# TERAK/UCSD PASCAL VERSION II.0 OPERATING SYSTEM

SOFTWARE RELEASE GUIDE

for the

TERAK GRAPHICS COMPUTER SYSTEMS

60-0133-001A

TERAK CORPORATION
SCOTTSDALE, ARIZONA

# TERAK/UCSD PASCAL VERSION II.0 OPERATING SYSTEM

SOFTWARE RELEASE GUIDE

for the

TERAK GRAPHICS COMPUTER SYSTEM

60-0133-001A

TABLE OF CONTENTS

RESERVED

# SECTION 1

# Introduction

## 1.1 Intended Audience

This document presumes familiarity with the Pascal language. Users unfamiliar with Pascal should review the Pascal User Manual and Report (Jensen & Wirth) or the Introduction to Pascal (Wilson & Addyman) for a discussion of the Standard Pascal language and the UCSD Pascal Users' Manual for a discussion of the UCSD implementation of Pascal.

## 1.2 Document Structure

This document contains the following sections:

## 1.3 Textual Conventions

| | |
|---|---|
| &lt;ESC&gt; | Indicates a key to be pressed; in this example, the ESC (escape) key. |
| CHDEMO | Capitalization indicates a file or program |
| &lt;CTRL&gt;3 | Indicates the Control key is to be pressed with the following key (in this case "3") pressed immediately after. |

## 1.4 New Release Information

The release of the Terak/UCSD Pascal Version II.0 differs substantially from previous Terak releases. Most importantly, all three graphics systems, GDC, monochrome and color are supported by one software distribution kit. There is only one minimum configuration QB bootable disk supplied from which the user can build a system-specific work disk by adding or changing files from other disks in the distribution kit as needed.

Also, software support to incorporate Winchester disk mass storage is included, along with software needed to add a Terak Touchpanel to the color system monitor.

Hardware device drivers are no longer linked to the interpreter file; instead, they exist as separate files and are loaded at boot time.

New interpreters that are smaller than the standard interpreters have been added to enable larger application programs to be executed.

Gemini is now loaded at boot time, which eliminates the need for a SYSTEM.STARTUP on 8600 color systems. LOAD86, a program to load Gemini, has been added.

LINE24 no longer loads a character set; instead, it selects font 1 to be the console font.

The user of a previous release of UCSD Pascal Version II.0 may notice the absence of STUDENT.SYSTEM, Turtle Graphics, Data Media Emulator and QX-related files in this release. Terak no longer supports these applications.

# SECTION 2

# PRODUCT OVERVIEW

## 2.1 Product Description

The Terak/UCSD Pascal Version II.0 Operating System is a single-user system intended for use on the Terak 8510 or 8600 Graphics Computer System. A variety of system and program development utilities support the basic operating system and Pascal compiler. The Terak/UCSD Pascal operating system uses the Terak monochrome graphics terminal as the console device and floppy disks for mass storage in the 8510 system. The graphics display is used as the console in the 8600 and a Winchester disk drive can be added to increase mass storage in either an 8510 or 8600 system.

The operating system consists of a pseudo-code interpreter with a set of device drivers. It also includes a file manager and utilities for data manipulation and program creation.

The interface between the system hardware, the system software and the user is provided by the operating system. It maintains continuity, reviews the status of input/output devices and provides automatic accounting of system status. In addition, the operating system accepts, processes and executes instructions entered through the keyboard.

UCSD Pascal differs from the standard Pascal in significant ways, specifically in the areas of file and I/O handling and extended language features. The <u>UCSD Users' Manual</u> provides discussions of the major differences between the UCSD Pascal and standard Pascal.

The device drivers for the system hardware devices are an integral part of the Pascal operating system. They provide the interface between each of the peripheral devices and the operating system and are generally invisible to the user. They will be of concern only if a peripheral device not supported by the standard system software is added to the user's system. Such support can be provided directly from the user's program without the need for a driver if performance is not critical. Otherwise, a new driver may be written or one of the existing drivers modified to accommodate the new device.

The device drivers included in this software release are

- o A Console Terminal Emulator for 8510/a and 8510/b systems.

- A Console Terminal Emulator for 8510/c systems.
- A Gemini loader for 8600 color systems.
- A floppy disk driver for 8510/a.
- A floppy disk driver for 8510/b and 8510/c.
- A Winchester disk driver for 8518.
- A line printer driver.
- A serial port input driver.
- A serial port output driver.
- A virtual disk (extended memory) driver.

The utilities provided with this system provide several functions. Some provide text editing and file mangement; others link or compile programs; still others allow formatting of blank or degaussed diskettes. The major utilities provided are as follows.

- An editor which allows the user to create and modify text files.
- A file maintenance utility which can be used to update and delete files, transfer files between devices, and perform other tasks required to manipulate and maintain disks.
- A librarian used for storing and cataloging often used subprograms or modules.
- A linking program used to collect various program modules and create a composite program.
- A patch/dump program used to patch specific disk blocks and dump files to the console or line printer. Data can be transmitted in ASCII and Hexadecimal characters or in Octal or decimal words or bytes.
- A Terak utility, FORMAT, which allows formatting of blank or degaussed diskettes.
- A Pascal compiler used to translate text files into executable P-code format.

## 2.2 Volume Assignment

The Terak/UCSD Pascal II.0 Software System allows 12 slots for I/O devices such as printers, keyboard and disk or diskette drives. The 12 I/O slots are allocated as the system boots up and configures itself, based on the driver available on the disk and the hardware configuration. See the tables that follow for the allocation of I/O slots for the system configuration.

The UCSD Pascal Users' Guide, FILEHANDLER, contains a detailed discussion on the requirements for assigning Volume names.

| DRIVER NAME | UNIT NUMBER | VOLUME ID | DESCRIPTION |
|---|---|---|---|
| (interpreter) | 1 | CONSOLE: | Screen and keyboard with echo |
| (interpreter) | 2 | SYSTERM: | Keyboard without echo |
| TKEML | 3 | GRAPHICS: | The graphics aspect of the 8510/a or 8510/b monochrome screen. Also provides console emulator |
| HTKEML | None | None | Provides console emulator for 8510/c high resolution graphics system |
| LOAD86 | None | None | Load Gemini in 8600 color system |
| QBA | 4, 5, 9, & 10 | <volume name>: | QB disk drive for 8510/a |
| QBB | 4, 5, 9, & 10 | <volume name>: | QB disk drive for 8510/b or 8510/c |
| PRINTER | 6 | PRINTER: | The line printer on serial Unit 1 |

| | | | |
|---|---|---|---|
| REMIN | 7 | REMIN: | The remote input on serial unit 2 |
| REMOUT | 8 | REMOUT: | The remote output on Serial unit 2 |
| DW | 9, 10, 11, & 12 | <volume name>: | Winchester logical disk slots |
| Memory | 11 | <volume name>: | Virtual disk using extended memory on an LSI 11/23 |

## 2.3 Console Emulator

The console display is provided through a software console display emulator. Software controls all console functions including motions of the cursor, character echo and scrolling of the screen. The emulator code is resident within one of the driver files or Gemini. Once the driver or Gemini is loaded from the user's working diskette, the display characteristics of the emulator resident within these files will automatically provide console display control.

A detailed discussion of the characteristics of the console display emulators and a listing of the control codes and escape sequences for each of the console display emulators provided can be found in the Appendices of this document.

The Terak (TK) emulator is the standard Terak console display emulator for the Terak/UCSD Pascal II.0 software system. Control codes and escape sequences are recognized to support user control of the screen such as cursor addressing, panning, click, invisible cursor and single axis cursor addressing. The TK emulator provides several functions not supported by the Data Media (DM) emulator and requires approximately 644 additional bytes of memory space. Terak no longer supports the Data Media emulator.

### 2.3.1 System Console Emulator

The emulator now supported under Version II.0 is the TK console emulator. Programs utilizing functions of the DM console emulator must be modified.

The TK emulator varies slightly between the monochrome, 8600 and GDC systems. The control code and escape sequences for each system are listed in Appendix C1.2 of this document.

## 2.4 UCSD Pascal Commands

The standard UCSD Pascal keystroke commands (as discussed in the UCSD Pascal Users' Manual shipped with the Software Kit) are implemented differently in some cases. Use the list below as a reference to translate functions.

| To Get This Function | Type Key(s) |
|---|---|
| ESC | <ESC> |
| DEL | <DEL> |
| RET | <RET> |
| EOF | <ETX> |
| BS | <BS> |
| ETX | <ETX> (Editor accept) |
| BREAK | <CTRL><NULL> (on numeric keypad) |
| DOWN ARROW | <DOWN ARROW> |
| UP ARROW | <UP ARROW> |
| LEFT ARROW | <LEFT ARROW> |
| RIGHT ARROW | <RIGHT ARROW> |
| LF | <LINE FEED> |
| TAB | <TAB> |
| Alternate Character Set Toggle | <DC1> |
| ALPHA LOCK Toggle | <DC2> |
| STOP OUTPUT Toggle | <DC3> or <CTRL><S> |
| FLUSH ALL OUTPUT Toggle | <CTRL><F> |

## 2.5 Distribution Kit Descriptions

### 2.5.1 Software Configuration

The Terak/UCSD Operating System Software is shipped on single-sided, single density disks. Each kit contains Terak/UCSD Pascal user documentation and information about the Terak supporting utilities.

Only the first disk in the Software Kit contains a complete bootable system configuration. A user-suitable working Pascal disk should be configured by each individual user from the contents of the entire Software Kit to suit the needs of that user. Descriptions of the files needed on a working system disk are presented in Section 3.

## 2.5.2 Media Kit Disk Descriptions

| NAME | CONTENTS |
|---|---|
| PSYS1: | Operating System<br>The only bootable disk in the media kit, this disk boots on all Terak QB systems. However, an 8600 system will boot as a very simple 24-line alphanumeric terminal. |
| PSYS2: | Additional system files<br>Alternate interpreting drivers and graphics libraries. Boot files. |
| PSYS3: | Utilities (mostly from UCSD Softech) |
| PSYS4: | Utilities (mostly from Terak) |
| PSYS5: | Character sets and loaders |
| PSYS6: | Monochrome demos |
| PSYS7: | Color demos |

## SECTION 3

## INSTALLATION PROCEDURES

### 3.1 Pre-Installation Considerations

Before installing the software, gather these materials:

- o the Terak/UCSD Pascal Version II.0 Software Kit (includes a Pascal System Disk, UCSD Utilities, Terak Utilities, Character Sets and Demonstrations).
- o a sufficient quantity of single-sided, single density blank disks to back up the distribution disks.

The disks should be examined for physical damage. Remove each disk from its protective cover and check for punctures, deep scratches or any other major flaws.

### 3.2 Initiating Disk Installation

1. Turn on the power switch for the 8510 (and 8515 if present).
2. Insert Disk 1, the Pascal System Disk, into unit 0.
3. Close the drive door. The red LED will light up and remain on while the disk is being accessed.
4. After the disk has been successfully booted, check to ensure the following message appears on the Terak monitor screen:

```
COMMAND: E(DIT,R(UN,F(ILE,C(OMP,L(INK,X(ECUTE,A(SSEM,D(EBUG?[II.O]

                 Welcome PSYS1, to
                 U.C.S.D. Pascal System II.0
                 Current Date is xx-xxx-xx
```

The command menu is displayed across the top of the screen with the operating system logo in the center.

### 3.3 Magnetic Validity Testing

After the above screen display has appeared, the disks should then be tested for magnetic validity by using the appropriate bad block scanning techniques on the following page.

For the System Disk:

| Step | User Input | System Response (screen display) |
|---|---|---|
| 1. | F | Filer command line at top of screen |
| 2. | B | Bad block scan of what vol ? |
| 3. | #4:<CR> | Scan for 494 blocks (Y/N) ? |
| 4. | Y | 0 bad blocks (if no damage) |
| 5. | E | Dir listing of what vol ? |
| 6. | #4:<CR> | The disk directory appears |

For All Other Distribution Disks:

| Step | User Input | System Response (screen display) |
|---|---|---|
| 1. | Insert distribution disk into drive unit 1. Close door. | |
| 2. | B | Bad block scan of what vol ? |
| 3. | #5:<CR> | Scan for 494 blocks ? (Y/N) |
| 4. | Y | 0 bad blocks (if no damage) |
| 5. | E | Dir listing of what vol ? |
| 6. | #5:<CR> | The disk directory appears |
| 7. | Remove disk. Repeat steps 1-7 for remaining disks in Kit. | |

If any physical or magnetic damage is detected, contact Terak Customer Service.

### 3.3.1 Magnetic Examination - Single Drive Systems

Turn the system power ON. Insert the System Diskette into disk drive 0 and close the drive door. If the bootstrap process does not automatically initiate, depress the BOOTSTRAP switch. After the operating system logo appears, proceed as follows.

| Step | User Input | System Response (screen display) |
|---|---|---|
| 1. | F | Command level prompt line appears |
| 2. | B | Bad block scan of what vol ? |
| 3. | #4:<CR> | Scan for 494 blocks ? (Y/N) |
| 4. | Y | 0 bad blocks (see note below) |
| 5. | L | Dir listing of what vol ? |
| 6. | #4:<CR> | The disk directory appears for the diskette in drive 0 |
| 7. | Remove disk from drive 0. Repeat steps 2-6 each of the remaining disks. | |

NOTE: If you receive a message listing bad blocks or if a diskette's directory does not display properly, contact Terak Customer Service.

## 3.4 Distribution Disks - Backup Procedure

When disk examination is complete, backup all distribution disks on formatted single density diskettes using the following procedure:

| Step | User Input | System Response (screen display) |
|---|---|---|
| 1. | F | (to invoke the Filer) |
| 2. | T | Transfer what files? |
| 3. | #4:,#5: | Transfer 494 blocks ? (Y/N) |
| 4. | Y | |

This will copy the entire contents of the diskette in drive 0 to the diskette in drive 1 including any bootstrap information.

## 3.5 Building a Working System Disk

To build a working system disk, follow these steps.

1. Examine the Terak Media Kit for defects.
2. Copy the Terak Media Kit.
   a. Format 7 blank diskettes (single-sided, single density).
   b. Copy - volume to volume transfer (including boot information): T(ransfer #4:, #5:.
   c. Store Terak Media Kit in safe place.
3. Build User-Specific Work Disk.
   a. Format a blank diskette; dual density media may be preferred, especially with an 8600 system.
   b. Zero the directory of the newly formatted blank diskette before transferring needed files.
   c. Copy necessary files. For an 8600 color system, rename SYSTEM.COLORFPU or SYSTEM.COLORFIS to SYSTEM COLORGRA; for the 8510/c GDC system, rename SYSTEM.GDCFPU or SYSTEM.GDCFIS to SYSTEM.GDCLIB.
   d. Copy bootstrap.

See the suggested minimum work disk configuration in Section 3.8.

## 3.6 Formatting Disks (Single, Dual and Quad Densities)

The Terak Utility FORMAT is used to format 8-inch 3740 Series floppy disks. The formatting procedure creates a data shell or "super structure" which is used to locate physical disk sectors. Formatting also initializes the media attribute identifier area of the disk. The program has the ability to format the three density attributes supported by Terak.

Note that double or quad density formatting requires disks certified for that specific recording density. Media originally formatted in single density may be unreliable if re-formatted in double density.

The standard number of logical blocks for the different density floppies after formatting and zeroing are as follows:

| Single | Double | Quad |
|---|---|---|
| 494 | 1140 | 2280 |

### 3.6.1 Formatting Procedure

When executed, the FORMAT program will request a media attribute selection by the following prompt:

```
0 = Single-sided, Single-density (494 Blocks)
1 = Single-sided, Double-density (1140) Blocks)
2 = Double-sided, Double-density (2280) Blocks)
S = Stop
?
```

Type the number corresponding to the media attributes desired. The program will then prompt for the drive unit for placement of the blank diskette.

```
Drive Number (0,1,2 or 3)
?
```

Type the number of the disk drive to be used (i.e., 0, 1, 2 or 3). The number appears on a label on the drive. The program will then prompt

```
PLACE DISK IN DRIVE # - TYPE ANY KEY WHEN READY
```

(# is the number of the drive unit entered)

Place the disk in the appropriate drive and close the door. Type any key. The program will display the message "Cylinder (x10): 0" and update with 1-7 as the formatting proceeds. When completed, the program will respond

```
FORMAT COMPLETED - NO ERRORS DETECTED
```

followed by

```
0 = Single-sided, Single-density (494 Blocks)
1 = Single-sided, Double-density (1140) Blocks)
2 = Double-sided, Double-density (2280) Blocks)
S = Stop
?
```

At this point, the format procedure may be repeated for another disk. If all disks have been formatted, type "S" to return to the operating system. The system will then respond

```
?REBOOT ADVISED?
```

After formatting or reformatting one or more diskettes, it is recommended that the system be rebooted to ensure proper determination of all disk attributes that might have been changed during the Format process.

#### 3.6.1.1 Bad Block Scan

To ensure that the disk is valid and ready to use, use the bad block scanning technique included in the magnetic validation procedure.

#### 3.6.1.2 Error Messages

There are several error messages that can occur during the formatting process. All messages are preceded by "FORMAT ABORTED BECAUSE", followed by the reason identified by the FORMAT program. The following table suggests remedies for these situations.

| MESSAGE | CHECK |
|---|---|
| DRIVE NOT READY | 1. Power on?<br>2. Cables Connected?<br>3. Disk inserted?<br>4. Door Closed? |
| WRITE PROTECT ON | Is a tab over the write protect notch of the disk? |
| SINGLE SIDED DISK | Attempt was made to format a single-sided disk as a dual-sided disk or the drive being used is single-headed. |
| WRITE FAILURE | Reattempt formatting. |
| FORMAT FAILURE | May indicate a faulty disk; reattempt formatting. |
| CONTROLLER BOARD FAULT<br>EXECUTE PHASE SYNC FAULT<br>RESULT PHASE SYNC FAULT | Reboot and reattempt formatting. If error persists, a controller board failure may be indicated. |
| SEEK FAILURE<br>RECALIBRATION FAILURE | Reboot and reattempt formatting. If error persists, a failure in the drive may be indicated. |

## 3.7 Installing a Directory (Z(eroing)

After formatting, use the following steps to install a directory on the work diskette.

<u>USER INPUT</u>

F (to invoke the Filer)
Z (to make a directory or zero an existing directory

| SCREEN DISPLAY | COMMENTS |
|---|---|
| Zero dir of what vol? | See Volume assignments for proper selection. |
| Duplicate dir? | Type Y or N. See the <u>UCSD Users' Manual</u> for duplicate directions. |

| SCREEN DISPLAY | COMMENTS |
|---|---|
| # of blocks on the disk? | Select appropriate number based on formatting. |
| New vol name? | Assign name - see the UCSD Users' Manual for naming conventions. |

The user can now transfer individual files as needed to the work disk.

## 3.8 Suggested Minimum Work Disk Configuration

| MONOCHROME<br>8510/a or 8510/b | HIGH RESOLUTION<br>8510/c | COLOR (8600) |
|---|---|---|
| SYSTEM.11.2(#)* | SYSTEM.11.2(#)* | SYSTEM.11.2(#)* |
| SYSTEM.11.23(#)* | SYSTEM.11.23(#)* | SYSTEM.11.23(#)* |
| SYSTEM.PASCAL(#) | SYSTEM.PASCAL(#) | SYSTEM.PASCAL(#) |
| SYSTEM.CONFIG(#) | SYSTEM.CONFIG(#) | SYSTEM.CONFIG(#) |
| DRIVER.QBX**(#) | DRIVER.QBX**(#) | DRIVER.QBX**(#) |
| DRIVER.TKEML(#) | DRIVER.HTKEML(#) | DRIVER.LOAD86(#) |
| DRIVER.PRINTER | DRIVER.PRINTER | DRIVER.PRINTER |
| SYSTEM.CHARSET(#) | SYSTEM.CHARSET(#) | SYSTEM.MISCINFO(#) |
| | | GEMINI.M86(#) |
| MONOCHR.LIBRARY*** | GDC.LIBRARY*** | COLOR.LIBRARY*** |
| | SYSTEM.GDCLIB | SYSTEM.COLORGRA |
| SYSTEM.ASSMBLER | SYSTEM.ASSMBLER | SYSTEM.ASSMBLER |
| SYSTEM.COMPILER | SYSTEM.COMPILER | SYSTEM.COMPILER |
| SYSTEM.EDITOR | SYSTEM.EDITOR | SYSTEM.EDITOR |

| MONOCHROME 8510/a or 8510/b | HIGH RESOLUTION 8510/c | COLOR (8600) |
|---|---|---|
| SYSTEM.FILER | SYSTEM.FILER | SYSTEM.FILER |
| SYSTEM.LINKER | SYSTEM.LINKER | SYSTEM.LINKER |
| SYSTEM.SYNTAX | SYSTEM.SYNTAX | SYSTEM.SYNTAX |
| 11.OPCODE | 11.OPCODE | 11.OPCODE |
| 11.ERRORS | 11.ERRORS | 11.ERRORS |
| LIBMAP.CODE | LIBMAP.CODE | LIBMAP.CODE |
| LIBRARY.CODE | LIBRARY.CODE | LIBRARY.CODE |
| FORMAT.CODE | FORMAT.CODE | FORMAT.CODE |

* 11/2 file is required for LSI 11/2 processor; 11/23 file is required for LSI 11/23 processor. Having both will enable the disk to be used on either processor.

** You should use DRIVER.QBA for 8510/a systems and DRIVER.QBB for 8510/b or 8510/c systems.

*** Can be renamed to SYSTEM.LIBRARY for convenience.

(#) Essential for the disk to boot.

## 3.9 File Transfer Procedure

Use the following procedure for transferring files to the system work disk.

| Step | User Input | System Response/Comments |
|---|---|---|
| 1. | Insert Pascal System disk into drive unit 0. Close door. Boot. | |
| 2. | Insert disk last formatted into drive 1. | |
| 3. | F | Filer command menu appears |
| 4. | T | Transfer what file? |
| 5. | Insert disk containing files to be transferred in unit 0. | Files will be transferred FROM selected disk. See file listing above for determining disk to be inserted. |
| 6. | #4:filename,#5:$ | Enter file from mandatory listing.<br>Screen display will appear:<br><br>volname:filename--->volname:<br>filename<br>Name of disk in unit 0 and file name transferred appear to the left; name of disk in unit 1 and file name appear on the right. |

Repeat steps 4-6 for each essential file first, using the appropriate disk in unit 0.

## 3.10 Installing the Bootstrap Program

Bootstrap installation is executed via the program BOOTER.

### 3.10.1 Bootstrap Installation Procedure

| Step | User Input | System Response/Comments |
|---|---|---|
| 1. | Insert Pascal system disk into drive unit 0.<br><br>Insert the PSYS2: Disk into drive unit 1. | |
| 2. | Q | To quit Filer. Command line appears. |
| 3. | X | Execute what file ? |
| 4. | #5:BOOTER<CR> | The following message appears:<br><br>This program is a general bootstrap mover.<br>To copy a boot from one unit to another, type unit number for the destination disk and the volume name of the source disk.<br>If the source disk is a disk file, then type the name of the disk file.<br>Unit to write boot to [4,5,9,10,11,12]: |
| 5. | Remove Pascal System Disk. Insert work disk. | |
| 6. | 4<CR> | File to be written as boot: |
| 7. | #5: BOOTSTRAP<CR> | Boot transferred successfully |
| 6. | IMMEDIATELY REBOOT SYSTEM | |

If all steps are completed successfully, the system work disk is now fully operational and ready for use.

If the bootstrap installation fails, the user should repeat all steps for building a working system disk. If failure again occurs, see the troubleshooting section of the Terak 8510 Installation Guide.

# SECTION 4

# SOFTWARE SYSTEM OVERVIEW

The files provided with the software system can be grouped into several categories by looking at the file name. If you examine the diskette directories of the seven software system diskettes, you will see several other catagories of file names including LIBRARY files, CHARSET files and COLORSET files. The file name prefix (or suffix) helps to identify the general category to which a file belongs.

## 4.1 File Categories

The major categories of files present on the software system diskettes and their functions are listed below.

| | | |
|---|---|---|
| SYSTEM | - | The software that comprises the operating system is contained in files prefixed by the word "SYSTEM". A complete discussion of the interpreter files and a brief description of each of the SYSTEM files is provided later in this chapter. A complete discussion of the use of the SYSTEM files is provided in the UCSD Pascal II.0 Users' Manual. The user who is unfamiliar with the operation of UCSD Pascal should refer to this manual for information on the operation of the various SYSTEM files. |
| DRIVER | - | These files contain the device drivers. A complete discussion of driver files is provided later in this chapter. |
| SPECIAL SYSTEM FILES | - | An additional group of files which do not carry the SYSTEM prefix are required for operation of the system. They provide the interface between the hardware and UCSD Pascal software systems, contain specialized libraries, and provide error and operating system codes for the machine level programmer. The following files are included in this group: 11.OPCODES, 11.ERRORS, GEMINI.M86, COLORGRA.TEXT, GDCLIB.TEXT plus the graphics library files (COLOR.LIBRARY, GDC.LIBRARY and MONOCHROME.LIBRARY). |
| BOOTSTRAP FILES | - | A copy of each of the Terak system bootstrap files is provided on the software system diskettes. These files are BOOTSTRAP and DW.BOOTSTRAP. |

| | |
|---|---|
| CHARSET - | The CHARSET files are alternate character sets for use with the 8510/a, 8510/b and 8510/c Graphics Computer Systems. |
| COLRSET - | The COLRSET files are alternate character sets for use with the 8600 Color Graphics Computer System. |
| TEXT AND CODE FILES - | The balance of the files provided with the software system carry the suffix TEXT or CODE and are either utility programs, or demonstration programs. |
| WINCHESTER FILES - | Two files for use with a Winchester disk drive are provided, DWBOOT.CODE and DWSETUP.CODE. |

## 4.2 System Interpreter File Characteristics

UCSD Pascal uses an interpreter-based implementation of Pascal. Technically, this means that the compiler emits code for a pseudo-machine which is emulated at runtime by a program written in the machine language of the host processor. This technique makes hardware variations transparent to the user.

The compiler, program editor, stand-alone operating system and the various utilities are written in Pascal and run using the interpreter.

The Terak systems are available with two different processors: LSI 11/2 and LSI 11/23. Because of the differences in the instruction sets of the two processors, two different interpreters are provided.

To enable execution of very large application programs, a pair of interpreters is provided that are smaller than the standard interpreter. The smaller interpreters do not have the ability to compile Pascal programs.

## 4.3 Driver Files

The driver files are loaded by a configuration program (SYSTEM.CONFIG) as the system boots up.

Driver files are loaded into memory only if hardware for that driver is present. For example, if you boot a DRIVER.DW file on a machine without a Winchester, that driver will not be loaded into memory.

If the hardware associated with two drivers is present, both will be loaded into memory. NOTE: This can result in a wasted memory. For example, if the disk has both DRIVER.QBA and DRIVER.QBB on it, both will be loaded, but only the one closer to the end of the disk will be used.

If a disk has two drivers that use the same UCSD unit, then the driver closer to the end of the disk will be accessed through that unit. For example, if you have DRIVER.QBB (Units 4, 5, 9, and 10) and DRIVER.DW (Units 9, 10, 11, and 12) on your disk and the DRIVER.DW is after DRIVER.QBB, units 9 and 10 will be assigned to DRIVER.DW.

## 4.4 Character Set File Characteristics

### 4.4.1 8510 Monochromatic Character Set Format

The character set of the 8510 Monochromatic Graphics Computer System must be initialized by the system software. It is fetched from a system file named SYSTEM.CHARSET which must be present on the diskette used to boot the system. If the system power has not been turned off since the last character set load and the bootstrap finds no file SYSTEM.CHARSET, the previously loaded character set is retained.

The standard English language character set is present in the file SYSTEM.CHARSET on the system disk provided with the Terak UCSD Pascal II.0 Software System. A number of additional character sets are provided. Using the system FILER, the standard character set can be replaced with one of the alternates by transferring the chosen character set file to the working diskette and renaming it SYSTEM.CHARSET.

The 8510 monochromatic character set file carries the extension .CHARSET. The utilities for working with a monochromatic character set will use this extension as the default. A monochromatic character set file contains a packed image of 192 character templates. Each template is a 10-byte field representing 10 rows (top row first) of 8 pixels each. The right-most pixel corresponds to the least significant bit (of the byte) of a character. Each half of the character set file contains a packed array of 96 character templates, covering character codes 40 (octal) thru 177 in the first half (the standard set) and codes 240 thru 377 in the second half (the alternate set). In general, each character in the alternate character set is the video reverse of the corresponding character in the standard set, but this need not be the case, as each half is created and modified separately by the Monochromatic Character Set Editor (CHEDIT).

Monochromatic character sets have a fixed 8-bit by 10-bit template. The 96 alternate characters may be accessed from a program by setting bit 8 of the character to be displayed. In the text editor, typing the DC1 key will toggle between the standard and alternate character sets.

| | |
|---|---|
| Bytes 1-960: | These bytes contain the standard character set. Each 10 bytes represents a printable character between 32 (blank) and 127 (delete). The character matrix is a fixed 8 bits by 10 bits; if a bit in a byte = 1, then a pixel in the character is illuminated. In the 10 by 8 bit array, 1,1 is the upper right corner and 10,8 is the lower left. |
| Bytes 961-1024: | Filler for an even block boundary. |
| Bytes 1025-1985: | These bytes contain the alternate character set. Each 10 bytes represent a printable character between 160 (alternate blank) and 255 (alternate delete). As with the standard character set, the character matrix is a fixed 8 bits by 10 bits; if a bit in a byte = 1, then a pixel in the character is illuminated. In the 10 by 8 bit array, 1,1 is the upper right corner and 10,8 is the lower left. |
| Bytes 1986-2048: | Filler for an even block boundary. |
| Bytes 2049-3072: | If the character set file is 5 blocks in length, these bytes contain the graphics logo displayed on the monochrome monitor at boot time. |

### 4.4.2 8600 Color Character Set Format

The character set of the 8600 Color Graphics Computer System is loaded at bootstrap time with the Color Graphics Operating System. The 8600 color character set file carries the extension .COLRSET. The utilities for working with a color character set will use this extension as the default.

A color character set file contains a packed image of a variable number of character templates (1 - 256). Character templates are variable in size, 2..256 pixels in each direction, but must be uniform within a particular character set. Character templates are defined in a byte array. Enough contiguous bytes are used to encompass an entire row of the character template, with template bits beginning at the most significant bit of the first byte of each line. If a portion of a byte is not needed to complete a character template row, it is not used. If necessary, the next character row will begin on the next complete byte.

Information in the 8600 color character set is stored byte flipped (i.e., low order byte first; high order byte second). The color character set has the following format. NOTE: The 8600 color character set file is not compatible with the 8510 monochromatic character set.

| | |
|---|---|
| Bytes 1-2: | Font number. This number is a value that is normally used to indicate the font number (1-9) into which a character set is to be loaded. |
| Bytes 3-4: | Character width. This value is the width, in pixels, of the character set. This value also determines the byte width of a character. |
| Bytes 5-6: | Character height. This value is the height, in scan lines, of the character set. This value, multiplied by ROUND (width divided by 8), gives the total number of bytes that a character requires. |
| Bytes 7-8: | Character set offset. This value specifies the starting value of the character set; it is usually ASCII 32 or blank, the first printable character. |
| Bytes 9-10: | Number of characters. This value specifies how many characters are stored in this file. (The usual value is 96 for the 96 printable ASCII characters.) This value, multiplied by height X ROUND (width divided by 8), will equal the number of bytes in the rest of the file. |

| | |
|---|---|
| Bytes 11-n: | These bytes contain the bit patterns for the character set; if a bit in a byte = 1, then a pixel will be illuminated. Bytes define a character from the lower left, moving right. Character widths are rounded to the next higher byte boundary (i.e., a character that is 9 bits wide takes 2 bytes for each pixel row). |

# SECTION 5

# SOFTWARE SYSTEM FILE DESCRIPTIONS

Each of the files shipped with the Terak/UCSD Pascal Software System is listed in the following sections. A generalized description of each of the files and information on the intended use of the file is provided.

## 5.1 Interpreter Files

| | |
|---|---|
| SYSTEM.11/2 | Interpreter for use with the LSI 11/2 processor. |
| SYSTEM.11/23 | Interpreter for use with the LSI 11/23 processor. |
| NOCOMPILE.11/2 | Smaller version of the interpreter which cannot compile Pascal programs. It must be renamed to SYSTEM.11/2 to be used. |
| NOCOMPILE.11/23 | Smaller version of the interpreter which cannot compile Pascal Programs. It must be renamed to SYSTEM.11/23 to be used. |

## 5.2 Driver Files

| | |
|---|---|
| DRIVER.DW | Driver for 8518 Winchester. Units 9, 10, 11 and 12. |
| DRIVER.HTKEML | Console emulator for 8510/c High Resolution Graphics System. No units. |
| DRIVER.LOAD86 | Driver to load the file GEMINI.M86 in the 8600 Color System. Does not use LSI memory after loading. No units. |
| DRIVER.MEMORY | Virtual disk driver for extended memory on the LSI 11/23. This enables the extra memory to be used as a block structured device. To determine the number of blocks available, do a bad block scan for 10000 blocks. The number of the first bad block is the size of the virtual disk. This size depends on the amount of memory available (128K or 256K). Unit 11. |
| DRIVER.QBA | QB floppy disk system for Terak 8510/a. If this driver is used with a Terak 8510/b or 8510/c, it will be noiser and there will be an increased possibility of read-write errors. Units 4, 5, 9, and 10. |

| | |
|---|---|
| DRIVER.QBB | QB floppy disk system for Terak 8510/b or 8510/c. If this driver is used on a Terak 8510/a, there is a very high probability of error. The disk may not boot. Units 4, 5, 9, and 10. |
| DRIVER.PRINTER | Line printer driver for output to a serial printer connected to Serial Unit 1. Unit 6 or PRINTER: |
| DRIVER.REMIN | Remote input driver which takes input from Serial Unit 2. Unit 7 or REMIN: |
| DRIVER.REMOUT | Remote output driver to output to a serial device connected to Serial Unit 2. Unit 8 or REMOUT: |
| DRIVER.TKEML | Console emulator for Terak 8510/a or 8510/b Monochrome Graphics System. Also the graphics aspect of the monochrome system. Unit 3 or GRAPHICS: |

## 5.3 System Files

Files prefixed by the word "SYSTEM" contain the software that forms the Terak/UCSD Pascal II.0 Software System. The following files are the standard UCSD Pascal system files, but not all are included on all Terak systems. See the UCSD Pascal II.0 Users' Manual for a complete discussion of the operation of the SYSTEM files.

SYSTEM.ASSMBLER

This is the standard UCSD Pascal assembler described in the UCSD Pascal II.0 Users' Manual. The assembler allows the user to create executable routines written in a UCSD subset of MACRO-11. If the SYSTEM.ASSEMBLER is used, the 11.OPCODES and 11.ERRORS files must also be present on the user's working diskette.

SYSTEM.CHARSET

This is the default system character set for the 8510 Monochromatic Graphics Computer System and is automatically loaded into memory at bootstrap time. If the user is working with an 8510 system or a system with both monochrome and color displays, this file must be present on the working diskette.

As shipped, SYSTEM.CHARSET contains the English letter alphabet. A variety of files with other alphabets are provided. The user can replace the English letter alphabet or modify individual characters by using the 8510 Monochromatic Utilities: Character Set Editor (CHEDIT) and the Character Set Demo (CHDEMO).

SYSTEM.COLORGRA

This is the machine code for the 8600 Color Graphics routines. Users working on 8600 Color Graphics Computer Systems and using the Terak color graphics routines must have this file on the prefixed diskette. (See the P(refix command under the Filer in the UCSD Pascal II.0 Users' Manual.) SYSTEM.COLORGRA is copied and renamed from either SYSTEM.COLORFIS or SYSTEM.COLORPU as appropriate.

SYSTEM.COMPILER

This is the standard UCSD Pascal compiler described in the UCSD Pascal II.0 Users' Manual, COMPILER. SYSTEM.COMPILER is used to compile the user's Pascal source file. If SYSTEM.COMPILER is installed on the user's diskette, SYSTEM.LIBRARY and SYSTEM.SYNTAX must also be present.

SYSTEM.CONFIG

This is the boot time configuration program which reads and loads the driver in memory based on availibility of the driver and hardware.

SYSTEM.EDITOR

SYSTEM.EDITOR is the standard Screen Oriented Editor that is described in the USCD Pascal Users' Manual, SCREEN ORIENTATED EDITOR. Two other editors, the Large File Screen Editor (L2) and the Yet Another Line Oriented Editor (YALOE), are provided. Either of these additional editors can be renamed as the SYSTEM.EDITOR.

SYSTEM.FILER

SYSTEM.FILER is the standard UCSD Pascal file handler that is described in the UCSD Pascal Users' Manual, FILEHANDLER.

SYSTEM.GDCLIB

This is the machine code for the 8510/c GDC Graphics routines. Users working on the 8510/c Graphics Computer Systems and using the Terak GDC graphics routines must have this file on the prefixed diskette. (See the P(refix command under the Filer in the USCD Pascal II.0 Users' Manual.) SYSTEM.GDCLIB is copied and renamed from either SYSTEM.GDCFIS or SYSTEM.GDCFPU as appropriate.

SYSTEM.LIBRARY

This file contains previously compiled or assembled routines to be linked with other programs. On the 8510/a or 8510/b work disk, MONOCHR.LIBRARY contains the Monochromatic Graphics Library Unit. On the 8600 work disk, COLOR.LIBRARY contains the Color Graphics Library Unit. On the 8510/c work disk, GDC.LIBRARY contains the GDC GRAPHICS unit. All three libraries also contain the standard units for input/output and intrinsic functions.

Note: COLOR.LIBRARY contains the Terak Touchpanel handler. This software is in place for the user with a Terak Touchpanel installed on the color monitor.

SYSTEM.LINKER

This is the standard UCSD Pascal linker. SYSTEM.LINKER allows the user to combine precompiled files that have been written in either Pascal II.0 or in assembly language into the user workfile. See the UCSD Pascal II.0 Users' Manual, LINKER, for details.

SYSTEM.MISCINFO

This file contains details on the Terak terminal specifications, some general information about the system, and specific information about the terminal's various control keys. SYSTEM.MISCINFO, if present, is loaded into memory at bootstrap time. The system utility, SETUP, is used to generate and modify the SYSTEM.MISCINFO file. See the UCSD Pascal II.0 Users' Manual, SETUP in the Utilities section, for details. The SYSTEM.MISCINFO provided will set the 8600 display for 48 lines to take full advantage of screen editing functions.

SYSTEM.PASCAL

This is the standard Terak version of the UCSD Pascal operating system kernel. This operating system has the TK emulator bound in.

SYSTEM.STARTUP

This is a user-definable file which is understood as a codefile. If it is present on a diskette, the operating system will execute it automatically at each bootstrap or initialize. This allows the user to specify a program to be run before the main command line comes up and anytime thereafter by typing I at the main command level.

On 8600 Color Graphics Systems, LINE24.CODE might be used to set up the system for a 24 line screen.

SYSTEM.SYNTAX

This file contains all of the compiler error messages.

## 5.4 Special System Files

COLORGRA.TEXT

This is used by the operating system to access the Terak color graphics routine file (SYSTEM.COLORGRA). This file is compiled and included in the COLOR.LIBRARY as unit COLORGRA. Inspect the file COLORGRA.TEXT for more information.

11.OPCODES

This contains the assembler directives (pseudo opcodes). It must be present on the user's diskette if the SYSTEM.ASSEMBLER is used.

11.ERRORS

This contains the assembler syntax error messages.

It must be present on the user's diskette if the SYSTEM.ASSEMBLER is used.

GEMINI.M86

This is the Terak 8600 Color Graphics Operating System which must be down-loaded to the 8600 Display Electronics Unit for operation of the 8600. Gemini is self-sizing to take full advantage of the memory characteristics of the user's particular 8600 color graphics hardware.

BOOTSTRAP

This is the bootstrap image for Variable Density (QB) diskette drive systems.

DW.BOOTSTRAP

This is the bootstrap image for Winchester disk drive systems.

## 5.5 Character Set Files

CHDEMO - Monochromatic Character Set Loader and Demo Utility

The Monochromatic Character Set Demo Utility, CHDEMO, will load a character set into the hardware character generator. A workspace is provided where characters can be typed and observed in context. The user must specify the file name of the character set to be loaded. Any one of the alternate software system character sets can be specified or a user-created character set may be loaded.

CHDEMO.PROMPT - A menu prompt file used by the CHEDIT Utility Program. The user will not access this file directly.

CHEDIT - Monochromatic Character Set Editor

The Monochromatic Character Set Editor (CHEDIT) allows the user to create or edit character sets for the 8510 Monochromatic Graphics Computer System. Character sets for the 8510 which operate under the Terak/UCSD Pascal II.0 Software System have the .CHARSET extension.

Character sets for the 8510 which are defined by CHEDIT have a fixed template size of 8 pixels wide by 10 pixels high. Each character set can have a maximum of 192 characters, 96 standard characters and 96 alternate. All 192 characters are accessible for modification.

Alternate Monochromatic Character Sets

The following alternate monochromatic character sets are included with the software system. These character sets can be loaded by renaming XXXX.CHARSET to SYSTEM.CHARSET. SYSTEM.CHARSET is the default system character set loaded when the system is bootstrapped. All characters, unless specified otherwise, are 5 by 7 pixels with true descenders in an 8 by 10 template.

APL.CHARSET

ARABIC.CHARSET

CURSIVE.CHARSET

FORMS.CHARSET

GREEK.CHARSET

HEBREW.CHARSET

INVERSE.CHARSET

LARGE.CHARSET

MATH.CHARSET

OCRB.CHARSET

RUSSIAN.CHARSET

SMUDGE.CHARSET

SYSTEM.CHARSET

WESTERN.CHARSET

CSLOAD

This program is used to load a character set into a character font on the 8600 system. The user is prompted for the name of the character set file to be loaded. The extension .COLRSET is assumed. Next the user is asked for the font number into which the character is to be loaded. This is a number between 1 and 9, inclusive. Lastly, the user is asked how many characters to load. Normally this value is 96 for a full ASCII character set. The file specified will be displayed on the screen.

The source text contains a general subroutine procedure (LODCHR) for loading 8600 color character sets. This routine may be extracted and included in the system library for more general use.

Character Set Conversion UTILITY

The Character Set Conversion Utility, CONVCHAR, will convert 8510 monochromatic character sets to 8600 color character set formats. The user is prompted for an 8510 character set file name. The extension .CHARSET is assumed. The 8600 output file name is then requested; the extension .COLRSET is assumed. The user will be prompted as to which part of the 8510 monochromatic character set to convert. The S(tandard part is the first 96 ASCII characters of an 8510 monochromatic character set. The A(lternate part is the last 96 characters of an 8510 monochromatic character set.

The output 8600 color character set file has the format specified in 8600 Color Character Set Format.

Alternate Color Character Sets

The following alternate 8 x 10 template size character sets are included with the software system. These character sets can be loaded into Font 1 using the CSLOAD utility. The character set SYSTEM.COLRSET is equivalent to the default character set contained in Font 0.

APL.COLRSET
ARABIC.COLRSET
CURSIVE.COLRSET
FORMS.COLRSET
GREEK.COLRSET
HEBREW.COLRSET
INVERSE.COLRSET
LARGE.COLRSET
MATH.COLRSET
OCRB.COLRSET
RUSSIAN.COLRSET
SYSTEM.COLRSET
WESTERN.COLRSET

## 5.6 Utilities

The Terak/UCSD Pascal II.0 Software System contains two sets of utility programs - the standard UCSD Pascal utilities and a group of supporting utilities provided by Terak. The standard

UCSD Pascal utilities can be identified by a [u] following the routine name, the Terak supporting utilities are identified with a [t].

## Directory Utilities

The directory utilities allow the user to mark and copy the duplicate directory, and write the directory of a diskette to a file.

### COPYDUPDIR [u]

The Duplicate Directory Utility will copy the duplicate directory from the diskette into the primary directory location. The directory stored in the primary location will be destroyed.

### DIRECTORY [t]

The List Directory Utility will write the directory of a diskette, in regular or extended format, into a file entitled 'DIR.LST.TEXT'. DIRECTORY.TEXT can be called and will provide information concerning the diskette directory location and structure. DIRECTORY.TEXT also provides a programming example should it be necessary for a user program to access or alter a disk directory.

### MARKDUPDIR [u]

If a diskette is not currently maintaining a duplicate directory, MARKDUPDIR will mark the diskette so that a duplicate is maintained.

## Disk Utilities

Terak provides two diskette utilities with the Terak/UCSD Pascal II.0 Software System - A Bootstrap Copying Utility and a Diskette Formatting Utility - and two utilities for use with Winchester hard disk drive systems - A Winchester Bootstrap Copying Utility and a Winchester Disk Setup Utility.

### BOOTER [t]

The Bootstrap Copier Utility will copy the bootstrap code (QB.BOOT, QX.BOOT or Q2.BOOT) to the first two blocks of the volume specified. The user must specify the diskette from which the bootstrap is to be transferred and the volume on which it is to be written.

FORMAT [t]

The Terak Format Utility is used to format 8-inch IBM Series 3740 floppy diskettes. The Format Utility will operate ONLY on hardware systems with a QB diskette controller.

The formatting procedure creates a data shell, or 'super structure', on the diskette and is needed to locate physical diskette sectors. In addition, formatting initializes the media attribute identifier area of the diskette. The program has the ability to format the three diskette densities supported by Terak QB systems: Single-sided, single-density; single-sided, double-density; and double-sided, double-density.

DWBOOT [t]

The Winchester Bootstrap Utility (DWBOOT.CODE) allows the user to select the Winchester unit to be assigned as the system volume. This capability allows the user to specify one of the Winchester volumes as the system volume.

NOTE: You must copy the Winchester bootstrap to the volume you are booting before DWBOOT can be used.

DWSETUP [t]

Because UCSD Pascal allows only a maximum storage size of 32,767 blocks per device, a Logical Device Table has been provided which allows the user to designate one Winchester device as up to four separate logical devices. This provides for more effective use of the Winchester storage space.

The size of each logical device is determined by a table that is stored in the Winchester driver. The Logical Device Table can be modified using the program "DWSETUP.CODE".

NOTE: Only 77 file entries can be made in a directory, regardless of volume size.

Editors

When working with the Terak/UCSD Pascal Software System, the editor will be the principle tool for creating, reading and changing .TEXT files. The UCSD Pascal user may choose from three editors:

o The Screen Oriented Editor (SOE)

o The Large File Screen Editor (L2)

o Yet Another Line Oriented Editor (YALOE)

In addition, Terak provides a Graphics Editor Utility for use with 8510/a or 8510/b Monochromatic Graphics Computer Systems.

SYSTEM.EDITOR (SOE) [u]

The Screen Oriented Editor (SOE) is the standard UCSD Pascal SYSTEM.EDITOR and is the default invoked unless another is specified. The Screen Oriented Editor is fully documented in the UCSD Pascal II.0 Users' Manual. NOTE: If L2 or YALOE have been named SYSTEM.EDITOR, that editor will be invoked instead of SOE.

L2 [t]

The Large File Screen Editor (L2) can be used to both write and edit text. The L2 editor has two unique features that make it a particularly useful editing tool. First, an unedited copy of the workfile is held as a .BACK file while editing is carried out. This protects the original file from accidental loss. Second, the L2 editor efficiently manages the file between the diskette and the memory buffer, allowing the editing of much larger files than is possible with SOE without loss of data due to buffer overflow. A detailed discussion of the operating of the L2 editor is contained in **Section 6 - Using Utilities.**

YALOE [u]

Yet Another Line Oriented Editor is a text editor intended for use on systems that do not have dynamic screen terminals. YALOE is documented in the UCSD Pascal II.0 Users' Manual.

GREDIT [t]

The Graphics Edit Utility (GREDIT), for use with the 8510/a or 8510/b Monochromatic Graphics Computer System, can be used to create and display pictures. GREDIT is useful in situations where textual descriptions are not practical. This includes cases where the information is difficult to express in textual steps, where the overhead to the program in one time only code is intolerable, and where quick display results are required. GREDIT can also be used as an example in teaching the basics of graphics development.

CONCAT [t]

The Concatenate Utility allows the user to combine any number of input files into one output file. The user must specify the name of the output file followed by the name of the first input file.

When the first file has been transferred, another input filename is requested. When all files have been specified, the user can exit the program or start again by specifying a new output file.

## I/O Utilities

The programs listed in this group are a series of Terak supporting utilities that provide various communication functions for use in developing I/O routines between the Terak unit and another device.

### PRINTOUT [t]

The Printout Utility allows listing of a text file to a printer, or to a serial line unit. The unmodified code drives serial line unit number two. Options can be easily modified by the user at runtime to drive either a printer or a line unit.

### SERIAL.IO [t]

The Serial I/O Utility contains a group of I/O routines that can be used for communications, through a serial interface, between the Terak unit and another device.

### TERMINAL [t]

With the Terak unit connected through a standard RS232 serial port to a 'host' computer, the Terminal Utility allows the user to treat the Terak unit as a simple terminal device. With the Terminal Utility operating, the Terak unit will accept keyboard commands, send and receive files, and accept input from the host computer.

The Terminal Utility, in conjunction with the information in the Serial I/O Utility, can be used as an example when developing programs involving communications through a serial interface.

## Library Utilities

The UCSD Pascal Library Utilities allow the user to map and link Library or Code files.

### LIBRARY [u]

The UCSD Pascal Librarian Utility allows the user to link separately compiled Pascal units and separately assembled subroutines into a LIBRARY file. See the UCSD Pascal II.0 Users' Manual, LIBRARIAN UTILITY, for details.

LIBMAP [u]

The UCSD Pascal Library Map Utility produces a map of a LIBRARY, or CODE, file and lists the linker information maintained for each segment of the file. See the UCSD Pascal II.0 Users' Manual, LIBRARY MAP UTILITY, for details.

PATCH [u]

The UCSD Patch Utility allows the user to edit and alter files in Hex or ASCII format.

DISASM.II [u]

The UCSD Pascal P-Code Disassembler Utility reads standard code files and outputs symbolic pseudo-assembly code (P-Code), along with various statistics regarding opcode frequency, procedure calls, and data segment references.

The disassembler is helpful in making architecture improvements or debugging and optimizing programs. A complete discussion of the P-Code Disassembler Utility is provided in the UCSD Pascal II.0 Users' Manual, P-CODE DISASSEMBLER.

RT-11 to Pascal Conversion Utilities

Three utilities are available to convert RT-11 structured files to Pascal format.

EDITTORT11 [u]

Edit To RT-11 will copy Pascal structured files, in text form, to an RT-11 diskette. The file size that can be transferred is limited by the size of the edit buffer (approximately 13,000 bytes).

RT11TOEDIT [u]

RT-11 To Edit will convert RT-11 structured files, in text form, to a Pascal diskette. Files must be in text form, and are limited to approximately 13,000 bytes in size.

RT2TOPAS [t]

The RT-11 To Pascal Utility will copy RT-11 structured files, in either text or binary form, to Pascal structured files. The file size is limited to the largest block of contiguous free space on the Pascal diskette.

Terminal Handling Utilities

The UCSD Pascal Terminal Handling Utilities allow the user to examine and modify various aspects of the system terminal handling.

GOTOXY [u]

GOTOXY is a Pascal procedure embedded in the Operating System. It provides random addressing for the terminal's cursor. A GOTOXY file is provided with the Terak/UCSD Pascal II.0 Software System. If you choose to write your own GOTOXY, or modify the existing one, the GOTOXY file will provide a sample structure. See the UCSD Pascal II.0 Users' Manual, GOTOXY, for instructions on writing your own GOTOXY.

BINDER [u]

The UCSD Pascal Binder Utility is used to bind a user-created GOTOXY file to the software system. The use of the Binder Utility is documented in the UCSD Pascal II.0 Users' Manual, GOTOXY.

SETUP [u]

The UCSD Pascal Setup Utility is used to generate and modify the SYSTEM.MISCINFO file. SYSTEM.MISCINFO contains miscellaneous data about the hardware system, general information on the terminal, and specifics about the terminal control keys. A detailed discussion of the UCSD Setup Utility is provided in the UCSD Pascal II.0 Users' Manual, SETUP.

## 5.7 Demonstration Programs

A number of graphics demonstration programs are provided with the Terak/UCSD Pascal II.0 Software System. These demonstrations are provided as examples of the types of graphics application programs that can be developed. Each demonstration file is present as a .CODE file and a .TEXT file. The user can X(ecute the .CODE file to display a specific demonstration. The .TEXT file provides the program steps used to create the .CODE file, and can be used as an example of how to create a given graphics representation.

## 8510/a or 8510/b Monochromatic Demo Programs

The following monochromatic demonstration programs are provided with the Terak/UCSD Pascal II.0 Software System. Monochromatic demonstration files can be identified by an (M) preceding the file name.

### Animated Demonstrations

To view the following animated demonstrations, X(ecute the demonstration program desired and respond to any system prompts that appear.

M.BREAKOUT

Is an interactive single player paddle ball game. To play the M.BREAKOUT game, X(ecute M.BREAKOUT and respond to any system prompts that appear.

M.LIFEDEMO

Builds a growing life structure.

M.PLANETS

Displays the motion of a moon in front of a planet.

INVERSE.FOTO - FOTO file used in the execution of M.Planets.

MASK.FOTO - FOTO file used in the execution of M.Planets.

SPHERE2.FOTO - FOTO file used in the execution of M.Planets.

M.RATMAZE

Displays a non-interactive run thru a maze.

### Emulator Demonstration

This file contains an emulator control code demonstration. Backspaces, tab, linefeed and cursor addressing are covered. To view the demonstration, X(ecute M.CNTRL.TK.

### Graphic Demonstrations

To view the graphic demonstrations, X(ecute the selected demonstration and respond to any system prompts that appear.

M.LETTERS

Displays an English language alphabet and demonstrates various character sizes.

M.OIL

Displays a graph of oil drilling results over several years.

M.PANSINE

Displays a scrolling graph of SIN (X)/X.

M.SALES

Displays a graph of sales vs. time.

Pattern Generation Demonstrations

To view the pattern demonstrations, X(ecute the demonstration program you wish to view and respond to any system prompts that appear.

M.PATT1

Displays a variety of pattern fill and flood routines.

M.PATT2

Additional pattern generation routines.

M.STARS

Prompts the user to enter the number of points on a star; then generates and fills the star pattern.

M.SURFACE

Draws a topographical surface map.

8600 Color Demo Programs

The following color demonstration programs are provided with the Terak/UCSD Pascal II.0 Software System. Color demonstrations can be identified by a (C) preceding the file name.

To view the color demonstration programs, X(ecute the demonstration program desired and respond to any system prompts that appear.

C.COLORS

Is a color demonstration program that draws a circle containing the current color palette, then rotates through the balance of the 512 possible system colors.

C.DOODLE

Is a graphics drawing demonstration that allows the user to draw various shapes to the screen.

C.PIES

Is a graphics chart demonstration program that draws a pie chart, a bar chart, and a table.

C.CNTRL.TK

Emulator control code demonstration. A simple 24-line monochrome display on the color monitor demonstrating backspaces, tab, linefeed and cursor addressing.

RESERVED

# SECTION 6

# USING UTILITIES

## 6.1 Disk Utilities

The Disk Utilities include two Bootstrap Copying Utilities, a Diskette Formatting Utility and a Winchester Setup Utility.

### 6.1.1 Winchester Bootstrap Utility

| | |
|---|---|
| FUNCTION | The Bootstrap Program DWBOOT.CODE allows the user to select the unit on a Winchester to be assigned the system volume. |
| SHORTENED NAME | DWBOOT |

| TYPE | THE SYSTEM WILL RESPOND WITH |
|---|---|
| 1. X | Execute what file? |
| 2. DWBOOT | DWBoot Version II.0 1.0<br>A utility to soft boot off Winchesters<br><br>Enter number of unit to boot (0 to exit)<br>[9, 10, 11, 12]: |

Enter the number of the unit from which to boot. A zero (0) will exit the program.

### 6.1.2 Winchester Bootstrap Program Error Messages

| | |
|---|---|
| DWBOOT: Invalid unit | The unit number entered is not one of the allowed units. |

### 6.1.3 Winchester Setup Utility

| | |
|---|---|
| FUNCTION | The Winchester Setup Utility allows the user to designate one Winchester device as up to four logical devices. |
| SHORTENED NAME | DWSETUP |

Because the exact formatted size of a Winchester disk drive will vary from unit to unit due to the presence of hard (media) errors, it will be necessary for the user to determine the exact size in blocks of the Winchester disk by running the 8510 Winchester System Acceptance Test. See the <u>8518 Winchester Disk Drive Installation and User's Guide</u> for instructions. Once the size of the user's Winchester disk drive has been determined, the Logical Device Tables for the Winchester driver to be used must be modified as described below.

Because UCSD Pascal allows only a maximum storage size of 32,767 blocks per device, a Logical Device Table has been provided which allows the user to designate one Winchester device as up to four logical devices. This provides for more effective use of the Winchester storage space.

The size of each logical device is determined by a table stored in the Winchester driver. The Logical Device Table can be modified using the program "DWSETUP".

CAUTION

If there is more than one Winchester device in the user environment, care must be taken to insure that a Logical Device Table intended for use on one Winchester device is not used for another Winchester device with a different Table. Data will be lost or destroyed if this occurs.

If two separate Winchester drivers are being used on the same Winchester drive or if one Winchester driver is being used for more than one drive, the Logical Device Tables must match. Data may be lost or destroyed if there is a mismatch.

NOTE: [ ] indicates options
      ( ) indicates default values

```
    TYPE                THE SYSTEM WILL RESPOND WITH

1.  X                   Execute what file?

2.  DWSETUP             DWSETUP version II.0 1.1
                        A utility to change Logical Device
                        Size Tables in an interpreter file:

                        Enter name of interpreter (DRIVER.DW)
3.  <return>            Unit #9: is 4352 Blocks
                        Unit #10: is 4352 Blocks
                        Unit #11: is 4352 Blocks
                        Unit #12: is 4352 Blocks

                        Do you want to change [Y/N] (N)
```

Enter the name of the Winchester driver file whose Logical Device Size Table is to be modified (default is used in this example). A <return> will use the default, DRIVER.DW, <esc> <return> will exit the program.

```
4.  Y                   Enter Size of Winchester (17408)

5.  34816               34816 available blocks, enter size of
                        Unit #9: (4352)

6.  8704                26110 available blocks, enter size of Unit
                        #10 (4352)

7.  8704                17404 available blocks, enter size of Unit
                        #11 (4352)

8.  8704                Unit #12 is 8704 blocks

                        Size of Winchester 34816 Blocks

                        Unit # 9: is 8704 Blocks
                        Unit #10: is 8704 Blocks
                        Unit #11: is 8704 Blocks
                        Unit #12: is 8704 Blocks

                        Update the file [Y/N] (N)

9.  Y                   Updating the file...
```

### 6.1.4 Winchester Setup Error Messages

| | |
|---|---|
| DWSETUP: file not found - "filename" | The driver file specified was not found on the default disk. Respecify file. |
| DWSETUP: Not a DW driver | The file specified is not a DW driver file. |
| DWSETUP: value too large | The size of the volume specified exceeds the number of available blocks. |

### 6.1.5 Bootstrap Copier Utility

| | |
|---|---|
| FUNCTION | The Bootstrap Copy Utility will copy a bootstrap file from a source diskette to a target and write the bootstrap in the first two blocks of the target. |
| SHORTENED NAME | BOOTER |

The Bootstrap Copier Utility will operate on all Terak QB or QX Floppy Diskette Graphics Computer Systems. Use the Winchester Bootstrap utility to copy bootstraps on Winchester hard disk drives.

| TYPE | THE SYSTEM WILL RESPOND WITH |
|---|---|
| 1. X | Execute what file? |
| 2. BOOTER | This program is a general bootstrap mover. To copy a boot from one disk to another, type the unit number for the destination disk and the volume name of the source disk. If the source is a disk file, then type the name of the disk file.<br><br>Unit to write boot to [4,5,9,10,11,12]: |

Enter the number of the unit to which the bootstrap is to be written (#5 in this example).

3. #5: File to be written as boot?

Enter the name of the bootstrap file to be copied - BOOTSTRAP or DW.BOOTSTRAP

The system will write the bootstrap to the target designated above.

## 6.2 RT2PAS - An RT-11 to UCSD File Conversion Utility

The Terak utility RT2PAS allows the user to transfer binary or text files from an RT-11 structured disk to a Terak/UCSD Pascal structured disk. RT2PAS will ask the user if the RT-11 directory is to be listed and then prompt for the source file name. This file name must be a file in the listed directory and entered in upper case. The user is then prompted for the destination file name. It must be a valid UCSD p-System file name and must include volume, file and extension specifications.

Finally, the user is prompted for the format of the source file. Data may be transferred in either binary or text form. Text files that will be accessed by the Pascal editor or will be used as source code for compilation or assembly should be copied in TEXT mode. This is necessary because the format of a Pascal source file is different from that of an RT-11 source file. Other types of files should be transferred in BINARY mode.

RT2PAS is similar in function to the UCSD Utility, RT11TOEDIT. It is included in the Terak Pascal release because the command structure is compatible with a corresponding utility PAS2RT, included in the Terak RT-11 Software Kit. These two utilities allow transportability between RT-11 and Pascal.

To access the RT2PAS Utility, insert the p-System work disk in drive 0 and the RT-11 disk in drive 1. Use the steps on the following page.

### 6.2.1 Executing RT2PAS

| Step | User Input | System Response/Comments |
|---|---|---|
| 1. | Y | Display the directory (Y/N) |
| 2. | Complete RT-11 filename to be transferred | Enter source file title: |
| 3. | Pascal filename the file is to be called | Enter target file title: |
| 4. | B or T | Transfer mode: B)inary or T)ext |

## 6.3 DIRECTORY

DIRECTORY will write the directory of a Pascal disk, in regular or extended format, into a user-specified device or file. If the default output is accepted, a file name "DIR.LST.TEXT" will be created. The user will be prompted for the output file name. This file must be a fully specified Pascal file or device. The user will be prompted for the input volume to be listed. A volume name or unit number for a disk unit is the only valid input. The directory listing generated is the full extended directory listing with file types and block locations. If another volume is to be listed, the new directory listing will be appended to the all previous listings.

To run DIRECTORY, use the steps on the following page.

### 6.3.1 Executing DIRECTORY

| Step | User Input | System Response/Comments |
|---|---|---|
| 1. | X | Execute what file? |
| 2. | #5:DIRECTORY | Directory Listing Utility<br><br>Full Detail? |
| 3. | Y <CR> | |
| 4. | <CR><br><br>to accept the default name, or enter another file name or a device such as PRINTER: | Enter output filename (DIR.LST.TEXT)? |
| 5. | Volume name or unit number to be listed | Generate directory of what unit?<br><br>After listing directory of specified volume, DIRECTORY will prompt more. |
| 6. | Y or y <CR> (optional) | DIRECTORY will again prompt for volume to list. The output is appended to any previous listings generated. |

The output file or device is not closed until the user exits DIRECTORY.

The source code DIRECTORY.TEXT will provide information concerning the disk directory location and structure. DIRECTORY.TEXT also provides a programming example, should it be necessary for a user program to access or alter a disk directory.

## 6.4 Serial Interface Utilities

### 6.4.1 TERMINAL Utility

With the Terak 8510 system connected through a standard RS-232 serial interface to a "host" computer, the Terminal Utility allows the 8510 system to be used as a terminal device. With the Terminal utility operating, the 8510 system will accept keyboard commands, send and receive files and accept input from the host computer. Upon execution, the Terminal utility is in an "ASCII terminal emulation" mode. At this point if the user enters characters on the keyboard, the characters will be transmitted to the Terak serial port set for unit 1. The default duplex is FULL so characters must be echoed by the host computer to be displayed on the console screen. The command line at the top of the screen shows the commands available for the Terminal utility.

To enter command mode, type <CTRL><^EOM> (6 on the numeric keypad). The command line will show

```
Terminal:  S(end file, R(eceive file, T(erminal mode, Q(uit
```

To abort the Terminal utility, type <CTRL><^NULL> (9 on the numeric keypad). To reenter the normal keyboard transmission mode, enter T. To send a file from the 8510 to the host computer type S. To receive a file from the host computer to be written as a file on the 8510, type R.

#### 6.4.1.1 Sending a File

When the user requests to send a file, the Terminal utility will prompt for the name of the Terak file to be sent. The user is then prompted for the file name as it is to be saved on the host computer.

The Terminal utility source code must be modified to suit the particular host computer with which the user is communicating. The Pascal procedure SENDFILE contains a line of code marked with "<<<<<<<". All host specific areas of the Terminal utility source are marked with this notation. The user must change the string "HERE COMES" to a command that is accepted by the host computer for creation of a file on the host's file system. For example, the command "CREATE" is utilized on the Digital Equipment VAX and TOPS-10 operating systems. The Terminal utility will automatically append the host file name to the specified string.

#### 6.4.1.2 Receiving a File

When the user requests the receipt of a file, the Terminal utility will prompt for the name of the host file to be sent to the Terak. Then the user is prompted for the Terak file name that that file is to be written to.

The Terminal utility source code must be modified in the Pascal procedure "TAKEFILE" to suit the user's host environment. The text "PLEASE SEND" must be changed to a host command that will transmit a file from the host to a terminal. The Terminal utility will automatically append the host file name to that string. As the file is transmitted to the Terak, the characters are written to the disk file instead of to the console screen.

The commands included in the TERMINAL.TEXT source file will provide additional insight into the function and application of the TERMINAL Utility. There are documented lines where the more common alterations may be performed to adapt TERMINAL to a particular host.

### 6.4.2 SERIAL.IO Utility

The SERIAL.IO Utility contains a group of I/O routines that can be used for communications by means of a serial interface between the Terak system and another device. This utility is a subset of the routines used by TERMINAL.

### 6.4.3 PRINTOUT Utility

The Printout Utility allows listing of a text file to a printer or to a serial interface. The unmodified code drives serial interface number one with soft form feeds. The code can be easily modified to drive either a printer or serial line unit through the operating system handlers. The code may also be modified to generate hardware form feeds. Options that the user may modify are flagged in the source file PRINTOUT.TEXT.

To run the program, execute PRINTOUT. The user may specify the number of lines per page, headings, page numbers, line spacing, number of copies, and margins.

In addition, the following commands are recognized when imbedded in the TEXT file, as the first two characters (left justified) in any line. The commands are not printed.

| COMMAND | FUNCTION |
|---|---|
| ^F | Formfeed. Move to the top of the next page. |
| ^O or ^0 | Overstrike. Suppress the linefeed at the end of this line. |
| ^1 thru ^9 | Linefeeds. Advance one through nine blank lines. |
| ^P | Paragraph. If the current position is within 10 lines of the bottom of a page, this command will cause a formfeed. |
| ^S | Start single line spacing. |
| ^D | Start double line spacing. |
| ^T | Start triple line spacing. |
| ^Q | Start quadruple line spacing. |

### 6.4.4 CONCAT - File Concatenate Utility

The CONCAT Utility will fold any number of input TEXT files into one output file. This is useful for combining a large number of small files. To run the program, execute CONCAT. The output filename is requested, followed by a request for the first input file.

When the first file has been transferred, another input filename is requested. When all files have been specified and the next request for an input file appears, press <CR>. It is then possible to exit the program, by answering 'N' to the question "MORE?", or to start again with a new output file by answering 'Y'.

The .TEXT extension is assumed on all filenames entered and therefore need not be typed.

## 6.5 Character Set Utilities

### 6.5.1 CHDEMO - Monochrome Character Demo Utility

The Character Set Demo utility CHDEMO will load a character set into the 8510 character generator buffer. A workspace is provided in which characters can be typed to be observed in context.

To run the program, execute CHDEMO. A table listing the character sets included on the Character Sets Disk will be displayed. Any of those listed can be loaded by pressing <ESC> followed by the letter of the desired character set.

The last entry of the table, U)ser, is provided so that a character set not listed can be loaded. When <ESC>U is typed, a prompt appears requesting the name of file; the CHARSET extension is assumed. Type the file name followed by a <CR> and the character set will be loaded.

Press <ESC> twice to exit the program. This will leave the current character set loaded. It can be modified by using the Character Editor utility if desired.

Character sets may be loaded into memory from within any user program just as they are by CHDEMO. CHDEMO.TEXT is also included on the Character Sets Disk and contains a procedure entitled LOAD which loads user program character sets.

### 6.5.2 CHEDIT - Monochrome Character Editor Utility

CHEDIT is a character-set editor used to modify and store character sets. The editing will always be performed on the character set currently loaded into the character generator buffer.

To run the program execute CHDEMO.

In the left hand corner of the screen a grid is displayed, ten boxes high and eight boxes wide. Each represents one pixel in a character.

In the lower left hand side of the screen both the optional and standard portions of the character set are displayed.

The upper right hand portion of the screen is blank. The area in the lower right corner displays the character being modified.

### 6.5.2.1 CHEDIT Commands

| Function | Instructions/Comments |
|---|---|
| TO MODIFY A CHARACTER | Type "G" followed by the character desired. It will be displayed in the grid and also in the lower right portion of the screen. |
| CURSOR | The cursor is positioned in the upper left box of the grid and may be moved anywhere in the grid using the four arrow keys. |
| BOX | The box where the cursor is positioned can be modified with the Oppose (O), Add (A), and Zap (Z) commands. |
| ADD | Add a pixel (turn it on) to the character. |
| OPPOSE | Turn a pixel on if it is off or vice versa. |
| ZAP | Remove (turn off) a pixel. |
| INVERT | Turn on every pixel within the grid. |
| MOVE INTO | Move into, followed by Get and a character, will replace the character currently displayed with the new character specified. This does not produce the same result as a Get command alone. |
| LOWER CASE | To call the upper half of the standard character set (lower case), use the DC2 key. Typing it again will bring back upper case. To reach the alternate portion of the character set, the Exchange (X) command is provided. |

| | |
|---|---|
| | To replace the standard portion of the character set with the alternate one and vice versa, use the Move Into (M) command, followed by the Exchange (X) command. |
| EXIT | To exit the program, use either Quit (Q) or Keep (K). |
| QUIT | If Quit is used, the modified character set remains in memory, but the file SYSTEM.CHARSET is not affected; after rebooting, the SYSTEM.CHARSET will be restored. |
| KEEP | If Keep is used, the modified character set is written to a file NEWSET.CHARSET on the work disk. |

If NEWSET.CHARSET is renamed as SYSTEM.CHARSET, the modified character code will be loaded into memory whenever the system is booted.

To use CHEDIT to modify a character set, execute CHDEMO to load the character set file, then execute CHEDIT to begin editing the character set in memory.

### 6.5.3 COLOR.EDIT - Color Character Set Editor

This program allows the user to create or edit custom character sets for the 8600 color display. Character sets for the 8600, operating under the Terak/UCSD Pascal II.0 Software System, have the .COLRSET extension.

Character sets for the 8600, defined by COLOR.EDIT, can have a variable template size ranging from a minimum template of 4 pixels wide by 5 pixels high, to a maximum template of 64 pixels wide by 80 pixels high. The default template is 8 pixels wide by 10 pixels high.

Each character set can have a maximum of 128 characters. All 128 characters are accessible for modification.

Executing COLOR.EDIT will display the program date and version. COLOR.EDIT will then begin initializing a scratch file on the diskette; at least 80 contiguous blocks must be available for the scratch file, or COLOR.EDIT will exit with the error message "Cannot open scratch file". The default template and a menu of options will then be displayed. The following is a list of the menus in COLOR.EDIT and an explanation of the options.

## Main and Subsidiary Command Menus

G(et — a character. Before any modification can be done to an existing character, it must be fetched. The Get command will display the character in the template and display the character in actual size in four different colors. To get a character specify

O(rdinal - the base 10 ASCII value of the character

or

C(haracter - input a single character keystroke

M(odify — current character. This command will allow character level changes to be made. Operations on the displayed symbol are

| | |
|---|---|
| C(lear character | - clears all pixels that are set (lighted) in the character template |
| I(nvert) character | - clear or set all pixels in the character template according to previous conditions. |
| eX(pand | - character - specify an integer factor between 1 and 16, for modification of the width (x direction) and height (y direction). Expansion of a character may take it outside of the viewable area in the current template. Change the size of the template to view the entire symbol. |
| P(ixel | mode - enter the single pixel edit mode. Press the shift lock, then use the numeric keypad to move the blinking cursor. With pixel edit it is possible to |

| | |
|---|---|
| A(dd | - set (light) a pixel at the current cursor position. |
| C(lear | - clear (extinguish) a pixel at the current cursor position. |
| I(nvert | - change the state of the pixel at the current cursor location. |
| M(ove | - moves the cursor without adding, clearing or inverting pixels in the cursor path. |

eX(pand character set. For each character in the character set, expand it by an integer factor between 1 and 16 for width (X direction) and height (Y direction). Character expansion for an entire character set can be lengthy.

N(ew font specifies the file name of a character font to be edited. The .COLRSET extension is assumed. A file must be specified if an existing font is to be edited. When a new font is fetched, the current character templates are transferred to the scratch file. Any editing performed affects the scratch file, and not the original file, until the font is written out.

W(rite the font to a file; save the contents of the scratch file as an 8600 color character set. The output file will have the information written to it as specified in "Color Character Set Format". The font number is currently always 1. The width and height, in pixels, is the current template size. The offset character is 32 or blank. The number of characters is 96.

| | |
|---|---|
| W(rite | - to file will prompt for a file name. |
| U(pdate | - the current file will write to the file name last specified with N(ew font. If no name was specified, you will be prompted for a name. |
| R(eturn | - to editor will return to the character set editor without writing anything to any file. |

| | |
|---|---|
| C(hange | template size, specifies a new character template size. Prompts will ask for a decimal value for the character width (x size) and height (y size). Numbers between 4 and 64 are valid for the width; numbers between 5 and 80 are valid for the height. After the template size is changed, the character is painted into the new template, starting at the lower left corner. If the new template size is smaller than before, pixels will not be lost until a new character is fetched into the template. THIS DOES NOT CHANGE THE SIZE OF THE CHARACTER. |
| D(isplay | current font. The screen is cleared and the current character set, as it exists in the scratch file, is painted to the screen. Pressing <ESC> at any time will stop the display; pressing <SPACE> will return to the main command menu. If a character was in the template at the time of this command, it is written to the scratch file and the template is cleared. At the current time, if the template width is not a multiple of 8, the characters may be undecipherable. |
| Q(uit | to leave the Color Character Set Editor. If any changes have been made to a font and it has not been previously written to a file (i.e., with the W(rite command), then the prompt "Current font has not been saved. Do you wish to do so?" will appear. If "Y" is typed then a call to W(rite to font will be made. |

Example:

To edit an existing character set, make sure that there are at least 80 contiguous free blocks on the disk you will use as the prefixed volume. SYSTEM.COLORGRA must also be resident on the prefixed volume.

1. Type "X" to execute a file; enter COLOR.EDIT as the file to execute.

2. Type "N" for a new font. When prompted, enter the name of the .COLRSET file to be edited. The template size will change to reflect the size of the new font. Now enter "CURSIVE". Note: To create a new font, it is not necessary to edit an existing font.

3. To modify a character, the character must first be entered into working storage. To do this type "G" to get a character. When prompted with O(rdinal or C(haracter, type "C"

4. Type "A", for example, to get the character "A". As it is currently depicted, the character will be displayed in the template and an actual-sized representation, in four colors, will be seen in the lower left hand corner of the screen.

5. Next type "M" to modify; a cursor will appear in the lower left hand corner of the template. The current template may be cleared by typing "C", or it may be inverted by typing "I".

6. To add or delete selected pixels, type "P". In pixel mode, the cursor may be moved within the template. Depress the shift lock and use the numeric keypad to move the cursor to the desired location. To begin adding pixels, type "A"; a pixel will be added in the template and in the actual-sized representation at the current cusor location. To clear pixels, type "C"; a pixel will be cleared in the template and in the actual size representation, at the current cursor location. Typing "I" to invert the current state of the pixels will work in the same manner.

7. To perform a non-destructive move, type "M". Moving the cursor will not affect any of the pixels within the template. Typing "Q" will exit pixel mode and return to modify mode.

The character may be expanded by selected factors in the X (width) and Y (height) dimensions. By typing "X" in the Modify mode, the current character will be expanded by the factors specified. The algorithm for expansion is very simple and will not perform smoothing. Expansion of a character, beyond the template size will not lose the character, but only that portion of the character within the lower left portion of the displayed template will be visible. To view the rest of the character, the template size must be changed at the top command level. For example: type "X", then type 2<RET> to double the width; next, type 2<RET> to double the height. It will appear as though the character is not all there but that will be corrected.

Type "Q" to quit modify mode. From the main command level, type "C" to change template size. Enter 16 for the new width and 20 for the new height. After the template size is changed, the character "A" will be completely painted. Now type "D" to display the entire character set. Notice that the character spacing is the current template size (16 x 20 pixels). The character A will be twice as large and above the rest of the characters because the expansion works on blank pixels as well as filled pixels.

After the full character set is displayed, press <SPACE> to return to the full command level. At this time, type "Q" to quit the color character editor. When asked if you want to save the character font, type "N" unless you want to save it. There must be a block of space at least 2 blocks long for each 8 x 10 character set created. For each factor of expansion in any direction, multiply by the corresponding factor to determine how much space is needed. If you get the error message "ERROR: Cannot write block to output file", the editor will return to the main command level. Insert a disk with enough free space into a disk drive, then Q(uit and W(rite the font to that new disk.

## Rebuilding COLOR.EDIT

The 8600 Character Set Editor consists of a number of files. It is necessary to have GEMINI V2.0 or later to execute this version of the character editor. The files used to build the character set editor are as follows:

| | |
|---|---|
| CH.MAIN | The main enclosing program. This file will include all other needed files at compilation time. |
| CH.DECLARE | The CONST, TYPE and VAR declarations that are global to the editor. |
| CH.INIT | The editor initialization routine. Sets all global initial variables. |
| CH.UTILITY | Several utility procedures used within the editor. |
| CH.GETFILE | Procedures used to establish the scratch file and read and write a specified character set. |
| CH.GETCHAR | Procedures to read and write a specified character from the scratch file. |
| CH.KEYBD | Procedures to get and interpret keyboard input. This requires asynchronous input and can be modified to include a digitizer for input. |

| | |
|---|---|
| CH.DISPLAY | Procedures for displaying the template and character on the screen. |
| CH.DFONT | Procedures to display the entire character set on the screen. This procedure requires that the 8600 have the 64K Byte Memory option. |
| CH.CHANGE | Procedures to change the size of the character template. |
| CH.MODIFY | Procedures to do pixel level modification of the character displayed in the template. |
| COLOR.EDIT.CODE | A compiled and linked version of the 8600 character set editor. This program requires GEMINI V2.0 or later to run. |
| SYS820.COLRSET | A sample character set with a template size of 8 by 20. |
| COLORGRA.TEXT | The edited version of the COLORGRAPHICS unit interface to be used with the 8600 character set editor. This unit has been specially edited to have only those routines currently used by the editor. To rebuild the editor, it is necessary to compile this unit and rebuild the system library. |

Steps to Rebuild

After making changes in the 8600 Character Set Editor, rebuild the editor as follows:

1. Compile COLORGRA.TEXT.
2. Using LIBRARY.CODE, build a new system library, replacing the unit COLORGRA with the just compiled code file.
3. Compile CH.MAIN. All other needed files will be included during compilation. The disk that has CH.MAIN must be the prefixed disk.
4. Link the resulting code file against the SYSTEM.LIBRARY. Name the output file COLOR.EDIT.CODE. Be sure to note any changes and modifications in CH.MAIN when editing that file. Also it may be advisable to change the version number of the program.

## 6.6 Monochrome Graphics Editor (GREDIT) Utility

The Graphics Editor (GREDIT) Utility is used to create and display pictures. These pictures are very useful in situations where textual descriptions are not practical. These include cases where the information is difficult to express in textual steps, where the overhead to the program in one-time-only code is intolerable and where quick display results are required. GREDIT can also be used as an example for development of graphics programs.

GREDIT is used to create .FOTO files. These files may then be used as backgrounds for other programs, or may be further modified using GREDIT or other programs. GREDIT .FOTO files can be converted to hard copy if a suitable dot matrix printer is available with a graphics printing program.

### 6.6.1 GREDIT Conventions

#### Graphics Space

The graphics space is always 320 pixels (picture elements) wide by 240 pixels high. Two independent pictures can be maintained in working memory. Any .FOTO (picture) file on disks can be accessed.

#### Picture Files

Picture files are 19 logical blocks in size and contain the image of one picture. When written to a disk, the Picture file will have the extension .FOTO appended to the filename identification.

#### Cursor

The cursor appears on the graphic display as a steadily blinking pixel. The cursor, which overlays the picture, can be placed anywhere on the screen and is generally nondestructive. The brightness of the cursor is adjusted according to the state of the pixel overlaid. The cusor will appear brighter when over-laying a lighted pixel. The cursor may not be directed off the screen; motion will stop at the pixel nearest the screen border.

The cursor is a virtual pen that has four states: UP, DOWN, ERASER and COMPLEMENT. The effect on the display of most graphics commands depends upon the state of the cursor-pen. The cursor is always associated with a heading that is the absolute angle from which commands will begin their motion.

## Screen Addressing

The display is addressed in absolute Cartesian coordinates with the origin (0,0) at the center of the screen. The unit of length is the distance between the centers of adjacent pixels. The X coordinate range is -160 to 159. The Y coordinate range is -119 to 120. The unit of the angular measure is the degree. Angles are referenced to the positive X coordinate axis; counterclockwise rotations are related to positive angles. All numbers are integers; fractional lengths and angles are not supported. Therefore the effect of integer arithmetic truncation must be considered. Note that GREDIT is oriented to the display coordinates and not to the ultimate scaling of the quantities being represented.

## Commands

Commands are entered on the keyboard and are generally echoed in the character display which overlays the picture. Most commands are entered with a single keystroke and require subsequent data: parameters related to the command, sub-command selection, or affirmation that the command is desired. Most commands accept upper or lower case characters as the same command.

## Absolute Motion Commands

Absolute motion commands generate cursor motion independent of the present cursor position heading.

## Relative Motion Commands

Relative motion commands generate cursor motion that depends upon the present cursor position and heading.

## Command Prompts

When prompted by an asterisk, typing the <return> key will clear the character display, leaving the picture unaffected. Typing the ? or / key will blank the picture and display a summary of all operating commands supported by GREDIT. The picture is restored by pressing <space>.

## Penstate Commands

U(p will bring the cursor-pen up (will not draw on the display screen); the pixel overlaid by the cursor will not be affected. The initial state of the pen, when GREDIT is first run, is Up.

D(own will allow the cursor-pen to draw solid lines on the display screen; the pixel overlaid will be set (lighted).

E(raser will allow the cursor-pen to draw clear lines on the display screen; the pixel overlaid will be cleared (turned off).

C(omplement will allow the cursor-pen to reverse the state of pixels that it overlays. If a pixel overlaid by the cursor is set (lighted), it will be cleared; if the pixel is clear (turned off), it will be set.

## Motion Commands

J(ump moves the cursor from its present position to a new position at a given (X,Y) coordinate (along a straight line). Movement is in units relative to the screen origin. All pixels along that line, excluding the starting pixel but including the end pixel, are affected by the penstate.

L(eap moves the cursor from its present position to a new position at a given (X,Y) coordinate (along a straight line). Movement is in units absolute to the screen origin. All pixels along that line, excluding the starting pixel but including the end pixel, are affected by the penstate.

H(eading changes the angular heading of the cursor, relative to the horizontal screen axis. The new heading can be specified in rectangular form (X,Y) or polar form (angle). The cursor and display are not affected.

T(urn changes the angular heading of the cursor, relative to the present cursor heading. The cursor and display are not affected.

M(ove relocates the cursor from its current position to a new position along the present heading. The pixel-to-pixel distance is specified by the user. The heading is not affected. All pixels along the generated line, excluding the starting pixel but including the ending pixel, are affected by the penstate.

S(egment relocates the cursor from its current position to a new position along a circular path. The path starts tangential to present heading, turns at a given radius in units and continues for a given angle in degrees. All pixels along the generated line, excluding the starting pixel but including the ending pixel, are affected by the penstate. The heading is changed and will be tangential to the circular path at the ending pixel. Note that both the given radius and angle may be positive or negative, allowing four possible directions of circle generation from a given cursor heading.

Cursor arrow keys will move the present cursor in the direction indicated by the keys. If the cursor is "twisted", movement will occur 45 degrees counterclockwise from the direction indicated on the key. Typing the backslash key toggles the twist condition on and off. All pixels along the path of the cursor will be affected by the penstate with the exception of the starting pixel.

The cursor arrow keys have a repeating action that will cause the cursor to move (in the key direction) in steps based upon the elapsed time of the last keystroke. When this repeat time is greater than 0.25 seconds, single steps will be generated. When repeat time is 0.12 to 0.25 seconds, the cursor will step 5 times per keystroke; when the repeat time is less than 0.12 seconds, the cursor will step 10 times per keystroke.

W(alk can be used to move rectangular blocks of a picture to different locations within the same picture. The cursor should be positioned in the upper left corner of the rectangle to be moved. Define the area to be moved in terms of pixel width and height. A height of zero equals one unit high; a width of zero equals one unit wide. After defining the block to be moved, enter the coordinates, relative to current cursor position, of the rectangle destination. Then enter a motion command (Move, Jump or Leap). The block will be copied to the new location starting from the upper left corner. Care should be taken to ensure that the target block area does not overlap the block being moved. If the source and target areas do overlap, it may only be possible to duplicate only the part of the source area which does not overlap the target.

With the Walk command, the penstates are interpreted as follows:

| | |
|---|---|
| Up | Replace target block with source block. |
| Down | Set the pixels in the target block wherever the pixels are set in the source block (Inclusive OR). |
| Complement | Reverse the state of the pixels in the target block wherever the pixels are set in the source block (Exclusive OR). |
| Eraser | Clear the pixels in the target block wherever the pixels are set in the source block. |

The current heading and penstate are not affected.

## Function Generation Commands

### Pixel Control

Single pixels may be set or cleared at any time. Type the (.) key to set a pixel, and type the (,) key to clear a pixel. The cursor location, heading and pen state are not affected. This feature, combined with the cursor key motion, allows manual generation of special pixel patterns without affecting the penstate.

The user can delete or alter an area of the picture by pressing <DEL>. When the delete mode is entered, the penstate will be changed to ERASER, and the data between the starting and ending cursor position will be erased.

To leave delete mode, type any key other than the four cursor direction keys. Whatever key is used to exit will be interpreted as the next command, i.e., another <DEL> will reenter the delete mode at the current cursor position.

### Figure Generators

The P(olygon command initiates a repetitive sequence of Move and Turn commands. The three parameters entered are A(ngle, S(ide and I(ncrement. The generation sequence will proceed as follows:

- o M(ove along current heading by (Side) units.
- o T(urn Heading by (Angle) degrees. Add (Increment) units to Side.
- o Repeat until any key is typed.

The values for Angle, Side and Increment can be positive or negative integers. The Polygon figure will start at the current cursor location. If the Increment is zero, Polygon will generate closed figures. If the Increment is not zero, unbounded figures will result. When the Polygon is used, the Heading and cursor-penstate will be affected; the Heading will be whatever it was at the instant a key is typed to stop generation, and, if the penstate was Up before Polygon was invoked, it will be changed to Down.

The B(ox command moves the cursor through a rectangular path of given height and width. The width will be drawn along the present heading; the height will be drawn at heading + 90 degrees. If the penstate was Up before box was invoked, it will be changed to Down. Height and width may be positive, negative or zero. A zero value for height or width draws a line and returns the cursor to its original location. The Heading is not affected.

Arrowheads can be generated by typing the (^, 6) key. The arrowhead will point along the present heading. If the pen state was Up, it will be changed to Down. The cursor is left at its original position. Four arrowhead sizes are available: small, medium, large and extra large.

Character Generation

The A(SCII command allows the user to add characters to the graphic display. The template for the characters is taken from the character set buffer. The character size will be an 8x10 pixel box; the current cursor position will appear at the lower left hand corner of the box. Any character sets that have been defined and are in memory may be used. Press <DC2> to enter lower case characters; <DC1> to alternate between standard and alternate character sets; and <DC1> <ESC> to quit entering characters.

When using the ASCII command, the cursor arrow keys will move the cursor left, right, up or down by one character step. The Backspace key will step back one character space.

Press <Return> to place the cursor at the left column of the screen. The tab key will move the cursor in steps of 8 pixels. When characters are entered, an audible click will be heard and the character echoed in the character display.

With the ASCII command, the penstates have the following effects:

| | |
|---|---|
| Up | The contents of the 8x10 pixel target block will be replaced with the character typed (destructive overlay). |
| Down | The character typed is overlaid into the target block (Inclusive OR). |
| Complement | The state of the pixels in the target block where corresponding pixels are set in the character typed (Exclusive OR). |
| Eraser | Clear the pixels in the target block where the corresponding pixels are set in the character typed. |

## Lettered Memory

GREDIT has a lettered memory array which can be used to store screen locations, numbers or both. There are 26 memory cells, each called by a letter of the alphabet. The commands used to control this feature are Remember, Verify and Forget.

R(emember will store a Number at the present location:

*GREDIT will prompt, asking if the user wants to store a number or cursor location.

*Type L to store a cursor location, N to store a number.

*GREDIT will request the letter of memory (A-Z) in which the data is to be stored. If a number is to be remembered, its value is requested.

V(erify will display the contents of one cell or all cells of the lettered memory. Type one letter (A-Z) to view the contents of an individual memory cell; press <space> to view the contents of the entire lettered memory. When a stored location is verified, both of it coordinates will be printed and the location on the screen will be flagged by a flashing box. If a number is verified, it will be printed as stored.

F(orget will erase the entire contents of the lettered memory.

The contents of the lettered memory may be used for any of the motion or function commands. When a certain value is required, a letter into which a value has been stored can be typed in place of the number. The user may insert a minus sign (-) in front of the letter if the negative of a lettered value is desired. For example, if the value of 10 is stored in letter A and a cursor location 25,50 is stored in letter B, then a cusor leap to -A,X will produce a motion to absolute location -10,50.

## Control Commands

I(dentify wil print the present cursor location, its heading and the pen state. The location of the cursor in the picture will be preserved.

N(ew will perform both a Zap and a Forget.

GREDIT maintains two independent picture spaces, A and B. This allows the user to back-up progressive stages of photo editing (with the Duplicate command) and ensures that if the working picture is destroyed, a copy is available. (For long-range backup, output the picture to a file.)

The cursors, headings, picture content and lettered memory of the two pictures are completely independent. All commands, including Zap, New and Forget, affect only the picture currently displayed.

Two commands allow access to the dual picture capability:

X(change displays the alternate picture (A or B) and accesses the alternate lettered memory, headings and cursor. The contents, heading and lettered memory of both pictures are preserved.

K(ombine generally transfers the contents of the alternate picture (not displayed) into the one presently displayed. The contents of the alternate picture are not affected. The mechanism of combination is selected by one of the following commands.

> D(uplicate transfers the present picture, including lettered memory, cursor and heading into the alternate picture space.
>
> O(verlay sets (lights) pixels in the present picture wherever the corresponding pixels in an alternate picture are set. Lettered memory, cursor and heading are not affected.
>
> E(rase clears (turns off) the pixels in the present picture whenever the correponding pixels in an alternate picture are set. Lettered memory, cursor and heading are not affected.
>
> C(omplement reverses the state of the pixels in the present picture wherever the corresponding pixels in an alternate picture are set. Lettered memory, cursor and heading are not affected.
>
> R(everse inverts the state of all pixels in the displayed picture. The alternate picture is not referenced. Lettered memory, cursor and heading are not affected.
>
> I(ntersect clears all pixels in the displayed picture where the pixels in the alternate picture are cleared (AND). Lettered memory, cursor and heading are not affected.

## File I/O Commands

O(utput causes the present picture contents and GREDIT state to written into a file. The file extension will be .FOTO. Present picture contents and GREDIT states, including memory content, are not affected.

G(et causes the present picture to be replaced by the contents of the file the user specifies. The file extension .FOTO will automatically be added to the file name. The contents of the file requested, including the GREDIT state and lettered memory, will be loaded from the file.

Q(uit exits GREDIT and returns command to the operating system. Anything not stored into a file is lost.

Command Reference Card

```
*-----PENSTATE COMMANDS-------------*------ANGLES-----------*
* U(p draw ghost lines/dots         *        90             *
* D(own draw solid lines/dots       *         !             *
* E(raser draw clear lines/dots     *  180 __*__ 0.360      *
* C(omplement reversing lines/dots  *         !             *
*_____MOTION COMMANDS_______________*        270            *
* M(ove relative to cursor @heading *_______RANGES__________*
* T(urn heading by relative angle   *-160,120.......159,120 *
* J(ump to relative X,Y location    *   :              :     *
* L(eap to absolute X,Y location    *-160,0  0,0     159,0  *
* H(ead to absolute angle for Move  *   :              :     *
* P(olydgon generator B(ox generator*-160,-119......159,119 *
* W(alk blocks ^(rrowhead generator *----LETTERED MEMORY____*
* S(egment circle generator         * R(emember  V(erify    *
*------CONTROL COMMANDS-------------* F(orget cursor, mmbrs *
* N(ew pix & letters Z(ap pix only  *------SINGLE DOTS------*
* A(scii character pattern entry    * Period key sets dot   *
* I(dentify cursor and states       * Comma key clears dots *
* X(change with alternate picture   *-----CURSOR MOTION-----*
* K(ombine alt pix into this pix    * \ twists 45 degrees   *
* <DEL>ete, alter massive areas     * Cursor arrows move    *
* O(utput picture into disk file    * cursor per pen state, *
* G(et picture form disk file Q(uit * twist & repetition    *
*-----------------------------------*-----------------------*
```

## Foto File Structure

Of the 19 logical blocks of a .FOTO file generated by GREDIT, 18 and 3/4 contain the 320 by 240 bit array used to create the picture. The remaining space contains the information on the GREDIT environment. The sources for GREDIT are provided to

1) Define the FOTO file structure.

2) Provide Pascal procedures for direct graphics management.

3) Illustrate general Pascal programming procedures (especially for implementation dependent code).

The following program is a further demonstration. It will load a file 'PIXFILE.FOTO', display it and copy the contents of the data area into working variables. The TYPE and VAR declarations are the same used in GREDIT to create the file. There are several example .FOTO files available in the Software Kit.

```
PROGRAM INSPECT;
TYPE
  INKCOLORS = (INVISIBLE,WHITE,ERASER,COMPLEMENT)
  QUAD = 0..3;
  TERAKSCREEN = RECORD CASE QUAD OF
  0:  (I: PACKED ARRAY [0..2391] OF PACKED ARRAY [0..319]
                                 OF BOOLEAN);
  1:  (C: PACKED ARRAY [0..9599] OF CHAR);
                                (*THE PICTURE...*)
  2:  (D: PACKED RECORD      (*...BY FOUR DIFFERENT NAMES*)
          PIX: PACKED ARRAY [0..9599] OF CHAR;
          X,Y INTEGER;                   (*CURSOR LOC'N*)
          MARK: PACKED ARRAY ['A'..'Z'] OF RECORD
                      (*LETTERED MEMORY ARRAY*)
           X: INTEGER        (*32767 -> INVALID AS X,Y *)
           Y: INTEGER        (*NUMBER OR 32767 ->
                                  INVALID AS NUMBER*)
           END;
           ANGLE: INTEGER;        (*HEADING*)
           PEN: INKCOLORS END);   (*PENSTATE*)
   3: (Q: PACKED ARRAY[0..4863] OF INTEGER)
                               (*19 BLKS FOR BLK I/O*)

END; (*TERAKSCREEN*)

VAR
 X,Y,ANGLE: INTEGER;         (*COORDINATES OF A DOT*)
 PEN: INKCOLOR;
 F:        FILE;
 SCREEN:   TERAKSCREEN;       (*THE GRAPHIC UNIVERSE*)
 MARK:     PACKED ARRAY ['A'..'Z'] OF RECORD
            X: INTEGER;   (*BOTH X&Y=32767->INVALID AS X,Y PAIR*)
            Y: INTEGER    (*X=32767 BUT Y<32767->VALID NUMBER*)
           END;

BEGIN
  OPENOLD(F,'PIXFILE.FOTO');         (*PERMANENT NAME FOR DEMO*)
  IF BLOCKREAD(F,SCREEN.Q[0],19)<>19
     THEN WRITELN('*FILE READ ERROR*')
     ELSE
     BEGIN
       CLOSE(F);
       UNITWRITE(3,SCREEN.C[0],63);              (*TURN ON DISPLAY*)
       X:=SCREEN.D.X;                            (*GET X OF CURSOR*)
       Y:=SCREEN.D.Y;                            (*GET Y OF CURSOR*)
       MARK:=SCREEN.D.MARK;                      (*LOAD MARK ARRAY*)
       ANGLE:=SCREEN.D.ANGLE;                    (*LOAD HEADING*)
       PEN:=SCREEN.D.PEN;                        (*LOAD PENSTATE*)
       READLN(X)
       END;
END.
```

# SECTION 7

# PROGRAMMING

In this section a sample Pascal program is presented with complete directions for entering and compiling it, fixing the error that was purposely placed in it, recompiling it, and running it.

## 7.1 Creating a Workfile

| Step | User Input | System Response/Comments |
|---|---|---|
| 1. | Boot the system using working copy | |
| 2. | F | Invokes Filer |
| 3. | N | Clears current workfile (SYSTEM.WRK. TEXT). If it already exists, prompt will appear:<br><br>Throw away current file? |
| 4. | Y | Y = Yes |
| 5. | Q | Q = Quit (to exit Filer) |

## 7.2 Editing the Workfile

| Step | User Input | System Response/Comments |
|---|---|---|
| 1. | E | Invokes Editor |
| 2. | <CR> | Indicates no workfile is currently present. |
| 3. | I | To insert characters into the file |
| 4. | <CR> | After each line. The first line of the program should begin in column one. The series of indentations shown are optional but make program logic easier to follow. Once a line is indented, the next line will begin at the same column To type further to the left, depress <BACKSPACE>. <BACKSPACE> key can also be used to correct an error; back up, erase the error and type in the correct data. The program entered should be identical to Figure 5-1. |
| 5. | <EXT> | To end insert mode |
| 6. | Q | Q = Quit (to exit Editor) |
| 7. | U | U = Update (to update workfile) |

```
    PROGRAM EXAMPLE;
{ DECLARATION OF VARIABLE TYPES }
VAR NEWNO, ANSWER, SUM, NUM : REAL;
    OPERATOR : CHAR;

    EXFILE : TEXT;
    I, N : INTEGER;
BEGIN
    WRITE ('HOW MANY EQUATIONS? ');
READLN (N);
    { OPEN FILE FOR OUTPUT }
    REWRITE (EXFILE, "EXFILE.TEXT');

    FOR I := 1 TO N DO

       BEGIN
       ANSWER := 0;
       OPERATOR :=
       REPEAT
          READ (NEWNO);
          CASE OPERATOR OF
              '+' : ANSWER : = ANSWER + NEWNO;  { ADD }
              '-' : ANSWER : = ANSWER - NEWNO;  { SUBTRACT }
              '*' : ANSWER : = ANSWER              * NEWNO;  { MULTIPLY }
              '/' : ANSWER : = ANSWER / NEWNO   { DIVIDE }
          END;
          READ (OPERATOR)
       UNTIL OPERATOR = '=';
       { WRITE OUT TO FILE }
       WRITELN (EXFILE, ANSWER)
       END;
    { CLOSE THE FILE AND MAKE IT PERMANENT }
    CLOSE (EXFILE, LOCK);
    { OPEN FILE AGAIN, FOR INPUT THIS TIME }
    RESET (EXFILE, ;EXFILE.TEXT;);
    SUM := 0;
    {READ VALUES IN UNTIL WE COME TO END OF FILE MARKER }
    WHILE NOT EOF (EXFILE) DO
       BEGIN
       READ (EXFILE, NUM);
       SUM := SUM + NUM
       END;
    ANSWER := SUM / N;
    WRITELN ('AVERAGE IS', ANSWER);
    { CLOSE THE FILE AGAIN }
    CLOSE (EXFILE)
END.
```

Figure 7-1. Pascal Program Example

## 7.3 Editing the Workfile to Correct an Error

| Step | User Input | System Response/Comments |
|---|---|---|
| 1. | | The first line at the top of the screen gives an explanation of the error the compiler encountered |
| 2. | <SP> | To continue. The error is the lack of a semicolon at the end of the line above the line on which the cursor is positioned. |
| 3. | Using arrow keys, move cursor to end of that line | The cursor should be after the last character on the line.<br><br>(OPERATOR := 't' |
| 4. | I | To insert |
| 5. | ';' | To put in semicolon |
| 6. | <EXT> | To exit insert mode. |
| 7. | Q | Q = Quit (to exit Editor) |
| 8. | U | U = Update (to update workfile) |

## 7.4 Compiling the Workfile

| Step | User Input | System Response/Comments |
|---|---|---|
| 1. | C | To compile the workfile again. This time compilation should be successful and a message beginning "Smallest available space..." should appear. |

## 7.5 Running the Program

This program will calculate the results of some specified number of arithmetic expressions and then give the average of those results. Follow these steps to run the program.

| Step | User Input | System Response/Comments |
|---|---|---|
| 1. | R | "Running..." and then "HOW MANY EQUATIONS" will appear. |
| 2. | Number of equations program is to calculate | |
| 3. | <CR> | Program waits for the entry of the first equation. An example: "1+2. 3-5*. 42+ .01/2=" * means multiple, / means divide. When = is typed, program will print out result.<br><br>Program will wait for next equation entry. As before result will print when = is typed.<br><br>When desired number of equations have been entered, the average of all results will automatically display. |

## 7.6 Printing under the Terak/UCSD Pascal System

To write to the printer from the Pascal program, the printer must be connected to serial unit #1. Data Terminal Ready (DTR) protocol and XON/XOFF protocol are observed by the printer handler.

### 7.6.1 Printing with Pascal

Below is a sample Pascal program which will write to the printer.

```
PROGRAM print;
VAR
   fp: TEXT;
BEGIN  (*print *)
   rewrite (fp,  'PRINTER:');
   writeln (fp, 'This line will be printed on the printer.');
   close  (fp, LOCK)
END.
```

# APPENDIX A

## A.1 UCSD Segments and Units

Since the release of Version 1.5 in early 1979, UCSD Pascal and the UCSD p-System have supported several useful extensions to Standard Pascal that assist modular compilation and/or execution of programs.

### A1.1 External Procedures

Programs may declare procedures or functions EXTERNAL in the same way they may be declared FORWARD (i.e., a full first procedure/function declaration line is required). The program cannot be run until it has been L(inked; the linker expects to find the missing code in the SYSTEM.LIBRARY. EXTERNAL procedures are reserved for assembly language procedures. The entire process is described in the UCSD documentation.

The manner in which parameters are passed warrants clarification. The called routine will find the following items on the stack, in descending (normal "POP xx") order:

1) The return address (this must be saved).

2) Two words of zeroes that reserve space for the results of the function if the called routine is a function. These two words are skipped if the routine is a procedure.

3) The parameters being passed to the routine, in reverse order (i.e., the last declared parameter is popped off first).

### A1.2 Units

A separate UNIT can be constructed which contains text to perform related but functionally distinct processes within a program. This is then compiled separately and the calling program merely states "USES <unitname>;" at the second line of the program (right after the main program declaration). Normally, the code from the correctly compiled UNIT should be incorporated into the SYSTEM.LIBRARY where it is available to the linker at execution or run time. Any constant, type, variable, procedure or function declared in the INTERFACE portion of the unit is read first at the time the Pascal main program is compiled.

These declarations are copied verbatim into the text of the main program by the compiler. It is therefore important to avoid declaring the same constants, types, etc., within the main program even if used; an error will result because these variables are already there. (If this is unclear, set up a UNIT and a Pascal main program that uses it, then examine a listing of the main program after compilation.)

## A1.3 Segments

Within the main program, it is possible to declare certain procedures or functions as SEGMENT procedures/functions. The compilation process is unchanged, but when the program is actually run, the declared procedures are drawn into memory only when required and are overwritten by data or code at other times. This approach is essential if there is a complicated program that uses substantial memory and has one or more long procedures (most commonly an initialization routine) that are rarely used. Unlike some other Pascal implementations, all other code (not explicitly declared as a SEGMENT procedure/function) is always memory resident while the program is running. A program may have up to 255 declared segments. A unit may have up to 16 declared segments.

There is no simple way for one program to load another program and then pass execution. Feeding data directly to the program execution section would require compiling the routine (usually simply a menu) at the 0th level with the (*$U-*) compiler command. This may be important in that many business programs use separate procedures driven by a menu and the present Pascal system limits the user to six procedure segments.

Although external UNITS are given new segment numbers by the compiler, they are always resident in memory at run time. To have code in a unit work as a SEGMENT PROCEDURE or FUNCTION, the source must be included in the main file and declared a SEGMENT PROCEDURE or FUNCTION.

# APPENDIX B

## B.1 Emulator Control Codes and Escape Sequences

The characteristics of the software console display emulator (TK emulator) with the Pascal II.0 software system are listed in this section. A specialized version of the TK emulator is provided for use with the 8600 Color Graphics Computer System.

### B1.1 8510/a or 8510/b (Monochromatic) TK Emulator

The TK (monochromatic) emulator presents an "intelligent" terminal to the user. A cursor is displayed by the emulator as either code 177 (over spaces) or code 377 (over all others). The pattern stored in the 8510/a or 8510/b writeable character generator for the 377 character is updated as the video reverse of whatever character is being overlaid by the cursor. These two codes, 177 and 377, are therefore reserved for system use.

Control codes and escape sequences are recognized to support user control of the screen such as cursor addressing and field protection. The TK emulator provides normal or reverse scrolling, an audible click, invisible cursor and single axis cursor addressing.

Some of the single character control codes are useful for manual adjustment of the emulator state. The keyboard usually echoes all control codes to the emulator. Thus typing a <cntrl-L> (or cursor down arrow) key can cause code to be sent to the emulator: in this case, clearing the screen, blanking all graphics and unblanking all characters. This is also useful for <cntrl-D> (slow scrolling rate) and for <cntrl-E> (increase scrolling rate).

When a formfeed character is received, all graphics zones will be blanked and all character zones unblanked. Otherwise, the zone blanking bits of the Video Control Register (VCR) are never modified by the TK emulator. User program changing of the VCR noise, beep and generator enable bits is allowed. See the Terak 8510 Users' Guide for a description of the video control registers.

Entering PROTECT mode is equivalent to shifting to the upper (alternate) character set for printing all characters. All characters are trimmed to seven bits, thus precluding direct printing of characters in the upper character set and providing a selective erasure (protected fields) feature.

Until FORMAT mode is turned on, protected (upper character set) characters on the screen are treated the same as all other characters. When FORMAT mode is entered, the screen is prevented from scrolling. A line-feed at the bottom line or a vertical line at the top line will cause the cursor to "wrap around" the screen. While in FORMAT mode, the cursor will never overlay a protected (upper character set) character. It will be moved to the first available unprotected field. Neither EOL or EOS erase commands will affect protected characters. This allows preservation of a form, while erasing all data written onto the form. Note that PROTECT and FORMAT modes are usually exclusive, although both may be active at the same time.

When using cursor addressing, if either absolute coordinate is out of range (0..79 for X and 0..24 for Y), the current corresponding component of the cursor position will be used in its place. Thus, by sending 80 as a column position, the cursor can be positioned at an absolute column along the current row of the cursor. If both coordinates are out of range, the cursor will be blanked and all subsequent printing will be lost. The cursor can also be blanked by sending a CURSOR_OFF code. It is preferable to blank the cursor using the CURSOR_OFF code, as printing may continue with the cursor off.

Note that the cursor can be addressed into row 24. This is the 'hidden' row of characters supported by the 8510 hardware to allow display of 25 rows of characters while the screen is scrolling. The cursor can only be be addressed into row 24. All other cursor motion will skip between rows 23 and 0. When the cursor is in row 24, if scrolling is commanded by either linefeed or a vertical tab, the row will not be cleared before it scrolls into sight. This allows a user program to support scrolling of the display.

In the following emulator codes, "scroll mode" indicates that the screen is allowed to scroll upward or downward to accomodate the cursor motion. "Block mode" indicates that the scrolling is inhibited and that certain cursor motions must wrap around the screen between top and bottom rows or right and left columns of the screen.

## SINGLE CHARACTER COMMAND SEQUENCES

| DECIMAL CODE | NAME | FUNCTION |
|---|---|---|
| 4 | DECSCR | Decrease scrolling rate. |
| 5 | INCSCR | Increase scrolling rate. |
| 6 | CLICK | Emit a short click through the speaker. |
| 7 | BELL | Emit a 130 us, 780 Hz tone. |
| 8 | BAKSP | Move cursor left, print space. |
| 9 | TAB | Move cursor right to multiple of 8 columns. |
| 10 | LF | Cursor down (scrolls if not in FORMAT mode). |
| 11 | VT | Cursor up (scrolls if not in FORMAT mode). |
| 12 | FF | Complete reset and erase of screen. |
| 13 | CR | Cursor to column 1, same row. |
| 14 | SO | Print in upper case character set (ESC-]). |
| 15 | SI | Print in lower case character set (ESC-[). |
| 21 | CRSOFF | Do not display cursor. |
| 22 | CRSON | Display cursor. |
| 24 | CAN | Cancel escape sequence. |

## DOUBLE CHARACTER COMMAND SEQUENCE

| CODE | NAME | FUNCTION |
|---|---|---|
| ESC A | CRSUP | Cursor up one row. |
| ESC B | CRSDN | Cursor down one row. |
| ESC C | CRSRT | Cursor right one column. |
| ESC D | CRSLT | Cursor left one column. |
| ESC E | CLEAR | Clear unprotected fields to spaces and home cursor. Does not affect graphics. |
| ESC H | HOME | Cursor to column 0, row 0. |
| ESC J | ERSEOS | Clear unprotected fields (to spaces) from cursor to end of screen. |
| ESC K | ERSEOL | Clear unprotected fields (to spaces) from cursor to end of line. |
| ESC ] | PRTON | Flag all following printed characters as protected. Display them in video reverse, depending upon the character set. Implies printing in the upper half (upper 96 of 192) of the character set. Note that characters are not protected from overstriking or erasing until FORMAT mode is on. FORMAT and PROTECT modes are independent. |

| CODE | NAME | FUNCTION |
|---|---|---|
| ESC [ | PRTOFF | Exit PROTECT mode. Implies printing in the lower half (lower 96 of 912) character set. |
| ESC W | FMTON | Exit SCROLL mode. Protect all flagged as protected. Enter BLOCK mode. |
| ESC X | FMTOFF | Exit BLOCK mode. Allow modification of protected characters. Enter SCROLL mode. |

## QUADRUPLE CHARACTER COMMAND SEQUENCES

NOTE: ESC = 27 decimal, F is the second character. Third and fourth characters are, respectively, the row and column positions in excess 32 decimal notation.

| CODE | NAME | FUNCTION |
|---|---|---|
| ESC F | CRSAD | Preamble to cursor addressing. Next two characters Y+32, X+32. Address origin is X=0, Y=0. If either X or Y is out of range, outside 0..24 of 0..79, respectively, the current X or Y coordinate of the cursor will be used in its place. If both are out of range, all further printing will be lost until a command places the cursor onto the screen. Note that the cursor may be placed into the 25th (hidden) line. In this mode, scrolling will not clear new lines. |

## B1.2 8510/c (High Resolution System) TK Emulator

This emulator accepts characters from the hardware emulator data buffer of the 8510/c and places them into the GDC display buffer in a manner similar to a hard video terminal. A cursor is displayed as a video reverse block. Escape sequences are recognized to support cursor addressing, selective erasing, etc. This emulator models the 8510/b TK emulator (also the BEEHIVE model B-100), with the exception of the following: no blinking, no half intensity, no send screen command, no vertical smooth scrolling, no 25th row and no field protect (forms erasure); and with the inclusion of the following: reverse scrolling, horizontal panning, click, blankable cursor, and single axis cursor addressing.

When a FORM-FEED character is received, the character display will be cleared and the cursor homed. On receipt of a second FORM-FEED character in immediate sequence after the first, all graphics will be cleared. Otherwise, this emulator co-exists with a graphics library or user program access to the GDC by communicating ownership through the least significant two bits of the VCR. These two bits must be used by both the emulator and the graphics code to coordinate their use of the GDC.

In the following, "SCROLL MODE" indicates that the screen is allowed to scroll upward or downward to accommodate the cursor motion. Likewise, "PAN MODE" indicates that the screen is allowed to pan horizontally left or right to accommodate the cursor motion (within 128 columns). "BLOCK MODE" indicates that both scrolling and panning and certain cursor motions must wrap around between the screen borders.

A list of control codes and escape sequences appears on the following page.

## TK EMULATOR CONTROL CODES AND ESCAPE SEQUENCES

| CODE | NAME | FUNCTION |
|---|---|---|
| 6 | CLICK | Emit a short click thru the speaker. |
| 7 | BELL | Emit a beep. |
| 10 | BAKSP | Move cursor left; print space. |
| 11 | TAB | Move cursor right to mult of 8 columns. |
| 12 | LF | Cursor down (scrolls if not FORMAT mode). |
| 13 | VT | Cursor up (scrolls if not FORMAT mode). |
| 14 | FF | 1st: Erase and home character display and reset states. 2nd: Complete reset and erase of screen; clear graphics. |
| 15 | CR | Cursor to column 0, same row. |
| 16 | SO | Print in upper char set (see ESC-]). |
| 17 | SI | Print in lower char set (see ESC-[). |
| 20 | VIDREV | Toggle the video reverse display control bit. |
| 25 | CRSOFF | Do not display a cursor. |
| 26 | CRSON | Display cursor. |
| 30 | CAN | Cancel escape sequence. |
| 31 | PANON | Allow horizontal panning within columns 0..27. |
| 32 | PANOFF | Allow printing, cursor addressing, etc., only in columns 0..7. |
| ESC A | CRSUP | Cursor up one row. |
| ESC B | CRSDN | Cursor down one row. |
| ESC C | CRSRT | Cursor right one column. |
| ESC D | CRSLT | Cursor left one column. |
| ESC E | CLEAR | Clear and home; do not affect graphics. |
| ESC F | CRSADR | Preamble to cursor addressing... next two chars carry Y+32, X+32, where address origin is 0,0. If either Y or X are out of range (if Y is outside the range 0..23 or X is outside the range 0..79 - except in PAN mode where the Y range becomes 0..127) the current Y or X coordinate of the cursor will be used in its place. If both are out of range, all further printing will be lost until a command places the cursor into the screen. |

| CODE | NAME | FUNCTION |
|---|---|---|
| ESC H | HOME | Cursor to column 0, row 0. |
| ESC J | ERSEOS | Clear from cursor to end of screen. |
| ESC K | ERSEOL | Clear from cursor to end of line. |
| ESC ] | PRTOFF | Print in upper charset. |
| ESC [ | PRTON | Print in lower charset. |
| ESC W | FMTON | Exit SCROLL mode. Enter BLOCK mode. |
| ESC X | FMTOFF | Exit BLOCK mode. Enter SCROLL mode. |

RESERVED

# APPENDIX C

## C.1 8600 Color System Console Display Characteristics

The console display characteristics of the 8600 Color Graphics System are provided by the color operating system, GEMINI.M86. The control codes and operating sequences are outlined below.

### C1.1 Gemini Control Codes and Escape Sequences

Gemini presents an "intelligent" terminal to the user. Control codes and escape sequences are recognized to support user control of the screen such as cursor addressing. Gemini provides reverse scrolling, tones, invisible cursor and single axis cursor addressing. Some of the single character control codes are useful for manual adjustment of the emulator state. For example, typing a Control-L (or cursor down arrow) key will clear the console portion of the screen.

If either absolute coordinate is out of range (0..79 for X, 0..48 for Y) during cursor addressing, the current corresponding component of the cursor position will be used in its place. Thus, by sending 80 as a column position, the cursor can be positioned at an absolute row along the current column of the cursor. By sending 49 as a row position, the cursor can be positioned at an absolute column along the current row of the cursor.

The row, column addressing documented here is for the default character cell (8 x 10). Cursor addressing with alternate cell sizes is functionally the same, although limits will change. (The number of rows and columns is determined by cell size).

If both coordinates are out of range, the cursor will be blanked and all subsequent printing will be lost. The cursor can also be blanked by sending a CURSOR_OFF code. It is preferable to blank the cursor using this code, as printing may occur with the cursor off.

Note that the cursor can be addressed to Row 48. This is the "hidden" row of characters supported by the 8600 hardware to allow display of 49 rows of characters while the screen is scrolling. The cursor can only be addressed into Row 48. All cursor motion will skip between Rows 47 and 0. When the cursor is in Row 48 and scrolling is commanded by either a linefeed or a vertical tab, the row will not be cleared before it scrolls into sight. This allows a user program to support scrolling of the display.

In the following emulator control codes, "scroll mode" indicates that the screen is allowed to scroll upward or downward to accommodate the cursor motion. "Block mode" indicates that scrolling is inhibited and that certain cursor motions must wrap around between top and bottom rows, or right and left columns of the screen.

## SINGLE CHARACTER COMMAND SEQUENCES

| DECIMAL CODE | NAME | FUNCTION |
|---|---|---|
| 6 | TONE1 | Emit a short tone through the speaker. |
| 7 | TONE2 | Emit a 130 us, 780 Hz tone through the speaker. |
| 8 | BAKSP | Move cursor left; print space. |
| 9 | TAB | Move cursor right to multiple of 8 columns. |
| 10 | LF | Cursor down (scrolls if not in FORMAT mode). |
| 11 | VT | Cursor up (scrolls if not in FORMAT mode). |
| 12 | FF | Complete reset and erase of screen. |
| 13 | CR | Cursor to column 1, same row. |
| 21 | CRSOFF | Do not display cursor. |
| 22 | CRSON | Display cursor. |
| 24 | CAN | Cancel escape sequence. |

## DOUBLE CHARACTER COMMAND SEQUENCE

NOTE: ESC = 27 decimal, letters are the second character

| CODE | NAME | FUNCTION |
|---|---|---|
| ESC A | CRSUP | Cursor up one row. |
| ESC B | CRSDN | Cursor down one row. |
| ESC C | CRSRT | Cursor right one column. |
| ESC D | CRSLT | Cursor left one column. |
| ESC E | CLEAR | Clear to spaces and home cursor within the console zone. |
| ESC H | HOME | Cursor to column 0, row 0. |
| ESC J | ERSEOS | Clear (to spaces) from cursor to end of screen. |
| ESC K | ERSEOL | Clear (to spaces) from cursor to end of line. |

| CODE | NAME | FUNCTION |
|---|---|---|
| ESC W | FMTON | Exit SCROLL mode and enter BLOCK mode. |
| ESC X | FMTOFF | Exit BLOCK mode and enter SCROLL mode. |

## QUADRUPLE CHARACTER COMMAND SEQUENCE

Note: ESC = 27 decimal, F is the second character. Third and fourth characters are, respectively, the row and column positions in excess 32 decimal notation.

| CODE | NAME | FUNCTION |
|---|---|---|
| ESC F | CRSAD | Preamble to cursor addressing. Next two characters Y+32, X+32. Address origin is X=0, Y=0. If either X or Y is out of range, outside 0..24 of 0..79, respectively, the current X or Y coordinate of the cursor will be used in its place. If both are out of range, all further printing will be lost until a command places the cursor onto the screen. Note that the cursor may be placed into the 25th (hidden) line. In this mode, scrolling will not clear new lines. |

NOTE: When the <ctrl> and <shift> keys are simultaneously depressed, the number pad key (1-9 on the far right of the keyboard) are reserved for use by Terak and are not available for user application program use.

RESERVED

# APPENDIX D

## CURRENT REVISION LEVELS

| | |
|---|---|
| GEMINI | VERSION 2.5 |
| COLOR EDIT | VERSION 3.0 |
| CHEDIT | VERSION 3 |
| CHDEMO | VERSION 3 |
| CSLOAD | VERSION 1.2 |
| GREDIT | VERSION 01-06 |
| PRINTOUT | VERSION 02-02 |
| HELPFILE | VERSION 02-01 |
| DWBOOT | VERSION 1.0 |
| DWSETUP | VERSION 1.1 |

RESERVED

# APPENDIX E

## SYSTEM NOTES

Terak has a 4 channel serial interface PWB (the Quad Serial Interface). The following utilities are included in the p-System media kit to allow the user to set via software the parameters of each serial channel.

A total of 2 quad serial interface PWBs may be installed and these utilities allow for channels 0 through 7 to be set up as required.

There are three utilities provided for QSI support:

QSISET - loads internal table to QSI
QSIADJ - adjusts table inside QSISET
QSI - loads QSI directly from the keyboard

## Quad Serial Interface (QSI) Utility

### QSISET

When run, this program loads the data in the procedure calls to QSISET into the mode and baud registers of the Quad Serial Interface (QSI). These characteristics may be changed by editing directly, and reassembling the program. Also, this source could be included in a custom SYSTEM.STARTUP file to configure the QSI at bootstrap time. The procedure QSISET is only a conversion routine to translate the parameter list into the codes expected by the QSI. If desired, it may be trimmed to its essentials and included in a user program.

To simply adjust baud rates, edit the channel table at the end of this program, recompile it and execute it. QSI code can also be altered by using QSIADJ. It will always have to be executed to effect immediate change, and after rebooting the system. Note that QSISET wil ignore the absence of any QSI channels.

To conveniently set up the QSI at bootstrap time, rename the resulting QSISET.CODE file to SYSTEM.STARTUP and transfer it to the desired disk.

The default parameter setting of each QSI channel of this utility as it is shipped is the same as the QSI bootstrap: 9600 baud, 8 character bits, no parity and 1 stop bit.

### QSIADJ

Instead of editing the text file (QSISET.TEXT) and compiling, this menu type program is available to patch changes in Quad Serial Interface parameters held inside QSISET.CODE.

This program is the companion to QSISET.SAV. When run, QSIADJ will ask for the file name of an image of QSISET.SAV and will search for the QSI setup table. If found, it will be displayed in tabular form so the operator may conveniently change the parameters used by QSISET when it is run. QSIADJ does not directly address the interface, only the QSISET code file. Operation of QSIADJ is self-explanatory. If an immediate change to the Quad Serial Interface is needed, follow executing QSIADJ with executing QSISET or whatever image file was adjusted. Typically, it is named SYSTEM.STARTUP to effect QSI setup at boot time.

X(ecuting QSI.CODE is another method of setting QSI parameters. QSI changes the parameters immediately without saving a disk file and is therefore most useful for the temporary changes to the QSI parameters. When * (asterisk) appears, the user can type H for Help and then type the desired parameter changes. Use Control C (EXT) to exit.

The format of the command line and its operation are described below.

```
*UNITn:parameter=value
       or
*UNITn:parameter=value,parameter=value,...
       or
*HELP
```

where

n is the unit number, ranging from 0 to 7.
parameter is one of the following keywords:
RATE STOPBITS CHLENGTH PARITY
values depends on associated keyword. Valid combinations are:

| | |
|---|---|
| RATE:50 | PARITY:NONE (default) |
| RATE:75 | PARITY:EVEN |
| RATE:110 | PARITY:ODD |
| RATE:134.5 | |
| RATE:150 | CHLENGTH:5 |
| RATE:200 | CHLENGTH:6 |
| RATE:300 | CHLENGTH:7 |
| RATE:600 | CHLENGTH:8 (default) |
| RATE:1200 | |
| RATE:1800 | STOPBITS:1 (default) |
| RATE:2400 | STOPBITS:1.5 |
| RATE:3600 | STOPBITS:2 |
| RATE:4800 | |
| RATE:7200 | |
| RATE:9600 (default) | |
| RATE:19200 | |

The baud rate may be set independent of the other three parameters at any time. Since the other three are all the same write-only register, they must all three be set if any one is set. If only one or two parameters are specified on the command line, the remaining parameters will be set to the default values listed above.

For example, the command *UNIT3:CHLENGTH:7,PARITY:ODD will also set the number of stop bits to 1, since that is the default. The command *UNIT7:RATE=2400 will not affect the character length, parity or stop bit parameters.

If a non-existent is specified, a message will appear prompting the user to check the hardware configuration.

## Touchpanel Handler

The following routines are used to access the Touchpanel from Pascal.

FUNCTION TPENABLE(UNIT:INTEGER):BOOLEAN

This is a function that is passed to the SLU number that the Touchpanel is attached to and returns TRUE if the unit is selected and FALSE if the unit is not selected.

If the unit is 0, the Touchpanel is disabled. This must be done when the program is terminating.

NOTE: This function must be called before the Touchpanel is accessed. If the unit is greater than 7, the results will be unpredictable.

PROCEDURE INQTP(VAR X,Y, COUNT: INTEGER)

This procedure returns the x and y position of the last location the Touchpanel responded to.

Count is a the number of times the Touchpanel was touched since the last time INQTP was called. Count is -1 if TPENABLE has not been called or if TPENABLE has been called with a -1 parameter.