM1.TEXT        22  28-Sep-78     92    512  Textfile
DISASM2.TEXT        28  28-Sep-78    114    512  Textfile
DISASM.TEXT         34  11-Oct-78    142    512  Textfile
OPCODES.I5           5  24-Mar-78    176     96  Datafile
ASM1.TEXT           26  14-Nov-78    181    512  Textfile
ASM2.TEXT           30  14-Nov-78    207    512  Textfile
ASM3.TEXT           26  14-Nov-78    237    512  Textfile
ASM4.TEXT           38  14-Nov-78    263    512  Textfile
ASM5.TEXT           28  14-Nov-78    301    512  Textfile
ASM6.TEXT           26  14-Nov-78    329    512  Textfile
ASMZ80.TEXT         38  14-Nov-78    355    512  Textfile
LIBMAP.TEXT         26  18-Oct-78    393    512  Textfile
PASCALIO.TEXT       18  26-Oct-78    419    512  Textfile
< UNUSED >          57               437
15/15 files<listed/in-dir>, 427 blocks used, 57 unused, 57 in largest area
```

SYS2: contains the source for the core portion of the assembler and source (and opcode file) for the disassembler. The opcode file must be on the system disk when running the disassembler.

The source for the BASIC compiler and for all three text editors are not being distributed. These source files are available for a modest extra distribution fee.

Thank you for your patience in waiting for I.5 to arrive.

Keith Allan Shillington

terak
CORPORATION

LOW LEVEL GRAPHICS INTRINSICS

TERAK Publication Number 60-0024-I01

REV 1

14405 NORTH SCOTTSDALE ROAD • SCOTTSDALE, ARIZONA 85254 • (602) 991-1580

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

(REPLACEMENT PAGES)

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

REV 1

INTRODUCTORY INFORMATION

The Terak 8510/a supports bit mapped, raster scan graphics, refreshed directly from main memory. The display presented is a composite of the 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 (GRAPHIC:). 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]);
        5:( BLKS:ARRAY[0..18] OF ARRAY[0..255] OF INTEGER)
    END;(*CASE*)

VAR SCREEN :TERAKSCREEN;
```

These allocate one picture-full of memory to 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'); PIXOK:=BLOCKREAD(PIX,SCREEN,19)=19;
```

(which loads a binary file into the picture), 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 intrinsic graphic procedures.

The graphic procedures supplied with the Terak release of the UCSD Pascal system are documented here. 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. This can be accomplished, respectively, by either of 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 = <starting address of picture memory>,
and VCRVAL = <integer zone blanking variable>

The GARVAL parameter locates 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. VCR values from 0 to 63 cover all combinations of graphics and character zone blanking. Addresses loaded into the GAR must always be on even (integer) boundaries, and may be indexed from the array base. 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.

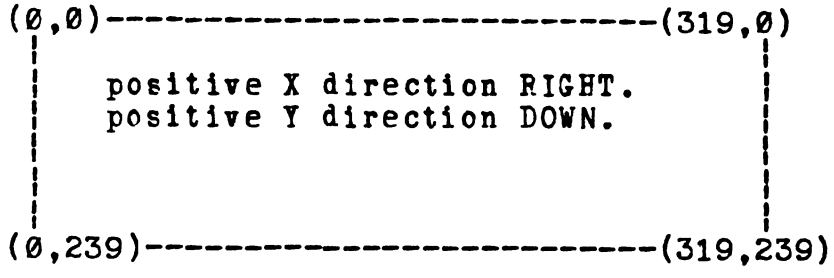
GRAPHICS PROCEDURE CALLS

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 (0,0) in the upper left portion of the display. X and Y locations of the screen should be addressed using the following scheme.



DRAWLINE

This procedure draws lines in one of five modes, into memory. 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	
0	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

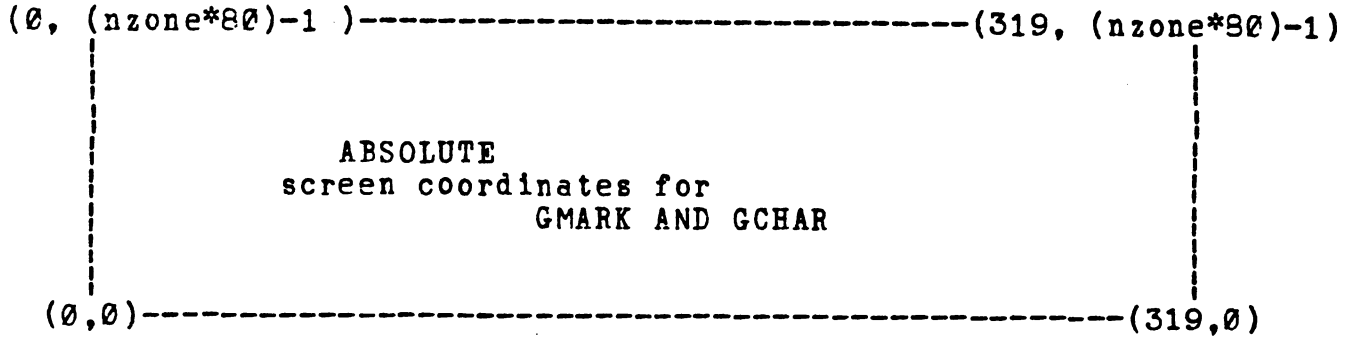
GCHAR & GMARK

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.



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.

```
                    0<=X<=319
                 0<=Y<=NZONE*80-1
              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 0<=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: 0 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;
```