Radio Shack

TRS-80

DISK/VIDEO INTERFACE

TRS-80®

DISK/VIDEO INTERFACE

CUSTOM MANUFACTURED FOR RADIO SHACK, A DIVISION OF TANDY CORPORATION

Contents

PART III/FORMAT OF THE DISKETTE

PART IV/APPENDICES

Introduction

Congratulations on selecting Radio Shack's Disk/Video Interface for use with your TRS-80 Portable Computer. Even though a CRT or a TV set is not necessary when working with the Portable Computer, using this Disk/Video Interface will help you realize the full potential of your Portable Computer as a home or business computer.

The Disk/Video Interface contains a 5-1/4 inch Floppy Disk Drive interface and an interface for connecting a Video Monitor or TV set.

- **Floppy Disk Drive Unit.** Allows you to use a 5-1/4 inch double-density floppy disk with your Portable Computer.
- **Floppy Disk Interface.** Controls the standard built-in Floppy Disk Drive Unit and can control an optional single-sided, double-density, 5-1/4 inch Floppy Disk Drive Unit.
- **Video Interface.** Allows connection to a Video Monitor or, with the built-in RF converter, to any standard television set with the cable and switch box supplied.

For further expansion of your Portable Computer system, Radio Shack offers the following optional equipment:

- A Floppy Disk Drive Unit to expand storage capacity.

About this manual.....

For your convenience, we've divided this manual into four sections.

Part I gives general information about the Disk/Vido Interface and should get you started with Disk-BASIC. Also, with the CRT connected to the Portable Computer, you will learn how to transfer the characters on the Liquid Crystal Display (LCD) of the Portable Computer to the monitor.

Part II provides detailed information on Disk-BASIC commands that drive the CRT and the floppy disk and utility programs to use the floppy disk.

Part III describes the file structure and format of the floppy disk which can be used for more detailed manipulation of diskette files.

Part IV, the Appendices, provides technical information that enables you to use the Disk/Video Interface more effectively.

We suggest you read this manual thoroughly. After you become familiar with the Disk/Video Interface, the Disk/Video Interface Quick Reference Guide will help keep you "up-and-running"

PART I/
GENERAL INFORMATION

1/Description of the Disk/Video Interface

Open the package and take out the Disk/Video Interface. Do not throw away the packing material or the box. They may be useful if you ever need to transport the Disk/Video Interface.

The Disk/Video Interface package includes:

- A Disk/Video Interface
- This Owner's Manual
- A Quick Reference Guide
- A System Diskette
- Cable and Switch Box for Standard TV Connection
- Cable for Connection to Portable Computer
- Adapter Connector for the I/O Bus on the Portable Computer
- Replacement Compartment Cover for the Portable Computer

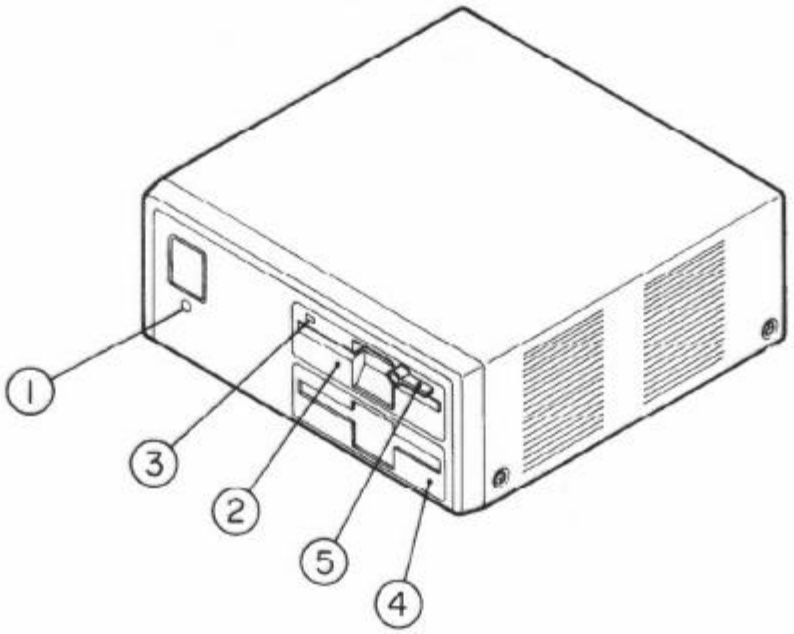


Figure 1-1. Disk/Video Interface (Front view)

(1) **LED Power Indicator.** Lights up when the Power Switch is on.

(2) **Drive 0.** This is the disk drive unit for the BASIC SYSTEM diskette.

(3) **Drive Select LED.** During access of the diskette, this LED lights.

(4) **Optional Disk Cover.** If you wish to add an optional disk drive unit, a qualified Radio Shack service technician will remove this cover when he does the upgrade for you.

(5) **Clamp Lever.** Turning this lever downward locks the disk drive unit into the operating position.

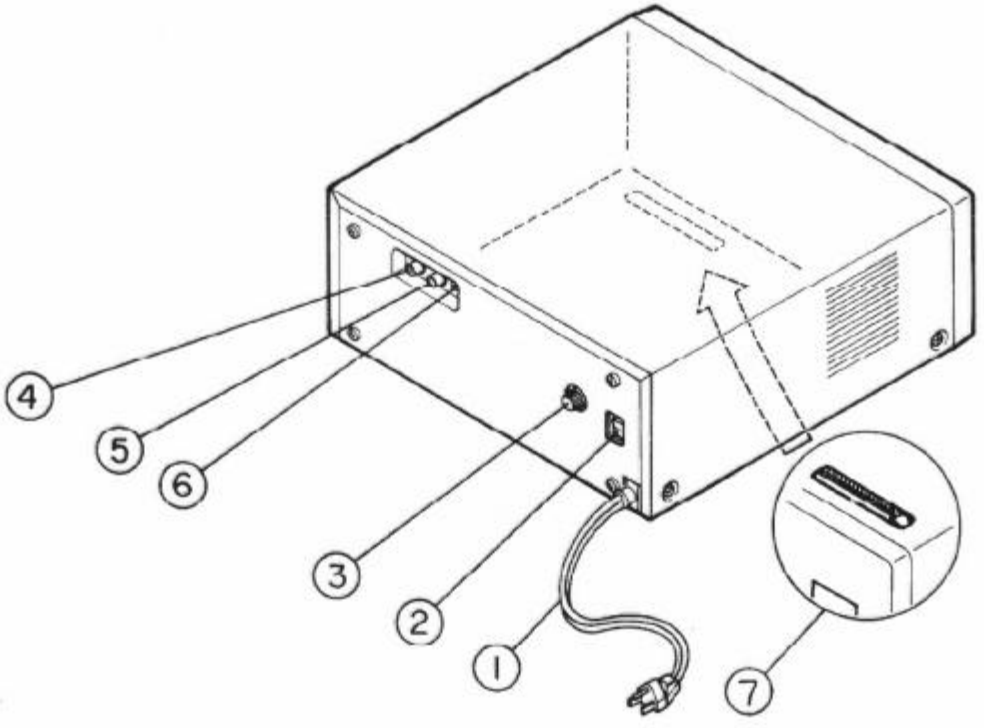


Figure 1-2. Disk/Video Interface (Rear view)

(1) **AC Power Cord.** Supplies AC power source to the Disk/Video Interface.

(2) **Power Switch.** Turn this switch on to supply AC power to the Disk/Video Interface.

(3) **Fuse Holder.** Contains a 250V/1A fuse. Remove the AC cord from the AC receptacle while inspecting/replacing the fuse.

(4) **Video Monitor Terminal.** Connect your video monitor for a 80 x 25 or 40 x 25 line display.

(5) **Home TV Terminal.** Provides RF output modulated to Channel 3 or Channel 4 of the TV frequency. Connect your home TV set to this terminal using the TV cable and switch box supplied.

(6) **Channel 3/Channel 4 Exchange Switch.** Select either Channel 3 or Channel 4 RF output, whichever is not used in your area.

(7) **System Bus Connector.** Connect the system bus connector of the Portable Computer using the attached cable.

2/Diskette

Always, handle your diskettes carefully. Take the same precautions you use with your music cassettes and high-fidelity phonograph records. A small indentation, dust particle or scratch can render all or part of diskette unreadable — permanently.

- Keep the diskette in its storage envelope whenever it is not in use.
- Always remove a diskette from the drive before turning the system on or off.
- Keep diskettes away from magnetic fields (transformers, AC motors, magnets, TVs, radios, etc.). Strong magnetic fields will erase data stored on a diskette.
- Handle a diskette by the jacket only. Do not touch any of the exposed surfaces. Do not try to wipe or clean the diskette surface; it scratches easily.
- Keep diskettes out of direct sunlight and away from heat.
- Avoid contaminating diskettes with cigarette ashes, dust or other particles.
- Do not write directly on the diskette jacket with a hard-point device, such as a ball-point pen or lead pencil. Use a felt-tip pen only.
- Store diskettes in a vertical file folder on a shelf where they are protected from pressure to their flat sides (just as phonograph records are stored).
- In very dusty environments, you may need to provide filtered air to the room where you use your computer.

Tips on Labeling Diskettes

Each diskette has a permanent label on its jacket. This label is for "vital staistics" that will never change. For example, to keep track of diskettes, it's a good idea to assign a unique number to each diskette. Write such a number on the permanent label. You might also put your name on the diskette, and record the date when the diskette was first put into use. Remember, use only a felt-tip pen for marking.

This "permanent" label is not a good place to record the contents of the diskette, since you may want to change the contents, and you do not want to be erasing or scratching out information on this label.

Figure 2-1. A Diskette

(1) **Storage Envelope.** While a diskette is not in use, keep it here.

(2) **Write Protect Notch.** When this notch is covered, the disk drive cannot write (change information) on the diskette. Do not pinch the tab into the notch when you apply it. If the tab becomes indented, the disk drive may not sense that the disk is write-protected. Leave the notch uncovered if you want to save or change information on the diskette.

(3) **Jacket.** The diskette is permanently sealed inside this protective jacket. Do not attempt to remove it.

(4) **Read/Write Window.** The disk drive accesses the diskette surface through this window. Do not touch the diskette surface.

(5) **Label.** To write on this label, use only a felt-tip pen. Any other writing instrument might damage the diskette.

3/Installation

Connecting to the portable Computer

Connect the Portable Computer and the Disk/Video Interface using the connection cable referred to in Figure 3-1. Make sure that cable connectors are firmly connected. Loose connections will cause trouble since data is transferred through this cable.

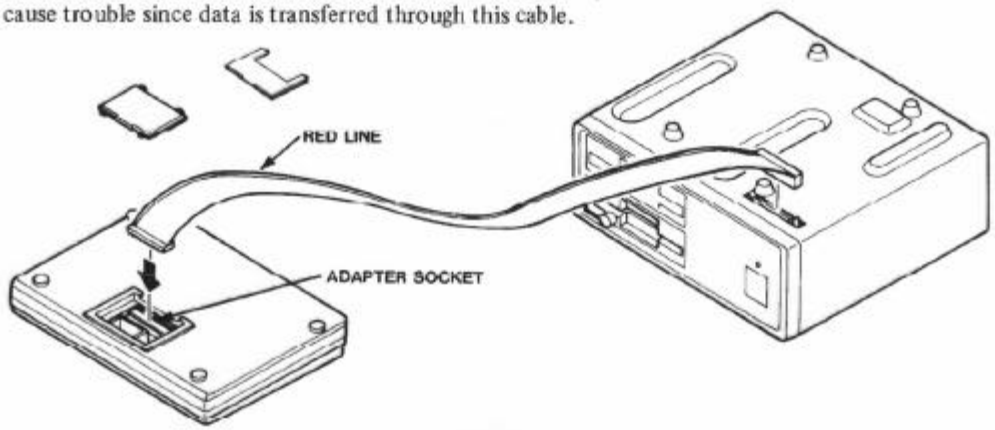


Figure 3-1. Cable Connection to the Portable Computer

1. Remove the compartment cover located on the bottom case or rear of the Portable Computer using a coin, etc.

2. You will find two integrated circuit (IC) sockets. Connect the adapter socket supplied with the unit into the upper IC socket. Be careful not to connect it to the other socket. Pull the release lever on the adapter socket up and insert one side of the cable connector into the adapter socket, then push the release lever down into the locked position. Make sure that pins 1-40 of the cable connector are connected to their corresponding numbers on the IC socket. You can find these numbers marked in white on the printed circuit board.

Important Notice: When you are going to insert the adapter socket or cable connector into the IC socket, apply force evenly on the adapter or cable connector. If the force is not applied evenly, pins on the connector may be damaged. If you mistakenly damage pins, carefully straighten the pins using pliers. Be careful – repeated bending and straightening of the pins will cause them to break off.

When connection is completed, attach the replacement compartment cover supplied with this unit to the bottom of the Portable Computer.

Always use this replacement cover and remove it only when connecting/disconnecting the Disk/Video Interface. This cover keeps out dust and dirt, which can cause poor connection to the IC socket: it also prevents the cable from being accidentally disconnected.

Once you have installed the adapter socket, leave it in the IC socket for ease of re-insertion and removing the connector cable.

3. Connect the other side of the cable to the System Bus Connector located on the bottom side of the Disk/Video Interface. Make sure the guide and guide slot match to insure correct connection.

Connecting the Video Monitor

To connect the Video Monitor, use the connection cable shown in Figure 3-2.

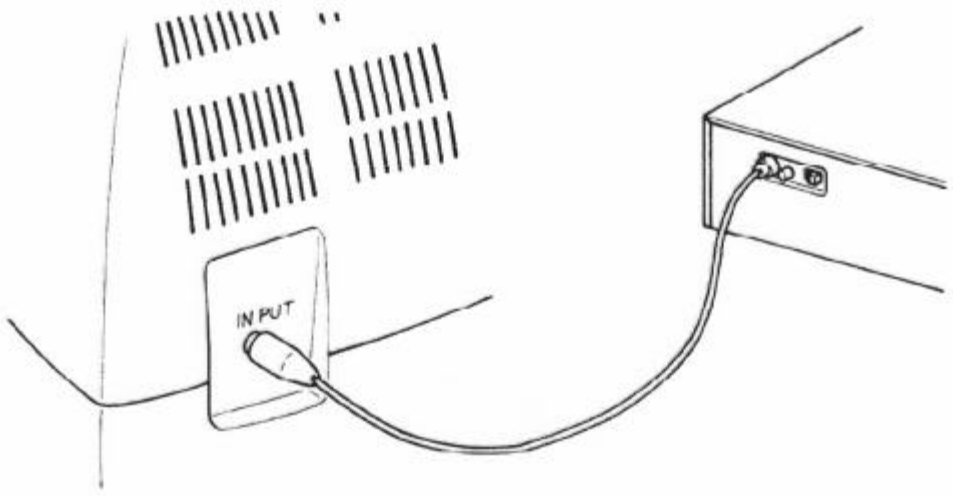


Figure 3-2. Connecting the Video Monitor

Connecting the Television Set

Connect the Disk/Video Interface to your home TV set using the cable connector and switch box as illustrated in Figure 3-3.

Set the Channel 3/Channel 4 Exchange Switch to the channel that is not being used by a broadcast station in your area.

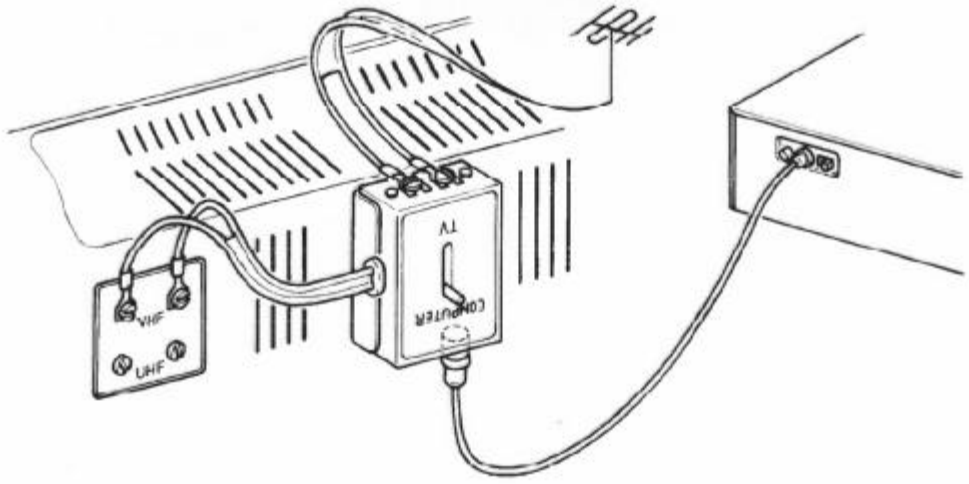


Figure 3-3. Connecting the Television Set

4/System Start-Up

Connect all cables according to the instructions in Chapter 3. Check that all the connections were made correctly. If you are ready, start the Disk/Video Interface according to the procedure below:

Turning the Power On

1. Turn the Power Switch of the Portable Computer on.

2. Turn the Power Switch of the Disk/Video Interface on.

This power up sequence is very important. If you do it in reverse, the unit will not function properly.

A message appears on the CRT for a few seconds:

```
Please wait !
```

Another message appears:

```
Insert system diskette.
```

and the Disk Select LED for Drive 0 lights.

Loading Portable Computer Disk Operating Software (DOS) into the Disk/Video Interface

1. Insert the System Diskette into Drive 0 with the label edge up. Be sure the edge with the write protect notch is the first to enter the drive.

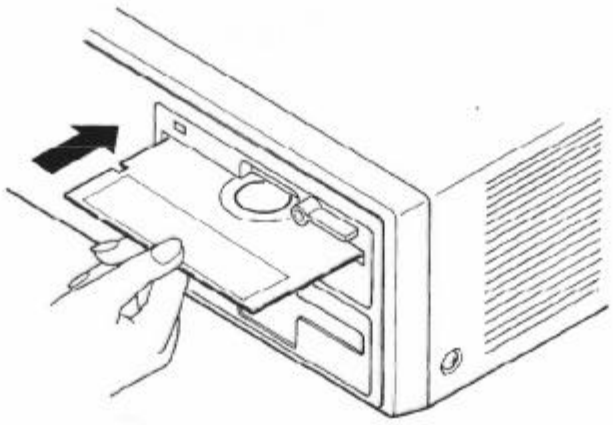


Figure 4-1. Inserting the System Diskette

2. Confirm that the diskette is fully inserted in the drive unit.

3. Turn the Clamp Lever downward.

A new message `Now, getting ready.....` appears and then the following message appears on the CRT:

```
CORP. 1983, MICROSOFT, CORP.
   ALL RIGHTS RESERVED
LICENSED TO TANDY CORPORATION
     VERSION 01.00.00
```

and the disk select LED turns off. If this message does not appear, you probably didn't turn the Clamp Lever correctly. Release the Clamp Lever once, and turn it downward again.

If the above message does not appear even after the operation above, chances are that the diskette inserted in Drive 0 is not the system diskette. If so, the message:

```
In Drive0 is NOT a SYSTEM DISKETTE.
```

will appear. Check the diskette and re-insert system diskette.

If any mulfunction has occurred in Drive 0, the following message:

```
There is no drive.
```

will appear.

Through these steps, all the programs (i.e., DOS) necessary to control the Disk and CRT are stored in the RAM of the Disk/Video Interface.

Loading Disk-BASIC into the Portable Computer

Before you can use Disk-BASIC, you have to transfer/load the Disk-BASIC program recorded on the System Diskette into the RAM of the Portable Computer.

There are two ways to do this. Whichever step you take, you must first load DOS into the RAM of the Disk/Video Interface. Then:

- Warm start
  Turn the Power Switch of the Portable Computer off and then on again. Or, if you prefer, simply press the RESET switch.

- Cold start (see "Important Notice" below)
  Press and hold (CTRL) (PAUSE) and the RESET switch, in that order. Release the RESET switch while continuing to press down the (CTRL) and (PAUSE) keys. When the Main Menu reappears on the LCD display, release the keys.

When you start the system (either warm or cold), the Disk Select LED turns on for a moment.

Important Notice: When you do a cold start, all the programs and data stored in the RAM of the Portable Computer are erased. You have to save programs and data to cassette tape in advance or you will lose everything you had stored in memory.
A warm start will not erase the stored data but it requires at least 4,500 free bytes. If there are not, you must either KILL some file(s) or cold start is required.

Once you have loaded Disk-BASIC into the Portable Computer, it is kept there unless you clear all the memory (cold start). But since Disk-BASIC is not loaded as "file", it does not appear as a file in the menu of the Portable Computer.

Displaying Characters on the CRT

After you've loaded DOS and Disk-BASIC into the RAM of the Disk/Video Interface and the Portable Computer, you are ready to change the display device from the LCD of the Portable Computer to the CRT (except Menu, which is always displayed on the LCD).

1. Move the Cursor to BASIC on the Menu and press (ENTER). The Portable Computer is now set in BASIC Mode.

2. Type:

   SCREEN 1,1 (ENTER)

   When you enter this command, the display of the Portable Computer is switched to the CRT. On the bottom line of the CRT, a LABEL line indicating Function Keys from F1 to F8 is displayed. If the LABEL line is not required, you can just type

   SCREEN 1 (ENTER)

   The LABEL line can be displayed or removed using the LABEL key just like on the LCD.

   If you want to move the display to the LCD again, type the command:

   SCREEN 0,1 (ENTER)

3. If you want to display characters in the 80-column x 25-line mode, type the command (in BASIC mode):

   WIDTH 80 (ENTER)

   To return to the 40-column x 25-line display, type:

   WIDTH 40 (ENTER)

CAUTION:

When you disconnect your Portable Computer from the Disk/Video Interface, the system's power must be **OFF**.

When you turn the system's power off, the system must be in menu mode. Make it a habit to press (F8) before turning the power off.

5/Making a BACKUP of Disk-BASIC

A BACKUP duplicates information from one diskette onto another diskette.

Your Disk-BASIC diskette contains utility programs called FORMAT, BACKUP. SNG and BACKUP. FORMAT is for initialization, BACKUP. SNG is for a single-drive system BACKUP, and BACKUP is for a two-drive system BACKUP.

To make a BACKUP, the first step is to format a blank diskette.

Making a Data Diskette (FORMAT)

The FORMAT utility program takes a diskette and initializes or "formats" it.

If the diskette was previously formatted, all prior information will be lost.

1. In the Disk-BASIC mode, type:

   RUN"Ø:FORMAT" (ENTER)

2. Disk-BASIC will start the formatter program and ask you a series of questions:

```
This utility formats diskettes.
  - All data will be lost -

Which drive will be used (Ø or 1)?
```

3. If you are using a single-drive system, remove the System Diskette from Drive 0 and insert a blank diskette; then type:

   Ø (ENTER)

   If you are using a two-drive system, insert a blank diskette into Drive 1; then type:

   1 (ENTER)

4. Next, Disk-BASIC will ask you:

```
Put the diskette to be formatted in
        Drive X

Press ENTER when ready.
```

   Press (ENTER)

5. Disk-BASIC will now format the diskette. After formatting, Disk-BASIC will display the message:

```
FORMAT COMPLETE;
number of flawed tracks: Ø
```

6. If you are using a single-drive system, remove the formatted diskette from Drive 0 and insert the Disk Operating System diskette.

Single-Drive BACKUP (BACKUP. SNG)

The DOS diskette will be referred to as the ***Source*** , and the blank one you just formatted will be called the ***Formatted*** during BACKUP.

1. Start Disk-BASIC as explained in the previous section.
   The copyright message will be displayed, for example:

```
TRS-80 MODEL 100 software
Corp. 1983 Microsoft
XXXXX Bytes Free
```

   A BACKUP requires at least 8500 free bytes. If the number of free bytes is less than 8500, BACKUP cannot be started. In this case, you will have to increase the number of free bytes.

   To do this, save some or all the files in the Portable Computer to cassette tape. Then you can load BACKUP.

2. Type: `RUN"Ø:BACKUP.SNG"` (ENTER)

3. Disk-BASIC now loads and starts BACKUP.SNG; it will then ask you:

```
SINGLE DRIVE BACKUP UTILITY

COPY ALL (Entire Diskette) or
SYSTEM (System Files Only)

Enter A (ALL) or S (SYSTEM)?
```

4. Type: `A` (ENTER)

5. Next, Disk-BASIC asks:

```
 TO COPY ALL:
Put the source diskette in the drive.
Press ENTER when ready.
```

6. Press: (ENTER)

7. Disk-BASIC asks you:

```
You have to swap diskettes n time(s).
Will you proceed (Y/N)?
```

8. Type: `Y` (ENTER)

9. Then Disk-BASIC informs you:

```
Ø READING
```

   for a few seconds.

10. Disk-BASIC will prompt you to swap source and formatted diskettes many times during the BACKUP process. This operation should be repeated until BACKUP is completed.

11. When BACKUP is completed, Disk-BASIC will display the message:

```
COPY COMPLETED.
```

Two-Drive BACKUP (BACKUP)

This section applies to two-drive systems only.

1. In the Disk-BASIC mode, type:

```
RUN"Ø:BACKUP" (ENTER)
```

2. Disk-BASIC now loads and starts BACKUP; it will then ask you:

```
BACKUP UTILITY

COPY ALL (Entire Diskette)
or SYSTEM (System Files Only)

Enter A (ALL) or S (SYSTEM) ?
```

3. Type: A (ENTER)

4. Next, Disk-BASIC asks:

```
Enter Source Drive (Ø or 1)?
```

5. Specify the drive which contains the original Disk-BASIC diskette. If you load the source diskette in Drive 0, type:

```
Ø (ENTER)
```

6. Disk-BASIC will ask you:

```
 Copy All from Drive Ø to Drive 1
Put the SOURCE diskette in Drive Ø and
a FORMATTED diskette in Drive 1.

Press ENTER when ready.
```

7. Now, the duplication process will begin and then the message "Copying..." blinks on the display. When the BACKUP is completed, Disk-BASIC will display the message:

```
COPY COMPLETE.
```

The duplication process is now completed. We suggest you save the original Disk-BASIC and use the duplicate as your working copy. If anything happens to the working copy, you can make another BACKUP from the original.

When executing FORMAT, BACKUP.SNG or BACKUP, press (F8) key (in response to any prompt) to terminate the utility and returns to the Menu.

Tips

It is better to back up system information on all the diskettes you want to use, even if you want to use certain diskettes to store data only. Since data cannot be written on the system track, you will lose no memory space if system tracks are recorded on the diskette.

6/Using the Floppy Disk in TEXT

With the Disk/Video Interface connected to the Portable Computer, you can save/load **TEXT** data to/from the floppy disk.

This section explains the procedure to make a simple sentence and save it to the disk, and then load it back into the RAM of the Portable Computer.

1. Enter **MENU** on the LCD of the Portable Computer and move the Cursor to **TEXT**. Press (ENTER).

2. The Portable Computer asks you:

   ```
   Files to edit?
   ```

   Type in the filename of your text:

   ```
   TEST1 (ENTER)
   ```

   In this case, **TEST1** is the filename. Of course, you can use any other filename, provided the filename is less than 6 characters and starts with an alphabetic character (not numeric).

3. After assigning the filename, you can type in any message. For example, type:

   ```
   You are living in the computer age. (ENTER)
   ```

4. In order to load this message on the floppy disk, press (F3). You will see a message "`Save to`" on the bottom line of the Portable Computer display. Type:

   ```
   0:TEST1 (ENTER)
   ```

Important Notice: In the above command, the first digit **0** specifies the number of the disk drive and **TEST1** specifies the filename. (The filename here need not be the same as the one used for the "File to edit?" prompt.) If you do not type the number of the disk drive, then the computer will SAVE the file to cassette. (Even if a cassette recorder is not connected, data will still be output to the cassette port.) The Portable Computer cannot perform other functions until SAVE is complete.

5. To see the message saved on the floppy disk on the display, press (F2). You will find a message "`Load from`" at the bottom of the display. Type:

   ```
   0:TEST1 (ENTER)
   ```

The message saved in floppy disk under the name of **TEST1** is loaded. Then, the display becomes:

```
You are living in the computer age.
You are living in the computer age.
```

The message in the first line is the data actually typed in, and the second line is the data loaded from the floppy disk.

Tips

When using the CRT or TV set as an output device, and you want to insert some lines, you may find it takes too long to scroll down the screen to add the new text. To avoid this, insert a couple of carriage returns/line feeds (press (ENTER)) before inserting. After text has been inserted, press the delete key to eliminate extra carriage returns. Another way is to move the displayed text so the part you wish to insert appears at the bottom of the screen.

7/Quick Instructions for Using Disk-BASIC

This section will give you step-by-step procedures for saving a program to a disk file and loading a program for a disk file.

For programming information, refer to the Disk-BASIC section of this manual.

Saving a Program

Suppose you have a BASIC program in the Portable Computer memory. Try saving that program onto disk. Load the program by either LOADing it in the BASIC mode, or positioning the Cursor over the program name in the Main Menu and pressing (ENTER). Be sure the "OK" prompt is showing. Type:

```
SAVE"Ø:PRGRAM" (ENTER)
```

Disk-BASIC now saves the program in Drive 0 to a file arbitrarily named **PRGRAM**. Note that the filename here need not be the same as the one you assigned it originally.

Loading a Program

For this sample session, load the program just saved. First, type:

```
NEW (ENTER)
```

to delete the resident program from memory. (This will prove that it can be retrieved from the disk file.)

Now, type:

```
LOAD"Ø:PRGRAM" (ENTER)
```

and Disk-BASIC will load the specified program. List it or run it for verification.

For further information on using Disk-BASIC, refer to Part II of this manual.

PART II/
DISK-BASIC

8/Portable Computer Disk-BASIC Overview

As shown in Figure 8-1, conceptually, Portable Computer BASIC is a core function of the larger, more functional Portable Computer Disk-BASIC.

Because of this structure, when using the Portable Computer Disk-BASIC, you can use not only most of the functions of the Portable Computer BASIC, but also the characteristic functions of Disk-BASIC together.

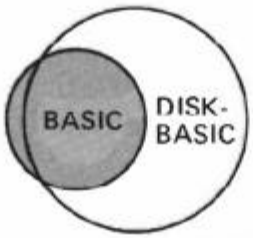


Figure 8-1. Structure of the Floppy Disk

How Disk-BASIC Uses the RAM

To operate the Disk/Video Interface using the Portable Computer, the first step is to load Disk-BASIC. Figure 8-2 shows the Memory Map, before and after loading Disk-BASIC.

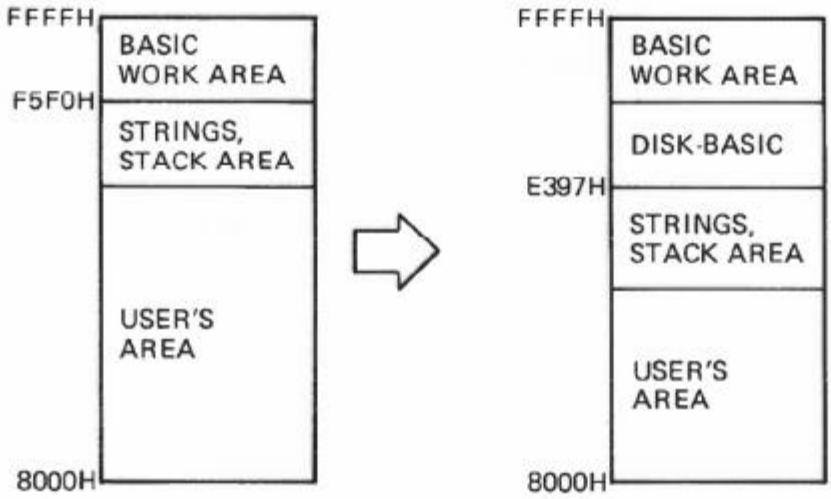


Figure 8-2. Memory Map

As shown in Figure 8-2, by loading Disk-BASIC, the memory area used for BASIC expands to the lower address, i.e., the memory size in the RAM you can use decreases.

Important Notice: The second parameter of the CLEAR command specifies the area in the RAM that the user can use freely. If you assign a figure over 58263 (E397 Hex) as the second parameter, FC (illegal function call) error will occur. Be careful when using the POKE command.

Filenames

A filename consists of a **name** and a **file extension.** A name consists of 6 alphanumeric characters. The first character must be an alphabetic letter. A file extension consists of 3 characters and you can use any combination of alphabetic and numeric characters.

Between the name and the file extension, Disk-BASIC inserts either a ".", "blank" or "*" as a partition mark. At the same time, this character indicates an attribute of the file. In Disk-BASIC, file extension is regarded as part of the filename.

" . " stands for BASIC file
" " stands for ASCII file
"*" stands for Machine Language file

To create a BASIC file or a Machine Language file, you have to save them using the **SAVE** or **SAVEM** commands in BASIC. You can create an ASCII file in two ways: 1) by saving a file in TEXT mode for accessing it as an input file to BASIC, or 2) by saving it in BASIC mode using the "A" option.

9/Disk-BASIC Functions

Disk-BASIC provides a powerful set of commands, statements and functions relating to Disk I/O and CRT control. They are divided into two categories:

1. File or CRT manipulation. Deals with the file or CRT as a unit, rather than with the contents of the file.

2. File or CRT access. Prepares data files for I/O by reading or writing to the file or CRT.

COMMAND	OPERATION
KILL	Deletes a program or data file
LOAD	Loads a BASIC program
LOADM	Loads a machine language program
LFILES	Displays filenames on the disk
MERGE	Merges a BASIC program (in ASCII format)
NAME	Changes the name of the file
RUN "program"	Loads and executes a BASIC program
RUNM "program"	Loads and executes a machine language program
SAVE	Saves the resident BASIC program
SAVEM	Saves a machine language program
SCREEN	Assigns the console to the specified device
WIDTH	Sets the screen width

Table 9-1. File or CRT Manipulation

STATEMENT	DESCRIPTION
CLOSE	Closes open files
DSKO$	Writes a string on the specified sector
INPUT #	Reads from the disk in sequential mode
LINE INPUT #	Reads a line of data in sequential mode
OPEN	Opens a file or CRT for use (creates the file on the disk, if specified)
PRINT #	Writes to the disk in sequential mode

FUNCTION	DESCRIPTION
CSRLIN	Gets the vertical coordinate of the Cursor
DSKI$	Gets a string from the specified sector
EOF	Checks to see if the end-of-file has been encountered during the last read
INPUT$	Gets a string of characters from the file
LOC	Gets current record number
LOF	Returns number of the last record in the file
POS	Returns the current Cursor position
CHR$ (27) + code	Controls the CRT display (ESCAPE character...begins a control sequence)

Table 9-2. File or CRT Access

File and CRT Manipulation

Commands

KILL
Deletes a Program or Data File

KILL *"drive#:filename. file extension"*

KILL deletes a specified file from the disk. Both filename and file extension must be specified.

Example

```
KILL"0:GAME"
```

Filename "GAME" is deleted from the disk in Drive 0. If you save files by assigning file extensions using the SAVE command, you must specify the file extension when "KILLing" the file.

LOAD
Loads a BASIC Program from the Disk

LOAD *"drive#:filename"* [**,R**]

LOAD loads a BASIC program file from the disk into the memory. LOAD closes all the open files. However, with the **R(un)** option, all the data files which have been already opened remain OPEN and the program starts to run after it is loaded.

Examples

```
LOAD"0:DEMO",R
```

Filename "DEMO" is loaded from Drive 0 and executed because of the **R** option.

LOADM
Loads a Machine Language Program

LOADM *"drive#:filename"*

LOADM loads the specified saved program from the disk into the memory at the address specified when the program was saved on the disk originally.

Example

```
LOADM"0:BBB"
```

LFILES
Displays Filenames on the Disk

LFILES *drive #*

LFILES displays the filenames on the specified drive, with the size of each file in clusters. A cluster is the minimum unit of allocation for a file with a size of 2.25K bytes (9 sectors).

Example

```
LFILES 0
```

On entering the above command, the following message is displayed:

```
SYSTEM VER 01.00.00 (or  NO SYSTEM)
FORMAT.     1     BACKUP.     2
BACKUP.SNG 2
155.25 K AVAILABLE
```

The first 6 characters stand for filename and the seventh character shows an identification of the file type, i.e.:

For machine language file: "*"
For BASIC program file: " "
For BASIC program file with ASCII type: " "
and Text files

These character I.D. codes are automatically assigned during execution of the SAVE command.

The next 3 characters show the file extension. This file extension appears blank if no extension was used to save the program.

The last figure shows the number of "clusters" used for the file. (See Chapter 11 for more information regarding clusters.)

The number in the last line such as 155.25K shows the free memory size.

MERGE
Merges a BASIC Program (in ASC II Format)

MERGE *"drive #:filename"*

MERGE merges the lines from a program file saved on the disk in the ASCII format into the program currently resident in the memory. If any lines in the file being merged have the same line number as lines in the program in the memory, the lines from the file will replace the corresponding lines in the memory. After the MERGE command, the MERGEd program resides in the memory and BASIC returns to command level.

Example

```
MERGE"0:AAA"
```

NAME
Changes the Name of the File

NAME *"drive#:old filename.file extension" AS "drive#:new filename.file extension"*

NAME changes the name of the file. The old filename must already exist and the new filename must not already exist. Otherwise, an FF or FE error will result.

Example

```
NAME"1:GAME" AS "1:SPACE"
```

RUN Program
Loads and Executes a BASIC Program

RUN *"drive#:filename"* [**,R**]

RUN loads a BASIC program from the disk and runs it. RUN closes all open files. LOAD with the R option does the same thing except all the data files remain OPEN.

Example

```
RUN"0:TEST"
```

Filename "TEST" is loaded from Drive 0 into the main storage, and runs.

RUNM Progrm
Loads and Executes a Machine Language Program

RUNM *"drive#:filename"*

RUNM loads a machine language program file from the disk into the appropriate location of the memory that was specified when the file was saved and runs it. RUNM closes all open files. The machine language program to be loaded must be independent of BASIC (i.e., one that can be called directly in Menu mode).

Note: Execution ADRS must have been specified when the program was saved.

Example

```
RUNM"0:DDD"
```

SAVE
Saves the Resident BASIC Program

SAVE *"drive#:filename"* [**,A**]

SAVE saves a BASIC program file on the disk. Use the A option to save the file in ASCII format. Otherwise, BASIC saves the file in a compressed binary format. ASCII files take up more space, but some types of access require that the files be in ASCII format. For example, a file intended to be MERGEd must be saved in ASCII format. Programs saved in ASCII format may be read as BASIC data files or text files.

Example

```
SAVE"0:GAME",A
```

Filename "GAME" is saved onto Drive 0 as an ASCII file. File extension during execution of a SAVE command is optional.

SAVEM
Saves a Machine Language Program

SAVEM *"drive#:filename", start address, end address, entry address.*

SAVEM writes the program stored, beginning at the start address and ending at the end address on the disk under the specified filename. Entry address is optional. If omitted, BASIC considers the entry address to be the same as the start address.

Example

```
SAVEM"0:CCC",50000,50256,50000
```

SCREEN
Assigns the Console to the Specified Devivce

SCREEN *console, function key display switch*

SCREEN assigns the console on the specified devices.

Console 0: LCD'
Console 1: CRT

Function key display switch 0 does not display the contents of function keys; switch 1 does display the contents of function keys.

Console is the output device for the PRINT and PRINT@ statements, the LIST command, and TEXT, ADDRESS, SCHEDL, and TELCOM functions (except Menu).

The default value of console is 0 (LCD) and function key display switch is 0 (does not display).

Example

```
10 SCREEN 1,1
20 CLS
30 PRINT "SPRING"
```

The first 5 characters of each function key definition is displayed on the bottom line of the CRT screen, and "SPRING" is printed on the first line.

WIDTH
Sets the Screen Width

WIDTH 40 or 80

This command sets the width of the screen at 40 or 80. The width set will stay effective in TELCOM, TEXT, ADDRESS and SCHEDL mode, too.

Important Notice: When the Disk/Video Interface is turned on, be sure to enter this command or what is displayed on the CRT may not be what you intended.

Example

```
WIDTH 40
```

The CRT display is set at 40 characters per line.

Statements

CLOSE
Closes Open Files

CLOSE *[file number list]*

This command closes the files specified in file number list. The file number is the number under which the file was opened. If omitted, all open files are closed.

Important Notice: When you remove the diskette from the drive, be sure that the files in the diskette have been already closed.

Example

```
CLOSE 1,2,3
```

Files associated with the file numbers 1, 2 and 3 are closed.

DSKO$
Writes a String on the Specified Sector

DSKO$ *drive, track, sector, switch, expression*

Writes the string on the specified sector. When (switch) is zero, the first half of the sector is accessed; and when (switch) is one, the latter half of the sector is accessed. The maximum length for the string is 128 characters. Disk-BASIC fills the entire 128-character length with null codes (ASCII 00) if the string is less than 128 characters.

Example

```
DSKO$ 0,5,12,0,"ABCDEF"
```

The string data "ABCDEF" is written on track-5, first half of sector-12 in Drive 0.

IMPORTANT NOTE: DSKO$ writes data to any position on the diskette, whether the position is used or not, **EVEN ON THE SYSTEM TRACK.** Refer to the "Tips on manipulation for files on diskette" before you attempt to use this statement.

INPUT#
Reads from the Disk in Sequential Mode

INPUT# *file number, variable list*

INPUT# reads data items from the disk file and assigns them to the variable list. The type of data in the file must match the type specified by the variable list.

The first character encountered that is not a space, carriage return or line feed is assumed to be the beginning of the data. The numeric data terminates on a space, carriage return, line feed or comma.

String data can be stored in two ways: 1) Inside the double quotation marks("), or 2) just as a "bare" string. If you use the double quotation marks, Portable Computer will read the data as one string until it encounters the next double quotation mark.

If the first character of the string is not a quotation mark, the string is a "bare" string, and will terminate on a comma, carriage return, line feed or after 255 characters have been read. When you want to store a string that contains a comma, carriage return or line feed, use double quotation marks.

A normal PRINT# statement followed by a double quotation mark will store the "bare" string on the disk. If you want to store a string with a double quotation mark, use "PRINT# CHR$(34)" (34 is the ASCII code for quotation mark), or save data in the TEXT mode, using a quotation mark at the top of the string.

If end-of-file is reached when a numeric or string data is being INPUT, the data is terminated.

LINE INPUT#
Reads a Line of Data in Sequential Mode

LINE INPUT# *file number, string variable*

LINE INPUT# reads an entire line of up to 254 characters without delimiters from a disk file to a string variable. The file number to assign is limited to the file already opened. The string variable is the name of a string variable to which the line will be assigned.

LINE INPUT# reads all the characters in the disk file up to the carriage return not preceded by line feed. Then the line feed/carriage return sequence is ignored (not read by LINE INPUT#). The next LINE INPUT# reads all the characters up to the next carriage return.

If a line feed/carriage return in this sequence is encountered, it is preserved, i.e., the line feed/carriage return codes are read as a part of the string. (When you press (ENTER), carriage return/line feed is sent in this sequence. If you want to send line feed/carriage return in this sequence, you must send CHR$(10) CHR$(13).)

LINE INPUT# is especially useful if each line of a file has been broken into fields, or if a BASIC program saved in ASCII mode is being read as data by another program.

OPEN
Opens a File or CRT for Use

OPEN *"device or drive#:filename" FOR mode AS file number*

OPEN creates a buffer for a file on the given device.

CRT: Opens the CRT as a file and sets the mode to sequential output. If the CRT is assigned as the device, it is not necessary to specify filename.

Disk Drive: Allows I/O to a disk file and sets the mode that will be used with the file.

Mode is one of the following:

OUTPUT: Specifies sequential output mode.

INPUT: Specifies sequential input mode.

File number is an integer expression whose value is between 1 and the maximum number of the files specified in a MAXFILES statement. File number, once assigned when OPENed, will remain the same until it is CLOSEd and must be referenced each time you access the file.

You must OPEN a file before you can do any I/O statements to the file, such as one of the following statements, or any statement or function requiring a file number:

PRINT#, PRINT#USING, INPUT#, LINEINPUT#, INPUT$

A file cannot be opened for input if it is already opened for output; or the file opened for input cannot be re-opened for output, unless you close it first.

Example

```
10 OPEN "CRT" FOR OUTPUT AS 1
```

Opens CRT and assigns it as file number "1" for output.

Example

```
10 OPEN"0:TEST1" INPUT AS 2
```

Opens TEST1 file on Drive 0 as file number "2" for input.

PRINT#
Writes to the Disk in Sequential Mode

PRINT# *file number,*

This statement writes data sequentially to the specified file. When you first open a file for sequential output, a pointer is set to the beginning of the file. Therefore, your first PRINT# places data at the beginning of the file. At the end of each PRINT# operation, the pointer advances, so the values are written in sequence.

A PRINT# statement creates a disk image similar to what a PRINT to display creates on the screen. Remember this, and you will be able to set up your PRINT# list correctly for access by one or more INPUT# statements. PRINT# does not compress the data; it writes ASCII codes data. Also you can use the field specifier of PRINT USING statement in the form of PRINT#n, USING.

Functions

CSRLIN
Gets the Vertical Coordinate of the Cursor

CSRLIN

The value returned will be in the range from 0 to 24 (if the console is CRT or TV).

Example

```
10 CLS
20 PRINT
30 PRINT
40 PRINT"CURRENT CURSOR LINE IS";CSRLIN
```

DSKI $
Gets a String from the Specified Sector

DSKI$ *(drive, track, sector, switch)*

DSKI$ is a complementary function to DSKO$ statement. DSKI$ returns the contents of the first half of the sector (128 bytes) when the switch is "0", and the latter half when the switch is "1" to a string variable.

Example

```
PRINT DSKI$ (1,5,12,0)
```

EOF
Checks to See if the End-of-File Has Been Encountered During the Last Read Operation

EOF *(file number)*

Returns "–1" (true) when the end of a disk file is detected. Use EOF to test end-of-file during a read operation.

Example

```
10 MAXFILE=2
20 OPEN"0:SDATA" FOR OUTPUT AS 2
30 PRINT#2, "ABCDE":CLOSE
40 OPEN"0:SDATA" FOR INPUT AS 1
50 IF EOF(1) THEN PRINT "EOF":CLOSE:END
60 LINE INPUT#1, A$
70 PRINT A$
80 GOTO 50
```

Data is transferred from disk 0 until the EOF marker is detected.

INPUT$
Gets a String of Characters from the File

INPUT$ *(numeric expression, file number)*

INPUT$ returns a string of a length given in a numeric expression from the file opened under file number. The numeric expression must be in the range of 1 to 255.

Example

```
10 A$=INPUT$(5,1)
```

Inputs five characters from the file opened as file #1, and assigns the input string to A$.

LOC
Gets Current Record Nnmber

LOC *(file number)*

LOC is used to determine the current record number, i.e., the number of the last record read since the file was opened.

Example

```
PRINT LOC(1)
```

LOF
Returns Number of the Last Record in the File

LOF *(file number)*

This function provides you with the number of the last record, i.e., the highest numbered record in the file. A record means one sector on the diskette.

Example

```
10 OPEN"0:SDATA" FOR INPUT AS 1
20 PRINT LOF(1)
30 END
```

POS
Returns the Current Cursor Position

POS *(dummy numeric expression)*

This command returns the current position of the Cursor on the display.

Example

```
10 CLS
20 PRINT
30 PRINT"ABC"
40 PRINT POS(0)
50 END
```

CHR$(27)+code
Controls the CRT Display

CHR$;*"character"*

CHR$(27) is the "ESCape" code. This code notifies the Portable Computer that a special control code is on its way. The next character sent determines which control on the CRT is selected.

CHR$(27); "A"	Cursor up
CHR$(27); "B"	Cursor down
CHR$(27); "C"	Cursor forward
CHR$(27); "D"	Cursor backward
CHR$(27); "E"	Clears display
CHR$(27); "H"	Moves Cursor to the home position
CHR$(27); "J"	Erases up to the end of the page
CHR$(27); "K"	Erases up to the end of the line
CHR$(27); "L"	Inserts a line
CHR$(27); "M"	Deletes a line
CHR$(27); "P"	Cursor ON
CHR$(27); "Q"	Cursor OFF
CHR$(27); "T"	Sets the system line (bottom line)
CHR$(27); "U"	Resets the system line
CHR$(27); "V"	Locks the screen (hold mode)
CHR$(27); "W"	Unlocks the screen
CHR$(27); "Y"; "x"; "y"	Sets Cursor (refer to note below)
CHR$(27); "j"	Clears display (same as CHR$(27) "E")
CHR$(27); "p"	Enters the reverse mode
CHR$(27); "q"	Exits the reverse mode

Note: CHR$(27); "Y"; "x"; "y"
This code specifies location of the cursor directly by "x" and "y" coordinates (the corresponding ASCII character code numbers in decimal – see Part 4 Character Code Tables). Note that, however, the actual values given to the Portable Computer are "x-32" and "y-32". For instance, assume you assign "!" in "y" and "#" in "x". ASCII code for "!" is 33 and for "#" is 35, so, by deducting 32, each value becomes 1 and 3. Therefore, the cursor is located in column 3 on line 1.

Example

```
10 CLS
20 PRINT"01234"
30 PRINT"1":PRINT"2":PRINT"3":PRINT"4"
40 PRINT CHR$(27);"Y";"!";"#";"A"
50 END
```

After excuting this program, the result on the CRT will be:

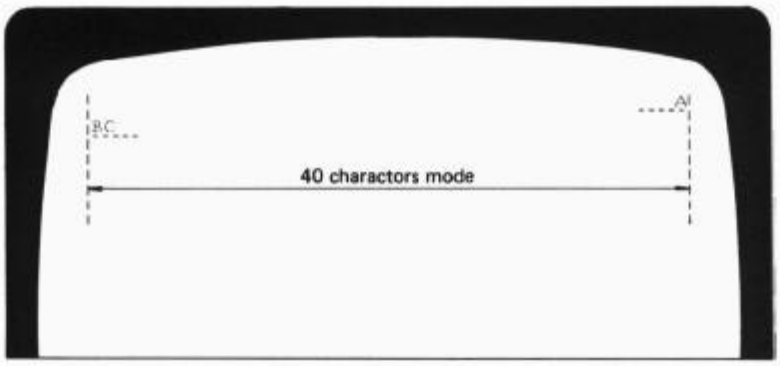
```
01234
1   A
2
3
4
```

Tips on Using PRINT@ Command

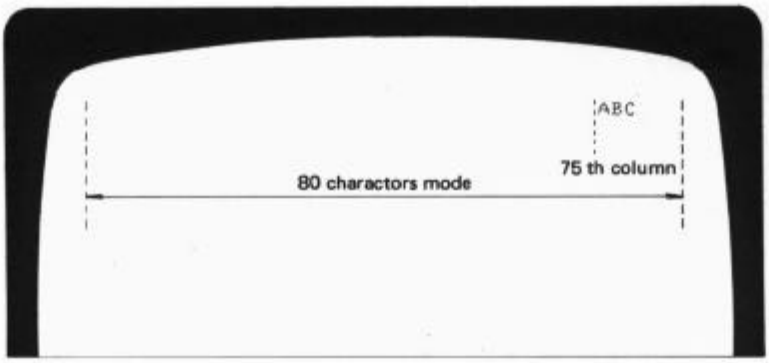
When you write data into the RAM or diskette using the command "PRINT@" etc., Disk-BASIC recognizes the numeric expression of the first parameter in this command as ESC code, line number and column number according to the condition of console width. So, if the console is in the 80 characters mode, the parameter of PRINT@ command will be written on the RAM or diskette as the following form:

PRINT@75; "ABC" → PRINT + ESC + "Y" + line #0 + column #75 + "ABC"

When this data is read from the RAM or diskette in the 40 characters mode, resultant display on the CRT will be:



That is, the first character of the string is displayed on the last column in the first line instead of the 75th column from the home position.



Note: The PSET and PRESET commands on your Portable Computer function only on the LCD: they do not function on the CRT or TV screen.

Before you switch the display device from the video display to the LCD by SCREEN command, you must terminate the last print line with a carriage return.

You should not switch the display device if the last PRINT command ended with a semi-colon.

Tips on Manipulation for Files on Diskette

We've described how to access files on the diskette . . . here's some hints and tips to manipulate the data on the diskette.

PRINT# writes data on the diskette sequentially. You can use TAB or USING along with PRINT#, just like with PRINT.

```
PRINT#1, TAB(22);A$
PRINT#1, USING"$$####.##",B
```

However, when you INPUT# from file, you will have to use some tactics if you want to retrieve as PRINT#ed. For, in the first example above, if you just INPUT#, leading blank will be ignored. Supposing LEN(A$)=5, use INPUT$(27,n) (n is the number of file you assigned when OPENed), or use LINEINPUT#. In the second example, the stored data is not longer the numeric data (the $ on top is not numeric) so you have to INPUT# as string data. You had better not use comma to show unit of thousand, because INPUT# regards the comma as terminator of data.

As explained to you previously, when you PRINT# the string, the double quotation mark (") is not attached, and, remember, the comma in the string too is regarded as data terminator. So, when you are storing a string such as:

When you think computer, think small.

You must PRINT# as

```
PRINT#1, CHR$(34);"When you think
computer, think small.";CHR$(34)
```

or, in TEXT mode, type

```
When you think computer, think small.
```

and SAVE using the filename you want to use when running BASIC program including "INPUT#".

Is there no way to input the string (or numeric data) including comma? Yes, there is. Use LINEINPUT#.

This statement reads data up to where you pressed ENTER, even there is comma in the string. Or, as we've said before, you can even include linefeed and carriage return in the string to be retrived by LINEINPUT#. Try the following:

```
PRINT#1,"This is one line";CHR$(10);CHR$
(13);"This is another line"
```

And use LINEINPUT# to read this string.

All these statements/commands access disk file sequentially. On some occasion you may want to access disk randomly. Use DSKO$ and DSKI$ for this purpose. BUT USE THESE WITH CARE!

For proper use of DSKO$ and DSKI$, you must first be well acquainted with the format of the diskette. Go through Part III and after you're confident that you get the picture, return to this point.

Essentially the DSKO$ is analogous to POKE and DSKI$ is to PEEK. The POKE/PEEK writes/reads on/from RAM, but DSKO$/DSKI$ do same on/from diskette. POKE cannot write anything on ROM, of course, but on diskette there is no such parts as "Read Only Memory". So DSKO$ can write to any portion on the diskette, EVEN ON THE SYSTEM TRACK. This is the reason why we tell you to use this statement with great care.

DSKO$ does not read directory or FAT. So it writes data on the track/sector you specified, whether it is used or not. Also directory and FAT remains same when you write data using DSKO$ – there is data on track/sector you specified, but directory/FAT do not reflect this.

For convenience, first create the "dummy" file using OPEN statement and PRINT# dummy data to prepare enough capacity for the data you want to put with DSKO$. Then CLOSE this file. Now the directory/FAT have the data for this dummy file. Use DSKI$ to read the data in the directory to see where on the diskette is this dummy file.

Return to Part III re structure of the directory and FAT.

Suppose our dummy file is named "DUMMY". Following is one of the method to get the data for DSKO$ (Track number and Sector number).

```
10 N=1:M=0
20 FOR L=0 TO 15
30 A$=MID$(DSKI$(0,20,N,M),1+16*L,11)
40 IF LEFT$(A$,6)="DUMMY" THEN 60 ELSE NEXT L
50 IF M=0 THEN M=1 ELSE M=0:N=N+1:GOTO
20
60 B$=RIGHT(A$,1):B=ASC(B$):'Obtain
cluster number
70 TR=INT(B/2):'Track number=integer
part of 1/2 cluster #
80 IF TR=B/2 THEN SC=1:'If cluster # is
even sector starts from 1
90 IF TR<>B/2 THEN SC=10:'If cluster #
is odd sector starts from 10
100 PRINT TR, SC
```

You can check the length of dummy by DSKI$ing the FAT.

The other way might be, for instance you write on newly backed up diskette using DSKO$, when you know there is no data on the diskette other than system and directory.

Check and note how many byte to which track sector you wrote using DSKO$. Then, write the information to directory and FAT using DSKO$. If you don't do this, Disk-BASIC might create a file on the sector you put data with DSKO$.

Here's what you may want to do. You've written 256-byte data on track 5, sector 12. Let's use "TEST" as filename. First find the blank portion of directory. In this example we are assuming you are using new diskette, so there should be three files, FORMAT, BACKUP and BACKUP.SNG.

One file uses 16 bytes in the directory, so total 48 bytes are used. DSKO$ writes 128 bytes at one time, so DSKI$ing 128 bytes, put the information to 49th byte and on using MID$ statement, then return the data using DSKO$.

The informations to be put on directory are filename, extension, attribute and cluster number. The cluster number can be obtained by doubling the track number and add 1 if the sector number is between 10 and 18. So the cluster number in our case is 11.

```
10 A$="TEST   "+"     "+CHR$(0)+CHR$(11)
20 B$=DSKI$(0,20,1,0)
30 MID$(B$,49,16)=A$
40 DSKO$ 0,20,1,0,B$
```

will put the necessary information into directory. Now we must also write to FAT. Use the same method as above.

Sector 12 we used is the third sector of cluster 11. So we have to put C3 hex, which is 195 decimal, on 11th byte.

```
50 A$=CHR$(195)
60 FOR M=16 TO 18
70 B$=DSKI$(0,20,M,0)
80 MID$(B$,12,1)=A$
90 DSKO$ 0,20,M,0,B$
100 NEXT M
```

This completes the operation. You may want to save this or similar program so you can load any time you want.

10/Utility Programs

FORMAT

The FORMAT program lets you prepare data diskettes (either new diskettes or diskettes which contain data or programs of being no longer needed), leaving a maximum amount of space for your program and data files.

FORMAT takes a blank (new or magnetically erased) diskette, records track/sector boundaries on it, initializes it and creates a directory.
First, load the FORMAT program from the diskette (or wherever it is stored) and run it.

```
This utility formats diskettes.
- All data will be lost -

Which drive will be used (0 or 1)?
```

The program prompts you for the drive selection.
Type in 0 or 1 and press (ENTER).
You will be prompted:

```
Put the diskette to be formatted in
    Drive n.
```

(Replace n with specified drive #.)

```
Press ENTER when ready.
```

Upon pressing (ENTER), FORMAT will be started.

When you attempt to format a diskette in which any data is already stored, following message will be displayed:

```
The diskette contains data.
Use this diskette or not (Y/N)?
```

Enter "Y" to format the diskette or "N" to terminate this utility.
During formatting, following message will blink on the CRT:

```
Formatting !
```

When the formatting is completed, this message is displayed:

```
FORMAT COMPLETE;
number of flawed tracks: 0
```

If any System Tracks are flawed, FORMAT cannot be continued and the message;

```
FORMAT FAILED.
```

is displayed.
If a diskette is not inserted into the drive or the clamp lever on the disk drive is not locked, even when the specified drive is accessed, following message will be appeared:

```
Format aborted; no diskette.
```

Furthermore, the write protect notch on the diskette is covered, the message;

```
Format aborted; Write-Protected.
```

will be displayed.

BACKUP

This system is operative only when two floppy disk drives are used together and not operative with only one drive. First load the BACKUP program from the diskette (or wherever it is stored) and let it run. How the following will be displayed:

```
BACKUP UTILITY
COPY ALL (Entire Diskette)
or SYSTEM (System Files Only)
Enter A (ALL) or S (SYSTEM) ?
```

ALL COPY listerally copies the entire diskette (tracks 0 through 39).
SYSTEM COPY copies only the system area (tracks 0 and 1) to another diskette.
If you want to copy the entire contents of the diskette, enter "A". If you want to copy the system area only, enter "S".
When "A" or "S" command has been input, the following message will appear on the display:

```
Enter Source Drive (0 or 1)?
```

Assign the drive # of the source diskette and press (ENTER) key.
The following message will be displayed:

```
Copy All from Drive (x) to Drive (y).
Put the SOURCE diskette in Drive (x)and
a FORMATTED diskette in Drive (y).

Press ENTER when ready.
```

Press (ENTER) key. If data or program has already been stored in the destination diskette, the following message will be displayed. If copying does not commence:

```
Destination contains data
Use this diskette (Y/N)?
```

Entering "N" will ceases this utility.
Entering "Y" will commences copying.

During copying process, the following message will flash on and off on CRT;

```
Copying...
```

When copy has been correctly made:

```
COPY COMPLETE.
```

BACKUP. SNG

First load the BACKUP.SNG program from diskette (or from wherever it is stored) and make it run. Now the following should be indicated:

```
SINGLE DRIVE BACK UP UTILITY
COPY ALL (Entire Diskette) or
SYSTEM (System Files Only)
Enter A (ALL) or S (SYSTEM)?
```

ALL COPY literally copies the entire contents of the diskette (tracks 0 through 39).
SYSTEM COPY copies only the system area (tracks 0 and 1) to another diskette.
If you want to copy the system area only, enter "S".
When "A" or "S" command is entered, the following message will appear on the display:

```
TO COPY (ALL or SYSTEM)
Put the source diskette in the drive.
Press  ENTER  when ready.
```

When (ENTER) key is pressed, the system will investigate the amount of free bytes on the Portable Computer side and indicate the number of times you have to swap the diskette to finish the copying. (n is the number of times you have to swap diskettes on the destination side and on the source side.) Then the following message will appear on the display:

```
You have to swap diskette n time(s)
Will you proceed (Y/N)?
```

If you want to start copying the diskette, enter "Y" and, if you want to cease copying the diskette, enter "N".
Entering "Y" will be let the contents of the source diskette stored in memory during reading, the following is indicated on the CRT or LCD:

```
Ø READING
```

When the process to read the diskette is finished, the following will appear on the display to place you to put the diskette you want to copy from:

```
Put the FORMATTED diskette in the drive
Press ENTER when ready.
```

After you have put the diskette and press ENTER key, the contents of the source diskette will be written from the memory. During writing, the CRT or LCD display will indicate the following:

```
1 WRITING
```

Hence, Disk-BASIC will prompt you to swap source and destination diskettes in the previously assigned number of times during the BACKUP process.

Since the diskette used for the first reading process in the BACKUP operation is identified as the "Source" diskette, and the diskette used for first writing process in the BACKUP operation is identified as the "Destination" diskette. When inserting the source diskette into the drive, if the writing process is executed, then the system gives the following message:

```
NOT DESTINATION!
Put the FORMATTED diskette in the drive
Press ENTER when ready
```

In reverse case, when inserting the destination diskette into the drive during reading process, following message will be given:

```
NOT SOURCE!
Put the SOURCE diskette in the drive
Press ENTER when ready
```

During BACKUP, if the back-up is disturbed on a track due to some reason, e.g. scratch on the diskette, etc. the system gives the following message.

```
ABORTED: < Destination's Flaw >
```

When the correct procedures are followed, the following will appear on the display:

```
COPY COMPLETED.
```

Important Notice for BACKUP and BACKUP. SNG utilities

(1) When writing from a diskette which contains data or program, the system will indicate the following:

```
Destination contains data.
Use this diskette (Y/N)?
```

(2) When writing into a protected diskette, the system will give the following message and this utility is ended:

```
ABORTED: < Write-Protected >
```

(3) When attempting to access the drive (for read/write process), if the diskette is not inserted, or the lever in not set down, or not ready, the system will give following message and this utility is ended:

```
ABORTED: < Drive Error >
```

(4) If some mulfunction occurs accidentally during Read and Write process, it gives the following message and ceases the utility:

```
BACKUP FAILED: < R/W Error >
```

PART III/
FORMAT OF THE DISKETTE

11/Format of the Diskette

Part III will provide you with the information necessary to effectively use the floppy disk. These informations may not be so useful for you to use Disk-BASIC because Disk-BASIC handles data on the diskette as a logical conception so called "File".

But to use the DSKO$ statement and DSKI$ function effectively, you have to understand the physical data configuration of the floppy disk. This information will help you develop a wider range of applications for the Portable Computer which will enable you to create better and more effective programs.

Physical Configuration of a Diskette

As shown in Figure 11-1, the **tracks** of a diskette are distributed as concentric circles. The outside edge track is numbered 0. Moving toward the center the number increases, and the centermost track number is 39. Therefore, the total track number count is 40 (tracks 0 through 39).

Each Track is divided into smaller units called **sectors**. Disk-BASIC handles sectors each time it controls the Diskette.
In order for Disk-BASIC to access multiple sectors efficiently, sectors are written in a definite interleave pattern.
The number of sectors count from 1 through 18. One sector has a maximum of 256 bytes. Therefore, one diskette contains:

256 bytes x 18 sectors x 40 tracks = 184,320 bytes

of memory capacity. A small percentage of this space is used by system software; therefore, not all of this space is available to the user.

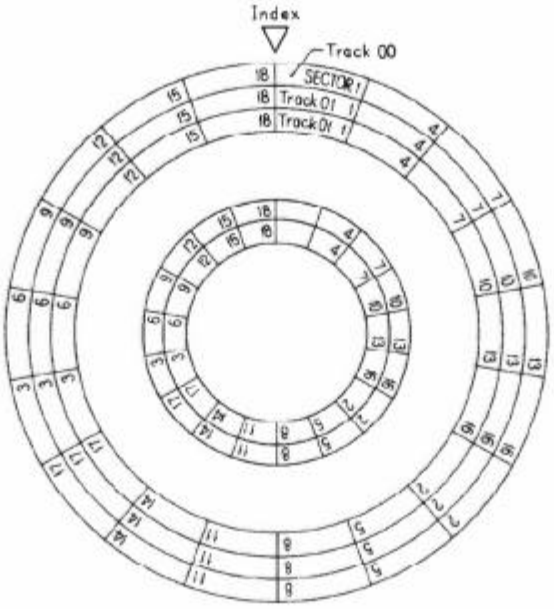


Figure 11-1. Physical Configuration of a Diskette

Cluster

Although the physical configuration of a diskette has been explained, you should not be concerned with this theory when you use the Disk Drive Unit. You can operate the Disk Drive Unit freely by assigning the filenames. Conversion of the filename is executed by Disk-BASIC automatically.

Although the minimum access unit of the disk is 1 sector, when the program or data is handled as a file, 9 sectors are accessed as a minimum access unit. This minimum access unit of 9 sectors is called a **Cluster**.

Figure 11-2 shows the logical configuration of a cluster. Please note that, as explained, sectors 1 through 18 are not distributed sequentially on a track; an actual cluster is not distributed as this figure shows. This is the reason why this figure is titled "logical configuration".

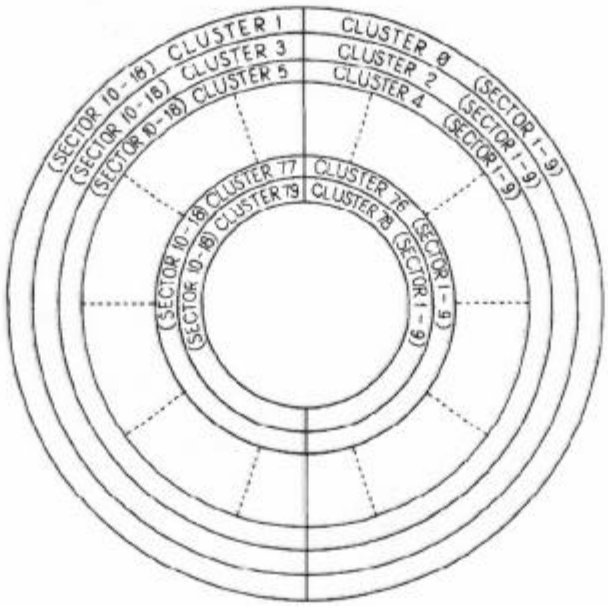


Figure 11-2. Logical Configuration of a Diskette

Format of a System Diskette

Figure 11-3 shows the format of a System Diskette. On tracks 0 and 1 of the System Diskette, the Disk Operating Software of Disk-BASIC and the Disk/Video Interface are loaded. On track 20 a directory, the record of the loading address of each file, is loaded.

The system track and directory are very important and, ordinarily, the user is not allowed to access them directly. However, if you use the DSKO$ statement and DSKI$ function, you can access them directly. For this reason, you have to be careful when using DSKO$ or DSKI$.

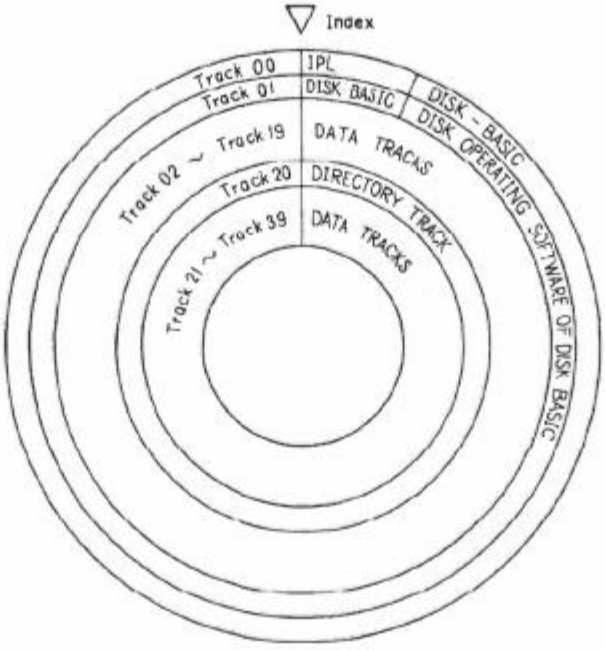


Figure 11-3. Format of a System Diskette

System Track

Figure 11-4 shows the format of system tracks 0 and 1. When the Disk/Video Interface reads DOS, it reads 128 bytes of the second half of sector 1 on track 0 first; then, it reads the data on sector 2 to 14 of track 1. Also, when the Portable Computer reads Disk-BASIC, it reads 128 bytes of the first half of sector 1 on track 0; then, it reads the other 17 sectors on track 0 and sector 1 of track 1.

Thus, on sector 1 of track 0, the first program to read DOS and Disk-BASIC is loaded. This kind of program is named IPL (Initial Program Loader). The first half of sector 1 of track 0 is the IPL for Disk-BASIC, and the second half is the IPL for DOS.

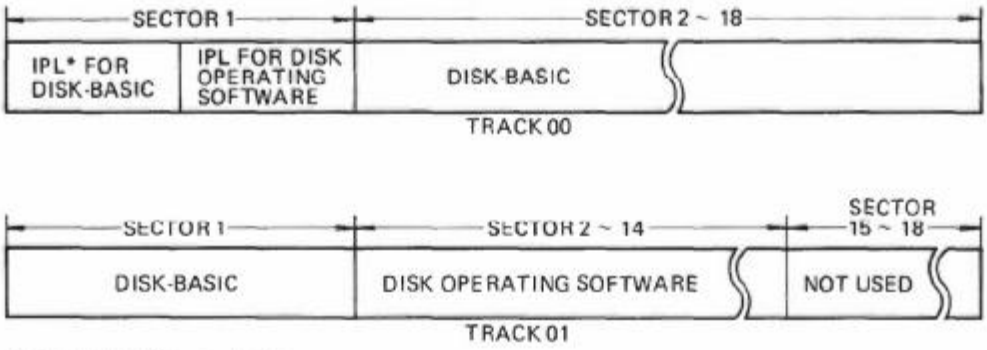


Figure 11-4. Format of System Tracks 0 and 1

Directory Track

Figure 11-5 shows the format of track 20, the directory track. Sectors 1 through 15 are the directory, and sectors 16, 17 and 18 are called FAT (File Allocation Table). The figure shows the allocation of each cluster.

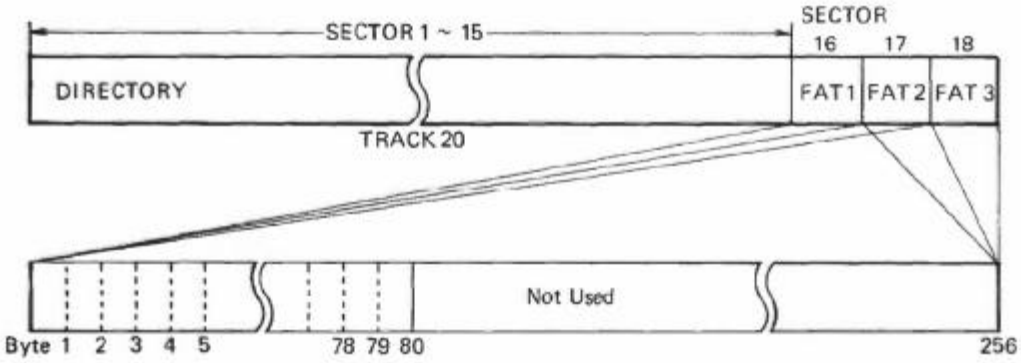


Figure 11-5. Format of the Directory Track

Directory

The directory is a kind of index to investigate the address of a certain file on the diskette on which you are going to access data.

In the directory, filenames, head cluster numbers and attributes corresponding to the files loaded on the diskette are recorded. For any one file, the directory shows this information using a 16-byte record. Figure 11-6 shows the contents of this one record.

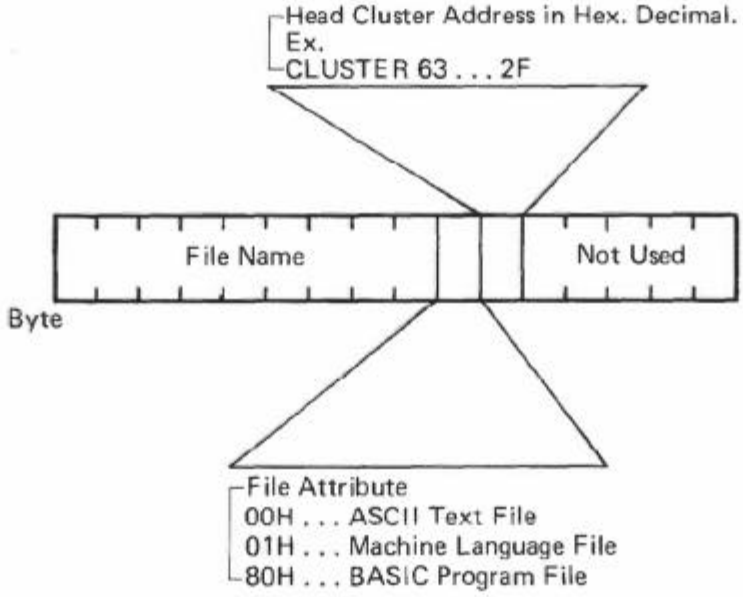


Figure 11-6. Format of the Directory

FAT (File Allocation Table)

One byte in FAT corresponds to one cluster in the diskette and shows the allocation of each cluster. As such, the actual memory size that is used by FAT is the first 80 bytes of sector 16, 17 and 18, they are all the same contents, that is, one track can contain 2 clusters (remember, one track consists of 18 sectors, and one cluster is made up of 9 sectors), 40 tracks per one diskette, for a total of 80 clusters on one diskette.

Allocation indicates whether or not the cluster is used yet, etc. When the diskette is FORMATed, Disk-BASIC sets "flags" on the FAT that show that Disk-BASIC uses the clusters for system tracks 0 and 1 and directory track 20. As you store other data or programs onto disk, Disk-BASIC writes the cluster allocations for those data/programs into FAT.

The value of FAT byte for unused clusters is FF hex (255 decimal). For the cluster that cannot be used (system track and damaged track if any) the value is FE hex (254 decimal). For the clusters that are already used (data is stored), the value is decided as follows.

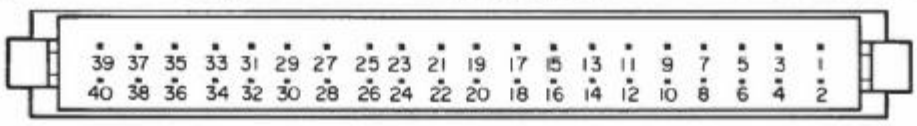
a) If the data is continued to the next cluster, the value shows the next cluster number in hex.
b) If the cluster is the last one in the file, the value is in the range of C1 to C9 hex (193 to 201 decimal). The lower digit of hex notation (1 through 9) shows the number of the last sector used in the cluster.

PART IV/ APPENDICES

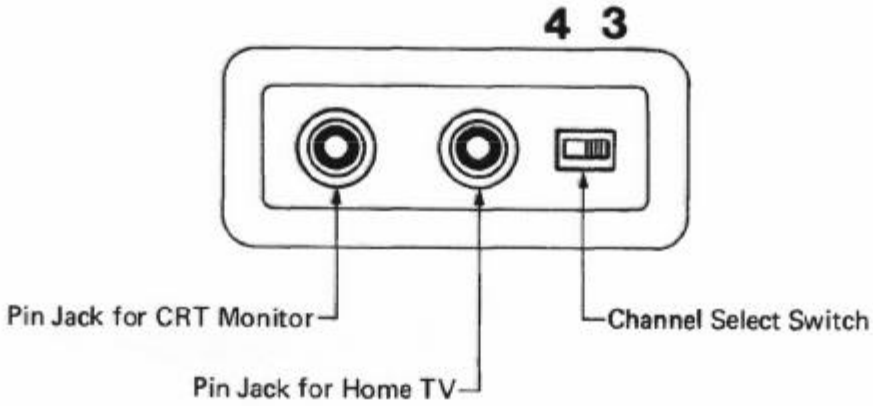
Appendix A/Connector Pin Assignments

SYSTEM BUS CONNECTOR

Pin No.	Symbol	Description
1	VDD	+5V Power Supply from TRS-80 Model 100
2	VDD	+5V Power Supply from TRS-80 Model 100
3	GND	Logic Ground
4	GND	Logic Ground
5	AD1	Address Data Signal bit 1
6	ADϕ	Address Data Signal bit ϕ
7	AD3	Address Data Signal bit 3
8	AD2	Address Data Signal bit 2
9	AD5	Address Data Signal bit 5
10	AD4	Address Data Signal bit 4
11	AD7	Address Data Signal bit 7
12	AD6	Address Data Signal bit 6
13	A 9	Address Signal bit 9
14	A 8	Address Signal bit 8
15	A 11	Address Signal bit 11
16	A 10	Address Signal bit 10
17	A 13	Address Signal bit 13
18	A 12	Address Signal bit 12
19	A 15	Address Signal bit 15
20	A 14	Address Signal bit 14
21	GND	Logic Ground
22	GND	Logic Ground
23	$\overline{WR}$*	Write enable Signal
24	$\overline{RD}$*	Read enable Signal
25	Sϕ	Status ϕ Signal
26	IO/$\overline{M}$*	I/O or Memory Signal
27	S 1	Status 1 Signal
28	ALE*	Address Latch enable Signal
29	Yϕ	I/O Controller enable Signal
30	CLK	2.54MHz Clock Signal
31	RESET*	TRS-80 Model 100 Reset Signal
32	Ⓐ*	Memory or I/O Access enable Signal
33	$\overline{INTA}$	Interrupt acknowledge Signal
34	INTR	Interrupt request Signal
35	GND	Logic Ground
36	GND	Logic Ground
37	NC	No connection
38	RAM RST	TRS-80 Model 100 RAM reset Signal
39	NC	No connection
40	NC	No connection

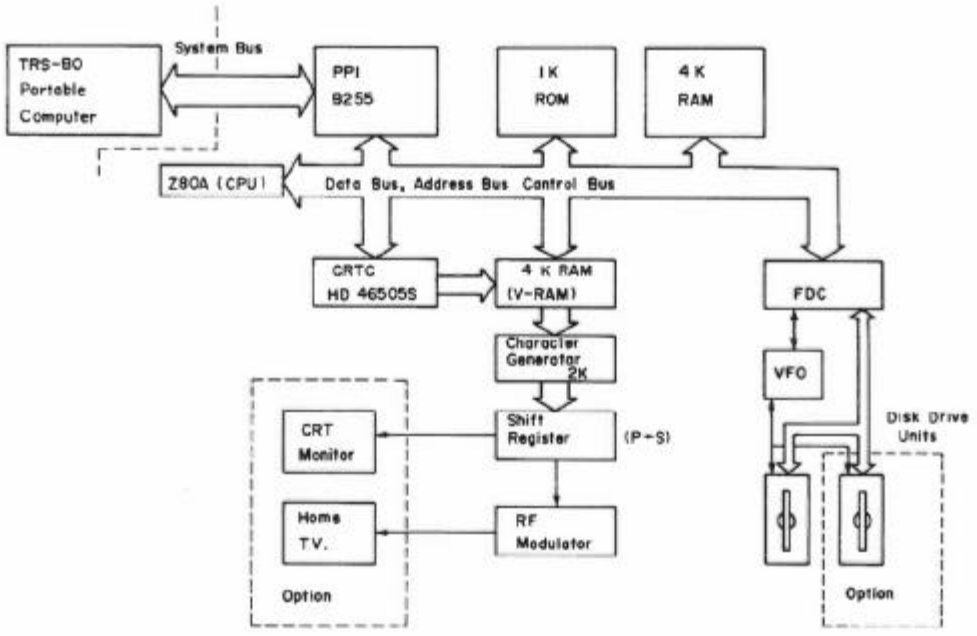


RF Modulator



Appendix B/Technical Information

System Block Diagram



Character Code Tables

Decimal	Hex	Binary	Printed Character
32	20	00100000	
33	21	00100001	!
34	22	00100010	"
35	23	00100011	#
36	24	00100100	$
37	25	00100101	%
38	26	00100110	&
39	27	00100111	'
40	28	00101000	(
41	29	00101001	)
42	2A	00101010	*
43	2B	00101011	+
44	2C	00101100	,
45	2D	00101101	-
46	2E	00101110	.
47	2F	00101111	/
48	30	00110000	0
49	31	00110001	1
50	32	00110010	2
51	33	00110011	3
52	34	00110100	4
53	35	00110101	5
54	36	00110110	6
55	37	00110111	7
56	38	00111000	8
57	39	00111001	9
58	3A	00111010	:
59	3B	00111011	;
60	3C	00111100	<
61	3D	00111101	=
62	3E	00111110	>
63	3F	00111111	?
64	40	01000000	@
65	41	01000001	A
66	42	01000010	B
67	43	01000011	C
68	44	01000100	D
69	45	01000101	E
70	46	01000110	F
71	47	01000111	G
72	48	01001000	H
73	49	01001001	I
74	4A	01001010	J
75	4B	01001011	K
76	4C	01001100	L
77	4D	01001101	M

Decimal	Hex	Binary	Printed Character
78	4E	01001110	N
79	4F	01001111	O
80	50	01010000	P
81	51	01010001	Q
82	52	01010010	R
83	53	01010011	S
84	54	01010100	T
85	55	01010101	U
86	56	01010110	V
87	57	01010111	W
88	58	01011000	X
89	59	01011001	Y
90	5A	01011010	Z
91	5B	01011011	[
92	5C	01011100	\
93	5D	01011101	]
94	5E	01011110	^
95	5F	01011111	_
96	60	01100000	`
97	61	01100001	a
98	62	01100010	b
99	63	01100011	c
100	64	01100100	d
101	65	01100101	e
102	66	01100110	f
103	67	01100111	g
104	68	01101000	h
105	69	01101001	i
106	6A	01101010	j
107	6B	01101011	k
108	6C	01101100	l
109	6D	01101101	m
110	6E	01101110	n
111	6F	01101111	o
112	70	01110000	p
113	71	01110001	q
114	72	01110010	r
115	73	01110011	s
116	74	01110100	t
117	75	01110101	u
118	76	01110110	v
119	77	01110111	w
120	78	01111000	x
121	79	01111001	y
122	7A	01111010	z
123	7B	01111011	{

Decimal	Hex	Binary	Printed Character
124	7C	01111100	|
125	7D	01111101	}
126	7E	01111110	~
128	80	10000000	Ç
129	81	10000001	ü
130	82	10000010	é
131	83	10000011	â
132	84	10000100	ä
133	85	10000101	à
134	86	10000110	å
135	87	10000111	ç
136	88	10001000	ê
137	89	10001001	ë
138	8A	10001010	è
139	8B	10001011	ï
140	8C	10001100	î
141	8D	10001101	ì
142	8E	10001110	Ä
143	8F	10001111	Å
144	90	10010000	É
145	91	10010001	æ
146	92	10010010	Æ
147	93	10010011	ô
148	94	10010100	ö
149	95	10010101	ò
150	96	10010110	û
151	97	10010111	ù
152	98	10011000	ÿ
153	99	10011001	Ö
154	9A	10011010	Ü
155	9B	10011011	ø
156	9C	10011100	£
157	9D	10011101	Ø
158	9E	10011110	×
159	9F	10011111	ƒ
160	A0	10100000	á
161	A1	10100001	í
162	A2	10100010	ó
163	A3	10100011	ú
164	A4	10100100	ñ
165	A5	10100101	Ñ
166	A6	10100110	ª
167	A7	10100111	º
168	A8	10101000	¿
169	A9	10101001	®
170	AA	10101010	¬
171	AB	10101011	½
172	AC	10101100	¼
173	AD	10101101	¡
174	AE	10101110	«

Decimal	Hex	Binary	Printed Character
175	AF	10101111	»
176	B0	10110000	¥
177	B1	10110001	Ä
178	B2	10110010	Ö
179	B3	10110011	Ü
180	B4	10110100	¢
181	B5	10110101	–
182	B6	10110110	ä
183	B7	10110111	ö
184	B8	10111000	ü
185	B9	10111001	ß
186	BA	10111010	¨
187	BB	10111011	ª
188	BC	10111100	µ
189	BD	10111101	º
190	BE	10111110	-
191	BF	10111111	£
192	C0	11000000	à
193	C1	11000001	è
194	C2	11000010	ì
195	C3	11000011	ò
196	C4	11000100	ù
197	C5	11000101	-
198	C6	11000110	ê
199	C7	11000111	ì
200	C8	11001000	â
201	C9	11001001	í
202	CA	11001010	ó
203	CB	11001011	ú
204	CC	11001100	ÿ
205	CD	11001101	ñ
206	CE	11001110	a
207	CF	11001111	ô
208	D0	11010000	À
209	D1	11010001	È
210	D2	11010010	Ì
211	D3	11010011	Ò
212	D4	11010100	Ù
213	D5	11010101	Ì
214	D6	11010110	È
215	D7	11010111	Ê
216	D8	11011000	A
217	D9	11011001	I
218	DA	11011010	O
219	DB	11011011	U
220	DC	11011100	Y
221	DD	11011101	U
222	DE	11011110	E
223	DF	11011111	A
224	ED	11100000	

Decimal	Hex	Binary	Printed Character
225	E1	11100001	(upper left)
226	E2	11100010	(upper right)
227	E3	11100011	(lower left)
228	E4	11100100	(lower right)
229	E5	11100101	
230	E6	11100110	
231	E7	11100111	(upper)
232	E8	11101000	(lower)
233	E9	11101001	(left)
234	EA	11101010	(right)
235	EB	11101011	
236	EC	11101100	
237	ED	11101101	
238	EE	11101110	
239	EF	11101111	
240	F0	11110000	

Decimal	Hex	Binary	Printed Character
241	F1	11110001	
242	F2	11110010	
243	F3	11110011	
244	F4	11110100	
245	F5	11110101	
246	F6	11110110	
247	F7	11110111	
248	F8	11111000	
249	F9	11111001	
250	FA	11111010	
251	FB	11111011	
252	FC	11111100	
253	FD	11111101	
254	FE	11111110	
255	FF	11111111	

Disk BASIC Error Codes

Code	Message	Meaning
1	NF	NEXT without FOR.
2	SN	Syntax Error.
3	RG	RETURN without GOSUB.
4	OD	Out of Data.
5	FC	Illegal Function call.
6	OV	Overflow.
7	OM	Out of Memory.
8	UL	Undefined line.
9	BS	Bad Subscript.
10	DD	Double Dimensioned Array.
11	/0	Division by Zero.
12	ID	Illegal Direct.
13	TM	Type Mismatch.
14	OS	Out ot String Space.
15	LS	String Too Long.
16	ST	String Formula Too Comples.
17	CN	Can't Continue.
18	IO	I/O Error.
19	NR	No RESUME.
20	RW	RESUME without Error.
21	UE	Undefined Error.
22	MO	Missing Operand.
23-49	UE	Undefined Error.
50	IE	Internal Error.
51	BN	Bad File Number.
52	FF	File Not Found.
53	AO	Already Open.
54	EF	Input Past End of File.
55	~~NW~~ NM	Bab Filename.
56	DS	Direct Statement in File.
57	FL	Too Many Files.
58	CF	File Not Open.
59	AT	Bad Allocation Table.
60	DN	Bad Drive Number.
61	TS	Bad Track/Sector.
62	FE	File Already Exists.
63	DF	Disk Full.
64-255	UE	Undefined Error.