SOFTECH
MICROSYSTEMS

UCSD Pascal™

One Pascal For All Microcomputers

Installation Guide

MILWAUKEE COMPUTERS INC.

16235 W. RYERSON ROAD • NEW BERLIN, WI 53151

Installation Guide
MILWAUKEE COMPUTERS INC.

There is More Brewing in Milwaukee Than Just Beer

$995.00 (QTY 1)

MC-1000 SERIES COMPUTERS

Our new **MC-1000 Computer System** is proof that the age of affordability in computers has arrived. Thanks to single board design we can offer not only the **lowest priced** computer in its class but also the **most reliable.** ***For information and ordering call:***

(414) 784-2312

ALL STANDARD

- **64K Memory**
- **2-5¼" Disk Drives (225K Total Storage)**
- **Real Time Clock**
- **2-RS-232 Interfaces**
- **Parallel Printer Port (Centronics Type)**
- **Extensive Self Diagnostics (In Rom)**
- **2 Year Limited Warranty**

OPTIONS

- **Disk Capacity to 1.8 MBYTE**
- **UCSD* Pascal, Basic, Fortran**

*TM U. Of California

MILWAUKEE COMPUTERS INC.

16235 W. RYERSON ROAD ★ NEW BERLIN, WI 53151 ★ (414) 784-2312

Circle 313 on inquiry card.

Milwaukee Computers Inc.

16235 W. Ryerson Road, New Berlin, WI 53151 • (414) 784-2312

TWO YEAR LIMITED WARRANTY
FOR
MC-1000 SERIES COMPUTERS

WARRANTY — Milwaukee Computers Inc. warranties that all MC-1000 Series Computers shall be free from defects in materials and workmanship. Milwaukee Computers Inc. will repair any and all such defects provided the buyer or ultimate user ("Customer") notifies Milwaukee Computers Inc. of the defect within two (2) years from the date of delivery. (90 days on disk drives)

PROCEDURE — To have your MC-1000 Series Computer repaired under the terms of this warranty follow the following procedure.

1. Contact Milwaukee Computers Inc. by letter or telephone with a description of the hardware problem. Milwaukee Computers Inc. will at that time assign a 'Return Authorization Number'.

2. Return the unit in it's original packing material to Milwaukee Computers Inc. prepaid. Be sure to mark clearly on the outside of the container your assigned 'Return Authorization Number'.

3. Milwaukee Computers Inc. will repair or replace all defective parts within 24 hours of receipt. We will return the MC-1000 Series Milwaukee Computer prepaid.

Note: Your warranty card MUST be sent to us within 10 days of receiving your computer.

The Computer Mart

PERSONAL BUSINESS COMPUTERS AND SUPPLIES

5550 So. Packard Ave.
Cudahy, WI 53110
1-414-483-4023

SINGLE DRIVE SYSTEMS

MC-100 **795**
64K high speed RAM, real time clock, 2 RS-232 interfaces, one parallel printer port (Centronics type), self diagnostics in ROM and 102.4K disk storage on a single 5.25 disk drive (Note the MC-100 is for turnkey use only)

MC-200 **995**
Same as the MC-100 except disk storage is increased to 204.8K

MC-300 **1195**
Same as the MC-100 except disk storage is increased to 409.6K

MC-400 **1395**
Same as the MC-100 except disk storage is increased to 819.2K

DUAL DRIVE SYSTEMS

MC-1100 **995**
64K high speed RAM, real time clock, 2 RS-232 interfaces one parallel printer port (Centronics type), self diagnostics in ROM and 204.8K disk storage on dual 5.25" disk drives

MC-1200 **1195**
Same as the MC-1100 except disk storage is increased to 409.6K

MC-1300 **1495**
Same as the MC-1100 except disk storage is increased to 819.2K

MC-1400 **1795**
Same as the MC-1100 except disk storage is increased to 1638.4K

SOFTWARE

UCSD p-SYSTEM (tm) Version IV **95**
Operating system, file handler, interpreter, editors, assembler, debugger, as well as a UCSD Pascal (TM) symbolic debugger, linker, utilities and documentation package

UCSD Basic **95**
Basic Compiler and Runtime Unit (with Basic reference guide)

UCSD Pascal (tm) **145**
Pascal Compiler with documentation

Fortran-77 **145**
Fortran-77 Compiler and Runtime Unit (with reference guide)

(UCSD p-System & UCSD Pascal are trademarks of the Regents of U. of C.)

Milwaukee Computers Inc.

16235 W. Ryerson Road, New Berlin, WI 53151 • (414) 784-2312

TECHNICAL DESCRIPTION
MC-1000 SERIES COMPUTER SYSTEMS

Processor	: 6502
Reset	: On power up. Manual switch also provided.
Clock	: 1 MHz
Bus	: Internal. Expansion port provided.
Power	: +5 volts. +12 volts for disk drives only.

MEMORY

RAM : 64K utilizing 64K x 1 dynamic RAMS
63, 488 bytes usable.

ROM : 1896 bytes devoted to bootstrapping, SBIOS, and diagnostics.

Memory map :

Address (hex)	Function
0000 - F7FF	63,488 bytes RAM
F800 - F87F	128 bytes expansion bus
F880 - F897	24 bytes system I/O
F898 - FFFF	1,896 bytes ROM
	65,536 bytes total (64K)

INPUT/OUTPUT

2 RS-232 serial ports using standard DB-25 connectors. Baud rates switch selectable from 300 - 19,200 baud in multiples of two. Character format 8 bits, no parity, 1 start bit, 1 stop bit (can be altered).

1 Centronics standard parallel port using Centronics compatible connector and printer software.

1 expansion port TTL compatible using 1 DB-25 serial type connector. Lines provided include 8 data lines, 1 clock line, 1 read/write line, 1 port select line, 7 low order address lines, 2 interrupt lines, and 1 data direction line.

Realtime clock accurate within 0.2%. No battery backup provided.

DISK STORAGE

5.25 inch floppy, soft sectored, 512 bytes per sector.

Track format	: 5 sectors/track single density (FM)
	10 sectors/track double density (MFM)
Data transfer rate	: 15,625 characters/sec. (FM)
	31,250 characters/sec. (MFM)

Data per disk	**Model**	**Format**
102.4 K bytes	**MC-100, MC-1100**	Single sided, 40 tracks, FM
204.8 K bytes	**MC-200, MC-1200**	Single sided, 40 tracks MFM
409.6 K bytes	**MC-300, MC-1300**	Single sided, 80 tracks, MFM
819.2 K bytes	**MC-400, MC-1400**	Double sided, 80 tracks, MFM

UNPACKING AND SET-UP PROCEDURE

The first thing to do when you receive your MC-1000 series computer is to make copies of the disks. Obtain any 5-1/4 inch soft sectored disks (double density disks are recommended for the MC-1200 and MC-1300, and double density - double sided disks for the MC-1400). After following the power-up procedure, the p-System promptline should appear on the terminal. Type 'x'. The following should appear on the screen (your responses are shown after the question marks, and comments are shown in angle brackets). After each response, type a carriage return:

```
        Execute what file ? initdisk

    (At this point, remove the disk labeled 'BOOT1' from the disk
    drive and replace it with a blank disk.)

        Unitnumber (4 or 5) ? 4

        Starting track number ? 0

        Ending track number ? (Your response depends on which model
    you have; see below)

        Done initializing.
```

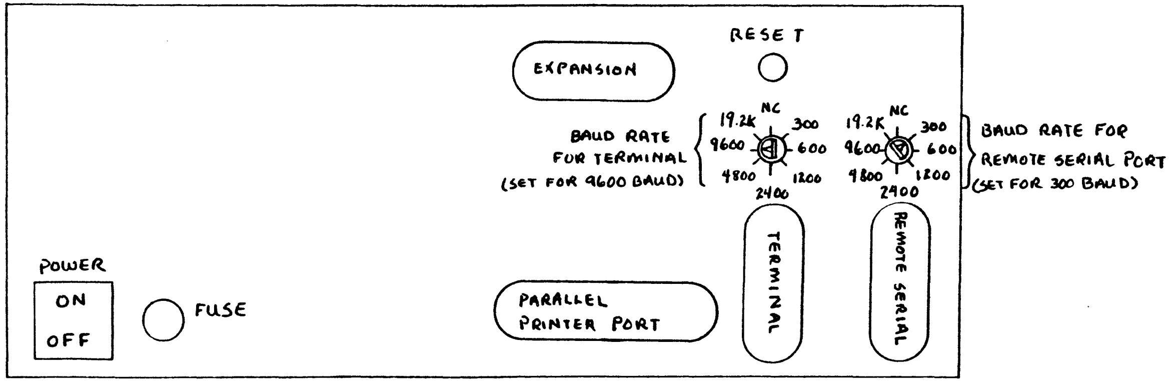
MODEL	Ending track#	Approx. time
MC-100 and MC-1100	39	15 sec.
MC-200 and MC-1200	39	15 sec.
MC-300 and MC-1300	79	35 sec.
MC-400 and MC-1400	159	70 sec.

After the disk is initialized, the system promptline should appear. Type 'f' for F)iler. When the Filer promptline appears, type 't' for T)ransfer. The Filer prompts 'Transfer what file ? '. If you have a single drive system, type '#4:, #4:' and put the disk to be copied in the drive. If you have a dual drive system, type '#4:,#5:' and put the disk to be copied in the left hand drive and the blank (initialized) disk in the right hand drive. Now enter a carriage return. On a single drive system, the Filer will occasionally prompt you to exchange disks when necessary.

If you wish to be able to bootstrap the new disk, you must use the 'booter' utility. Q(uit the Filer, then type 'x'. In response to 'Execute what file ?', type 'booter'. On a dual drive system, copy the bootstrap from unit 4 to unit 5. On a single drive system, specify from unit 4 to unit 4. The program will ask you to exchange disks if necessary.

DIAGRAM 'A'

'BACK OF MC-1000 SERIES'



Pinouts for I/O Board

S-232 (Terminal)	2nd RS-232 (Remote)	Parallel Port	Expansion Port
1. GND	1. GND	1. STROBE^	1. GND
2. Rx data	2. Tx data	2. D0	2. NMI^
3. Tx data	3. Rx data	3. D1	3. IRQ^
4. CTS^	4. RTS^	4. D2	4. Bus Enable
5. RTS^	5. CTS^	5. D3	5. R/W
6. +5V	6. NC	6. D4	6. 02 CLK
7. GND	7. GND	7. D5	7. GND
8. +5V	8. NC	8. D6	8. Data Dir.
9. NC	9. NC	9. D7	9. D0
10. NC	10. NC	10. ACK^	10. D1
11. NC	11. NC	11. BUSY	11. D2
12. NC	12. NC	12. PAPER	12. D3
13. NC	13. NC	13. SELECT	13. D4
14. NC	14. NC	14. NC	14. D5
15. NC	15. NC	15. NC	15. D6
16. NC	16. NC	16. GND	16. D7
17. NC	17. NC	17. GND	17. NC
18. NC	18. NC	18. GND	18. A0
19. NC	19. NC	19. GND	19. A1
20. NC	20. +5V	20. GND	20. A2
21. NC	21. NC	21. GND	21. A3
22. NC	22. NC	22. GND	22. A4
23. NC	23. NC	23. GND	23. A5
24. NC	24. NC	24. GND	24. A6
25. NC	25. NC	25. GND	25. NC
		26. GND	
		27. GND	
		28. GND	
		29. GND	
		30. GND	
		31. PRIME^	
		32. FAULT^	
		33. NC	
		34. NC	
		35. NC	
		36. GND	

Note

e ^ symbol denotes
egative logic.

POWER-UP PROCEDURE

(1) Turn on the terminal. Some time may be required for the terminal to warm up, depending on model (note that on some terminals, you must press a key on the keyboard at this point or the system will not bootstrap).

(2) Place a bootstrap diskette (such as the supplied disk labeled "BOOT1") into the left disk drive (drive A).

(3) Turn on the computer. A short message will appear on the terminal, and then the system will begin to bootstrap.

(4) After 45 to 60 seconds, the UCSD p-System welcome message should appear on the terminal. Refer to the UCSD p-System Users Manual for further information on the operating system software.

TEST MODE

The test mode of the MC-1000 series computers is entered by simply turning on the power (or pressing the reset button) when there are no disks in the drives. There are several single key commands available in the test mode:

A - Turns on drive A motor and designates drive A as the current drive.

B - Turns on drive B motor and designates drive B as the current drive.

S - Selects designated drive.

D - Deselects designated drive.

H - Homes drive head to track zero.

O - Steps drive head one step out (toward track zero).

I - Steps drive head one step in (away from track zero).

P - Prints a '*' for each index pulse when this key is held down (provided your terminal has auto-repeat).

W - If disk is write protected, prints 'Y', if not prints 'N'.

M - Performs a thorough memory test. The test takes about 8.5 minutes, and then recycles, printing a '*' on the terminal for each cycle. If an error occurs, a hexadecimal error pattern and address will be displayed on the console. If any error occurs, contact Milwaukee Computers.

E - Exits test mode (same effect as pressing the reset switch).

SOFTWARE

The standard p-System software is discussed in the supplied manuals. This section deals with programs written and supplied by Milwaukee Computers, Inc.

The INITDISK utility allows you to initialize a disk, or a portion of a disk. If you plan to create a new disk, you must initialize it first using the INITDISK utility. INITDISK asks you for the unitnumber of the disk drive containing the blank disk (beware of accidentally destroying already created system disks). Unit 4 is drive A (the left hand drive), and unit 5 is drive B (the right hand drive). The track ranges for different MC-1000 series computers are listed in the unpacking and setup procedures in this manual.

The RS-232 utility is intended to be used when the hardware specifications of your RS-232 compatible device will not allow using an 8 bit data word with 1 stop bit (for example, parity may be required), or when a baud rate of 75 or 150 is required. To achieve a baud rate of 75 or 150, set the baud rate select switch to 300 baud or 600 baud, respectively, and use the RS-232 utility to divide the effective baud rate by 4. Note that a UNITCLEAR on the RS-232 affected will reset it to the default (8 bits, 1 stop bit, baud rate divided by 1).

REALTIME is a Pascal program demonstrating the use of the real time clock. Enter a whole number from 1 to 32000 and the program will report the time required to count from 1 up to the specified value. Enter zero to end the program.

Compatibility: Milwaukee Computers MC-1000 series computers are, of course, completely software compatible with each other and with other computers using the UCSD p-System. Disks intended for the MC-1300 can also be read on the MC-1400. MC-1400 disks can be read on the MC-1300, provided no important information is recorded on the bottom side of the disk (block number greater than 789). In order to transfer files between other MC-1000 series computers, the serial port (REMOUT: and REMIN: under the p-System) may be used.

Note: A text file is included with the supplied software called 'SETUP.INFO.TEXT' which may help you to configure the system for your terminal.

The following is a list of block ranges for different MC-1000 series computers:

MC-100, MC-1100	0 - 194	(195 blocks)
MC-200, MC-1200	0 - 389	(390 blocks)
MC-300, MC-1300	0 - 789	(790 blocks)
MC-400, MC-1400	0 - 1589	(1590 blocks)

Milwaukee Computers MC-1000 Series Computers

Physical Memory Map

Address (hex)	
0000 - F7FF	62K RAM (Random Access Memory)
F800 - F87F	128 bytes expansion I/O
F880 - F881	6850 ACIA for first RS-232 (terminal)
F882 - F883	6850 ACIA for second RS-232 (remote)
F884 - F887	6821 PIA for Centronics parallel port
F888 - F88B	8253 Timer for disk motor delay and real time clock
F88C - F88F	6821 PIA for disk controller
F890 - F891	6852 SSDA for disk I/O
F892 - F897	unused I/O addresses
F898 - FFFF	1,896 bytes EPROM (for bootstrap, SBIOS, and test)

Logical Memory Map

Address (hex)	
0000 - 005F	Zero page memory reserved for SBIOS use
0060 - 007F	Reserved for use by 6502 assembled routines in RAM
0080 - 00FF	Zero page memory used by operating system
0100 - 01FF	6502 stack
0200	address of p-System Interpreter
0200 - F7F9	RAM dedicated to p-System
F7FA	6502 NMI vector points to this location
F7FD	6502 IRQ vector points to this location
F898	Address of ROM-based SBIOS
FFFA	6502 NMI jump vector
FFFC	6502 Reset jump vector (points to ROM bootstrap)
FFFE	6502 IRQ jump vector

Configuration of Parallel Port PIA

A side			B side		
PA0	I	Busy	PB0	O	Data 0
PA1	I	Acknowledge^	PB1	O	Data 1
PA2	I	Paper	PB2	O	Data 2
PA3	I	Select	PB3	O	Data 3
PA4	I	Fault^	PB4	O	Data 4
PA5	O		PB5	O	Data 5
PA6	O	Strobe^	PB6	O	Data 6
PA7	O	Prime	PB7	O	Data 7

Notes: The "I" or "O" stands for Input or Output.
The ^ symbol denotes negative logic.

Timer 0 of the 8253 programmable interval timer is currently unused. It has a 250 KHz input and may be used for any purpose desired.

SOFTECH
MICROSYSTEMS

A SUBSIDIARY OF SOFTECH

UCSD p-SYSTEM and UCSD PASCAL

A PRODUCT FOR MINI- AND MICRO-COMPUTERS

Version IV.0

INSTALLATION GUIDE

First edition: February 1981

SofTech Microsystems, Inc.
San Diego 1981

9494 Black Mountain Rd., San Diego, CA 92126 (714) 578-6105

TWX: 910-335-1594

III.5.5 Appendix E -- Sample SCREENTEST Log

This is a sample of a SCREENTEST log for a terminal that has some problems.

```
     1 test_DLE_expansion: expansion not happening properly
     2 test_DLE_expansion: expansion not happening properly
     3 test_DLE_expansion: expansion not happening properly
     4 test_DLE_expansion: expansion not happening properly
     5 test_DLE_expansion: expansion not happening properly
     6 test_DLE_expansion: expansion not happening properly
     7 test_DLE_expansion: expansion not happening properly
     8 test_DLE_expansion: expansion not happening properly
     9 test_keyboard: backspace key not correct
    10 tes_keyboard: line feed key not correct

***** End Diagnostic; 10 errors encountered.
```

11.4 The BOOTER Utility

BOOTER is a utility which transfers a bootstrap from one disk to another. In normal System use, bootstraps are copied only when an entire disk is copied using the T(ransfer command in the Filer. If you have created a System disk by T(ransferring individual files to a new disk, BOOTER must be used. On many hardware configurations, T(ransfer is incapable of copying a bootstrap, and BOOTER must be used in any case (if you have such hardware, you will be told about this situation in the supplemental literature).

The code for BOOTER is on the Utilities disk under the name BOOTER.CODE or ABOOTER.CODE. To copy a bootstrap, eX(ecute the codefile.

On PDP-11, LSI-11, and 9900 systems, ABOOTER prompts for the name of the disk on which the bootstrap will be written, and the name of a file from which the bootstrap is to be read (if only a disk name is given, the first two blocks of that disk will be copied). Only two blocks are transferred: from the input disk or input file to the first two blocks of Track 0 of the output disk.

On Z80, 8080, and 6502 systems, BOOTER prompts for two disk names, and copies all of Track 0 from the input disk to the output disk.

III. TERMINAL HANDLING

III.1 Introduction

You should read this chapter if you are new to the System, want to change or improve the way the System handles your terminal, or want to convert to a new variety of terminal.

The first thing you will be concerned with is SETUP, a utility program that modifies some terminal handling information stored in a file called SYSTEM.MISCINFO. The next thing to tailor is GOTOXY, an intrinsic Pascal UNIT within the Operating System that provides random addressing for your terminal's cursor. The System comes with its own defaults, but for more convenient or more efficient use of your console, you will want to specify your own characteristics. Changing SYSTEM.MISCINFO with SETUP does not require much knowledge or preparation. Changing the GOTOXY procedure requires a little more familiarity with your terminal, and a knowledge of UCSD Pascal.

To tailor terminal handling to your own needs, you will first run SETUP. SETUP creates a file called NEW.MISCINFO which contains information about your own terminal. You will then go into the Filer, change SYSTEM.MISCINFO to a backup file, and change the name of NEW.MISCINFO to SYSTEM.MISCINFO. After this, you reboot or I(nitialize: the new SYSTEM.MISCINFO is loaded into main memory, and your terminal is now controlled according to the information in this file. To see if you have run SETUP correctly, you might want to run the SCREENTEST diagnostic immediately, or you might want to wait until you have bound in a new GOTOXY. To create your own GOTOXY, you will write a Pascal procedure that does cursor addressing, create a codefile by C(ompiling it, and bind the codefile into the Operating System by using the Librarian utility. After binding, you should reboot, and then test the terminal handling by running SCREENTEST.

SCREENTEST checks that characters are being sent and received properly, and that the Screen Oriented Editor interface will work. If you encounter problems, it is easy to go back into SETUP and change your specifications, or modify your GOTOXY procedure and bind it in again.

If you don't feel confident, you might do a little more reading. Check your own terminal manual, and the following portions of the Users' Manual: the UNITWRITE intrinsic (Section VI.2.36), the introduction to the Screen Oriented Editor (Sections IV.0 and IV.1), and glance over the description of YALOE (Yet Another Line Oriented Editor, described in Chapter V). YALOE can be used on virtually any terminal, but the Screen Oriented Editor, which is more convenient and is usually used as the System editor, requires GOTOXY.

This chapter describes the care and feeding of SETUP, SCREENTEST, and GOTOXY. Users who wish to do more involved screen handling may use the

Operating System's Screen Control Unit, which is described in the Internal Architecture Guide.

III.2 SETUP

SETUP is provided as a System utility (on the Utilities disk) called SETUP.CODE. SETUP changes a file that contains details about your terminal, and a few miscellaneous details about the System in general. SETUP can be run, and the data changed, as many times as you desire. After running it, it is important to reboot (or I(nitialize) so that the System will start using the new information. It is also important to back up old data, at least until after you have run SCREENTEST, so that you can climb back out of any hole you dig for yourself!

The file that SETUP uses to store all of this information is called SYSTEM.MISCINFO. Each System initialization loads it into main memory. New versions of SYSTEM.MISCINFO are created by SETUP, and are called NEW.MISCINFO. Backups are created by renaming or copying SYSTEM.MISCINFO with the Filer.

SYSTEM.MISCINFO contains three types of information:

> Miscellaneous data about the System,
>
> General information about the terminal, and
>
> Specific information about the terminal's various control keys.

Section III.5.4 (Appendix D) contains a sample session with SETUP. You might look this over before you actually use the program.

III.2.1 Running SETUP

SETUP is a utility program, and is run like any other compiled program: type X for eX(ecute, and then answer the prompt with 'SETUP'<return>. It will display the word 'INITIALIZING' followed by a string of dots, and then the prompt:

```
SETUP: C(HANGE T(EACH H(ELP Q(UIT [D1]
```

(The '[D1]' is the SETUP version number, and may be different for your particular System.)

To invoke any command, just type its initial letter.

H(ELP gives you a description of the commands that are visible on any promptline where it appears.

T(EACH gives a detailed description of the use of SETUP. Most of it is concerned with input formats. They are mainly self-explanatory, but if this is your first time running SETUP, you should look through all of T(EACH.

C(HANGE gives you the option of going through a prompted menu of all the items, or changing one data item at a time. In either case, the current values are displayed, and you have the option of changing them. If this is your first time running SETUP, the values given are the system defaults. You will find that your particular terminal probably requires more sophisticated specifications.

Q(UIT has the following options:

H(ELP),

M(EMORY) UPDATE, which places the new values in main memory,

D(ISK) UPDATE, which creates NEW.MISCINFO on your disk for future use,

R(ETURN), which lets you go back into SETUP and make more changes, and

E(XIT), which ends the program and returns you to the System promptline.

Please note that if you have a NEW.MISCINFO already on your disk, D(ISK) UPDATE will write over it.

Section III.2.2 contains a detailed description of the data items in SYSTEM.MISCINFO. An abbreviated list of all the data items, together with the System-supplied defaults, is in Section III.5, along with a list of sample settings for a variety of terminals (Appendices A and B for this chapter).

When you use SETUP to change your character set, don't underestimate the importance of using keys you can easily remember, and making dangerous keys like BREAK, ESCAPE, and RUBOUT hard to hit.

Once you have run SETUP, you should always backup SYSTEM.MISCINFO under some other name (OLD.MISCINFO is one suggestion; you might want to name your backups according to different terminals, e.g., TTY.MISCINFO, IQ120.MISCINFO, VT52.MISCINFO, etc.), then change the name of NEW.MISCINFO to SYSTEM.MISCINFO and reboot or I(nitialize. It is indeed possible to update to memory alone, and go on using the System without rebooting, but the results may not always be what you wanted, and the backup security is more risky. In general, M(EMORY) UPDATE is a Q(UIT option that you will use only when experimenting. If you do get into a bind, remember that the current in-memory SYSTEM.MISCINFO can be saved by running SETUP and doing a D(ISK) UPDATE _before_ you change any data items.

When you reboot or I(nitialize, the new SYSTEM.MISCINFO will be read into main memory and its data used by the System, provided it has been stored under that name on the System disk (the disk from which you boot).

The only thing SETUP will not arrange for you, as far as terminal handling goes, is telling the System how to do random addressing for your terminal's cursor. This is a feature that the Screen Oriented Editor requires. To learn how to support this capability, see the section on GOTOXY.

III.2.2 Miscellaneous Notes for SETUP

The STUDENT bit, one of SYSTEM.MISCINFO's data items, should always be set to FALSE.

The HAS 8510A bit is always FALSE.

On the PDP-11, LSI-11, 8080, 9900, 6502, 6809, and Z-80 systems HAS WORD ORIENTED MACHINE is always FALSE.

HAS BYTE FLIPPED MACHINE is FALSE for all IV.0 systems except the 9900.

SETUP and the Manual refer to PREFIXED [DELETE CHARACTER]. This refers to the backspace function: read it as PREFIXED [BACKSPACE]. On most terminals it will be FALSE.

Your terminal should be set to run in full duplex, with no auto-echo.

Don't use terminal functions that do a "Delete and close up" on lines or characters -- not all terminals have these functions, and so they are supplied through the Screen Oriented Editor's software.

In general, if SETUP prompts for a feature that your terminal does not have, set the item to NUL (zero).

If you have a DEC VT-52 and a backspace won't move the cursor on the console, this is because you have KEY TO DELETE CHARACTER set to '_', the "rubout character". This is a printing character, so the Operating System does not echo a cursor move; the contents of memory *are* updated correctly. One workaround is to use the V(erify key to display the actual file contents, but to fix this for good use SETUP to change KEY TO DELETE CHARACTER to control-H or left-arrow -- BACKSPACE should be set to the same character as well.

III.2.3 The Data Items in SYSTEM.MISCINFO

The information in this section is very specific, and you may skip it on first reading. If you have a question about a certain data item, look in this section. Default values are shown, and sometimes our recommendations. When no suggested values are given, you should consult your own terminal's documentation. The items are ordered according to SETUP's menu. (See Section III.5.1, Appendix A.)

If you are using a hardcopy terminal or a storage screen rather than a CRT, you can ignore all the data items that are only used by the Screen Oriented Editor and leave them set to their defaults. In particular, if you are in doubt about a particular item, it is safest to leave it set to NUL. Always leave items set to NUL which concern features that your terminal does not have (ERASE LINE, for instance); the software will take care of these situations.

Please note that SETUP frequently makes a distinction between a character which is a key on the keyboard, and a character which is sent to the screen from the UCSD System; on some terminals, the same function may be performed by two different characters. On other terminals, the key pressed and the character sent for a given function may be the same, but in any case, when you run SETUP you must be explicit and answer all questions, even if the information is redundant.

There are a few characters which you cannot change with SETUP. These are CARRIAGE RETURN (<return>), LINE FEED (<lf>), ASCII DLE (control-P), and TAB (control-I). It is assumed that <return>, <lf>, and TAB are consistent on all terminals. ASCII DLE (data link escape) is used as a blank compression character. When sent to an output textfile, it is always followed by a byte containing the number of blanks which the output device must insert. If you try to use control-P for any other function, you will run into trouble. More information on DLE is given in the sections below on GOTOXY and SCREENTEST.

BACKSPACE ↑ 4 NAK

When sent to the screen, this character should move the cursor one space to the left. Default: ASCII BS.

EDITOR ACCEPT KEY

This key is used by the Screen Oriented Editor. When pressed, it ends the action of a command, and accepts whatever actions were taken. Default: ASCII NUL. Suggested: ASCII ETX (control-C or "Home").

EDITOR ESCAPE KEY

This key is used by the Screen Oriented Editor. It is the opposite of the EDITOR ACCEPT KEY - when pressed, it ends the action of a command, and ignores whatever actions were taken. Default and Suggested: ASCII ESC (control-[).

EDITOR EXCHANGE-DELETE KEY ETB ↑W

This key is also used by the Screen Oriented Editor. It operates only while doing an eX(change, and deletes a single character. Default: ASCII US (control-_).

EDITOR EXCHANGE-INSERT KEY ENQ ↑E

Like the EDITOR EXCHANGE-DELETE KEY, this only operates while doing an eX(change in the Screen Oriented Editor: it inserts a single space. Default: ASCII RS (control-^).

ERASE LINE

When sent to the screen, this character erases all the characters on the line that the cursor is on. Default: ASCII NUL.

ERASE SCREEN FF ↑L

When sent to the screen, this character erases the entire screen. Default: ASCII NUL.

ERASE TO END OF LINE

When sent to the screen, this character erases all characters from (and including) the current cursor position to the end of the same line. Default: ASCII NUL.

ERASE TO END OF SCREEN

When sent to the screen, this character erases all characters from (and including) the current cursor position to the end of the screen. Default: ASCII NUL.

HAS 8510A

May be TRUE or FALSE. Should be TRUE if and only if your hardware system is a Terak 8510a. Default: FALSE.

HAS BYTE FLIPPED MACHINE

May be TRUE or FALSE. On PDP-11, LSI-11, 8080, Z-80, and 6502 processors this bit is FALSE. On the 6800, 9900, and the GA440 system, it is TRUE. In general, it is TRUE only for implementations in which the IPC (Instruction Program Counter) is segment-relative. Default: FALSE.

HAS CLOCK TRUE

May be TRUE or FALSE. If your hardware has a line frequency (60 Hz) clock module, such as the DEC KW11, setting this bit TRUE will allow the Pascal system to optimize disk directory updates. It also allows you to use the TIME intrinsic: see Section VI.2 in the Users' Manual. If your hardware doesn't have a clock this must be FALSE. (Adaptable System users must write their own clock-handler; until it is installed, this item must be FALSE.) Default: FALSE.

HAS LOWER CASE

May be TRUE or FALSE. It should be TRUE if you do have lower case and want to use it. If you seem stuck in upper case even if this bit is TRUE, remember there is a soft alpha-lock: see KEY TO ALPHA LOCK. Default: FALSE.

HAS RANDOM CURSOR ADDRESSING TRUE

May be TRUE or FALSE. If your terminal is not a CRT, this should be FALSE. Default: FALSE.

HAS SLOW TERMINAL

May be TRUE or FALSE. When this bit is TRUE, the system's promptlines and messages are abbreviated. It is suggested that you leave this set at FALSE unless your terminal runs at 600 baud or slower. Default: FALSE.

HAS WORD ORIENTED MACHINE

May be TRUE or FALSE. If sequential addresses on your processor reference sequential 16 bit words, this should be TRUE. For PDP-11, LSI-11, 8080, Z-80, 9900, 6800, and 6502 systems, this should be FALSE. For the GA440 system it should be TRUE. Default: FALSE.

KEY FOR BREAK STX ↑P

When this key is pressed while a program is running, the program will terminate immediately with a runtime error. Default: ASCII NUL. Suggested: a key that is difficult to hit accidentally.

KEY FOR FLUSH SYN ↑V

This key may be pressed while the System is sending output (writing to the file OUTPUT). The first time it is pressed, output is no longer displayed, and will be ignored ("flushed") until FLUSH is pressed again. This can be done any number of times; FLUSH functions as a toggle. Note that processing continues while the output is ignored, so using FLUSH causes output to be lost. Default and suggested: ASCII ACK (control-F).

KEY FOR STOP

This key may be pressed while the System is writing to OUTPUT. Like FLUSH, it is a toggle. Pressing it once causes output and processing to stop, pressing it again causes output and processing to resume, and so on. No output is lost; STOP is useful for slowing down a program so the output can be read while it is being sent to the terminal. Default and suggested: ASCII DC3 (control-S).

KEY TO ALPHA LOCK

This character, when sent to the screen, locks the keyboard in upper case (alpha mode). It is usually a key on the keyboard as well. Default: ASCII DC2 (control-R).

KEY TO DELETE CHARACTER

Deletes the character where the cursor is, and moves cursor one character to the left. Default and suggested: ASCII BS (control-H or "Backspace").

KEY TO DELETE LINE

Deletes the line that the cursor is currently on. Default and suggested: ASCII DEL ("Rubout").

KEY TO END FILE

Sets the intrinsic Boolean function EOF to TRUE when pressed while reading from the System input files (either KEYBOARD or INPUT, which come from device CONSOLE:). Default and suggested: ASCII ETX (control-C or "Home").

KEY TO MOVE CURSOR DOWN LF ↑J
KEY TO MOVE CURSOR LEFT NAK ↑U
KEY TO MOVE CURSOR RIGHT ACK ↑F
KEY TO MOVE CURSOR UP SUB ↑Z

These keys are recognized by the Screen Oriented Editor, and are used when editing a document to move the cursor about the screen. If your keyboard has a vector pad, we suggest using those keys for these functions. If you have no vector pad, you might select four keys in the same pattern (such as, for example, '.','K',';', and 'O', in that order) and use them as your vector keys, prefixing them or using the corresponding ASCII control codes. Default (in order): ASCII LF, ASCII BS, ASCII FS, ASCII US.

LEAD IN FROM KEYBOARD

On some terminals, pressing certain keys generates a two-character sequence. The first character in these cases must always be a prefix, and must be the same for all such sequences. This data item specifies that prefix. Note that this character is only accepted as a lead in for characters where you have set PREFIXED[<itemname>] to TRUE. An example of this is in Appendix B below. Default: ASCII NUL.

LEAD IN TO SCREEN

Some terminals require a two-character sequence to activate certain functions. If the first character in all these sequences is the same, this data item can specify this prefix. This item is similar to the one above. The prefix is only generated as a lead in for characters where you have set PREFIXED[<itemname>] to TRUE. An example of this is in Appendix B below. Default: ASCII NUL.

MOVE CURSOR HOME SOH ↑A

When sent to the terminal, moves the cursor to the upper left hand corner of the screen (position (0,0)). If your terminal doesn't have a character which does this, this data item must be set to CARRIAGE RETURN; you will not be able to use the Screen Oriented Editor. Default: ASCII CR ("Return").

MOVE CURSOR RIGHT ACK ↑F

When sent to the terminal, moves the cursor nondestructively one space to the right. If your terminal doesn't have this function, you will not be able to use the Screen Oriented Editor. Default: '!'.

MOVE CURSOR UP SUB ↑Z

When sent to the terminal, moves the cursor vertically up one line. If your terminal doesn't have this function, you won't be able to use the Screen Oriented Editor. Default: ASCII NUL.

NON PRINTING CHARACTER

The character that will be displayed on the screen when a non-printing character is typed or sent to the terminal while using the Screen Oriented Editor. Default and suggested: '?'.

PREFIXED [<itemname>] FALSE

If any two-character sequence must be generated by a key or sent to the screen, the System will recognize that if you set PREFIXED[<itemname>] to TRUE. See the explanaticns for LEAD IN FROM KEYBOARD and LEAD IN TO SCREEN. An example of the use of two-character sequences is given in Appendix B.

SCREEN HEIGHT

The number of lines in your display screen, starting from 1. If you are using a hardcopy terminal, this should be set to 0. Default: 24 (base ten).

SCREEN WIDTH

The number of characters in one line on your display, starting from 1. Default: 80 (base ten).

STUDENT

May be TRUE or FALSE. On IV.0 Systems, should always be FALSE. Default: FALSE.

VERTICAL MOVE DELAY 0

May be a decimal integer from 0 to 11. Many terminals require a delay after vertical cursor movements. This delay allows the movement to be completed before another character is sent. This data item specifies the number of nulls that the System sends to the terminal after every CARRIAGE RETURN, ERASE TO END OF LINE, ERASE TO END OF SCREEN, CLEAR SCREEN, and MOVE CURSOR UP. Default: 5 (base ten).

III.3 GOTOXY

When you have tailored SYSTEM.MISCINFO with SETUP, you should write your own GOTOXY. GOTOXY is a Pascal UNIT embedded in the Operating System. It provides random addressing for your terminal's cursor. There is a GOTOXY that is provided with the System we ship, (the source for this code, along with other examples, is in Appendix C below), but as it is a general routine for any terminal, it is not fast. When you create your own GOTOXY, you will write a Pascal procedure, compile it, then bind it into the Operating System using the utility LIBRARY.

If you are not yet ready to write your own GOTOXY, you should skip down to the next section, which describes SCREENTEST.

If you intend to do all your work on a line-oriented terminal, you never need to write a GOTOXY.

Before you write your own GOTOXY, you should understand the I/O intrinsic UNITWRITE, which is described in Section VI.2 of the Users' Manual. In Section III.5.3 (Appendix C) of this Installation Guide are a few sample versions of GOTOXY, including the source for the GOTOXY code which comes with the System, and the SAMPLEGOTO.TEXT that is also on your System disk. You should look this appendix over.

III.3.1 Writing Your Own GOTOXY

III.3.1.1 A Discussion

You may write GOTOXY using either YALOE or the Screen Oriented Editor, whichever you find more convenient.

The purpose and the calling protocol of GOTOXY are quite simple. The procedure is given two parameters, X and Y. They must be in that order, and they must be of type INTEGER. The procedure should position the terminal's cursor at co-ordinates (X,Y), where (0,0) is home (the upper left hand corner of the screen). That is all it should do.

To get your GOTOXY to run at all, there are a few things that are required.

First, the name of your unit must be GOTOXY. The name of the procedure itself must be something different.

Second, you must include the pseudo-comment {$U-}. This Compiler option allows you to use the predeclared name GOTOXY as the name of your unit -- it will become part of the Operating System. This comment must be the first line of your source code. If it does not look like one of the following lines:

```
(*$U-*)
{$U-}
```

... your GOTOXY will not compile. In particular, there must be no spaces within the comment, and the 'U' must be capitalized.

Finally, the code for GOTOXY should be compiled as a UNIT, as shown in the next section.

Your procedure should check that the values of X and Y are within bounds. . If they are off the screen, change them to a value that is on the screen (such as the nearest location along the border -- this is what all the sample procedures do).

You will need to move the cursor by a WRITE to the terminal, a repeated set of WRITEs within a loop, or a UNITWRITE of a vector. Using UNITWRITE is recommended: it can speed up your terminal handling by about 10%. (Although if you use UNITWRITE, you cannot redirect console output.)

To summarize, your GOTOXY should contain, in order:

1. The pseudo-comment '{$U-}',
2. In the program body, a check to make sure that X and Y are on the screen,
3. A section that fills an array with all the characters you must send to the terminal, and
4. The actual write to the terminal, preferably with UNITWRITE.

Please note: some terminals take a bias on X and Y. That is, for example, sending (X+32,Y+32) actually positions the cursor at (X,Y). If your terminal is capable of this, you should include these offsets in your procedure. This will eliminate any problems you might run into with the ASCII DLE (control-P) character, which is always interpreted as a blank-compression character. You don't want to send this value as a cursor control character. See the section below on SCREENTEST.

The following section contains a more detailed description of GOTOXY. Section III.5.3 (Appendix C) contains specific examples for a variety of terminals.

III.3.1.2 A Recipe for GOTOXY

This section walks you through a sample GOTOXY, and demonstrates the best way of writing a GOTOXY. To see some more specific examples, see Appendix C (Section III.5.3).

The sample program here is commented like a Pascal program.

```
{$U-}              { ALWAYS include this compiler directive. }
UNIT GOTOXY;

INTERFACE

PROCEDURE AGOTOXY(X,Y: INTEGER);


IMPLEMENTATION

PROCEDURE AGOTOXY;

CONST  TELL_LENGTH_MINUS_1 = 3,
        OFFSET = 32;
{ You may have to change these, depending on your terminal. }

VAR    TELL: PACKED ARRAY [0..TELL_LENGTH_MINUS_1]
              OF 0..255;

BEGIN
  IF X>79 THEN X:=79
    ELSE IF X<0 THEN X:=0;
  IF Y>23 THEN Y:=23
    ELSE IF Y<0 THEN Y:=0;
  {  This range-checking is necessary.  The actual
      screenwidth and height may be different for you. }

  { These first elements of TELL must contain
    the characters which tell your terminal to
    position the cursor at (X,Y):
    fill in the blanks...           }
  TELL[0] := ____;
  TELL[1] := ____;
  ...
  { The actual X and Y values are usually the
    last things in the array;
    the order may be different on your terminal. }
  TELL[TELL_LENGTH_MINUS_1 - 1]    := Y+OFFSET;
  TELL[TELL_LENGTH_MINUS_1]       := X+OFFSET;

  UNITWRITE(1,TELL,TELL_LENGTH_MINUS_1 + 1)
END {AGOTOXY};

END {UNIT GOTOXY}.
```

III.2 Binding GOTOXY

The first thing to do, once you have written your own GOTOXY, is to compile it to a codefile. Any filename will do, provided its suffix is .CODE. Choose a name you will remember.

A common error is incorrectly entering the comment '{$U-}'. If this is not the first line in your source file, if the comment contains spaces that are not shown in this manual, or any other variances, your GOTOXY will not compile. You will get the error message 'GOTOXY predeclared' when you try to compile.

You should also make sure that the STUDENT bit in SYSTEM.MISCINFO is set to FALSE -- otherwise GOTOXY binding will not work, and you will get the message "No proc in seg table" when you try to reboot the System.

III.2.1 Using LIBRARY to Bind GOTOXY

First, back up your System disk. If the binding works, all will be well, and you will have a functioning System with a new (and hopefully functioning) GOTOXY. If the binding does not work, your System may be destroyed. Make sure you have a backup.

The LIBRARY is a utility program which is shipped on the Utilities disk under the name LIBRARY.CODE. To run it, eX(ecute LIBRARY.

The first prompt LIBRARY gives you is:

```
        Output file? NEW.PASCAL
```

... the underlined portion is a sample response. Choose any unambiguous name that suits you -- this new output file will become the new Operating System if all goes well. Be sure you have enough room on your disk for the new System: most Systems are from 70 to 100 blocks long. If there is not enough room on your disk, either use the Filer's K(runch command to create more room, or use another disk with more room.

LIBRARY then asks:

Input file? MYGOTO.CODE

... the underlined portion is a sample response. This should be the file that contains your compiled GOTOXY procedure. It will be displayed in slot 0 of the input file. You must move it to a slot in the output file (this new slot must be greater than 15).

Type 'T'. The INTERFACE part of your unit will not be copied.

Type '0'. LIBRARY prompts:

Copy from slot 0?

... type a space. LIBRARY prompts:

Copy to which slot? 16

... respond with a number greater than 15 (as shown).

Now type 'N' for N(ew. This causes a repeat of the prompt:

Input file? SYSTEM.PASCAL

... type in the name of your Operating System, as shown. This is the new input file

Finally, type 'E' for E(very. This will cause all of the slots in SYSTEM.PASCAL to be transferred to the output file, except for GOTOXY, which will not be destroyed because it is already there.

Before using E(very, your screen should look more or less like this:

```
Library: N(ew, 0-9(slot-to-slot, E(very, S(elect, C(omp-unit, F(ill,?
[IV.0z]

Input file: SYSTEM.PASCAL
  0 u KERNEL      1481     9 u SCREENOP    918      18 u DEBUGGER   187
  1 s PRINTERR     695    10 s SEGSCINI    416      19 s EXTRALEX  4872
  2 s INITIALI    1358    11 u SOFTOPS     559      20 u SYSCMND    119
  3 s GETCMD      2779    12 u OSUTIL      511
  4 u HEAPOPS      314    13 u REALOPS     752
  5 u EXTRAHEA     736    14 u CONCURRE    140
  6 u EXTRAIO      772    15 s USERPROG   1549
  7 u PASCALIO     304    16 u FILEOPS    2146
  8 u STRINGOP     259    17 u GOTOXY       31

Output file: NEWSYS.CODE
  0                        9                        18
  1                       10                        19
  2                       11                        20
  3                       12
  4                       13
  5                       14
  6                       15
  7                       16 u GOTOXY       29
  8                       17
```

... note that there is a GOTOXY in the SYSTEM.PASCAL that is shipped. This will be abandoned by the E(very command, since you have already put a GOTOXY in the output file.

Typing 'Q' for Q(uit causes the changes you have made to be saved in your output file.

Once you are out of LIBRARY, use the Filer to change the name of SYSTEM.PASCAL to something like OLD.PASCAL, and NEW.PASCAL (or whatever you have called your new output file) to SYSTEM.PASCAL. Then bootstrap your System again; the new GOTOXY will be in effect.

If at any point while using LIBRARY, you think you have made a mistake, A(bort will exit without recording any changes. When modifying the Operating System, it is far better to be safe than sorry.

Note: While using LIBRARY on the Operating System, <u>never</u> move slot 0 or slot 15.

III.2.2 Problems

If your newly created System will not bootstrap at all, it may be because you moved the USERPROG segment when you used LIBRARY. USERPROG must be at slot fifteen in SYSTEM.PASCAL. Boot your System's backup, and try again.

If the System starts to boot, but halts with the message 'No unit in seg table', it may also mean that the STUDENT bit is on in your SYSTEM.MISCINFO file. The STUDENT bit must be FALSE when you compile your GOTOXY. Boot your System's backup, change the STUDENT bit to FALSE, recompile your GOTOXY, and use LIBRARY again.

For more information on LIBRARY, see Section VIII.5 in the Users' Manual.

Once LIBRARY has been successfully run, and the System successfully rebooted, you should run SCREENTEST to make sure the Screen Oriented Editor interface will work. SCREENTEST is described immediately below.

III.4 SCREENTEST

Now that you have changed your SYSTEM.MISCINFO with SETUP (or your GOTOXY, or both), you will want to test the results. SCREENTEST is a utility which accomplishes that. Like SETUP, it is largely self-explanatory. SCREENTEST checks that the Interpreter and Operating System are sending and receiving characters correctly, that the control keys are set up correctly, and that the Screen Oriented Editor will interface to the terminal as it is supposed to.

When you run SCREENTEST, it will display patterns on the screen and ask you if they are correct. You will need to be seated at your terminal while SCREENTEST is running; it takes roughly five minutes.

SCREENTEST will also output a report of errors to any file you specify. If you do encounter problems, you will need this report to help track them down, especially if you require assistance from your supplier's support group.

III.4.1 Running SCREENTEST

Type X for eXecute, and enter 'SCREENTEST'<return>. It will respond by displaying a heading, telling you that all questions must be answered with either 'Y' or 'N' (either upper or lower case; all other characters are ignored), and will then prompt you for the name of an error log file.

If you hit <return> instead of specifying a log file name, no error report will be generated. You may want to do this if you are running SCREENTEST for the first time and don't anticipate any problems. If you do have trouble, you can run it again, this time with a log. Sending the log to 'PRINTER:' may suit your needs if you have a hardcopy device, otherwise you can save it on a disk file named 'LOG.TEXT' or something similar. (The .TEXT suffix is necessary if you want to look at it with the Editor.)

If your terminal is set up correctly, you should be able to answer 'Y' to all of the yes/no questions that SCREENTEST asks. If there is any problem with the questions about individual characters, SCREENTEST will tell you immediately. The log file will also contain a record of all problems. A sample log is in Section III.5.5 (Appendix E).

III.4.2 Results of SCREENTEST

SCREENTEST consists of twelve individual tests. Their names follow:

```
test_basic
test_clr_screen
test_gotoxy
test_clr_line
test_erase_eol
test_etoeos
test_home
test_single_vectors
test_scroll
test_DLE_expansion
test_keyboard
test_normal_keys
```

Each of these tests may generate error messages. While the text of each error message is fairly clear, some further explanation follows. The error messages are grouped by the nature of the problems -- what you must check in order to solve them. They are further grouped under the name of the test that generates them. This information is included in the error log. If you find yourself at a loss and decide to consult Pascal Support, you will need to refer to this log.

III.5.2.1 Problems that can be Fixed by Changing SETUP

If you get any of these error messages, check your SETUP values. To the right of each error message listed below is a suggestion as to which key or character value might be in error. These suggestions won't always pinpoint your problem, but they will tell you what you should check first. It may be the case that changing SETUP does not fix your problem. Some special cases are described at the end of this section. If these don't cover your particular problem, you should probably ask for help.

```
test_clr_screen:

        screen not cleared              -> is ERASE SCREEN OK?
        cursor not left at (0,0) afterwards
                                        -> is MOVE CURSOR HOME OK?


test_clr_line:

        didn't clear enough - (x,y)
            (where x and y are the cursor co-ordinates)
                                        -> is ERASE LINE OK?
        Clearing one line affected another
                                        -> is ERASE LINE OK?


test_erase_eol:

        sc_erase_to_eol didn't work
                                        -> is ERASE TO END OF LINE OK?


test_etoeos:

        sc_eras_eos didn't work
                                        -> is ERASE TO END OF SCREEN OK?


test_home:

        cursor didn't go home
                                        -> is MOVE CURSOR HOME OK?
```

test_single_vectors:

sc_right didn't work	-> is MOVE CURSOR RIGHT OK?
sc_left didn't work	-> is BACKSPACE OK?
sc_up didn't work	-> is MOVE CURSOR UP OK?
sc_down didn't work	-> this shouldn't happen; call Pascal Support!

test_keyboard:

<key> not correct	-> is <key> OK? <key> means one of the following: KEY TO MOVE CURSOR DOWN KEY TO MOVE CURSOR LEFT KEY TO MOVE CURSOR RIGHT KEY TO MOVE CURSOR UP BACKSPACE EDITOR ACCEPT KEY EDITOR ESCAPE KEY KEY TO DELETE LINE KEY TO END FILE

test_normal_keys:

Can't type these - <list>	-> <list> means a list of any standard printing characters; this usually means that a standard character is being interpreted as a special key, which usually happens when HASPREFIX is incorrect -- it should be FALSE for a key which needs no prefix, or TRUE for a key which does need one; check your own terminal manual;

III.5.2.2 Problems that can be Fixed by Changing GOTOXY

test_gotoxy:

```
        gotoxy(0,0) did not go home
        gotoxy(screenwidth-1,screenwidth) not ok
        box not correctly drawn
        exhaustive_gotoxy_check: first pass not ok
        exhaustive_gotoxy_check: top line not ok
                                            -> all these problems relate to your
                                               GOTOXY procedure; if you find any
                                               discrepancies, you will have to
                                               change it; refer to the previous
                                               section in this document for a
                                               description of using GOTOXY,
                                               and to the first paragraph in
                                               the miscellaneous notes below;
```

III.5.2.3 Other Problems

test_basic:

not all characters written out

-> there is a problem with the Pascal system intrinsic UNITWRITE, or, if you are using the Adaptable System, with the SBIOS. You should call Pascal Support; disregard the rest of SCREENTEST's results until this particular problem is cleared up;

test_scroll:

sc_down at bottom didn't scroll properly

-> there is a note below about scrolling;

test_DLE_expansion:

expansion not happening properly

-> there is a problem in your Interpreter's terminal handling; this may be hardware-related; it is still possible to run with improper DLE expansion -- you may encounter off-by-one errors and the like in your output and your editing; this is the case with Terak systems; DLE is an ASCII character used as a blank-compression code to save space in output strings;

III.5.3 Miscellaneous Notes on SCREENTEST Problems

The System interprets an ASCII DLE or chr(16) (base ten) within a textfile as a blank-compression code (this is its standard use). It can lead to problems if GOTOXY ever writes out a chr(16) as an X or Y value. If you run into this problem, check whether your terminal can handle an offset on X and Y values, that is, whether sending it X+32 and Y+32 will position the cursor at (X,Y) (the value 32 is just an example). If so, this will fix your problem. If not, you will have to modify GOTOXY so it catches this situation; see above.

ERASE LINE will have difficulty if there are bugs in the screen emulator for memory-mapped screens. This is applicable primarily to Terak systems. In particular, Teraks have trouble with blank-compression sequences (DLE-expansions) of 64 or longer.

Some terminals will not scroll at all, or scroll two lines at a time. The IV.0' System's Screen Oriented Editor unfortunately cannot handle these terminals -- you must use YALOE for SYSTEM.EDITOR.

Use your judgement when interpreting the results of SCREENTEST: if something is reported as an error, but the Screen Oriented Editor performs to your satisfaction, do not worry about the SCREENTEST evaluation.

III.5 Appendix A -- SETUP Menu and Defaults

In the defaults shown below, 'T' means true and 'F' means false as per the input conventions in SETUP. The numbers shown are in base ten, literal characters are quoted, and ASCII abbreviations are used for nonprinting characters. When you use SETUP, these values are shown in several formats, so the meaning is clear. {Note: must add the eX(change INSERT CHAR and DELETE CHAR items.}

BACKSPACE	BS
EDITOR ACCEPT KEY	NUL
EDITOR ESCAPE KEY	ESC
EDITOR EXCHANGE-DELETE KEY	US
EDITOR EXCHANGE-ACCEPT KEY	RS
ERASE LINE	NUL
ERASE SCREEN	NUL
ERASE TO END OF LINE	NUL
ERASE TO END OF SCREEN	NUL
HAS 8510A	F
HAS BYTE FLIPPED MACHINE	F
HAS CLOCK	F
HAS LOWER CASE	F
HAS RANDOM CURSOR ADDRESSING	F
HAS SLOW TERMINAL	F
HAS WORD ORIENTED MACHINE	F
KEY FOR BREAK	NUL
KEY FOR FLUSH	ACK
KEY FOR STOP	DC3
KEY TO ALPHA LOCK	DC2
KEY TO DELETE CHARACTER	BS
KEY TO DELETE LINE	DEL
KEY TO END FILE	ETX
KEY TO MOVE CURSOR DOWN	LF
KEY TO MOVE CURSOR LEFT	BS
KEY TO MOVE CURSOR RIGHT	FS
KEY TO MOVE CURSOR UP	US
LEAD IN FROM KEYBOARD	NUL
LEAD IN TO SCREEN	NUL
MOVE CURSOR HOME	CR
MOVE CURSOR RIGHT	'!'
MOVE CURSOR UP	NUL
NON PRINTING CHARACTER	'?'
PREFIXED [DELETE CHARACTER]	F
PREFIXED [EDITOR ACCEPT KEY]	F
PREFIXED [EDITOR ESCAPE KEY]	F

PREFIXED [ED EXCH-DELETE KEY]	F
PREFIXED [ED EXCH-ACCEPT KEY]	F
PREFIXED [ERASE LINE]	F
PREFIXED [ERASE SCREEN]	F
PREFIXED [ERASE TO END OF LINE]	F
PREFIXED [ERASE TO END OF SCREEN]	F
PREFIXED [KEY TO DELETE CHARACTER]	F
PREFIXED [KEY TO DELETE LINE]	F
PREFIXED [KEY TO MOVE CURSOR DOWN]	F
PREFIXED [KEY TO MOVE CURSOR LEFT]	F
PREFIXED [KEY TO MOVE CURSOR RIGHT]	F
PREFIXED [KEY TO MOVE CURSOR UP]	F
PREFIXED [MOVE CURSOR HOME]	F
PREFIXED [MOVE CURSOR RIGHT]	F
PREFIXED [MOVE CURSOR UP]	F
PREFIXED [NON PRINTING CHARACTER]	F
SCREEN HEIGHT	24
SCREEN WIDTH	80
STUDENT	F
VERTICAL MOVE DELAY	5

III.5.2 Appendix B -- Sample SETUPs for Some Terminals

Here is a list of SYSTEM.MISCINFO data items followed by some sample values for four popular terminals. Some items in the SETUP menu haven't been included; these are data items that refer to your processor configuration, not your terminal.

These examples represent what we consider reasonable layouts for a few different keyboards, but we don't guarantee that they work for your particular hardware, or match your individual taste.

Terminals:	LSI ADM-3A	HAZELTINE 1500/1510	SOROC IQ120	HEATH H19
Data Items:				
BACKSPACE	left-arrow	backspace	ctrl-H	ctrl-H
EDITOR ACCEPT KEY	ctrl-C	ctrl-C	home	ctrl-C
EDITOR ESCAPE KEY	esc	esc	esc	ctrl-[
ERASE LINE	NUL	NUL	NUL	l
ERASE SCREEN	ctrl-Z	ctrl-\	'*'	E
ERASE TO END OF LINE	NUL	ctrl-O	T	K
ERASE TO END OF SCRN	NUL	ctrl-X	Y	J
HAS LOWER CASE	TRUE	TRUE	TRUE	TRUE
HAS RAND CURS ADDR	TRUE	TRUE	TRUE	TRUE
HAS SLOW TERMINAL	FALSE	FALSE	FALSE	FALSE
KEY FOR BREAK	ctrl-B *	break **	break	break
KEY FOR FLUSH	ctrl-F	ctrl-F	ctrl-F	ctrl-F
KEY FOR STOP	ctrl-S	ctrl-S	ctrl-S	ctrl-S
KEY TO ALPHA LOCK	ctrl-R	NUL	ctrl-R	ctrl-R
KEY TO DELETE CHAR	ctrl-H	backspace	l-arrow	ctrl-H
KEY TO DELETE LINE	rubout	shift-DEL	rubout	DEL
KEY TO END FILE	ctrl-C	ctrl-C	ctrl-C	ctrl-C
KEY TO MV CURS DOWN	ctrl-J	ctrl-K	d-arrow	B
KEY TO MV CURS LEFT	ctrl-H	backspace	l-arrow	D
KEY TO MV CURS RGHT	ctrl-L	ctrl-P	r-arrow	C
KEY TO MV CURS UP	ctrl-K	ctrl-L	u-arrow	A
LEAD IN FROM KEYBD	NUL	NUL	NUL	esc
LEAD IN TO SCREEN	NUL	~	esc	esc
MOVE CURSOR HOME	ctrl-^	ctrl-R	ctrl-^	H
MOVE CURSOR RIGHT	ctrl-L	ctrl-P	r-arrow	C
MOVE CURSOR UP	ctrl-K	ctrl-L	u-arrow	A
NON PRINTING CHAR	'?'	'?'	'?'	'?'
PREF [DELETE CHAR]	FALSE	FALSE	FALSE	FALSE
PREF [ED ACCEPT KEY]	FALSE	FALSE	FALSE	FALSE
PREF [ED ESCAPE KEY]	FALSE	FALSE	FALSE	TRUE

PREF [ERASE LINE]	FALSE	FALSE	FALSE	TRUE
PREF [ERASE SCREEN]	FALSE	TRUE	TRUE	TRUE
PREF [ERASE TO EOLN]	FALSE	TRUE	TRUE	TRUE
PREF [ERSE TO EOSCN]	FALSE	TRUE	TRUE	TRUE
PREF [KEY DEL CHAR]	FALSE	FALSE	FALSE	FALSE
PREF [KEY DEL LINE]	FALSE	FALSE	FALSE	FALSE
PREF [KEY MV CRS DN]	FALSE	FALSE	FALSE	TRUE
PREF [KEY MV CRS LT]	FALSE	FALSE	FALSE	TRUE
PREF [KEY MV CRS RT]	FALSE	FALSE	FALSE	TRUE
PREF [KEY MV CRS UP]	FALSE	FALSE	FALSE	TRUE
PREF [MOVE CRS HOME]	FALSE	TRUE	FALSE	TRUE
PREF [MOVE CURS RT]	FALSE	FALSE	FALSE	TRUE
PREF [MOVE CURS UP]	FALSE	FALSE	FALSE	TRUE
PREF [NONPRINT CHAR]	FALSE	FALSE	FALSE	FALSE
SCREEN HEIGHT	24	24	24	24
SCREEN WIDTH	80	80	80	80
STUDENT	FALSE	FALSE	FALSE	FALSE
VERTICAL MOVE DELAY	5	5	10	10

* The BREAK key can also be used, but it's perilously close to RETURN.
** Break is also control-@ on Hazeltines.

Terminals:	DEC VT-52	HEWLETT/ PACKARD	DATA- MEDIA
Data Items:			
BACKSPACE	backspace	backspace	backspace
EDITOR ACCEPT KEY	ctrl-C	ctrl-C	cntrl-C
EDITOR ESCAPE KEY	esc	esc	esc
ERASE LINE	ctrl-@	cntrl-@	ctrl-@
ERASE SCREEN	ctrl-@	cntrl-@	ctrl-L
ERASE TO END OF LINE	K	K	ctrl-]
ERASE TO END OF SCRN	J	J	ctrl-K
HAS LOWER CASE	TRUE	TRUE	TRUE
HAS RAND CURS ADDR	TRUE	TRUE	TRUE
HAS SLOW TERMINAL	FALSE	FALSE	FALSE
KEY FOR BREAK	ctrl-@	cntrl-@	cntrl-@
KEY FOR FLUSH	ctrl-F	ctrl-F	ctrl-F
KEY FOR STOP	ctrl-S	ctrl-S	ctrl-S
KEY TO ALPHA LOCK	ctrl-R	ctrl-R	ctrl-R
KEY TO DELETE CHAR	ctrl-H	backspace	backspace
KEY TO DELETE LINE	del	del	del
KEY TO END FILE	ctrl-C	ctrl-C	ctrl-C
KEY TO MV CURS DOWN	B	d-arrow	d-arrow
KEY TO MV CURS LEFT	D	l-arrow	l-arrow
KEY TO MV CURS RGHT	C	r-arrow	r-arrow
KEY TO MV CURS UP	A	u-arrow	u-arrow
LEAD IN FROM KEYBD	esc	cntrl-A	ctrl-@
LEAD IN TO SCREEN	esc	esc	ctrl-@
MOVE CURSOR HOME	H	H	ctrl-Y
MOVE CURSOR RIGHT	C	C	ctrl-\
MOVE CURSOR UP	A	A	ctrl-_
NON PRINTING CHAR	'?'	'?'	'?'
PREF [DELETE CHAR]	FALSE	FALSE	FALSE
PREF [ED ACCEPT KEY]	FALSE	FALSE	FALSE
PREF [ED ESCAPE KEY]	TRUE	FALSE	FALSE
PREF [ERASE LINE]	FALSE	FALSE	FALSE
PREF [ERASE SCREEN]	FALSE	FALSE	FALSE
PREF [ERASE TO EOLN]	TRUE	TRUE	FALSE
PREF [ERSE TO EOSCN]	TRUE	TRUE	FALSE
PREF [KEY DEL CHAR]	FALSE	FALSE	FALSE
PREF [KEY DEL LINE]	FALSE	FALSE	FALSE
PREF [KEY MV CRS DN]	TRUE	FALSE	FALSE
PREF [KEY MV CRS LT]	TRUE	FALSE	FALSE
PREF [KEY MV CRS RT]	TRUE	FALSE	FALSE
PREF [KEY MV CRS UP]	TRUE	FALSE	FALSE
PREF [MOVE CRS HOME]	TRUE	TRUE	FALSE

PREF [MOVE CURS RT]	TRUE	TRUE	FALSE
PREF [MOVE CURS UP]	TRUE	TRUE	FALSE
PREF [NONPRINT CHAR]	FALSE	FALSE	FALSE
SCREEN HEIGHT	24	24	24
SCREEN WIDTH	80	80	80
STUDENT	FALSE	FALSE	FALSE
VERTICAL MOVE DELAY	0	0	0

III.5.3 Appendix C -- GOTOXY Source Examples

The following example is shipped on your System disk as SAMPLEGOTO.TEXT. It is about as simple a GOTOXY as can be written. It is not the code which is shipped in your Operating System: that is the next example, which on one hand is a much more general program, and on the other hand is also much longer. Since GOTOXY is a frequently used I/O routine, you want it to be efficient: it should be tailored to your particular terminal. This brief example works for a DEC VT-52. For an efficient example, see the Datamedia sample.

```
(*The following is a sample gotoxy procedure for the VT-52*)
(*$U-*)
UNIT GOTOXY;

INTERFACE
PROCEDURE AGOTOXY(X,Y:INTEGER);

IMPLEMENTATION
PROCEDURE AGOTOXY;
BEGIN
  IF X<0 THEN X:=0;
  IF X>79 THEN X:=79;
  IF Y<0 THEN Y:=0;
  IF Y>23 THEN Y:=23;
  WRITE (CHR(27),'Y',CHR(Y+32),CHR(X+32));
END;
END.
```

This example works for a DEC VT-50. It uses WRITEs embedded in WHILE loops, and is not fast.

```
{$U-}
UNIT GOTOXY;

INTERFACE
PROCEDURE AGOTOXY(X,Y: INTEGER);

IMPLEMENTATION
PROCEDURE AGOTOXY;
BEGIN
   {Check the input data to see that it is within the screen
     dimensions.  On some smarter terminals, if a cursor position
     command is sent for a position that does not exist, the
     results are unpredictable.}
IF X < 0 THEN X := 0
ELSE
  IF X > 79 THEN X := 79;
IF Y < 0 THEN Y := 0
ELSE
  IF Y > 11 THEN Y := 11;
   {For a DECscope VT-50, GOTOXY needs to be implemented by:}

   {Send the cursor home, 0,0}
WRITE(CHR(27),'H');

   {While TAB is meaningful, use it to move the cursor}
WHILE X > 8 DO
  BEGIN
    WRITE(CHR(9));
    X := X-8;
  END;

   {Finish off what portion of the x coordinate could not be
     absorbed with the TAB characters.}
WHILE X > 0 DO
  BEGIN
    WRITE(CHR(27),'C');
    X := X-1
  END;

   {Send line-feeds to access the y coordinate.}
WHILE Y > 0 DO
```

```
  BEGIN
    WRITE(CHR(10));
    Y := Y-1
  END
END;

END.
```

This example is for a Datamedia 1520, and demonstrates the quickest form of GOTOXY: using a UNITWRITE to send one single command stream to the terminal. As mentioned above, this method can speed up your terminal I/O by as much as 10%; we recommend it.

```
{$U-}
UNIT GOTOXY;

INTERFACE
PROCEDURE AGOTOXY(X,Y: INTEGER);

IMPLEMENTATION

PROCEDURE AGOTOXY;
VAR
  T: PACKED ARRAY[0..2] OF CHAR;
BEGIN
  T[0] := CHR(30);  {chr(30) is an ASCII RS, which is Datamedia's
                              absolute cursor address flag.}

   {Set appropriate character for x coordinate.}
  IF X < 0 THEN T[1] := CHR(32)        {Note the offset of 32.}
  ELSE
    IF X > 79 THEN T[1] := CHR(32+79)
    ELSE
      T[1] := CHR(X+32);

   {Set appropriate character for y coordinate.}
  IF Y < 0 THEN T[2] := CHR(32)
  ELSE
    IF Y > 23 THEN T[2] := CHR(32+23)
    ELSE
      T[2] := CHR(Y+32);

   {Send the cursor where it belongs.}
  UNITWRITE(1,T,3)       {1 is the device number of CONSOLE:}
END;

END.
```

Here are two more examples using UNITWRITE. They are for a Soroc and a Hazeltine terminal, respectively.

```
(*$U-*)
UNIT GOTOXY;

INTERFACE
PROCEDURE AGOTOXY(X,Y: INTEGER);

IMPLEMENTATION
PROCEDURE AGOTOXY;

(* FOR A SOROC IQ 120 *)

VAR TELL: PACKED ARRAY [0..3] OF 0..255;

BEGIN
  IF X>79 THEN X:=79
    ELSE IF X<0 THEN X:=0;
  IF Y>23 THEN Y:=23
    ELSE IF Y<0 THEN Y:=0;
  TELL[0] := 27;             (* LEAD-IN FOR SOROCS *)
  TELL[1] := ORD('=');
  TELL[2] := 32+Y;           (* NOTE THE OFFSET *)
  TELL[3] := 32+X;
  UNITWRITE(1,TELL,4)
END;

END.
```

```
{$U-}
Unit gotoxy;

Interface
Procedure agotoxy(x,y: integer);

Implementation
Procedure agotoxy;

{gotoxy for the Hazeltine 1500 and 1510}

var tell: packed array [0..3] of 0..255;

Begin
   if x>79 then x:=79
      else if x<0 then x:=0;
   if y>23 then y:=23
      else if y<0 then y:=0;
   tell[0] := 126;             {the lead-in for a Hazeltine}
   tell[1] := 17;              {also a DC1}
   if x<30 then
      tell[2] := x+96          {different offset for these terminals}
   else
      tell[2] := x;
   tell[3] := y+96;
   unitwrite(1,tell,4)
End;

End.
```

III.5.4 Appendix D -- Sample SETUP Session with Comments

The following is a sample of part of a session with SETUP. The data is being changed from the System defaults to the specifications for a Soroc terminal, as in Appendix B above. All underlined text like this is user input, and all text enclosed in curly brackets {like this} is commentary. Angle brackets <these> are used to enclose the names of non-printing characters {like <return>}. All else is SETUP's output to the terminal.

```
{To begin, you must eXecute SETUP}

XSETUP<return>
INITIALIZING...........................
.........................
SETUP: C(HANGE T(EACH H(ELP Q(UIT [D1]

{H(ELP tells you about the other commands, and T(EACH
 describes the use of SETUP.  Now is the most profitable
 time to use these commands.
 Suppose you have read H(ELP and T(EACH, and decide
 to change data items by going through the menu.
 You must hit C for C(HANGE.}

C   {Note: these single-character commands don't echo.}
CHANGE: S(INGLE) P(ROMPTED) R(ADIX)
    H(ELP) Q(UIT)

{H(ELP) describes the commands on this particular line,
 R(ADIX) allows you to change the base of the numbers
 you enter, and Q(UIT) returns you to the SETUP: prompt.
 What you want to do now is go through the prompted menu.}

P



FIELD NAME = BACKSPACE
 OCTAL  DECIMAL  HEXADECIMAL  ASCII  CONTROL
    10        8        8        BS       ^H
WANT TO CHANGE THIS VALUE? (Y,N,!)
<return>
WANT TO CHANGE THIS VALUE? (Y,N,!)
```

```
{<return> or <space> will cause this prompt to be repeated.
 ! causes an escape to the CHANGE: prompt.
 Since control-H (^H) is indeed the Soroc's backspace,
 you want to go on.}


N



FIELD NAME = EDITOR ACCEPT KEY
 OCTAL  DECIMAL  HEXADECIMAL  ASCII  CONTROL
     0        0           0    NUL       ^@
WANT TO CHANGE THIS VALUE? (Y,N,!)
Y
NEW VALUE: <home>

{When <home> or any other non-printing key
 is pressed, ? is displayed.}

 OCTAL  DECIMAL  HEXADECIMAL  ASCII  CONTROL
     3        3           3    ETX       ^C
WANT TO CHANGE THIS VALUE? (Y,N,!)
N



FIELD NAME = EDITOR ESCAPE KEY
 OCTAL  DECIMAL  HEXADECIMAL  ASCII  CONTROL
     0        0           0    NUL       ^@
WANT TO CHANGE THIS VALUE (Y,N,!)
Y
NEW VALUE: <return>

{Any unexpected input here causes the
 relevant section of T(EACH to be output,
 followed by this:}
C(ONTINUE)
{All characters are ignored except C, and
 then the prompt is repeated.}
```

```
C
NEW VALUE: <rubout>  {Again, a ? is echoed.}
 OCTAL  DECIMAL  HEXADECIMAL  ASCII
   177      127      7F         DEL
WANT TO CHANGE THIS VALUE? (Y,N,!)

{(Note that there is no corresponding control key.)
 DEL is not the key you meant, so you must
 change it again.}

Y
NEW VALUE: <esc>  {? is echoed.}
 OCTAL  DECIMAL  HEXADECIMAL  ASCII  CONTROL
    33      27      1B         ESC      ^[
WANT TO CHANGE THIS VALUE? (Y,N,!)
N  {This is what it should be.}



{The menu continues in this way for the rest of
 the data items.  Suppose you have gone ahead and
 answered all of the questions according to the
 Soroc specifications.  After the last data item,
 you again get the menu:}


CHANGE: S(INGLE) P(ROMPTED) R(ADIX)
    H(ELP) Q(UIT)

{You realize that you left the prefix for
 ERASE LINE at FALSE, when it should be
 TRUE.  You want to change just this one
 data item.}

S  {For S(INGLE)}
NAME OF FIELD: PREFIXED [ERASE]
DIDN'T FIND PREFIXED [ERASE]   {Oops}
NAME OF FIELD: PREFIXED [ERASE LINE]

FIELD NAME = PREFIXED [ERASE LINE]
 CURRENT VALUE IS FALSE
WANT TO CHANGE THIS VALUE? (Y,N,!)
Y
NEW VALUE: TRUE  {T would also work.}
 CURRENT VALUE IS TRUE
```

```
WANT TO CHANGE THIS VALUE? (Y,N,!)
N

CHANGE: S(INGLE) P(ROMPTED) R(ADIX)
    H(ELP) Q(UIT)
Q
SETUP: C(HANGE T(EACH H(ELP Q(UIT [D2]
Q  {You're through changing data now.}
QUIT: D(ISK) OR M(EMORY) UPDATE,
          R(ETURN) H(ELP) E(XIT)

{You want to do a disk update to create
 NEW.MISCINFO on your disk for future use.}

D
QUIT: D(ISK) OR M(EMORY) UPDATE,
          R(ETURN) H(ELP) E(XIT)
E

{And now you're done.  The Pascal system prompt
 will appear.}
```

This guide was written by David Berger, Randy Clark, Barry Demchak, and Bruce Sherman, and edited by Randy Clark. George Symons and John Tennant provided invaluable time and information. Mark Allen and Steve Koehler also contributed useful comments.