UCSD p-System*

- Editor
- Filer
- Utilities

Part Three: UCSD p-System Utilities

*UCSD p-System is a trademark of the Regents of the University of California.

1041570-1
[Part C]

UCSD p-System*

- Editor
- Filer
- Utilities

Part Three: UCSD p-System Utilities

This manual was developed by staff members of the Texas Instruments Education and Communications Center.

 Item involved met its quality assurance standards applicable to Version IV.0.

TABLE OF CONTENTS

SECTION 1: GENERAL INFORMATION

The UCSD p-System* Utilities software package includes 11 precompiled programs. These programs supply certain functions that are sufficiently useful to be included with the System, but are not used frequently enough to warrant their being established as System commands.

The Utility programs, written in UCSD Pascal*, are designed for use with the UCSD p-System for the TI-99/4 and TI-99/4A Home Computers. In addition to the computer and a TI Color Monitor (or a TI Video Modulator and a television set), the Utilities package requires the TI Memory Expansion unit, the TI P-Code peripheral, and the TI Disk Memory System (a TI Disk Drive Controller and one to three Disk Memory Drives). See the owner's manual for the P-Code unit for details.

The Utilities package contains these programs:

- DECODE -- "Decodes" codefiles to give you detailed information about their internal contents.
- DFORMAT -- Prepares a diskette for use with the p-System.
- XDISK and DSKTOCASS -- Allow you to transfer diskette files to a cassette tape.
- MARKDUPDIR and COPYDUPDIR -- Allow easy duplication of diskette directories as protection against loss of files.
- LIBRARY -- Provides a means for grouping separate compilations and separately assembled routines into a single file.
- MODRS232 -- Lets you modify the software switch options for a device connected to the computer via the TI RS232 Interface.
- PATCH -- Makes it possible to edit codefiles at the byte level or to dump files in various formats.
- RECOVER -- Attempts to recreate a damaged diskette directory.
- SETLTYPE -- Allows you to indicate to the P-Code peripheral if a program segment that is being loaded includes assembly language.
- SETUP -- Lets you change information related to your computer terminal.

*trademark of the Regents of the University of California

1.1 USING THIS MANUAL

This manual is designed for use both as an introduction to the p-System Utilities and as a reference document after you are familiar with these programs. Each utility or group of related utilities is discussed in a separate section of the manual.

Utility	Section
DECODE	2
DFORMAT	3
Diskette-to-Cassette Transfers:	4
XDISK	4.1
DSKTOCASS	4.2
Duplicate Directories:	5
MARKDUPDIR	5.1
COPYDUPDIR	5.2
LIBRARY	6
MODRS232	7
PATCH	8
RECOVER	9
SETLTYPE	10
SETUP	11

Each section gives a description of the inputs required by the utility program and, in some cases, a sample of the output of the program.

1.2 SET-UP INSTRUCTIONS

The steps involved in accessing a utility program are included in this section. Please read this material completely before proceeding.

1. Be sure the Memory Expansion unit, the P-Code peripheral, and the Disk Memory System are connected to the computer and turned on in the proper order. (Refer to the appropriate owner's manuals for details.)

2. Insert the Utilities diskette into the first disk drive, and place your data or backup diskette, if any, in the second or third drive, if available.

3. Now turn on the monitor and computer console. The p-System promptline now appears. (**Note:** If you turn on the computer before inserting a diskette in a disk drive, you must insert a diskette and then press **I** to initialize the System before you can proceed.)

4. To load a Utilities program, press **X** for X(ecute (execute). The System then asks:

   What file?

5. Type a number sign, the device number (see Section 1.2.1), a colon, and the name of the program. Then press <return>. For example, typing

   #4:PATCH

   and pressing <return> loads the PATCH utility if it's located in the first disk drive. Note that the suffix .CODE is automatically added to the filename. After a brief loading period, the utility's name appears, and the program begins to prompt you for the appropriate input information.

6. If you are using only one disk drive, remove the Utilities diskette, and insert your diskette into the drive. If you are using more than one drive, be sure that the correct diskette is in place in the second or third drive. Then begin entering the information requested by the utility program. (See the appropriate sections of this manual for details.)

1.2.1 Device Numbers

The following table lists the device number associated with each device. (For a more detailed description of the devices, refer to the UCSD p-System Filer manual.)

Number	Device
#1	Keyboard and display with echo
#2	Keyboard and display without echo
#4	first disk drive
#5	second disk drive
#6	9600 baud RS232 input/output
#7	300 baud RS232 input
#8	300 baud RS232 output
#9	third disk drive
#14	Operating System
#31	Audio cassette tape
#32	TI Solid State Thermal Printer

1.3 SPECIAL FUNCTION KEYS

For the most part, the Utilities programs require very few special key functions beyond the single keystrokes that access any promptline commands. The PATCH utility, which contains editing capabilities, is the major exception, and its editing keys are explained in Section 8. For a list of the other function keys, refer to the UCSD p-System P-Code manual.

SECTION 2: DECODE

The DECODE utility provides access to a codefile in symbolic form, giving you a detailed knowledge of the internal contents of the file. For instance, DECODE can help give you a better understanding of the step-by-step execution of a program's object code so that you can make efficient modifications.

DECODE gives you access to the following information.

- Names, types, global data size, and other general information about all code segments in the file.
- INTERFACE section text (if present) for all units in the file.
- Symbolic listing of any (or all) p-code procedures in any (or all) segments of the file.
- Segment references and linker directives associated with code segments.

If a program uses a unit (a separately compiled segment), the unit is decoded only if it is within the host file. DECODE does not search the diskette for units to decode. Assembly routines linked into a higher-level host are not disassembled when the host is decoded.

2.1 USER INSTRUCTIONS

To access the DECODE utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

What file?

If the Utilities diskette is in the first disk drive, type

#4:DECODE

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After DECODE is loaded, the first prompt asks for the input codefile (the suffix .CODE is automatically added, if necessary). If you have a single disk drive, remove the Utilities diskette at this point, and place your program diskette in the first drive. If you are using more than one drive, insert your program diskette into the second or third disk drive. Then type a number sign, the appropriate device number, a colon, and the name of the program, and press <return>. For example, typing #5:DEMO and pressing <return> loads a file named DEMO.CODE from the diskette in the second disk drive.

The next prompt asks for the name of a listing file to which DECODE's output is to be written. The file may be CONSOLE: (indicated by pressing <return>), REMOUT:, PRINTER:, or a diskette file. The following promptline, which indicates the first level of options available, is then displayed.

```
Segment Guide: A(ll), #(dct index)
 D(ictionary), Q(uit)
```

A(ll)	Disassembles all segments.
#(dct index)	Disassembles the specified segment (if present).
D(ictionary)	Displays the codefile's segment dictionary.
Q(uit)	Allows you to enter another file for analysis or to return to the System promptline.

Depending on the disassembly you want to perform, type the appropriate letter or number and press <return>.

2.2 A(ll)

If you press **A** for A(ll), a disassembly similar to the following is produced.

DECODE

```
Constant Pool for segment DEMO
Block:   2  Block offset:   0  Seg offset:   0
0:   0017 0000 4445 4D4F 2020 2020 0001 0014 0004 0000  _ _DEMO    _ _ _ _
 10:   0000                                              _

Block:   2  Block offset:  40  Seg offset:  40
  0:   0001 0000 000C                                    _ _ _

Segment: DEMO      Procedure:   1
Block:   2  Block offset:  26  Seg offset:   26
Data size:     0  Exit IC:    38
     Offset                                Hex code
     0(000):    LDCB        50             8032
     2(002):    SRO          1             A501
     4(004):    SCXG   ADD        1        7201
     6(006):    SLDO         1              30
     7(007):    LCDI       400              819001
    10(00A):    EFJ          4             D2F8
  Exit code:
    12(00C):    RPU          0             9600

Constant pool for segment ADDI
Block:   1  Block offset:   0  Seg offset:    0
0:  0014 0000 4144 4449 2020 2020 0001 0011 0004 0000  _ _ADDI    _ _ _ _
  10:  0000                                             _

Block:   1  Block offset:  34  Seg offset:   34
   0:  0001 0000 000C                                   _ _ _

Segment: ADDI      Procedure:   1
Block:   1  Block offset:  26  Segment offset:   26
Data size:     0  Exit IC:    31
     Offset                                  Hex code
     0(000):    SLDO         1              30
     1(001):    SLDO         1              30
     2(002):    ADI                         A2
     3(003):    SRO          1              A501
  Exit code:
     5(005):    RPU          0              9600
```

The first lines of the disassembled listing show the segment name, block number, block offset, and segment offset of the code in that segment.

The next lines contain a variable number of words. Each word is displayed as a hexadecimal number (most significant byte first). If the word consists of displayable characters, its character representation follows. The first word is the PROCEDURE DICTIONARY POINTER, followed by the RELOCATION LIST POINTER, the eight-byte segment name, and a variable number of words. The next-to-last word is the segment's EXIT IC, followed by its DATASIZE.

The disassembled code itself is displayed in blocks. The OFFSET column, which corresponds to the fourth column in a compiled listing, shows the offset in bytes from the beginning of the procedure (the count is in both decimal and hexadecimal). Then the p-code mnemonic is displayed, followed by the operands, if any, and finally the HEX CODE for that particular instruction.

Jump operands are displayed as offsets relative to the start of the procedure, rather than from the instruction program counter (IPC), to make the disassembly more readable. Thus, the operand shown is the offset of some line; in the example, the false jump (EFJ) on line 10 shows 4, which means line 4 (the SCXG instruction); the HEX CODE indicates that the offset is actually >F8, which is IPC-relative.

When the Segment Guide promptline reappears on the screen, you may select another option or press **Q** for Q(uit) (see Section 2.5).

2.3 #(dct index)

If you enter a segment number and press <return>, rather than using the A(ll) command, DECODE disassembles a single segment. A line similar to the following is displayed.

```
There are  2 procedures in ADDI.
Procedure Guide: A(ll), #(of procedure)
 L(ink info), C(onst pool), S(eg refs)
 I(nterface text), Q(uit)
```

Selecting A(ll) disassembles all of the procedures in the segment. Typing the number of a procedure and pressing <return> disassembles that procedure. The following options are also available if they are applicable to the program you are disassembling.

L(ink info)	Linker information.
C(onst pool)	Constant pool information.
S(eg refs)	Segment references -- a list of any other segments referenced by the disassembled segment.
I(nterface text)	A section that links a program to a unit by listing the procedure or variables that can be called.

If you select an option that is not applicable to your program, the display gives you a brief message and returns to the segment promptline.

2.3.1 L(ink info)

For example, if the segment has linker information and you press **L** for L(ink info), the following might be displayed:

```
Linker information for segment SOMESEG:

SOMEPROC  EXTPROC srcproc=4 nparams=0 koolbit=false
```

2.3.2 C(onst pool)

Pressing **C** for C(onst pool) displays the constant pool for the current segment. All of the constants (strings, reals, long integers, and case tables) of a segment are included in the constant pool.

2.3.3 S(eg refs)

If the segment has references to other segments and you press **S** for S(eg refs), the following might be displayed.

```
Segment references list for segment KERNEL:
14: ***                    5: SYSCMND
13: CONCURRE               4: DEBUGGER
12: PASCALIO               3: FILEOPS
11: HEAPOPS                2: SCREENOP
10: STRINGOP               0:
```

2.3.4 I(nterface text)

If the segment is a unit with interface text and you press **I** for I(nterface text), the following might be displayed.

```
Interface text for segment SOMEUNIT:

PROCEDURE A_PROC;
PROCEDURE ANOTHER_PROC(I:INTEGER);
FUNCTION A_FUNCTION:BOOLEAN;
IMPLEMENTATION
```

2.3.5 Q(uit)

When you have completed your procedure disassemblies, press **Q** and <return> to return to the Segment Guide promptline.

2.4 D(ictionary)

If you press **D** for D(ictionary) when the Segment Guide promptline is on the screen, a display similar to the following appears.

```
INX   NAME  START  SIZE  VERSION  M_TYPE  SG#  SEG_TYPE RL FMY_NAME or
                                                          DSIZE SGRF  HSG  TS
  0:  DEMO      2    24   IV_0     M_PSEUDO  2  PROG_SEG R      1   5    3   0
  1:  ADDI      1    24   IV_0     M_PSEUDO  3  PROC_SEG R  'DEMO  '
  2:                                             NO_SEG
  3:                                             NO_SEG
  4:                                             NO_SEG
  5:                                             NO_SEG
  6:                                             NO_SEG
  7:                                             NO_SEG
  8:                                             NO_SEG
  9:                                             NO_SEG
 10:                                             NO_SEG
 11:                                             NO_SEG
 12:                                             NO_SEG
 13:                                             NO_SEG
 14:                                             NO_SEG
 15:                                             NO_SEG
 (C):
 SEX: MOST significant byte first, NEXT PAGE:0
 Segment Guide: A(ll), #(dct index),
  D(ictionary), Q(uit)
```

DECODE's D(ictionary) display is a format of the codefile's segment dictionary, with the following information given.

INX — INDEX, a list showing each segment's number, according to DECODE.

NAME — The name of each segment.

START — Each segment's starting block (relative within the codefile).

SIZE — The length (in words) of each segment.

VERSION — The System version number of the segment.

M_TYPE — The machine type. Usually this is M_PSEUDO, indicating a p-code segment, but any segment containing machine code will display M_9900.

SG# — Segment number.

SEG_TYPE — Segment type. These can be NO_SEG, PROG_SEG, UNIT_SEG, PROC_ SEG, or SEPRT_SEG.

- NO_SEG — An empty segment "slot."
- PROG_SEG — A program segment.
- UNIT_SEG — A unit segment.
- PROC_SEG — A separate routine segment.
- SEPRT_SEG — An assembled segment.

RL — Relocate/Link. This column indicates whether or not the segment is relocatable and whether or not it needs to be linked. R indicates a relocatable segment; L indicates a segment that must be linked.

FMY_NAME or DSIZE, SGRF, and HSG — If the segment is declared within a program or unit, the FMY_NAME column contains its "family name"; i.e., the name of the program or unit. Otherwise, the DSIZE (data size), SGRF (segment reference number), and HSG (high segment number) columns are displayed, containing respectively the compilation module's data size, segment references, and maximum number of segments.

TS — Text size, showing the number of disk blocks in the interface text.

(C): — Copyright notice. (C): is followed by whatever copyright notice the codefile may have.

SEX: — The byte sex of the codefile, indicating whether the most- or least-significant byte is first.

The Segment Guide promptline is the last line on the screen.

2.5 Q(uit)

When you have finished with DECODE, press **Q** for Q(uit) when the Segment Guide promptline is on the screen. The program then asks

Continue?

If you press **Y**, the program returns to the "Input file" prompt, with the current filename given as a default. If you want to DECODE another file, enter its filename and proceed as before.

Pressing **N** in response to the "Continue?" prompt returns you to the p-System promptline.

Note: Pressing <return> is not required after a Y or N response.

SECTION 3: DFORMAT

The DFORMAT (Disk Format) utility formats (initializes) a diskette for use with the p-System. The formatting process gives the diskette a volume name of PASCAL and organizes the diskette into a single, long file also named PASCAL. In addition, the process provides the diskette with header information for each sector. Each diskette must be formatted before its first use. (Note that any diskette formatted with DFORMAT cannot be used with TI BASIC or TI Extended BASIC unless it is first re-initialized with the Disk Manager Solid State Software ™ Command Module.)

3.1 USER INSTRUCTIONS

To access the DFORMAT utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

```
What file?
```

If the Utilities diskette is in the first disk drive, type

```
#4:DFORMAT
```

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After DFORMAT is loaded, the screen displays the following prompt.

```
Enter the number of the unit
holding the diskette (4,5,9).
```

The number you choose depends on which drive contains the diskette to be formatted. Press **4** to format a diskette located in the first disk drive, **5** for the second disk drive, or **9** for the third disk drive. (**Note:** With a single disk drive, be sure to remove the Utilities diskette and insert the diskette to be formatted before proceeding.) If you type a number other than 4, 5, or 9, the program displays the error message "ERROR -- Unit must be 4, 5, or 9" and then asks again for the unit number.

Next, DFORMAT displays the prompt:

```
How many tracks do you want
on a side (normally 40)?
```

To specify the number of tracks you want on a side of the diskette, type a number from 1 through 99 and press <return>. If you enter a number outside of this range, you receive an error message and are asked again for the number of tracks.

After you specify the number, DFORMAT asks if you want to format the disk as single sided (S) or double sided (D). Depending on your disk system and the type of diskettes you are using, press **S** for single-sided diskettes or **D** for double-sided. If necessary, refer to the Disk Memory System owner's manual for more information.

Now DFORMAT displays a prompt asking if the disk should be formatted as single density (S) or double density (D). Again, depending on your configuration, press **S** for single-density diskettes or **D** for double-density diskettes.

You have now entered all required information. To help you check your entries, a list of your selections is displayed, telling you the specified unit number, the number of tracks per side indicated for your diskette, whether the diskette is single or double sided, and the density (single or double) you chose. You are now asked:

```
Ready to format the disk (Y/N)?
```

If the information is correct and the diskette to be formatted is inserted in the appropriate disk drive, press **Y** to begin the formatting process.

If the diskette cannot be formatted, an error message appears, followed by the prompt "Format another disk (Y/N)?". This prompt also appears when the formatting is completed or if you press **N** in response to "Ready to format the disk (Y/N)?". To enter the information required to format another diskette, press **Y** and DFORMAT asks again for the unit number. When you finish formatting diskettes, press **N** in response to "Format another disk (Y/N)?" and the System promptline is displayed.

Now you must establish the diskette directory with the Filer's Z(ero command (see the UCSD p-System Filer manual) before using the diskette with the System.

Note: Diskettes to be used with the p-System can also be formatted with the Disk Manager Module. However, as with DFORMAT, you must also establish the diskette directory with the Filer's Z(ero command before using the diskette with the System.

SECTION 4: DISKETTE-TO-CASSETTE TRANSFERS

Two programs, XDISK and DSKTOCASS, allow you to transfer codefiles from diskette to cassette tape for inexpensive storage and portability. The XDISK program converts a diskette to a format compatible with cassette storage, and the DSKTOCASS program transfers the file to a cassette tape.

The XDISK program creates an "executable disk" containing the directory and all the codefiles on a diskette. Any unused portions of blocks are omitted. The result is a compact, memory-image file of continuous code that can be stored on tape. The executable disk is transferred as a single file to the output diskette you specify. This diskette serves as a temporary storage device for the file until it is transferred to a cassette by the DSKTOCASS utility.

Note: Diskette-to-cassette transfer requires a minimum of two disk drives, and cassette files are limited to a size of 12,350 bytes.

4.1 XDISK UTILITY USER INSTRUCTIONS

The first step in the disk-to-cassette process involves using the T(ransfer option of the UCSD p-System Filer (see the Filer manual for details). Transfer to an initialized diskette the codefile or codefiles you want to convert to cassette form. This diskette is hereafter called the "program diskette" in this section.

Note: For ease of use, you may want to name the file XCASS.CODE when you make the transfer, since this is the default filename for executing TAPE: at the System level to load and run a file from cassette. You may also want to store only one program on each cassette. If so, the program diskette should contain only one codefile.

When you have completed your program diskette, exit from the Filer, place the Utilities diskette in the first disk drive, and insert the program diskette in the second drive.

To access the XDISK utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

What file?

DISKETTE-TO-CASSETTE TRANSFERS

If the Utilities diskette is in the first disk drive, type

```
#4:XDISK
```

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After XDISK is loaded, the program begins to prompt you for the required information. If you have two disk drives, remove the Utilities diskette at this point, and place in the first drive a temporary storage diskette, one with enough available storage space to contain the codefile(s) on the program diskette. If you are using three drives, insert your temporary storage diskette into the third drive. Then respond to the program's prompts, as follows.

Unit number of disk: Enter the number of the disk drive holding your program diskette. For example, if your program diskette is in the second disk drive, type 5 and press <return>.

Output file: Type a number sign, the device number of the drive containing the temporary storage diskette, a colon, and the filename you are assigning to the temporary file. Then press <return>. For example, typing

```
#4:CASSETTE
```

and pressing <return> creates a file named CASSETTE on the diskette in the first drive.

When you have answered the last prompt, the program converts the codefile(s) on your program diskette to the executable disk format and stores the file on your temporary storage diskette. A report of the process similar to the following is displayed.

```
Begin
File: CTEST. . . .
```

As each block is converted, a dot appears on the screen following the filename. When the executable disk is completed and stored on the temporary storage diskette, the XDISK program returns to the System promptline.

4.2 DSKTOCASS UTILITY USER INSTRUCTIONS

After the executable disk is created, you are ready to transfer the contents to a cassette tape. Attach your tape recorder to the outlet on the back of the console, and insert your cassette tape into the recorder. (See the User's Reference Guide for information about using cassette recorders as storage devices.)

At this point, if you are using only two disk drives, remove your program diskette, and reinsert the Utilities diskette into the second drive (the temporary storage diskette is in the first drive).

To access the DSKTOCASS program, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

What file?

If the Utilities diskette is in the second disk drive, type

#5:DSKTOCASS

and press <return>. If the Utilities diskette is in the first or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After the program is loaded, it begins to prompt you through the transfer process.

Disk file name — Type the device number of the drive containing the temporary storage diskette, a colon, and the filename you assigned to the codefile. Then press <return>. For example, typing

#4:CASSETTE

and pressing <return> identifies a file named CASSETTE on the diskette in the first drive.

REWIND TAPE
PRESS A KEY — Rewind the cassette in your tape recorder; or, if the recorder has a tape location counter, position the tape to the location at which you want the recording to begin. Then press any key on the console keyboard to go on.

SET TO RECORD
PRESS A KEY — Press down the Record button(s) on your recorder, and then press any key on the console to go on.

DISKETTE-TO-CASSETTE TRANSFERS

The transfer to cassette begins when you answer the last prompt, and a report similar to the following is displayed, one line at a time.

```
    Disk to Cassette Transfer
  Disk file name #5:CASSETTE

 Reading disk . . . .
 Writing to cassette
 Verify (y/n)
```

If you answer "yes" to the "Verify" prompt, the program prompts you through the verification routine. (If you respond with N for "no," the program gives you the "End" report and returns to the System promptline.)

REWIND TAPE PRESS A KEY	Rewind the tape to the proper starting position. Then press any key on the keyboard.
SET TO PLAY PRESS A KEY	Press down the Play button on your recorder, and then press any key on the console to go on.

When you answer this last prompt, the verification process begins, and a report similar to the following is displayed.

```
 Verifying
 Reading disk . . .
 Reading cassette
 Verify O.K.

 Verify (y/n)
```

If the verification has been successful, press **N.** The display shows

```
 End, press STOP on recorder
```

and returns to the System promptline. If the verification was not successful, press **Y** to try again.

When the "End" report appears, press the Stop button on your recorder, rewind your cassette tape, remove it from the machine, and label it with its filename and the date of the file transfer.

4.3 LOADING PROGRAMS FROM CASSETTE TAPE

The UCSD p-System supports cassette files through the TAPE: device. (See the UCSD p-System P-Code manual for information.)

If you execute TAPE:filename from the System level, the computer loads the file from the cassette and runs it under the specified filename. If you do not specify a filename, the computer looks for the name XCASS.CODE (the default filename) in order to run the program.

A cassette program, in the form of an executable disk, is loaded into a special area in memory and assigned to Unit #13. Since files loaded from diskettes may overlap this memory area and destroy your cassette-loaded program, be careful in chaining to or using data from diskette files in conjunction with a cassette program.

SECTION 5: DUPLICATE DIRECTORIES

Keeping a duplicate directory on a diskette can help you restore directory information that is lost or garbled, or restore a diskette or the files on it to some desired state. Two utilities, MARKDUPDIR and COPYDUPDIR, have been designed for duplicate directory maintenance.

5.1 MARKDUPDIR UTILITY USER INSTRUCTIONS

The MARKDUPDIR utility first marks a specific area on a diskette so that the area is cleared for the duplicate directory. Before marking the diskette, you must be sure that blocks 6-9 are free for use by checking to see where the first file starts (see the E(xt-dir command in the UCSD p-System Filer manual).

If the first file starts at block 6 or if the first file starts at block 10 with a 4-block unused section at the top, the diskette hasn't been marked. Therefore, to prevent the loss of information, you must re-arrange the files on the diskette to free blocks 6-9.

However, if the first file starts at block 10 and no unused blocks occur at the beginning of the directory, the diskette has been marked and a duplicate directory may already exist.

To access the MARKDUPDIR utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

What file?

If the Utilities diskette is in the first disk drive, type

#4:MARKDUPDIR

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After MARKDUPDIR is loaded, you are asked for the number of the drive which contains the diskette to be marked. Type 4 for the first disk drive, 5 for the second disk drive, or 9 for the third disk drive. Then press <return>. **Note:** Disk Drives 10-12 are not available for use with the TI Home Computer.

After the device number is specified, MARKDUPDIR checks the diskette to see if a duplicate directory exists. If so, you receive a message to that effect, including the prompt "Type <ret> to exit." Pressing <return> leaves MARKDUPDIR and displays the System promptline. If a duplicate directory does not exist, you receive the following message:

```
No duplicate directory on (disk name)
WARNING!
 Blocks 6 - 9 may not be free for use.
 Are you sure that they are free?
```

To leave MARKDUPDIR to check the diskette, press **N.** Next, press <return> to exit and the System promptline is displayed. If you have previously checked the diskette and know that blocks 6-9 are free, press **Y.** You then receive the prompt:

```
Do you want directories to be marked?
```

To leave MARKDUPDIR without creating a duplicate directory, press **N** and then press <return> to display the System promptline. To create a duplicate directory, press **Y.** After completing the duplication, the following message appears:

```
Directories are now marked as duplicate.
Type <ret> to exit.
```

Press <return> to leave MARKDUPDIR and return to the System promptline.

Note: The Z(ero command in the Filer can also be used to create a duplicate directory (see the UCSD p-System Filer manual). The difference between Z(ero and MARKDUPDIR is that Z(ero establishes the area for the duplicate directory before any information is stored on the diskette, while MARKDUPDIR creates a duplicate directory after the primary directory exists. With either method, both directories are automatically updated as files are added to the diskette.

5.2 COPYDUPDIR UTILITY USER INSTRUCTIONS

The COPYDUPDIR (Duplicate Directory Copier) utility copies the duplicate directory for a diskette into the primary directory location, blocks 2 through 5. Thus, if a duplicate directory exists, COPYDUPDIR can help you reconstruct a damaged primary directory.

DUPLICATE DIRECTORIES

To access the COPYDUPDIR utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

What file?

If the Utilities diskette is in the first disk drive, type

#4:COPYDUPDIR

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After COPYDUPDIR is loaded, you are asked for the number of the drive which contains the diskette with the directory to be copied. Type 4 for the first disk drive, 5 for the second disk drive, or 9 for the third disk drive. Then press <return>. **Note:** Disk Drives 10-12 are not available for use with the TI Home Computer.

After the device number is specified, COPYDUPDIR checks the diskette to see if a duplicate directory exists. If not, you receive a message to that effect, including the prompt "Type <ret> to exit." Pressing <return> leaves COPYDUPDIR and displays the System promptline. If a duplicate directory exists, you receive the following message:

```
Are you sure you want to zap
 the directory of (disk name): {blocks 2-5}?
```

To leave COPYDUPDIR without copying the directory, press **N.** Next, press <return> to exit and the System promptline is displayed. To copy the duplicate directory to blocks 2-5, press **Y.** After the copy takes place, you receive the message:

```
  Directory copy is complete.
Type <ret> to exit.
```

Press <return> to leave COPYDUPDIR and return to the System promptline.

SECTION 6: LIBRARY

The LIBRARY utility lets you group separate compilations and separately assembled routines into a single file so that they can be used conveniently by a program or series of programs. Manipulating a single library file takes less time than if the various pieces it contains were within individual files.

Libraries generally contain routines relating to a certain area of application or routines general enough to be used by many programs over a long period of time, such as a math or datafile management library.

Even if a file contains only a single unit or routine, it is treated as a library when the unit or routine is used by some external host.

When using LIBRARY, keep in mind that a "unit" is a separately compiled "segment," an executable piece of code containing one or more procedures. The segment could be stored on diskette or loaded into memory. A "compilation unit" is a program or unit and all the segments declared in it. The segment for the program or unit is the host segment of the compilation unit, while any segment routines declared within the host are called subsidiary segments.

Any units used by the host are <u>not</u> segments belonging to that compilation unit. Units accessed by the compilation unit generate "segment references" in the host segment. The segment references contain the names of all segments referenced by a compilation unit. The Operating System sets up a runtime environment with this information.

6.1 USER INSTRUCTIONS

To access the LIBRARY utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

```
What file?
```

If the Utilities diskette is in the first disk drive, type

```
#4:LIBRARY
```

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After LIBRARY is loaded, the screen displays the following prompt.

```
Output file?
```

With a single disk drive system, remove the Utilities diskette and insert your diskette; with more than one disk drive, insert your diskette in the second drive. Next, type the filename, including the appropriate prefix, and press <return>. **Note:** If you enter the name of an existing file, that file is replaced by the new file.

Next, the prompt "Input file?" appears. Type the name of the file containing the units to be copied to the new file and press <return>. The type, name, and length in words of each module in the file are listed with a corresponding number called a "slot" in the left column.

Slot numbers are used to specify which module to copy and also into which position the module should be placed in the output file. The four types of modules that can occupy the 16 slots in the library are as follows:

- u -- units
- p -- programs
- s -- segment routines
- a -- assembled routines

The LIBRARY promptline now appears at the top of the screen.

```
Library: N(ew,? [IV.0 z8]
```

The question mark at the end indicates that there are more commands than can fit onto one line of the display. Press **?** to display the additional commands. The remaining promptlines appear in the order shown here.

- Library: 0-9(slot-to-slot,? [IV.0 z8]
- Library: E(very, S(elect,? [IV.0 z8]
- Library: C(omp-unit, F(ill,? [IV.0 z8]
- Library: I(nput, O(utput,? [IV.0 z8]
- Library: T(ext, R(efs,? [IV.0 z8]
- Library: A(bort, Q(uit [IV.0 z8]

The commands available from the LIBRARY promptline are explained here in alphabetical order, according to the letter you type to access them.

A(bort	Exits the LIBRARY utility without saving the output file and returns to the p-System promptline.
C(omp-unit	(Compilation unit) Asks you "Copy what compilation unit?" and then copies the specified unit, along with any segment procedures it references, to the output file.
E(very	Copies the entire input file to the output file. If the slot for a code segment is already filled in the output file, the code is placed in the next available slot. If the output file contains an identical code segment, the segment is not copied.
F(ill	Copies all segments that are in the input file and referenced in the output file to the output file.
I(nput	Displays the remainder of the input file list. If the remaining slots are not filled, the same list is redisplayed.
N(ew	Lets you change the input file so that the output file can contain modules from several different files.
O(utput	Displays the remainder of the output file list. If the remaining slots are not filled, the same list is redisplayed.

Q(uit — Exits LIBRARY and saves the output file. When you select Q(uit, the prompt "Notice?" appears, giving you the chance to add a notice, such as a copyright statement, to the output file. To include a notice, type the information and press <return>. To save the file without a notice, simply press <return>. After the file is saved, the System promptline is displayed.

R(efs — (References) Lists the names of each entry in the segment references of all segments currently in the output file. The list also gives the names of all compilation units currently in the output file, even if the names do not occur in any of the segment references.

S(elect — Lets you select which modules to copy from the input file to the output file. For each code segment not already in the output file, LIBRARY prompts "Copy from slot #_?" To copy the module, press **Y**; to skip the module, press **N**; and to copy all modules beginning with the one specified, press **E**. If the corresponding slot in the output file is filled, the module is placed in the next available slot.

T(ext — Toggles to determine whether or not the interface sections of units are copied to the output file. (An interface section links a program to a unit by listing the procedures or variables that can be called.)

0-9(slot-to-slot — Transfers a specified slot in the input file to a specified slot in the output file. First, type a number from 0 through 9. The prompt "From slot #?" then appears with the cursor flashing after the number you typed. For a two-digit number, type the second digit now and press <return>. For a single-digit number, simply press <return>. Note that you can correct typing errors by pressing the <left-arrow> key before pressing <return>.

After you enter the slot number for the input file, you are asked for the output file. In response to the "To slot #?" prompt, type a one- or two-digit slot number and press <return>. If you press <return> without typing an output file slot number, no transfer occurs and the LIBRARY promptline is displayed.

6.1.1 Example for C(omp-unit, N(ew, and F(ill

Assume that you have the following program.

```
Program TEST;
USES SUPPORT;
Segment Procedure Seg;
.
.
.
Begin {TEST};
.
.
.
End.
```

Then, when LIBRARY asks for the input file, you type TEST. The following information is similar to what might appear.

```
Input file? TEST
  0 p TEST        xxxx
  1 s SEG         xxxx
```

Next, press **C** for C(omp-unit, and then enter TEST as the name of the compilation unit. TEST and SEG are now copied to the output file. At this point, press **N** for N(ew and enter the new input filename as SYSTEM.LIBRARY. The input file information would then show something similar to the following.

```
Input file? SYSTEM.LIBRARY
  0 u SUPPORT     xxxx
  1 u SPEECH      xxxx
  2 u SPRITE      xxxx
```

Now, if you press **F** for F(ill, the segment SUPPORT is copied to the output file since it is used by TEST.

SECTION 7: MODRS232

The MODRS232 (TI RS232 Interface device modification) utility lets you modify any of the software switch options for a device connected to the computer via the TI RS232 Interface. In addition, you can change the software switch options that control operations performed by the computer. By specifying any of these options, you can match the computer to the operating characteristics of the accessory attached to the interface unit. For a description of the software switch options and the correct specification format, refer to the TI RS232 Interface owner's manual.

Note: For your convenience, the Utilities diskette also contains the source program for MODRS232 so that you can change the characteristics of the RS232 Interface in a program. After you recompile the program, you must use the SETLTYPE utility (see the source program comments and Section 10 of this manual).

7.1 USER INSTRUCTIONS

To access the MODRS232 utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

```
What file?
```

If the Utilities diskette is in the first drive, type

```
#4:MODRS232
```

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After MODRS232 is loaded, the screen displays the following information and prompt.

```
Device attribute Modifier
  Printer:
  Remin:/Remout:
Which device:  [P, R, <esc> to escape]
```

To change the software switch options for a 9600-baud input/output device connected to the RS232 Interface, press **P.** For a 300-baud input or output device, press **R.** To leave MODRS232 and return to the System promptline, press <esc>.

If you press **P** or **R**, the screen displays the current software options for the indicated device. The first time you use MODRS232, the device name for PRINTER: is

RS232/2.BA=9600.DA=7.PA=O.EC

while the initial device name for REMIN:/REMOUT: is

RS232/1.BA=300.DA=7.PA=O.EC

Next, the program asks for the "new device name." To change the options, type the new name as a maximum of 43 characters, according to the format described in the TI RS232 Interface owner's manual. Then press <return>. Now the name is changed and the System promptline is displayed. To leave the device name unchanged, simply press <return>.

After pressing <return> to leave the program, you can press **U** for U(ser restart from the System promptline to run the program again. In this way, you can easily change more than one device name.

SECTION 8: PATCH

The PATCH utility allows you to view and alter files, and perform bit-manipulation and other such tedious (but sometimes useful) tasks.

There are two main facilities in PATCH: a mode for editing files at the byte level and a mode for dumping files in various formats. The byte-editing capability allows you not only to edit textfiles, but also to modify codefiles quickly and to create specialized test data. The dump capability provides formatted dumps in various radices and allows dumps from main memory.

8.1 USER INSTRUCTIONS

To access the PATCH utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

What file?

If the Utilities diskette is in the first disk drive, type

#4:PATCH

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After PATCH is loaded, the display shows the Edit mode promptline.

EDIT : D(ump, G(et, R(ead, S(ave, M(ix, T(ype, I(nfo, F(or, B(ack, ?

EDIT : V(iew, W(ipe, Q(uit, ?

The first promptline is displayed on one line; however, not all of the commands are within the boundaries of the screen. Press <screen right> to view the remaining prompts on the first line, or press **?** to see the second promptline. All of the commands are available, even if they are not currently visible.

If you have a single disk drive, remove the Utilities diskette at this point, and place your diskette in the drive. If you are using more than one drive, insert your diskette into the second or third drive.

When PATCH is first executed, you are in Edit mode. To perform an Edit option, type the appropriate letter and press <return>. **Note:** You can enter Dump mode by pressing **D**; you lose no information by switching back and forth between the two modes.

8.2 EDIT MODE

Edit mode allows you to open a file or device, read a selected block (specified by relative block number) into an edit buffer, view the buffer, modify it (via the Type mode), and write it back to the file.

The individual commands of Edit mode are explained below in alphabetical order, according to the letter you press to access them.

B(ack — (Backward) Reads the preceding block of the file into the buffer.

D(ump — Switches to Dump mode.

F(or — (Forward) Reads the next block of the file into the buffer.

G(et — Lets you open the file or device that you wish to use and then reads block zero into the buffer.

I(nformation — Displays information about the current file, including:

- The filename
- The file length
- The number of the current block
- The byte sex
- Whether or not the file is open
- Whether or not UNITREADs are allowed
- The device (unit) number (-1 if UNITIO is false)

M(ix — Changes the display format of the current block. Pressing **M** switches back and forth between the two formats.

Mixed — Displays printable ASCII characters and the hexadecimal equivalent of nonprintable characters.

Hex — Displays the block in hexadecimal digits.

Q(uit	Exits the PATCH utility.
R(ead	Reads a specified block from the current file.
S(ave	Writes the contents of the buffer to the current block.
T(ype	Enters Type mode, allowing you to edit the contents of the buffer.
V(iew	Displays the block currently in the buffer.
W(ipe	(Wipe display) Clears the block from the display.

Depending on the key you press, the program prompts you for the appropriate inputs. For example, if you press **G** for G(et, the following prompt is displayed.

FILENAME : <CR for Unit I/O>

Press <return> if you want to open one of the System files.

To open a diskette file, type a number sign, the device number, a colon, and the name of the file, such as

#5:DEMO.TEXT

and press <return>.

8.3 TYPE MODE

Type mode allows you to modify the contents of the buffer by moving the cursor on the screen and typing over existing information. If you decide that you don't want to keep the changes you have made in Type mode, don't save the buffer; instead, read the block over and try again.

Note: Type mode is interactive with the buffer. Any characters or hex codes you type in Type mode automatically replace the buffer contents in the position indicated by the cursor (a flashing rectangle), even though the buffer contents may not be displayed currently on the screen. For best results, press **V** for V(iew to display the buffer contents before pressing **T** for T(ype mode.

The promptline for Type mode is:

TYPE : C(har, H(ex, F(ill, U(p, D(own, L(eft, R(ight, <vector arrows>, Q(uit

The commands are explained here in alphabetical order, according to the letter you press to access them.

C(har	(Character mode) Exchanges bytes in the buffer for ASCII characters as you type them. Only printable characters are accepted. **Note:** You must press <etx> to escape from the C(har mode because Q is a printable character.
D(own	Moves the cursor down one row within the displayed block of data.
F(ill	Fills a portion of the current block with the same byte pattern. Either ASCII characters or hexadecimal digits are accepted for the pattern. When the block is filled, the cursor is positioned after the last byte.
H(ex	(Hex mode) Exchanges bytes in the buffer for hex digits as you type them. Press <etx> to return to the Edit mode. (Hex digits can be either upper- or lower-case letters.)
L(eft	Moves the cursor left one column within the displayed block of data.
Q(uit	Exits Type mode and returns to Edit mode.
R(ight	Moves the cursor right one column within the displayed block of data.
U(p	Moves the cursor up one row within the displayed block of data.
<vector arrows>	Operate like the arrow keys in the Editor. They perform the same actions as U, D, L, and R.
<etx>	Escapes from the C(haracter format in the Type mode.

Note that the cursor is always at a particular byte on the screen. When you move the cursor, it wraps around from side to side and from top to bottom, rather than moving off the screen.

8.4 DUMP MODE

You can generate dumps in decimal, hexadecimal, octal word, ASCII character (if printable), decimal byte (BCD), or octal byte format. Dump mode is also capable of flipping the bytes in a word before displaying it or of simultaneously displaying a line of words in both flipped and non-flipped form.

You can specify input to Dump mode from a diskette file or directly from main memory. The latter is primarily used to examine the Interpreter and/or the Basic Input/Output Subsystem (BIOS).

You can also specify the width of the output. A line may contain any number of machine words; nine words fill an 80-character line, and 15 fill a 132-character line.

When you enter Dump mode, the display shows a brief promptline:

D(o, Q(uit

and a list of format specifications. To modify a specification, type the letter of the item and then enter the specification, as follows:

A): Specifies the input volume (a diskette file or device).

B): Identifies the number of the block from which dumping starts. If A is a device, this number is not range-checked.

C): Indicates the number of blocks to print. If the number is too large, the dump stops when there are no more blocks to output.

E): Serves as a toggle. If True, a main memory dump is performed; if False, the file in A is dumped.

F): (Applies only if E is True.) Specifies an offset. The main memory dump may start with a byte that is past byte zero: 0 <= F <= 32767.

G): (Applies only if E is True.) Determines the number of bytes to print: 0 <= G <= 32767.

H): Opens the output file as a textfile.

I): Sets the width of the output line, in machine words: 1 <= I <= 15.

J): Displays each word as a decimal integer.

K): Displays each word as hexadecimal digits in byte order.

L): Displays each word as ASCII characters in byte order. Unprintable characters are displayed as hex digits.

M): Displays each word as an octal integer. M is the octal equivalent of J.

N): Displays each word as decimal bytes (BCD) in byte order.

O): Displays each word as octal digits in byte order.

S): Puts a blank line after the non-flipped version of a line.

T): Puts blank lines between different formats of a line.

Options J, K, L, M, N, and O have three associated Booleans that must be specified: USE, FLIP, and BOTH.

USE (First column) Specifies whether or not to use the format associated with that item.

FLIP Specifies whether or not to flip the bytes before displaying words in that format.

BOTH Specifies a simultaneous display of both flipped and non-flipped versions of the line. If BOTH is True, the value of FLIP is irrelevant.

When you have specified the file and format you want, press **D** for D(o to perform the dump. Pressing **Q** returns you to the Edit mode.

8.5 INPUT LIMITATIONS

All numbers accepted as input by PATCH are read as strings and then converted to integers. Only the first five characters of a string are processed, and any non-numeric character in the string is treated as a zero. If integer overflow occurs, the integer defaults to 32767. Since integer overflow can only be detected by the presence of a negative number, integers in the range 65536 to 98303 are effectively converted to modulo 32768.

8.6 EXIT FROM THE PATCH UTILITY

When you have completed your work with the PATCH utility, press **Q** for Q(uit when the Edit mode promptline is on the screen. The program immediately returns to the System promptline.

SECTION 9: RECOVER

The RECOVER utility attempts to recreate a damaged diskette directory. The program reads the existing directory, validates each entry it can find, removes erroneous entries, and lists the valid filenames on the screen. Then, if important files are still missing, you can instruct the program to search the diskette for files not accounted for in the partially reconstructed directory. If any codefiles or textfiles are found, RECOVER creates entries for them in the directory.

Note: Data files cannot be detected by RECOVER, since their format is not System defined. To recover data files, use the PATCH utility.

9.1 USER INSTRUCTIONS

To access the RECOVER utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

What file?

If the Utilities diskette is in the first disk drive, type

#4:RECOVER

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

RECOVER

After RECOVER is loaded, the program displays a series of prompts. If you have a single disk drive, remove the Utilities diskette at this point, and insert your program diskette in the drive. If you are using more than one drive, insert your program diskette into the second or third drive. Then respond to the following prompts.

RECOVER - Version II.0.6 ENTER TODAY'S DATE MM-DD-YY	Type a valid date and press <return>. If RECOVER finds any files that were not in the directory, the date you enter here is assigned to them. Entering an invalid date may cause RECOVER to stop with a value-range error. **Note:** Once a hyphen has been typed, it may not be backed over; thus, previous portions of the date may not be changed.
USER'S DISK IN DRIVE:	Type the number of the disk drive (4, 5, or 9) that contains the diskette to be recovered, and press <return>.
USER'S VOLUME ID:	Type the volume name to be recorded on the diskette. The name should be in upper-case letters. Lower-case letters are accepted as input; however, these are contrary to System standards and may cause unpredicatable results.
HOW MANY BLOCKS ON DISK?	This prompt is displayed only if the number of blocks recorded in the damaged directory is not a valid number. Your response should be 180.

At this point, RECOVER reads each entry in the diskette directory and checks its validity. Entries with errors are removed. Entries that are valid are saved, and RECOVER then displays a message similar to "ENTRY.NAME found" for each valid entry.

RECOVER now gives several yes/no prompts, which must be answered with upper-case letters. When all the directory entries have been checked and either saved or discarded, RECOVER prompts:

```
Are there still IMPORTANT files missing (Y/N)?
```

Yes Pressing **Y** in response to this prompt causes RECOVER to search the areas of the diskette that are still not accounted for by the partially reconstructed directory. If textfiles and codefiles are detected, RECOVER creates appropriate directory entries for them and lists them on the screen.

If RECOVER can't determine the original name of a file it has found, it creates a directory entry for DUMMY##.TEXT or DUMMY##.CODE (where ## represents two unique digits). If a codefile has a PROGRAM name, it is given that name; if this name would create a duplicate entry in the directory, digits are used (for example, RECOVER first restores SEARCH.CODE and then SEARCH02.CODE).

When RECOVER has completed a pass through the entire diskette, it prompts:

```
GO AHEAD AND UPDATE DIRECTORY (Y/N)?
```

Pressing **Y** causes the reconstructed directory to be saved. RECOVER then displays "WRITE OK" and terminates, returning to the System promptline. Pressing **N** exits RECOVER without doing anything.

No Pressing **N** in response to the "files missing" prompt causes RECOVER to prompt:

```
GO AHEAD AND UPDATE DIRECTORY (Y/N)?
```

Proceed as described above.

9.2 ANALYSIS OF RECONSTRUCTED DIRECTORIES

If RECOVER restores a textfile with an odd number of blocks, the end of the textfile was probably lost. Use the UCSD p-System Editor to make sure.

Recovered codefiles should be linked again if linking was necessary with the original file.

SECTION 10: SETLTYPE

The SETLTYPE (set language type) utility lets you indicate to the P-Code peripheral if a program segment that is being loaded includes assembly language.

With the p-System, if you load a program segment that does not contain assembly language, the segment is placed in either the computer's VDP memory or the Memory Expansion unit. In most cases, the location of the segment does not matter. However, some programs, such as the MODRS232 utility, actually modify the VDP memory in the computer. In this case, since only one pointer is maintained to indicate the address of the next byte to be read from VDP memory, the program should not be located in the VDP memory. If the program pointer to VDP memory is inadvertently changed, unpredictable results can occur.

To prevent this situation, use SETLTYPE to set a flag which tells the System that the program segment contains assembly language code. The segment is then loaded into the Memory Expansion unit.

10.1 USER INSTRUCTIONS

To access the SETLTYPE utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

```
What file?
```

If the Utilities diskette is in the first disk drive, type

```
#4:SETLTYPE
```

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After the program is loaded, you are asked

```
Object code file name?
```

Type the number sign, the number of the disk drive containing the diskette, a colon, and the name of the file containing the segments to be modified. Then press <return>. For example, entering #4:SYSTEM.LIBRARY indicates that you are going

to modify SYSTEM.LIBRARY, which is located in the first disk drive.

Next, the SETLTYPE promptline appears.

```
SETLTYPE: C(hange, L(ist, M(odify,?
```

The question mark at the end indicates that there are more commands than can fit onto one line of the display. Press **?** to display the additional promptline shown here.

```
SETLTYPE: Q(uit
```

10.2 C(HANGE

The C(hange command lets you either select another source file to modify or specify a different file to contain your program listing.

To access the C(hange command, press **C** from the SETLTYPE promptline. The following promptline then appears.

```
Change file: S(ource, L(isting, <esc>
```

These commands are explained here in the order in which they appear on the promptline. (After you enter the source or listing filename, the SETLTYPE promptline returns to the display.)

S(ource — Asks you for the object code file name. This is the same prompt displayed at the start of the program. Simply enter the name of the new file to be modified. With this command, you can modify as many files as you want without having to start SETLTYPE again.

L(isting — Lets you enter the name of the file where you want the listing to go. The prompt

```
Listing file name[CONSOLE:]?
```

is displayed. Simply type the name of the file for the listing, including the appropriate device number, and press <return>. **Note:** CONSOLE: is the default filename displayed when you start the program. Press <return> to accept CONSOLE: as the listing file.

<esc> — Leaves the C(hange command and returns the SETLTYPE promptline to the display.

10.3 L(IST

The L(ist command provides information about all the segments in the current file. The information given is the segment number, name, length, and type, and whether or not the segment contains assembly language.

To access the L(ist command, press **L** from the SETLTYPE promptline. After the listing is complete, the file information, similar to the following, is displayed.

```
SETLTYPE: C(hange, L(ist, M(odify,?
Segment listing of (filename)

 # Name      Leng  Seg   Code
 - ----      ----  ---   ----
 0 MAIN       158  UNIT  PSEUDO
 1 PRNTIT     200  UNIT  M_9900
 .
 .
 .
```

The information given is explained here.

#	A list showing each segment's number, according to SETLTYPE.
Name	The name of each segment.
Leng	The length (in words) of each segment.
Seg	Segment type. This can be NONE, PROG, UNIT, PROC, or SEPRT.

NONE	An empty segment "slot."
PROG	A program segment.
UNIT	A unit segment.
PROC	A separate routine segment.
SEPRT	An assembled segment.

Code	The type of code in the segment. This can be PSEUDO or M_9900.

PSEUDO	A segment consisting only of pseudo-code (p-code).
M_9900	A segment containing some assembly language code, which must therefore be loaded into the Memory Expansion unit.

10.4 M(ODIFY

The M(odify command lets you set the language code type of segments in a file. You can choose either to go through a prompted menu of all the segments or to change one segment at a time.

To access the M(odify command, press **M** from the SETLTYPE promptline. The following promptline then appears.

```
Modify: P(rompted, S(ingle, Q(uit
```

These commands are explained here in the order in which they appear on the promptline.

P(rompted — Displays the name and code type of the segments, one at a time, and gives you the opportunity to change the type. After the last segment is displayed, the program tells you the number of segments in the file and returns to the SETLTYPE promptline.

S(ingle — Allows you to select a single segment according to its number and change its type. After you complete a segment, the SETLTYPE promptline returns to the display.

Q(uit — Leaves the M(odify command and returns the SETLTYPE promptline to the display.

For each segment, the screen shows the segment name and its code type (PSEUDO or 9900), followed by these choices:

```
      1- Pseudo
      2- 9900
  <esc>- No Change
Set to which type?
```

To set a flag which indicates that the segment contains only pseudo code, press **1.** To set a flag which indicates that the segment includes some assembly language code, press **2.** To leave the flag unchanged, press <esc>.

10.5 Q(UIT

The Q(uit command leaves the SETLTYPE utility.

To access the Q(uit command, press **Q** from the SETLTYPE promptline. The System promptline then appears.

SECTION 11: SETUP

Although the P-Code peripheral is designed specifically to be used with the TI-99/4 or TI-99/4A Home Computer, you may occasionally need to use the p-System with a different computer terminal. If so, the SETUP utility lets you change information related to the terminal, such as the arrow keys, the Editor accept <etx>, and the Editor escape <esc>, by creating a MISCINFO file.

When the System is booted, it automatically reads a file called SYSTEM.MISCINFO which exists in the P-Code peripheral. However, you can override the P-Code file if a file named SYSTEM.MISCINFO is located in the first disk drive. The System then uses the SYSTEM.MISCINFO file for the terminal's characteristics.

SYSTEM.MISCINFO contains the following three types of information.

- Miscellaneous data about the System.
- General information about the computer terminal.
- Specific information about the terminal's various control keys.

The general procedure for changing the terminal's characteristics is as follows. First, load SETUP and make the necessary changes to the characteristics. Next, select the D(isk Update option of the Q(uit command (see Section 11.4) to leave SETUP and create a file named NEW.MISCINFO. Now use the Filer's C(hng command (see the UCSD p-System Filer manual) to change the name of the existing SYSTEM.MISCINFO file, if any, to OLD.MISCINFO. Then use the C(hng command again to change NEW.MISCINFO to SYSTEM.MISCINFO. The next time you initialize or boot the System, the characteristics are as you defined them.

11.1 USER INSTRUCTIONS

To access the SETUP utility, press **X** for X(ecute from the System promptline. Next, the program displays the following prompt.

What file?

If the Utilities diskette is in the first disk drive, type

#4:SETUP

and press <return>. If the Utilities diskette is in the second or third disk drive, include the appropriate prefix with the filename as explained in Section 1.2.

After the program is loaded and initialized, the SETUP promptline appears.

SETUP: C(HANGE T(EACH H(ELP Q(UIT [D4]

11.2 C(HANGE

The C(HANGE command lets you go through a prompted menu of all the items or change one data item at a time. In either case, the current values are displayed, and you have the option of changing them. If you are using SETUP for the first time, the values given are the system defaults.

As each item is displayed on the screen, its current value and the question "WANT TO CHANGE THIS VALUE? (Y, N, !)" also appear. Press **Y** to change the value, **N** to leave the value as it is, or **!** to return to the SETUP promptline.

If you press **Y**, SETUP prompts you for the new value. For numeric- or character-type items, the octal, decimal, hexadecimal, ASCII, and control default values are given. To change these items, press the new function key itself and then press <return>. If you prefer, you can type the ASCII mnemonic or the numeric value and press <return>. When entering a numeric value, keep in mind that the default radix is decimal. To enter an octal or hexadecimal value, first type O or H, respectively, followed by the value.

Boolean-type items are shown with a default of TRUE or FALSE. To change these items, simply type TRUE or FALSE and press <return>.

After the new value for the item is entered, you are asked again if you want to change it. This prompt gives you the opportunity to correct typing errors or change your mind.

To access the C(HANGE command, press **C** from the SETUP promptline. The following promptline then appears.

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

These commands are explained here in alphabetical order, according to the letter you type to access them.

H(ELP)	Gives a brief explanation of the other commands on the CHANGE promptline.
L(IST)	Lists the current values for all items (see Section 11.2.1).
P(ROMPTED)	Displays the items, one at a time, and gives you the opportunity to change them (see Section 11.2.1 for a list of the items).
Q(UIT)	Leaves the C(HANGE command and returns the SETUP promptline to the display.
R(ADIX)	Lets you change the default radix from decimal to octal or hexadecimal.
S(INGLE)	Allows you to select a single item and change its value (see Section 11.2.1).

11.2.1 Terminal Characteristics

The items available for changing a terminal's characteristics are listed in this section, along with an explanation of each one, including the default value displayed when SETUP is run for the first time.

BACKSPACE	Moves the cursor one space to the left. Default: ASCII BS (decimal 8)
EDITOR ACCEPT KEY	Exits a command in the Editor and accepts the changes. Default: ASCII ETX (decimal 3)

SETUP

EDITOR ESCAPE KEY	Exits a command in the Editor and ignores the changes. Default: ASCII ESC (decimal 27)
EDITOR EXCHANGE-DELETE KEY	Deletes a single character in the Editor's X(ch (exchange) mode. Default: decimal 131
EDITOR EXCHANGE-INSERT KEY	Inserts a single character in the Editor's X(ch (exchange) mode. Default: decimal 132
ERASE LINE	Erases all the characters on the line where the cursor is positioned. Default: ASCII NUL (decimal 0)
ERASE SCREEN	Erases the entire screen. Default: ASCII NUL (decimal 0)
ERASE TO END OF LINE	Erases all characters from the current cursor position to the end of the same line. Default: decimal 141
ERASE TO END OF SCREEN	Erases all characters from the current cursor position to the end of the screen. Default: decimal 142
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. In general, it is TRUE only for implementations in which the IPC (Instruction Program Counter) is segment-relative. Default: TRUE
HAS CLOCK	May be TRUE or FALSE. If your hardware has a line frequency (60 Hz) clock module, setting this item to TRUE allows the System to optimize disk directory updates. If your hardware doesn't have a clock, this must be FALSE. Default: TRUE
HAS LOWER CASE	May be TRUE or FALSE. The value should be TRUE if you have lower-case letters and want to use them. Default: TRUE
HAS RANDOM CURSOR ADDRESSING	May be TRUE or FALSE. If your terminal is not a CRT, this should be FALSE. Default: TRUE

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. Default: FALSE
KEY FOR BREAK	Stops an executing program with a runtime error. Default: decimal 130
KEY FOR FLUSH	Toggles to stop and start writing output to the screen. Default: decimal 135
KEY FOR STOP	Suspends the program until this key is pressed again. Default: decimal 142
KEY TO ALPHA LOCK	Acts as a toggle to convert lower-case letter input to upper-case and back again. Default: decimal 140
KEY TO DELETE CHARACTER	Moves the cursor one character to the left, deleting characters as it goes. ASCII BS (decimal 8)
KEY TO DELETE LINE	Deletes the line where the cursor is currently positioned. Default: decimal 143
KEY TO END FILE	Sets the intrinsic Boolean function EOF to TRUE when pressed while reading from the System input files. Default: ASCII ETX (decimal 3)
KEY TO MOVE CURSOR DOWN	Moves the cursor down one line. Default: decimal 138
KEY TO MOVE CURSOR LEFT	Moves the cursor to the left one character. Default: decimal 136
KEY TO MOVE CURSOR RIGHT	Moves the cursor to the right one character. Default: decimal 137
KEY TO MOVE CURSOR UP	Moves the cursor up one line. Default: decimal 139

SETUP

LEAD IN FROM KEYBOARD	Specifies a prefix which is the first character in a two-character sequence generated by pressing certain keys. This prefix must be the same for all such sequences. Note that this character is only accepted as a lead in for characters where you have set PREFIXED [(itemname)] to TRUE. Default: ASCII NUL (decimal 0)
LEAD IN TO SCREEN	Specifies a prefix which is the first character in a two-character sequence required to activate certain functions. If the first character in all these sequences is the same, this data item can specify it. The prefix is only generated as a lead in for characters where you have set PREFIXED [(itemname)] to TRUE. Default: ASCII NUL (decimal 0)
MOVE CURSOR HOME	Moves the cursor to the upper left-hand corner of the screen (position (0,0)). Default: decimal 130
MOVE CURSOR RIGHT	Moves the cursor to the right one character. Default: decimal 137
MOVE CURSOR UP	Moves the cursor up one line. Default: decimal 139
NON PRINTING CHARACTER	Defines the character that is displayed on the screen when a non-printing character is typed or sent to the terminal while you are using the Editor. Default: ASCII ? (decimal 63)
PREFIXED [(itemname)]	Determines whether or not the System recognizes a two-character sequence generated by a key or sent to the screen. Set PREFIXED [(itemname)] to TRUE to recognize a sequence. All defaults: FALSE

SCREEN HEIGHT	Specifies the number of lines on your display screen. If you are using a hardcopy terminal, this value should be set to 0. Default: decimal 24
SCREEN WIDTH	Specifies the number of characters on one line of your display. Default: decimal 40
STUDENT	May be TRUE or FALSE. On IV.0 Systems, this value should always be FALSE. Default: FALSE
VERTICAL MOVE DELAY	May be a decimal integer from 0 through 11. Many terminals require a delay after vertical cursor movements to allow 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: decimal 0

11.3 H(ELP

The H(ELP command describes each of the commands that are available on the SETUP promptline.

To access the H(ELP command, press **H** from the SETUP promptline.

11.4 Q(UIT

The Q(UIT command lets you save the new characteristics on the diskette or in memory and then leave the SETUP utility.

To access the Q(UIT command, press **Q** from the SETUP promptline. The QUIT promptline then appears.

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

These commands are explained here in alphabetical order, according to the letter you type to access them.

D(ISK) UPDATE — Creates the file NEW.MISCINFO on the Utilities diskette. This file contains the current characteristics specified in SETUP. Remember, the Filer must be used to change the name of this file to SYSTEM.MISCINFO before it can be accessed during booting or initializing.

E(XIT) — Leaves SETUP and returns the System promptline to the display.

H(ELP) — Gives a brief explanation of the other commands on the QUIT promptline.

M(EMORY) UPDATE — Stores the current characteristics specified in SETUP in memory until you exit from the System.

R(ETURN) — Returns you to the SETUP promptline in case you selected the Q(UIT) command accidentally.

11.5 T(EACH

Describes the use of SETUP, concentrating mainly on input formats. We recommend that you review this explanation the first time you use SETUP.

At the end of each screen of information, you are asked if you want to continue or quit. Pressing **C** for C(ONTINUE) displays the next screen in T(EACH, and pressing **Q** for Q(UIT) leaves T(EACH and displays the SETUP promptline.

SECTION 12: IN CASE OF DIFFICULTY

1. Be sure that the diskette you are using is the correct one. Use the L(dir (list directory) command in the Filer to check for the correct diskette or program.

2. Ensure that your Memory Expansion unit, P-Code peripheral, and Disk System are properly connected and turned on. Be certain that you have turned on all peripheral devices and have inserted the appropriate diskette before you turn on the computer.

3. If your program does not appear to be working correctly, end the session and remove the diskette from the disk drive. Reinsert the diskette, and follow the "Set-Up Instructions" carefully. If the program still does not appear to be working properly, remove the diskette from the disk drive, turn the computer and all peripherals off, wait 10 seconds, and turn them on again in the order described above. Then load the program again.

4. If you are having difficulty in operating your computer or are receiving error messages, refer to the "Maintenance and Service Information" and "Error Messages" appendices in your User's Reference Guide or UCSD p-System P-Code manual for additional help.

5. If you continue to have difficulty with your Texas Instruments computer or the UCSD p-System Utilities package, please contact the dealer from whom you purchased the unit or program for service directions.

THREE-MONTH LIMITED WARRANTY
HOME COMPUTER SOFTWARE MEDIA

Texas Instruments Incorporated extends this consumer warranty only to the original consumer purchaser.

WARRANTY COVERAGE

This warranty covers the case components of the software package. The components include all cassette tapes, diskettes, plastics, containers, and all other hardware contained in this software package ("the Hardware"). This limited warranty does not extend to the programs contained in the software media and in the accompanying book materials ("the Programs").

The Hardware is warranted against malfunction due to defective materials or construction. **THIS WARRANTY IS VOID IF THE HARDWARE HAS BEEN DAMAGED BY ACCIDENT OR UNREASONABLE USE, NEGLECT, IMPROPER SERVICE OR OTHER CAUSES NOT ARISING OUT OF DEFECTS IN MATERIAL OR CONSTRUCTION.**

WARRANTY DURATION

The Hardware is warranted for a period of three months from the date of original purchase by the consumer.

WARRANTY DISCLAIMERS

ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE THREE-MONTH PERIOD. TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR LOSS OF USE OF THE HARDWARE OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES, OR DAMAGES INCURRED BY THE CONSUMER OR ANY OTHER USER.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you in those states.

LEGAL REMEDIES

This warranty gives you specific legal rights, and you may also have other rights that vary from state to state.

PERFORMANCE BY TI UNDER WARRANTY

During the three-month warranty period, defective Hardware will be replaced when it is returned postage prepaid to a Texas Instruments Service Facility listed below. The replacement Hardware will be warranted for a period of three months from date of replacement. TI strongly recommends that you insure the Hardware for value prior to mailing.

TEXAS INSTRUMENTS CONSUMER SERVICE FACILITIES

Texas Instruments Service Facility
P. O. Box 2500
Lubbock, Texas 79408

Geophysical Services Incorporated
41 Shelley Road
Richmond Hill, Ontario, Canada L4C5G4

Consumers in California and Oregon may contact the following Texas Instruments offices for additional assistance or information.

Texas Instruments Consumer Service
831 South Douglas Street
El Segundo, California 90245
(213) 973-1803

Texas Instruments Consumer Service
6700 Southwest 105th
Kristen Square, Suite 110
Beaverton, Oregon 97005
(503) 643-6758

IMPORTANT NOTICE OF DISCLAIMER REGARDING THE PROGRAMS

The following should be read and understood before purchasing and/or using the software media.

TI does not warrant the Programs will be free from error or will meet the specific requirements of the consumer. The consumer assumes complete responsibility for any decisions made or actions taken based on information obtained using the Programs. Any statements made concerning the utility of the Programs are not to be construed as express or implied warranties.

TEXAS INSTRUMENTS MAKES NO WARRANTY, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE PROGRAMS AND MAKES ALL PROGRAMS AVAILABLE SOLELY ON AN "AS IS" BASIS.

IN NO EVENT SHALL TEXAS INSTRUMENTS BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE PURCHASE OR USE OF THE PROGRAMS AND THE SOLE AND EXCLUSIVE LIABILITY OF TEXAS INSTRUMENTS, REGARDLESS OF THE FORM OF ACTION, SHALL NOT EXCEED THE PURCHASE PRICE OF THE SOFTWARE MEDIA. MOREOVER, TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR ANY CLAIM OF ANY KIND WHATSOEVER BY ANY OTHER PARTY AGAINST THE USER OF THE PROGRAMS.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you in those states.