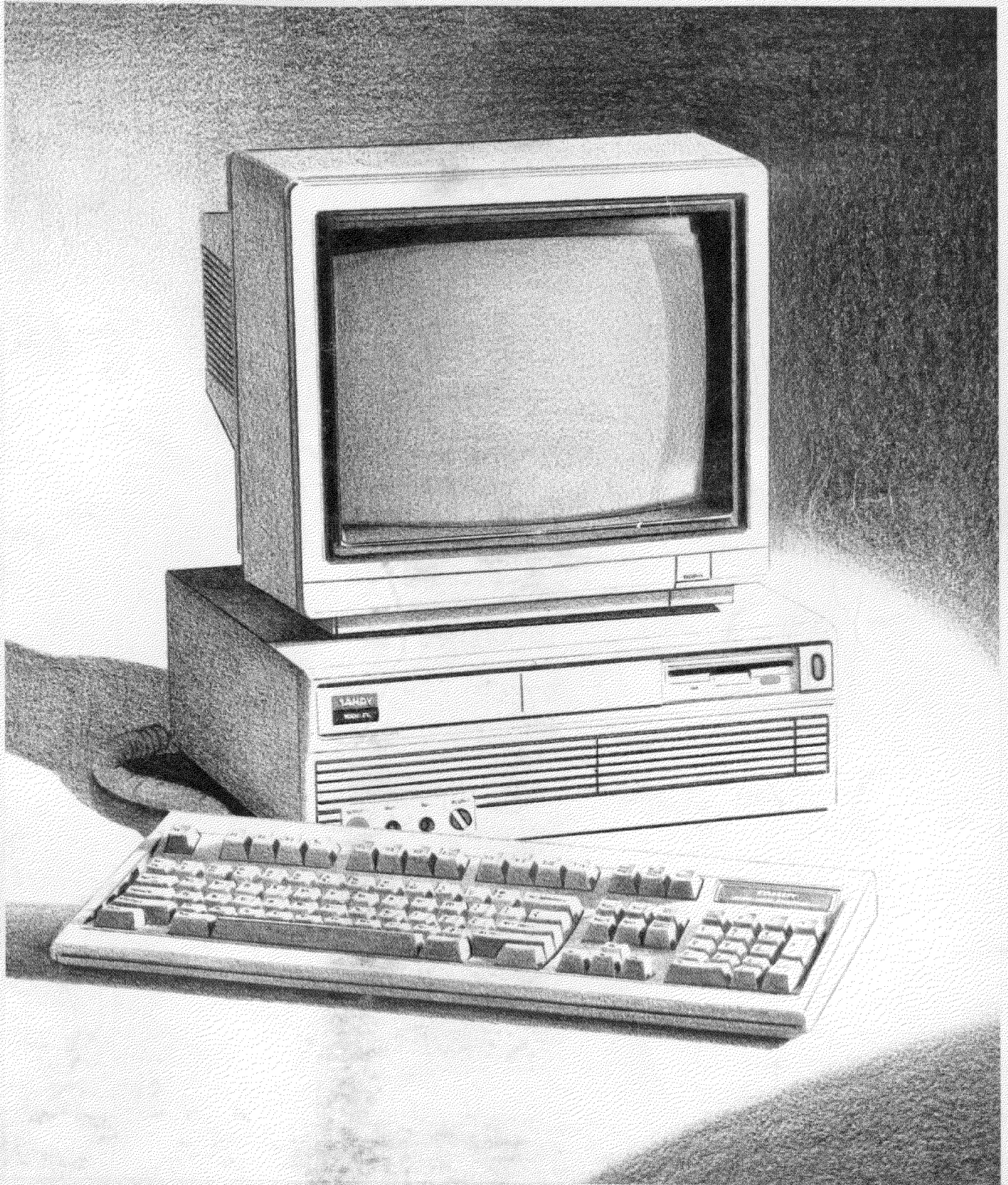


A Practical Guide to the **Tandy 1000 TL**



TANDY®

SERVICE POLICY

Radio Shack's nationwide network of service facilities provides quick, convenient, and reliable repair services for all of its computer products, in most instances. Warranty service will be performed in accordance with Radio Shack's Limited Warranty. Non-warranty service will be provided at reasonable parts and labor costs.

6/86

The FCC Wants You to Know

This equipment generates and uses radio frequency energy. If not installed and used properly, that is in strict accordance with the manufacturer's instructions, it may cause interference to radio and television reception.

It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna
- Relocate the computer with respect to the receiver
- Move the computer away from the receiver
- Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

Warning

This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

5/86

TERMS AND CONDITIONS OF SALE AND LICENSE OF TANDY COMPUTER EQUIPMENT AND SOFTWARE PURCHASED
FROM RADIO SHACK COMPANY-OWNED COMPUTER CENTERS, RETAIL STORES AND RADIO SHACK FRANCHISEES OR
DEALERS AT THEIR AUTHORIZED LOCATIONS

USA LIMITED WARRANTY

I. CUSTOMER OBLIGATIONS

- A. CUSTOMER assumes full responsibility that this computer hardware purchased (the "Equipment"), and any copies of software included with the Equipment or licensed separately (the "Software") meets the specifications, capacity, capabilities, versatility, and other requirements of CUSTOMER.
- B. CUSTOMER assumes full responsibility for the condition and effectiveness of the operating environment in which the Equipment and Software are to function, and for its installation.

II. LIMITED WARRANTIES AND CONDITIONS OF SALE

- A. For a period of ninety (90) calendar days from the date of the Radio Shack sales document received upon purchase of the Equipment. RADIO SHACK warrants to the original CUSTOMER that the Equipment and the medium upon which the Software is stored is free from manufacturing defects. **This warranty is only applicable to purchases of Tandy Equipment by the original customer from Radio Shack company-owned computer centers, retail stores, and Radio Shack franchisees and dealers at their authorized locations.** The warranty is void if the Equipment or Software has been subjected to improper or abnormal use. If a manufacturing defect is discovered during the stated warranty period, the defective Equipment must be returned to a Radio Shack Computer Center, a Radio Shack retail store, a participating Radio Shack franchisee or a participating Radio Shack dealer for repair, along with a copy of the sales document or lease agreement. The original CUSTOMER'S sole and exclusive remedy in the event of a defect is limited to the correction of the defect by repair, replacement, or refund of the purchase price, at RADIO SHACK'S election and sole expense. RADIO SHACK has no obligation to replace or repair expendable items.
- B. RADIO SHACK makes no warranty as to the design, capability, capacity, or suitability for use of the Software, except as provided in this paragraph. Software is licensed on an "AS IS" basis, without warranty. The original CUSTOMER'S exclusive remedy, in the event of a Software manufacturing defect, is its repair or replacement within thirty (30) calendar days of the date of the Radio Shack sales document received upon license of the Software. The defective Software shall be returned to a Radio Shack Computer Center, a Radio Shack retail store, a participating Radio Shack franchisee or Radio Shack dealer along with the sales document.
- C. Except as provided herein no employee, agent, franchisee, dealer or other person is authorized to give any warranties of any nature on behalf of RADIO SHACK.
- D. **EXCEPT AS PROVIDED HEREIN, RADIO SHACK MAKES NO EXPRESS WARRANTIES, AND ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE IS LIMITED IN ITS DURATION TO THE DURATION OF THE WRITTEN LIMITED WARRANTIES SET FORTH HEREIN.**
- E. Some states do not allow limitations on how long an implied warranty lasts, so the above limitation(s) may not apply to CUSTOMER.

III. LIMITATION OF LIABILITY

- A. **EXCEPT AS PROVIDED HEREIN, RADIO SHACK SHALL HAVE NO LIABILITY OR RESPONSIBILITY TO CUSTOMER OR ANY OTHER PERSON OR ENTITY WITH RESPECT TO ANY LIABILITY, LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY "EQUIPMENT" OR "SOFTWARE" SOLD, LEASED, LICENSED OR FURNISHED BY RADIO SHACK, INCLUDING, BUT NOT LIMITED TO, ANY INTERRUPTION OF SERVICE, LOSS OF BUSINESS OR ANTICIPATORY PROFITS OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OR OPERATION OF THE "EQUIPMENT" OR "SOFTWARE." IN NO EVENT SHALL RADIO SHACK BE LIABLE FOR LOSS OF PROFITS, OR ANY INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY BREACH OF THIS WARRANTY OR IN ANY MANNER ARISING OUT OF OR CONNECTED WITH THE SALE, LEASE, LICENSE, USE OR ANTICIPATED USE OF THE "EQUIPMENT" OR "SOFTWARE." NOTWITHSTANDING THE ABOVE LIMITATIONS AND WARRANTIES, RADIO SHACK'S LIABILITY HEREUNDER FOR DAMAGES INCURRED BY CUSTOMER OR OTHERS SHALL NOT EXCEED THE AMOUNT PAID BY CUSTOMER FOR THE PARTICULAR "EQUIPMENT" OR "SOFTWARE" INVOLVED.**
- B. RADIO SHACK shall not be liable for any damages caused by delay in delivering or furnishing Equipment and/or Software.
- C. No action arising out of any claimed breach of this Warranty or transactions under this Warranty may be brought more than two (2) years after the cause of action has accrued or more than four (4) years after the date of the Radio Shack sales document for the Equipment or Software, whichever first occurs.
- D. Some states do not allow the limitation or exclusion of incidental or consequential damages, so the above limitation(s) or exclusion(s) may not apply to CUSTOMER.

IV. SOFTWARE LICENSE

RADIO SHACK grants to CUSTOMER a non-exclusive, paid-up license to use the TANDY Software on **one** computer, subject to the following provisions:

- A. Except as otherwise provided in this Software License, applicable copyright laws shall apply to the Software.
- B. Title to the medium on which the Software is recorded (cassette and/or diskette) or stored (ROM) is transferred to CUSTOMER, but not title to the Software.
- C. CUSTOMER may use Software on a multiuser or network system only if either, the Software is expressly labeled to be for use on a multiuser or network system, or one copy of this software is purchased for each node or terminal on which Software is to be used simultaneously.
- D. CUSTOMER shall not use, make, manufacture, or reproduce copies of Software except for use on **one** computer and as is specifically provided in this Software License. Customer is expressly prohibited from disassembling the Software.
- E. CUSTOMER is permitted to make additional copies of the Software **only** for backup or archival purposes or if additional copies are required in the operation of **one** computer with the Software, but only to the extent the Software allows a backup copy to be made. However, for TRSDOS Software, CUSTOMER is permitted to make a limited number of additional copies for CUSTOMER'S own use.
- F. CUSTOMER may resell or distribute unmodified copies of the Software provided CUSTOMER has purchased one copy of the Software for each one sold or distributed. The provisions of this Software License shall also be applicable to third parties receiving copies of the Software from CUSTOMER.
- G. All copyright notices shall be retained on all copies of the Software.

V. APPLICABILITY OF WARRANTY

- A. The terms and conditions of this Warranty are applicable as between RADIO SHACK and CUSTOMER to either a sale of the Equipment and/or Software License to CUSTOMER or to a transaction whereby Radio Shack sells or conveys such Equipment to a third party for lease to CUSTOMER.
- B. The limitations of liability and Warranty provisions herein shall inure to the benefit of RADIO SHACK, the author, owner and or licensor of the Software and any manufacturer of the Equipment sold by Radio Shack.

VI. STATE LAW RIGHTS

The warranties granted herein give the **original** CUSTOMER specific legal rights, and the **original** CUSTOMER may have other rights which vary from state to state.

**A Practical Guide to the
Tandy 1000 TL**

Tandy 1000 TL BIOS:
©1984, 1985, 1986, 1987, 1988
Phoenix Software Associates, Ltd. and Tandy Corporation.
All Rights Reserved.

MS-DOS Software:
©1981, 1986 Microsoft Corporation.
Licensed to Tandy Corporation.
All Rights Reserved.

GW-BASIC Software:
©1983, 1984, 1985 Microsoft Corporation.
Licensed to Tandy Corporation.
All Rights Reserved.

DeskMate Spell Checker
©1986-88 Tandy Corporation; Microlytics, Inc; UFO Systems, Inc; Xerox Corp.
All Rights Reserved

All portions of this software are copyrighted and are the proprietary and trade secret information of Tandy Corporation and/or its licensor. Use, reproduction, or publication of any portion of this material without the prior written authorization of Tandy Corporation is strictly prohibited.

A Practical Guide to the Tandy 1000 TL:
©1988 Tandy Corporation.
All Rights Reserved.

Reproduction or use of any portion of this manual, without express written permission from Tandy Corporation and/or its licensor, is prohibited. While reasonable efforts have been made in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors in or omissions from this manual, or from the use of the information contained herein.

Tandy, Radio Shack, and DeskMate are registered trademarks of Tandy Corporation.

Microsoft and MS-DOS are registered trademarks of Microsoft Corporation.

GW is a trademark of Microsoft Corporation.

IBM is a registered trademark of International Business Machines Corporation.

PC/XT is a trademark of International Business Machines Corporation.

Apple is a trademark of Apple Computer Inc.

Hercules is a trademark of Hercules Computer Technology.

Contents

A Practical Guide to the Tandy 1000 TL

The Tandy 1000 TL Features	1
The Keyboard	5
Using Diskettes and the Diskette Drive	7
Care and Handling of Diskettes	7
Inserting and Removing Diskettes	7
Write-Protecting 3 1/2-inch Diskettes	8
Write-Protecting 5 1/4-inch Diskettes	8
Using Program Diskettes	9
Making Copies	9
Using MS-DOS	11
Entering MS-DOS Instructions	11
Changing Drives	12
Preparing Diskettes with Format	12
Duplicating Diskettes with Diskcopy	12
Duplicating Information with Xcopy	13
Options for the Tandy 1000 TL	15
Adding External Options	15
Adding Internal Options	16
Main Logic Board	17
Adding a New Drive	18
Hard Disk Card Installation	19
There's More	21
Changing the Computer Defaults	23
Using Setup	23
The Default Settings	24
Troubleshooting	27
Video Problems	27
Printer Problems	27
Caring for Your Equipment	28
Tandy 1000 TL Specifications	29
MS-DOS Quick Reference	33
MS-DOS Commands	33
Special Keys	51
Edlin Commands	52
Edlin Editing Keys	54
Debug Command Parameters	54
Debug Commands	56
GW-BASIC Quick Reference	61
Loading GW-BASIC	61

GW-BASIC Commands and Statements	63
Typing Keywords Using the ALT Key	101
Exponential Notation and Numeric Precision Characters	101
Operator Precedence	101
Video Modes	102
Enhanced Graphics Color Selection	103
Error Codes and Messages	104
Index	107

The Tandy 1000 TL Features

A computer is made up of physical parts known as *hardware*, such as the system unit, the monitor, and the keyboard. You can add optional hardware to your system such as a printer, a modem, or a mouse.

Your hardware runs *software*, programs that send instructions to the computer. Ordinarily, software is stored on diskettes.

Your Tandy 1000 TL is quite special. It has a great deal of software built right in.

This chapter explains many of the features of your Tandy 1000 TL. You don't need to learn all this information to operate your computer, but it can be helpful to have a general idea of how the parts work together.

Feature

What it Means to You

MS-DOS in ROM

The Tandy 1000 TL uses the MS-DOS *operating system*. MS-DOS is the software that controls the basic functions of IBM® PC, XT, or AT-compatible computers. Ordinarily, you have to load the operating system from a diskette or hard disk when you start up a computer.

Having the basic portion of MS-DOS in the Tandy 1000 TL's ROM means that all you have to do is turn on your computer and it is ready to go. Also, access to many of the commonly used MS-DOS functions is available without having to use the MS-DOS diskette.

DeskMate in ROM

If you do not want to work in DeskMate when you turn on your computer, you can tell your computer to bypass DeskMate and start in MS-DOS. See the section "Changing the Computer Defaults."

*You can also exit DeskMate by pressing **ESC**.*

DeskMate is software that provides a user-friendly environment and a collection of useful and fun application programs.

Because a large portion of DeskMate is also built into your computer, you can begin using DeskMate as soon as you turn on your computer.

You can run other IBM-compatible programs right from the DeskMate desktop, if you like, without having to exit to MS-DOS. See the provided DeskMate manuals for more information on DeskMate.

Speller in ROM

Because the Tandy 1000 TL's spelling checker is built-in, it is super-fast. You can use the spelling checker for DeskMate applications as well as other ASCII files.

**A dual-speed, 4/8 MHz
CPU chip**

The *central processing unit* is the brain of your computer. This is the chip that processes information.

Dual-speed processing lets you choose the appropriate speed (4 or 8 megahertz) for your programs. Most programs run better at 8 megahertz. However, some programs are speed-sensitive; your computer lets you run these programs at the slower speed.

**640K of RAM,
Expandable to 768K**

Random access memory is your computer's temporary memory. This is where programs, instructions, and information are kept while you are working. Turning off your computer erases the random access memory. Be sure that you always save your work on disk before you turn off your computer.

Application programs require different amounts of memory. Your computer comes with enough memory to run many programs. If you want more memory, or if some of your programs require more memory, you can easily expand to as much as 768 kilobytes. (One kilobyte equals 1,024 bytes, or characters of information.)

Built-in Video Support

Unlike many other computers, the Tandy 1000 TL comes ready to connect to a color monitor (and supports 16-color display) or a monochrome monitor. You do not need to purchase a video adapter card.

Special EEPROM Circuitry

The Tandy 1000 TL has a special *electronically erasable program-mable read-only memory* that gives your computer the ability to remember the way you want it to run.

The EEPROM stores such information as what monitor you have, how much memory you have, and whether you want your computer to start up in DeskMate, the built-in MS-DOS, or from a diskette. Your computer is set up at the factory for the most popular configuration; however, you can change this information any time.

**A Built-in 3 1/2-inch
Diskette Drive**

Your computer uses diskette drives to read programs and information from diskettes and to store programs and information on diskettes.

The 3 1/2-inch diskette drive lets you use diskettes that can store more information. These diskettes are also smaller, and not as susceptible to damage.

A Reset Button

You do not have to remember a complicated key sequence to reset your computer. You only have to press one button.

Music and Sound

The Tandy 1000 TL has a three-voice sound circuit with an analog-to-digital/digital-to-analog convertor, a built-in speaker, volume control, a microphone jack, and an earphone jack and is capable of sophisticated computer-generated music and sound.

A Full-feature, 101-key, Enhanced Keyboard

The keyboard that comes with your computer is sleek and easy-to-use. It has the industry-standard key arrangement and enhanced features found on more expensive computers.

SmartWatch

SmartWatch is a perpetual clock and calendar on a chip. It has a built-in battery so it can keep time even when your computer is turned off.

Built-in Serial Port

You don't have to buy an optional serial adapter card to connect a serial device to your computer. With the Tandy 1000 TL, you can immediately connect a serial mouse, modem, or a serial printer. Also, using the built-in serial port, you can connect your TL to another computer.

Built-in Parallel Port

Connect your parallel printer to your computer immediately. No extra adapter card is needed.

Built-in Joystick Ports

Simply plug in joysticks or a color mouse. No extra adapter card is needed.

Five IBM PC/XT-compatible, 10-inch Expansion Slots

You can add as many as five optional adapter cards to upgrade or customize your computer.

MS-DOS/BASIC Diskette

This diskette contains the complete MS-DOS version 3.30 operating system and version 3.20 of the GW-BASIC programming language.

Supplemental Programs Diskette

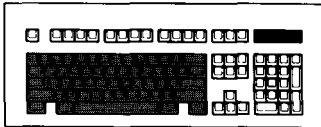
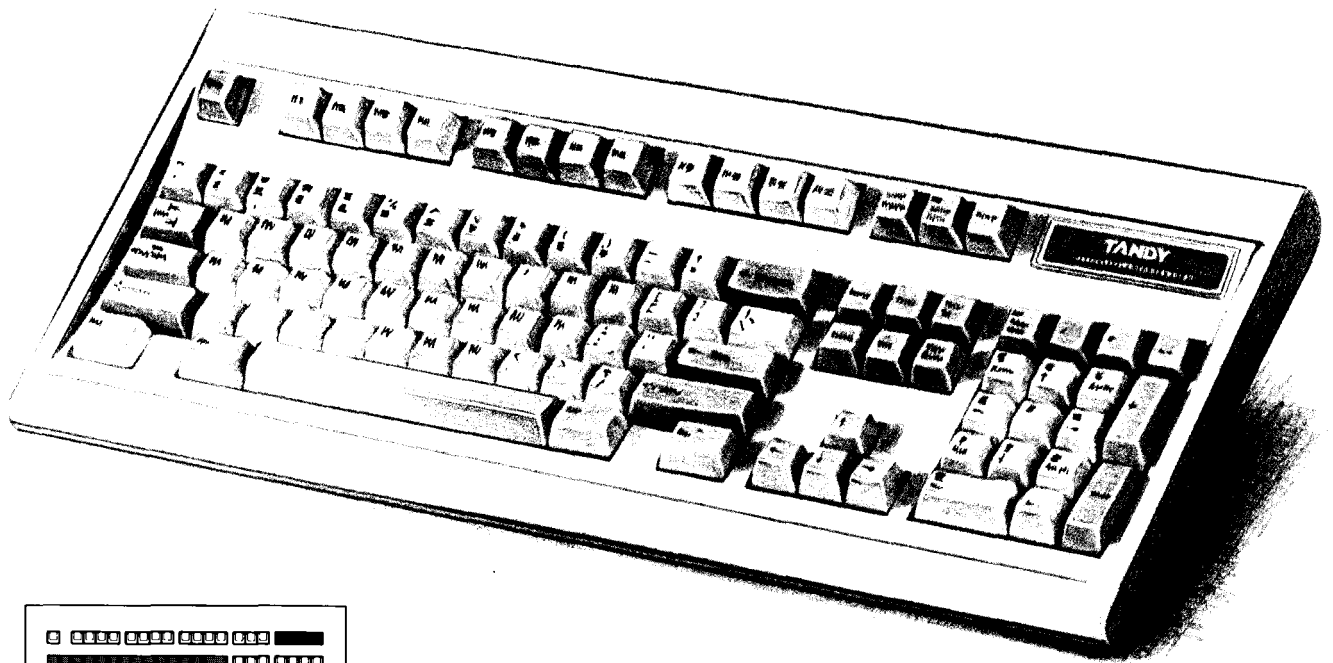
This diskette contains several hard disk setup commands and other utility programs.

DeskMate Diskettes

The DeskMate diskettes that come with your computer contain the applications and accessories that let you immediately make use of DeskMate programs and functions.

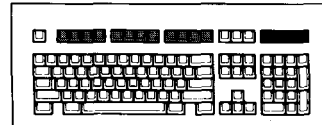
The Keyboard

Your computer's keyboard consists of four sections: the function keys, the typewriter keys, the cursor keys, and the numeric keypad.



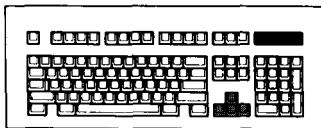
Typewriter Keys

The left side of the keyboard, below the function keys, is similar to the keyboard of a standard typewriter. However, when you hold down a character or number key, the keystroke repeats automatically until you release the key.



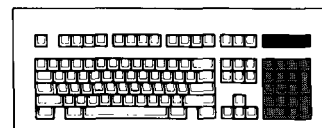
Function Keys

The 12 function keys at the top of the keyboard are program-specific. Their functions depend on the program you are running.



Cursor Keys

The cursor keys are clustered at the bottom of the keyboard, between the typewriter keys and the numeric keypad. Many programs use these keys to control the movement of the cursor (or highlight) on the screen.

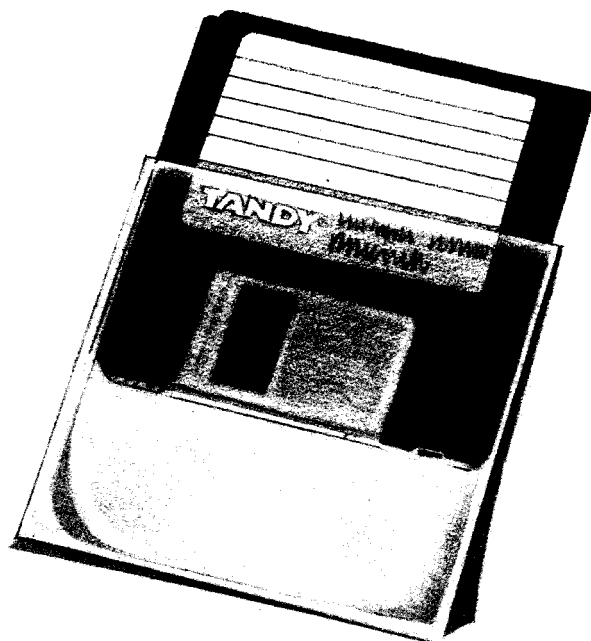


Numeric Keypad

The numeric keypad on the right side of the keyboard is arranged the same as a calculator keypad. Number keys are normally the shifted characters on the numeric keypad. (You hold down **SHIFT** and press a number.) Press **NUM LOCK** to use the keypad for extensive number entry. **NUM LOCK** works like the **SHIFT LOCK** key on a typewriter. When number lock is on, you can type numbers without pressing the **SHIFT** key.

Using Diskettes and the Diskette Drive

Care and Handling of Diskettes



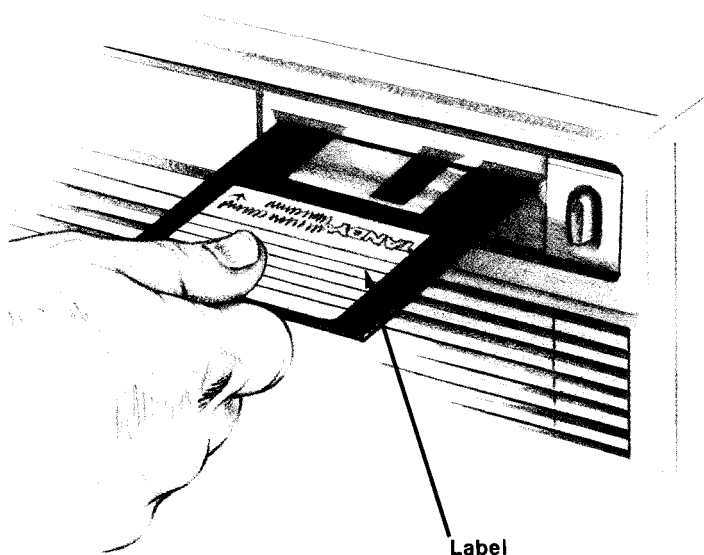
Diskettes store information, such as programs and the data you create, in *files*.

The diskette drive in the Tandy 1000 TL uses double-sided, 3 1/2-inch, 80-track diskettes (Cat. No. 26-417 and 26-418). These diskettes can store approximately 720 kilobytes (more than 730,000 characters) of information.

To protect your diskettes (and the information they contain), follow these guidelines:

- Keep diskettes away from magnetic fields (such as transformers, AC motors, magnets, speaker systems, TVs, and radios).
- Don't lay a diskette on top of or next to the computer system's console.
- Keep diskettes out of direct sunlight and away from heat.
- Keep diskettes away from dust.
- It's a good idea to make several copies of your diskettes (*backups*) to use as working diskettes. See the "Making Copies" section of this chapter.

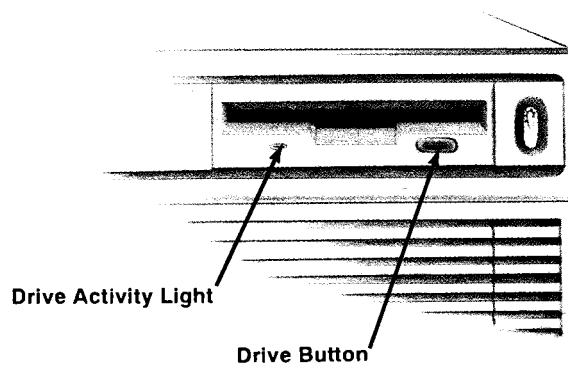
Inserting and Removing Diskettes



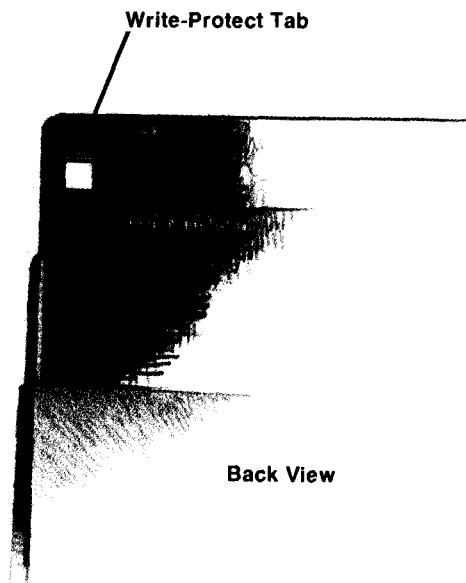
To insert a diskette into an empty drive, gently slide it, label side up, into the drive until the diskette clicks into place.

A drive's activity light is on whenever the diskette drive is active. Removing a diskette from a drive when the drive activity light is on can destroy data on the diskette.

Before removing a diskette from a drive, be sure the drive activity light is off. Then, push in the drive button. The diskette slides out of the drive.



Write-Protecting 3 1/2-inch Diskettes

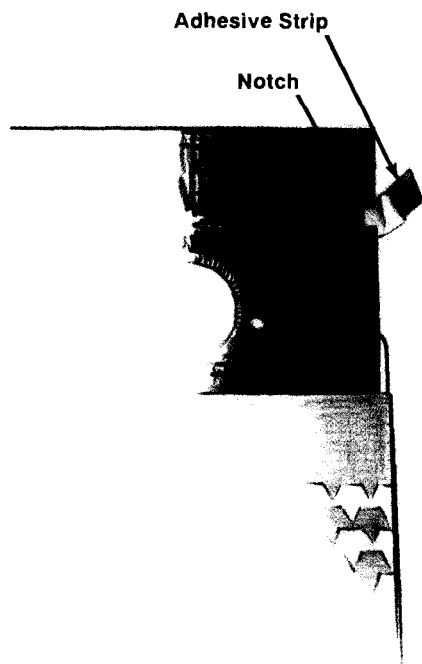


If you have important data on a diskette, you might want to make certain that your computer cannot erase or write over it. You can do this by *write-protecting* the diskette.

A 3 1/2-inch diskette has a small, square hole in the upper-right corner. This is the write-protect hole. From the back of the diskette, move the small tab (normally red or black) up so that the hole is open. This write-protects the diskette.

If you want to write to the diskette again, move the tab down so that the hole is completely covered.

Write-Protecting 5 1/4-inch Diskettes



If you add a 5 1/4-inch diskette drive to your computer, you use a different method to write-protect diskettes for that drive.

To write-protect a 5 1/4-inch diskette, fold a small adhesive strip over the notch cut in the side of the diskette as shown.

When this tab is in place, the computer can *read* information from the diskette, but cannot *write* information to the diskette. If you want to write to the diskette again, remove the strip.

Using Program Diskettes

The Tandy 1000 TL is capable of running thousands of application programs, such as word processing and spreadsheet programs. These programs come on diskettes.

You can run application programs right from the DeskMate desktop, or you can load them from MS-DOS. See the DeskMate manuals for instructions on running applications from the desktop.

If you want to run an application program from MS-DOS, press **ESC** at the DeskMate desktop. When you see the MS-DOS system prompt, **A>**, follow the instructions that came with your application program.



Making Copies

It's a good idea to make *backup* copies of your application program diskettes. Use the backups as your work diskettes, and put your original diskettes away for safekeeping.

Some application programs you buy are *copy-protected*. You cannot make copies of these diskettes. Check the program's manual for information on protecting the data on copy-protected diskettes.

You can make copies of specific files, or entire diskettes using either DeskMate or MS-DOS commands. Instructions for making copies with DeskMate are in the *DeskMate User's Reference*. Instructions for making copies with MS-DOS are included in the next chapter, "Using MS-DOS."

Using MS-DOS

*When you are in the DeskMate desktop, you can exit to MS-DOS by pressing **ESC**.*

*To re-enter DeskMate from MS-DOS, press **F12**.*

MS-DOS is an operating system. An operating system is software that manages your computer's activities. Operating system software must be in place before you can run application programs.

How much you need to know about your MS-DOS operating system depends on how you plan to use your computer. If you plan to primarily run applications from DeskMate, you don't need to know anything about MS-DOS.

On the other hand, if you plan to use advanced operating system features or create your own programs, you need to become quite familiar with the operating system.

The remainder of this chapter presents information on several basic procedures you might find handy, including:

- Preparing a diskette to store information
- Copying the operating system, program, and data files
- Duplicating a diskette

If you want to know more about MS-DOS and how to use its many powerful functions, you can purchase the *Tandy 1000 MS-DOS Reference Manual* and the *Tandy 1000 GW-BASIC Reference Manual*. Tandy also sells two other books designed to help you understand your operating system: *MS-DOS: The Basics, Vol. 1*, and *MS-DOS: Advanced Applications, Vol 2*.

The "MS-DOS Quick Reference" and the "GW-BASIC Quick Reference" sections of this manual provide a guide for using your operating system and the GW-BASIC programming language. However, these sections provide only a quick overview.

Entering MS-DOS Instructions

The MS-DOS instructions you give to the computer are called commands. You type commands at a *system prompt* (usually **A>**). You can type commands in either upper- or lowercase letters.

Changing Drives

If you have more than one drive, you can easily change from one to another. Drive A (the bottom drive) is considered the current drive unless you specify otherwise. If you have a hard disk drive (Drive C) it is considered the current drive.

If you have two diskette drives, you can change the current drive to Drive B by typing:

b:

Then press **ENTER**

The system prompt changes from A> to B>. Drive B is now the current drive.

Preparing Diskettes with Format

If you are formatting a diskette because you want to use Copy to duplicate an operating system diskette, type:

format a: /s and press ENTER.

Before you can use a new blank diskette, you must prepare it to hold information. To do this you use the Format command.

Exit any application program you are using.

When you see the system prompt, A>, insert a blank diskette.

Type **format**, followed by a space. Type the letter of the drive that contains the new diskette and a colon. Then, press **ENTER**.

For example, if you want to format a blank diskette that you have in Drive A, type:

format a:

Then, press **ENTER**

Caution: Formatting erases everything on a diskette; copy any important files to another diskette before formatting.

Duplicating Diskettes with Diskcopy

The Diskcopy command formats your blank diskette and copies the entire contents of another diskette (of the same type) to that blank diskette in one easy step.

Using Diskcopy with One Disk Drive

At the system prompt, A>, type:

```
diskcopy
```

Then, press **ENTER**

Insert the diskette you wish to copy into Drive A. Then, press any key to begin. MS-DOS will tell you when to swap the original diskette and the blank diskette.

Using Diskcopy with Two Disk Drives

At the system prompt, A>, type:

```
diskcopy a: b:
```

Then, press **ENTER**

Insert the diskette you wish to copy into Drive A and the blank diskette into Drive B. Then, press any key to begin.

Duplicating Information with Xcopy

The Xcopy command copies one file at a time. You can use Xcopy to back up an entire diskette or to duplicate selected files only. Use Xcopy to back up a diskette from one type of drive to another.

1. Prepare a blank diskette using the Format command.
2. Insert your MS-DOS diskette into Drive A.
3. If you have a two-drive system, insert the formatted blank diskette into Drive B.
4. At the system prompt, A>, type `xcopy` followed by a space. Type the letter of the drive that contains the original diskette, followed by a colon. Then, type the name of the file you want to copy. Next, type the letter of the drive that will contain the blank diskette, followed by a colon. If you have only one drive, use `b:` as your second letter anyway. Type another space, type `/w`, and then press **ENTER** to begin.

For example, type:

```
xcopy a:filename b: /w
```

Then, press **ENTER**

If you want to copy **all** the files on the original diskette, type:

```
xcopy a:*. * b: /s/e/w
```

Then, press **ENTER**

The `/e` and `/s` switches cause Xcopy to copy all files on a diskette, including any empty files or any files in subdirectories.

5. Xcopy prompts you to insert the diskette you want to copy from.
6. If you have a one-drive system, the Xcopy command tells you when to insert the Drive A (source) diskette or the Drive B (target) diskette.

Options for the Tandy 1000 TL

You can expand your Tandy 1000 TL's capabilities in many ways. This section describes some of the more popular accessories and *upgrades*.

Adding External Options

In most instances, adding external options is as easy as connecting a few cables.

Option	What it Means and What You Need
A Printer	You can print the data you create, from pictures to budgets. Because the Tandy 1000 TL has a printer port, you do not need to buy an adapter card to connect a printer. The printer (parallel) port is located on the back panel. Tandy has a full line of printers, from fast and economical dot matrix printers to typeset-quality laser printers.
A Serial Mouse	A mouse is a device that you roll on your desk top to control the actions of some computer programs. If you do a lot of drawing or word processing, a mouse (with a program that recognizes it) can really help. You can connect a serial mouse to the Tandy 1000 TL's built-in serial port.
Joysticks	Joysticks can make many games and educational programs easier and more fun. The back panel of the Tandy 1000 TL has connections for two joysticks. You do not need to purchase an adapter card.
A Modem	<p>Adding a modem to your computer system lets you communicate with other computers over the phone lines. This means you can share information with other computer users.</p> <p>With the appropriate software, you can use your computer as a terminal to a larger computer system.</p> <p>It also means that you can connect to <i>on-line</i> services such as PC-Link, that offer computer shopping, encyclopedia references, news services, stock prices, electronic mail, games, and so on.</p>
A Microphone	<p>You can plug a microphone into your computer to record music and sound. Tandy carries several microphones suited for this purpose.</p> <p>Please note that if you wish to plug in a stereo or other line audio output into the microphone jack, you will need to move a jumper on the main logic board from E6-E7 to E7-E8. See the main logic board illustration later in this section.</p>

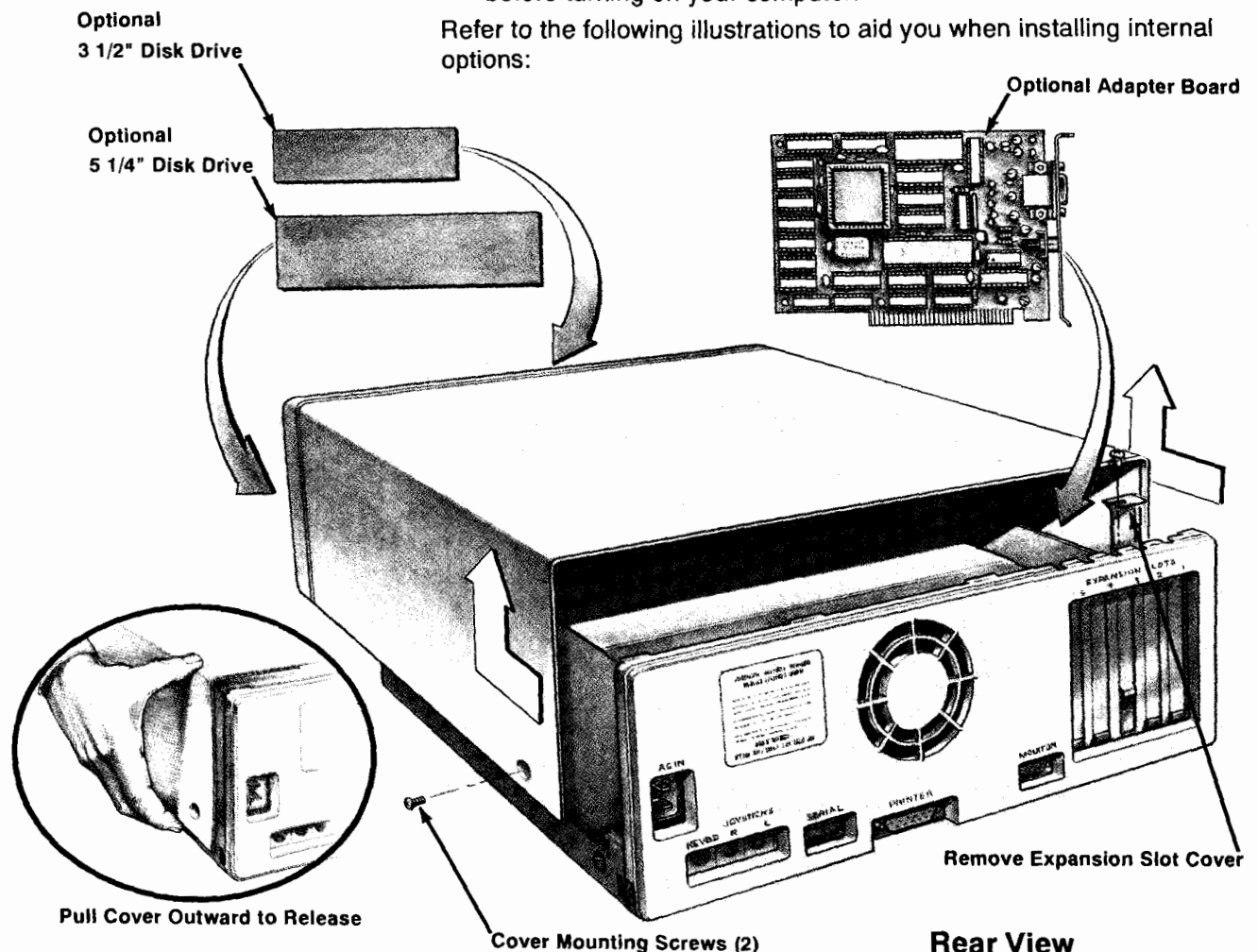
Adding Internal Options

If you want to handle the installations yourself, all of Tandy's add-on products come with complete instructions. If you do not feel comfortable doing this kind of work, your nearest Radio Shack Service Center can do the job and guarantee the work.

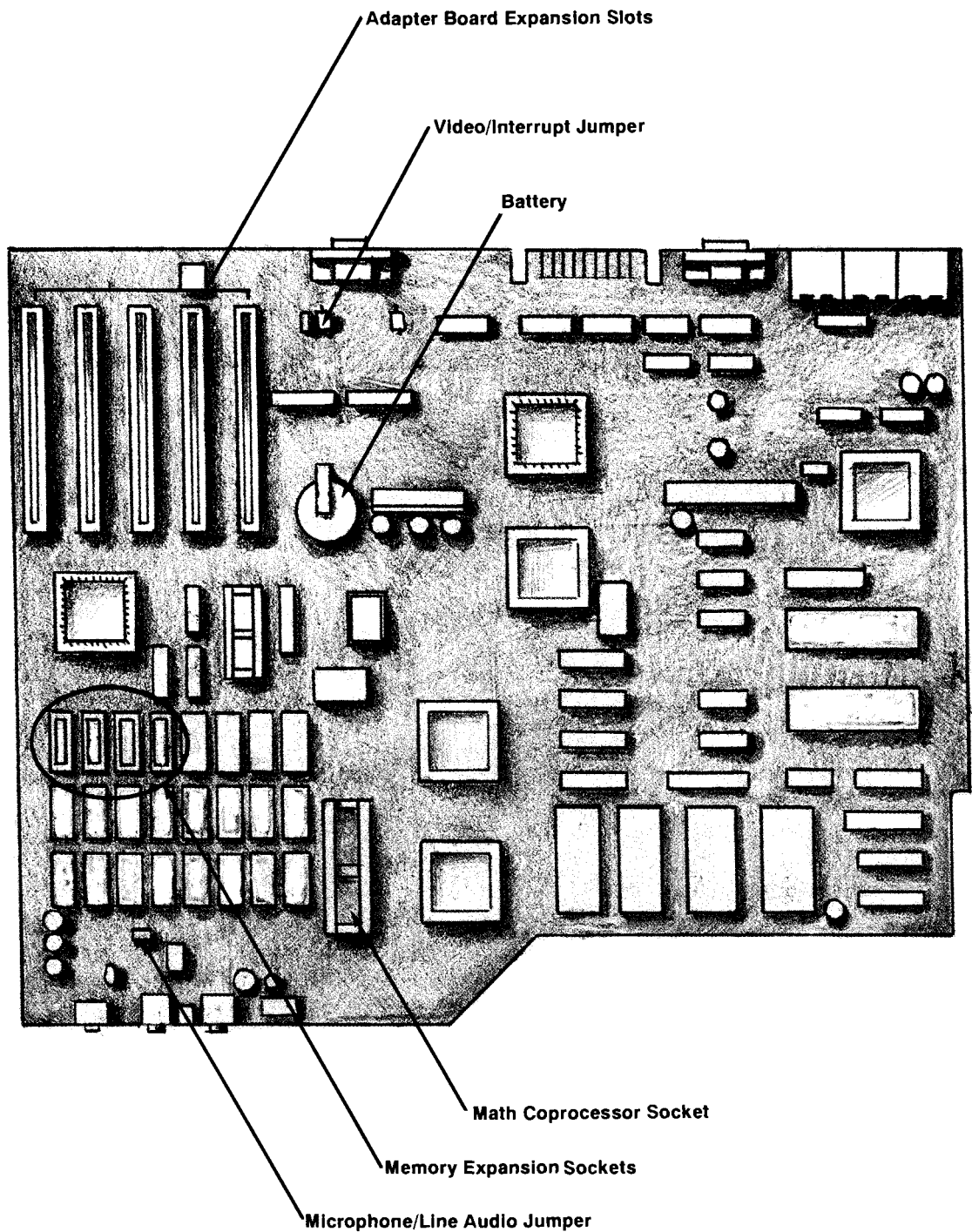
When adding internal options, do so in the following order:

1. Turn off the power to your computer.
2. Unplug the unit from the electrical outlet.
3. Remove the cover.
4. Install any chips that go on the main logic board.
5. Adjust the jumper settings that need to be changed on the main logic board.
6. Install any additional drives.
7. Install any adapter boards.
8. Replace the cover and be sure that screws are tightly in place before turning on your computer.

Refer to the following illustrations to aid you when installing internal options:



Main Logic Board

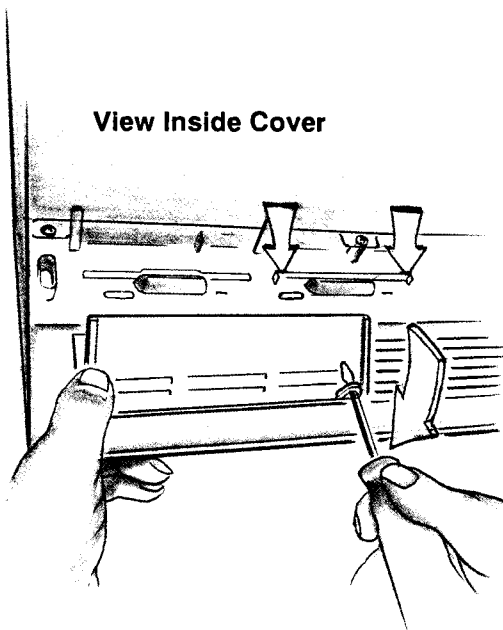


Adding a New Drive

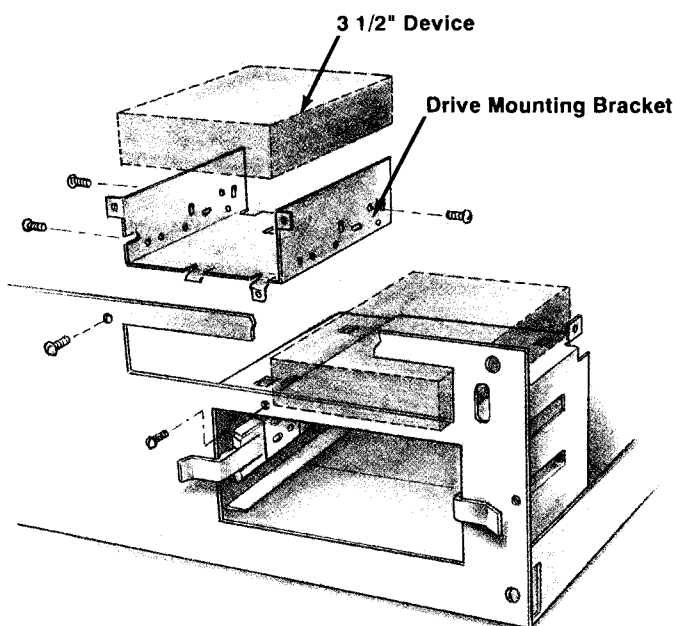
Press tab through slot to remove the panel for a 3 1/2" drive.

Use a screwdriver as a lever to break away the lower panel.

View Inside Cover

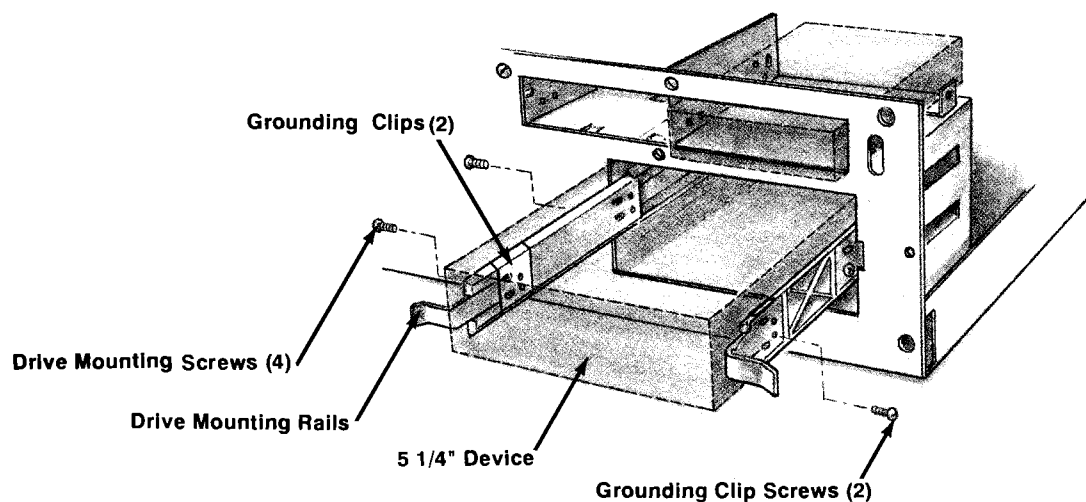


Installing a Second 3 1/2" Disk Drive



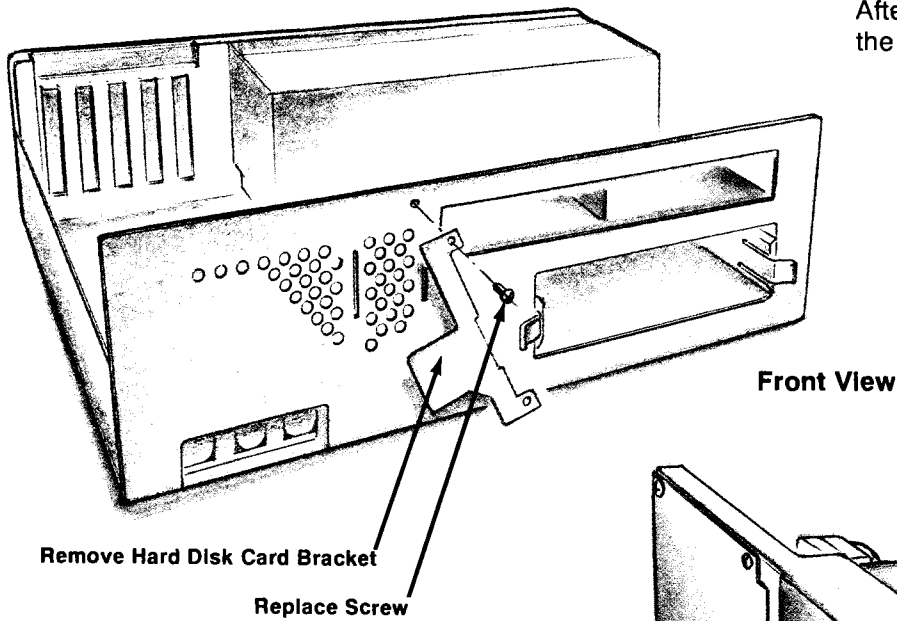
Front View

Installing a 5 1/4" Disk Drive

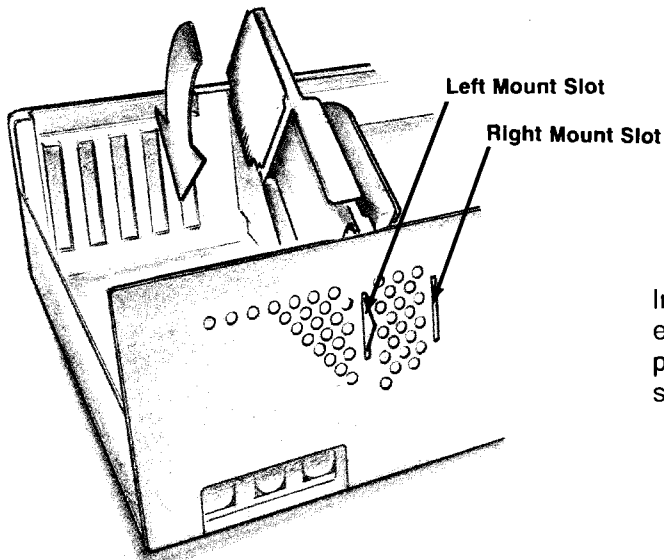
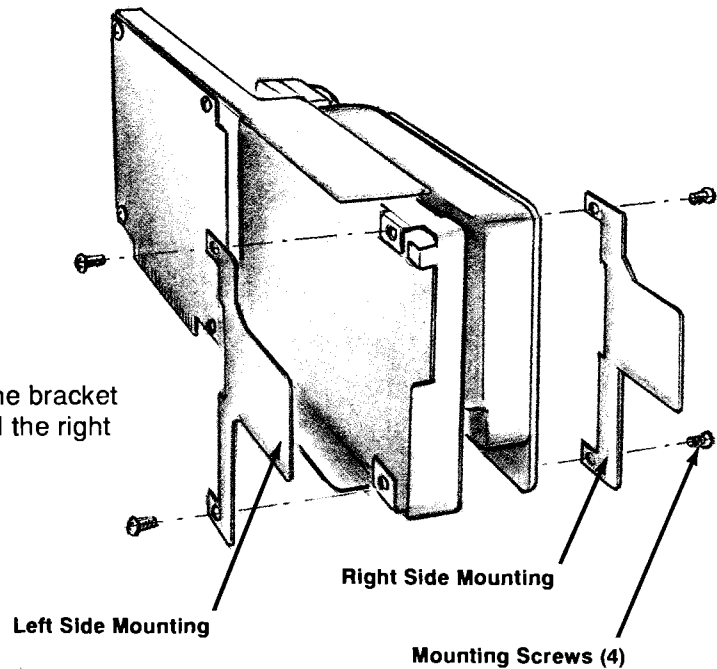


Hard Disk Card Installation

After you remove the cover, remove the hard disk card bracket.



Attach the bracket to the hard disk card. The bracket attaches on the left side of some cards and the right side of other cards.



Insert the hard disk card bracket first, so the bracket slips into the appropriate mounting slot. Then, press the card down into the right-most expansion slot.

Option	What it Means and What You Need
Memory Kits	<p>You can expand your TL to 768K <i>on-board</i> memory. Memory chips are plugged into sockets on the main logic board.</p>
Math Coprocessor	<p>Some programs that do a great deal of calculating benefit greatly from a Math Coprocessor. The coprocessor can perform highly accurate mathematical calculations while your computer is performing another set of instructions.</p> <p>All you have to do is plug the coprocessor chip into the TL main logic board.</p>
Diskette Drives	<p>Some computer programs are much easier to use with more than one drive. Also, copying the contents of diskettes is much quicker and easier with two drives. Radio Shack can supply you with either a 3 1/2-inch or 5 1/4-inch internal drive.</p>
A Hard Disk Card, 20- or 40-Megabytes	<p>Hard disks hold much more data than a diskette, and they operate much faster. One 20-megabyte hard disk card can contain as much information as more than 55, 5 1/4-inch diskettes.</p> <p>To install a hard disk card you need to use the bracket stored on the front of your computer, under the cover. See the illustrations at the beginning of this section.</p>
Expanded Memory	<p>Programs designed to use expanded memory can run faster and more efficiently. You can also use expanded memory as a <i>virtual</i> or RAM drive. That is, you can set aside a portion of memory to act like a disk drive, only much faster.</p> <p>Installing a memory expansion board in your Tandy 1000 TL lets you add up to two megabytes of expansion memory.</p>
A Modem Board	<p>Installing a modem board allows your computer to communicate with other computers through the phone lines. With an internal modem board you do not have to use your serial port to connect a modem.</p>
TandyLink	<p>The TandyLink Adapter board lets you connect your computer to a workgroup. In a workgroup environment, you can share information among the members of the group without ever leaving your desk.</p>

Enhanced Video

With an enhanced video adapter board and a compatible monitor, your TL can generate high-resolution (EGA and VGA) graphics.

The video output of the Tandy 1000 TL operates as a Color Graphics Adapter (CGA) with 640 x 200 resolution. Its monochrome video output is compatible with both the MDA text adapter and the Hercules video adapter (720 x 348 resolution). If the programs you run require higher resolution, or if you want your computer's graphics to look better, Tandy can provide a variety of high-resolution video cards and monitors.

There's More

This chapter discussed only a few of the options available for your Tandy 1000 TL. Other options you can add might include:

- A MIDI Interface Adapter for connecting musical synthesizers (keyboards) to your connector
- A tape or disk cartridge drive to use for archiving important information
- An answering machine board to turn your computer into a personal voice mail system
- A drawing tablet for creating graphics
- A FAX board to enable your computer to transmit and receive images
- A CD-ROM drive to let your computer access the enormous storage capabilities of read-only disks

For further information, talk to your store sales personnel and obtain a free copy of the *Tandy Computer Catalog & Software Reference Guide*.

Changing the Computer Defaults

The Tandy 1000 TL is a versatile computer that has the ability to *remember* certain settings that determine the way it works. Some of the settings affect:

- Whether your computer prompts you for the date and time when it starts
- Whether your computer is to check the status of its memory every time it starts
- Whether you want the computer to start with programs built into its memory or to start from programs on a diskette
- Which drive you want to use as Drive A and which drive you want to use as Drive B (if you have more than one diskette drive)
- The speed at which the computer runs

The TL stores these settings in a special memory chip called an EEPROM. The EEPROM retains its contents, even when your computer is turned off.

The TL's original EEPROM settings are called *default* settings. If you decide to change these defaults, the TL stores the changes you made and remembers these new settings, until you change them again. Normally, you only have to make the changes once, instead of every time you start your computer.

You change the settings in the EEPROM using the Setuptl program. The following pages describe each setting. If you are satisfied with the default settings, you do not need to use the information in this section.

Using Setup

To view the Setup screen, first exit DeskMate. (From the desktop, press **ESC**.)

Place the MS-DOS/GW-BASIC Diskette in Drive A, and type:

`setuptl`

Then, press **ENTER**

In a moment, the following Setuptl display appears.

VIDEO DISPLAY	COLOR	MONOCHROME
AUTOMATIC PROMPT FOR DATE AND TIME	NO	YES
MEMORY DIAGNOSTICS ON START-UP	NO	YES
PRIMARY START UP DEVICE	ROM	DISK
INITIAL START-UP PROGRAM	MSDOS	DESKMATE
COMPUTER SPEED	FAST	SLOW
NUMBER OF DISK BUFFERS (2-17)	10	
MAXIMUM # OF OPEN FILES (8-23)	10	
CHECK FOR CONFIG.SYS ON DRIVE	NO	
CHECK FOR AUTOEXEC.BAT ON DRIVE	NO	
DISKETTE DRIVE A DESIGNATION	TOP	BOTTOM
ESC TO QUIT, F1 TO UPDATE. USE ↑ AND ↓ TO MOVE, ← AND → TO CHANGE F10: FACTORY SETTINGS		

Use the up and down arrow keys to move to the line you want to change. Use the left and right arrow keys to make the changes. If you are changing a number, such as the NUMBER OF DISK BUFFERS, pressing the right arrow increases the value of the number and pressing the left arrow decreases the value of the number. Otherwise, use the left and right arrows to highlight a choice. When you have selected the correct number or choice, use the up or down arrow to move to the next line you want to change. When you have completed all selections, press F1 to store them.

If you decide you want to return to the default settings, press F10.

The Default Settings

The following information explains each setting. The default setting is indicated in boldface type.

VIDEO DISPLAY

COLOR MONOCHROME

Select **MONOCHROME** if your monitor is capable of only two colors (black and white, black and green, black and amber). You can also toggle your computer between color and monochrome monitor from the keyboard. If, when you start your computer, no message appears on the screen, press **CTRL-ALT-SHIFT-V** to switch from color to monochrome.

AUTOMATIC PROMPT FOR DATE AND TIME **NO** **YES**

The Tandy 1000 TL has the SmartWatch built in. Select YES if you want the date and time prompt to appear when you start your computer with the MS-DOS operating system.

MEMORY DIAGNOSTICS ON START-UP **NO** **YES**

Select YES if you want your computer to take time to test the integrity of its memory chips each time it starts.

PRIMARY START UP DEVICE **ROM** **DISK**

When you start your computer, it can run programs stored in memory or programs stored on a disk. Select DISK if you do not want to start with either the built-in DeskMate or the built-in MS-DOS.

INITIAL START-UP PROGRAM **MSDOS** **DESKMATE**

Select MS-DOS if you do not want your computer to run DeskMate automatically when it starts.

COMPUTER SPEED **FAST** **SLOW**

Select SLOW if your initial program was written for older computers and will not work properly with your computer's faster speed.

NUMBER OF DISK BUFFERS (2-17) **10**

Change this number if one or more of the programs you run require more than ten disk buffers. (The program documentation will tell you.) If more than one program requires additional disk buffers, set the value to the highest number requested.

MAXIMUM # OF OPEN FILES (8-23)

10

Change this number if one or more of the programs you run needs to open more than 10 files at one time (the program documentation will tell you). If more than one program requires additional open files, set the value to the highest number requested.

CHECK FOR CONFIG.SYS ON DRIVE

NO

Select the location of the Config.sys file that you want your computer to use when it starts. If you have not created a Config.sys file, or if the programs you run have not created a Config.sys file, leave the setting at NO. Your computer then uses the Config.sys file that resides in memory. Press → to change to A:. Press → again to change to C:. Your programs' manuals will tell you if they need a Config.sys file.

CHECK FOR AUTOEXEC.BAT ON DRIVE

NO

Select the location of the Autoexec.bat file that you want your computer to use when it starts. If you have not created an Autoexec.bat file, or if the programs you run have not created an Autoexec.bat file, leave the setting at NO. Your computer then uses the Autoexec.bat file that resides in memory. Press → to change to A:. Press → again to change to C:. Your programs' manuals will tell you if they need an Autoexec.bat file.

DISKETTE DRIVE A DESIGNATION

TOP

BOTTOM

Select whether you want the top or the bottom drive to be Drive A.

Troubleshooting

Video Problems

If you do not get an image on your monitor, be sure you have the cables connected properly. See *First Things First*.

If you are using a monochrome monitor and you get no image on your screen, you need to restart your computer using a special key sequence. Press and hold CTRL-ALT-SHIFT-V.

If you attempt to run some game software designed for an original Tandy 1000 (Cat. No. 25-1000) or an IBM PC Jr., and the program appears to stall, you may need to move the Video/Interrupt Jumper from E2-E3 to E1-E2. If you then experience a rolling screen, you may have another device (such as a hard disk card) trying to use the same interrupt (5). If possible, change the interrupt setting or the conflicting device.

Be sure and check to see if the brightness and contrast controls are properly adjusted.

Printer Problems

If your printer fails to operate properly, see *First Things First* for help. If you make all the proper printer connections and still have trouble, check that:

- The printer power light is on and the printer is getting electrical power.
- The printer cable is connected to the computer with the top of the cable up.
- The paper is loaded properly.
- The printer ribbon is inserted properly.
- The application program you are using is properly set up for use with your printer. Refer to your printer's manual.
- All of your printer switches are set properly. Refer to your printer's manual.
- The printer is ready, not off line, out of paper, out of ribbon, and so on.

Caring For Your Equipment

Generally, computers and accessories are quite durable and do not need special care. Everyday use, making mistakes while working, and minor bumps will not harm the unit. So, don't be afraid to explore and try things.

Following are some suggestions to help you keep your computer and equipment clean and running well.

- Protect them from spills and hard bumps.
- Use a diskette head cleaning kit to clean your diskette drives regularly.
- Install a surge protector between your equipment and its power supply.
- To clean your equipment, occasionally wipe the surface with a slightly damp cloth.
- Putting dust covers over your equipment when it is not in use can help prolong its life and reduce the need for repairs. This is especially true if the equipment is located in a dusty area.
- Use special anti-static cloths to clean display screens.

Tandy 1000 TL Specifications

Processor:	80286, 8 or 4 megahertz
Size:	Length 354 mm (13.9 in.) Width 432 mm (17 in.) Height 140 mm (5 1/4 in.) Weight 7.82 kg (17 1/4 lb)
Power Requirements:	United States: 120 VAC, 60 Hz International only: 120 VAC / 240 VAC, 50, 67 watts
Heat Output:	363 Btu/hr
Environment: Air Temperature:	Operating 14°C — 30°C (55°F — 85°F) Storage -40°C — 65°C (-40°F — 149°F)
Humidity:	Operating 20% to 80% (non-condensing) Storage 10% to 80% (non-condensing)
Internal Disk Drive:	Unformatted Capacity 1 megabyte Formatted Capacity 720 kilobytes Number of Heads 2 Number of Cylinders 80
Average Access Time:	93 ms (including settling time) Track to Track 4 ms
Motor Starting Time:	500 ms
Rotation Speed:	300 RPM
Media standard:	3 1/2-inch, double-sided, 80 track

MS-DOS

Quick Reference

MS-DOS Commands

This chapter contains an alphabetical reference to all of the MS-DOS commands. At the beginning of each command description is a syntax line that gives the format for entering the command. Each syntax line observes the following notations:

bold	in syntax lines, indicates information that you type exactly as it appears, such as command names, switches, and options.
screen type	indicates words and characters you type to perform a specified function.
<i>lowercase italics</i>	represent variable words, letters, characters, or values.
[] (square brackets)	indicate optional parameters.
... (ellipsis)	indicates that you can repeat a parameter, along with any applicable punctuation.
(vertical bar)	indicates an either/or situation.
SMALL CAPS BOLD	indicates a key combination, such as CTRL-Z. Press and hold the first key, and press the second while holding down the first.

The commands in this section are labeled as either internal or external

Internal	these commands load into your computer's memory on startup. Your computer does not need to access a disk to use them.
External	these commands are on disk. Your computer must be able to access the disk and directory where these commands reside before you can use them.

append [;] [pathname [;pathname]...] [/e]/[x]

(External) Sets a data file path, which tells MS-DOS the drives and directories in which to search for data files. Including a semi-colon (Append;) sets the NUL data path and causes MS-DOS to search only the current directory. Append with no parameters displays the current data path.

/e	causes Append to store appended directories in the MS-DOS environment.
/x	extends the search path to include all subdirectories within the specified directory.

```
append B:\sales\region1;a:
```

assign [drive1: = drive2:...]

(External) Reassigns the drive letter *drive1:* to *drive2:*. Use this command to run application programs from drives other than those for which they were written.

<i>drive1:</i>	the original target drive.
<i>drive 2:</i>	the substitute target drive.

```
assign a=c b=c
```

attrib [+r|-r] [+a|-a] pathname [/s]

(External) (1) Sets the read-only and archive attributes of the file specified by *pathname*, or (2) displays the attributes if you omit the optional parameters.

+r sets the read-only mode.
-r disables the read-only mode.
+a sets the archive attribute.
-a clears the archive attribute.
/s causes Attrib to apply to all subdirectories as well as the specified directory.

```
attrib +r b:\mydir\myfile.txt
attrib *.*
```

autofmt [/b]

(External) Initializes a hard disk for use. If the hard disk is Drive C, Autofmt installs the system files to make it the boot disk. If the hard disk is larger than 32K, Autofmt partitions it as necessary.

/b causes Autofmt to prompt you for bad sector information.

```
autofmt
```

backup [source pathname] [target drive:][/s] [/m] [/a] [/t] [/d:mm/dd/yy] [/t:hh:mmx] [/l:filename]

(External) Backs up one or more files from one disk to another formatted disk. Backup can copy between disks of different media, for example from hard disk drives to floppy disk drives. It can also copy from one diskette to another, even if the diskettes have a different number of sides and sectors.

source pathname specifies the files to back up. It can be an entire drive, a directory name, or a filename.

target drive: the drive to receive the files. If the target is a diskette drive, Backup places the files in the root directory. If it is a hard disk drive, Backup places the files in a subdirectory called Backup.

/s copies all files in the specified directory and in the directories below it.

/m copies only those files modified since the last backup.

/a adds the files to be backed up to those already on the target disk, instead of erasing the existing files.

/f formats the target disk if it is not already formatted.

/d:mm/dd/yy copies only those files created on or after the specified date.

/t:hh:mmx backs up only those files modified on or after the specified time. The time must be less than 13:00. x is either a (for a.m.) or p (for p.m.).

/l:filename creates a backup log entry in the specified file or—if you omit *filename*—in a file called Backup.log in the root directory of the files being backed up.

```
backup c:\store\sales.dat a: /a
```

break [on|off]

(Internal) Turns the CTRL-C check on or off. Displays the current setting of CTRL-C if you omit On and Off.

```
break off
```

cache [*buffer size*][*/a*][*/e*]**cache** [*drive:*][*/8*][*/9*][*/c*][*/g*][*/h*][*/o*][*/r*][*/s*][*/t:seconds*][*/w*]

(External) Establishes a RAM buffer for storing data coming from or going to a disk.

buffer size the size of the buffer in kilobytes. Default = 30K.

/a causes the buffer to reside in *expanded* RAM if you have an expanded memory board compatible with Lotus/Intel/Microsoft expanded memory specifications.

/e causes the buffer to reside in *extended* RAM if you have extended RAM installed.

After you install Cache, you can again issue the Cache command with the following options:

drive: the drive letter of the drive for which the following parameters are effective. Default = current drive.

/8 sets for a diskette formatted 8 sectors per track.

/9 sets for a diskette formatted 9 sectors per track.

/c clears the accumulated statistics. (See */s*.)

/g displays the current Cache status.

/h sets for a high-capacity diskette.

/o toggles Cache on and off.

/r resets all data in the buffer.

/s displays the Cache statistics.

/t:seconds sets the time in seconds that data stored in the buffer from diskette drives remains valid.

/w toggles disk saves to the buffer on and off. This is only valid for buffers smaller than 100K.

```
cache
cache a: /t:1 /w
```

chcp [*nnnn*]

(Internal) Displays or changes the current code page for the command processor.

nnnn the code page to display or select. Chcp accepts one of the two prepared system code pages. An error message appears if you select a code page that is not prepared. If you type the Chcp command without a code page, it displays the active code page and the prepared code page.

The following are valid code pages:

437	United States
850	Multilingual
860	Portuguese
863	French-Canadian
865	Nordic

```
chcp
chcp 863
```


chdir [pathname]

cd [pathname]

(Internal) Changes the current or home directory of the specified drive to the directory specified by *pathname*. Displays the pathname of your current directory if you omit *pathname*.

```
chdir \bin\user
chdir b:\user
```

chkdsk [pathname] [/f] [/v]

(External) Checks the MS-DOS disk in the current or specified drive for errors. You can redirect Chkdsk's output to a file by adding *>pathname2* to the end of the command.

pathname specifies either an entire drive or an individual file. If you specify a file, Chkdsk displays information about both the drive and the file.

/f fixes errors (if possible) and updates the disk. (Do not redirect the output from Chkdsk if you use /f).

/v displays messages and error details while Chkdsk is running.

```
chkdsk a:\sales\jrsales>b:\sales\jrerrr
```

cls

(Internal) Clears the screen.

```
cls
```

command [pathname] [device] [/e:size] [/p] [/c string]

(External) Starts a new command processor.

pathname specifies the drive and directories in which the command processor is to look for the Command.com file if it needs to reload the transient portion of the file into memory.

device specifies a different device for input and output. It can be:

aux to specify an auxiliary device, usually RS232 Serial Port 1.

com1 to specify RS232 Serial Port 1.

com2 to specify RS232 Serial Port 2.

con to specify the console (keyboard input, screen output).

/e:size specifies the environment size, in bytes. *Size* is in the range 128-32768. The default is 128.

/p tells the command processor not to exit to a higher level.

/c string tells the command processor first to execute the command or commands specified by *string*, then to return. The /c switch is valid only as the last parameter.

```
command /c chkdsk b:
```

comp [drive:][pathname1][drive:][pathname2]

(External) Compares the contents of two files or sets of files.

```
comp c:*.asm b:*.bak
```

copy source pathname [target pathname] [/a] [/b] [/v]

(Internal) Copies one or more files to the same directory as the source (giving them different filenames) or to another directory (giving them the same or different filenames). To leave the filename the same, omit the filename from *target pathname*. If you omit /a and /b, Copy uses /b.

- /a Source file: treats the file as an ASCII file (text or data file). Target file: adds an EOF character to the end of the file.
- /b Source file: treats the file as a binary file (program file). Target file: does not add an EOF character to the end of the file.
- /v verifies the sectors written to disk.

```
copy memos.txt /a b:corr.txt
```

copy target pathname + source pathname1 [+source pathname2...] [/a] [/b] [/v]

(Internal) Adds one or more files to the end of the existing file specified by *target pathname*. If you omit /a and /b, Copy uses /a.

- /a Source file: treats the file as an ASCII file (text or data file). Target file: adds an EOF character to the end of the file.
- /b Source file: treats the file as a binary file (program file). Target file: does not add an EOF character to the end of the file.
- /v verifies the sectors written to disk.

```
copy b:read.dat + write.dat + print.dat
```

copy source pathname1 [+source pathname2...]target pathname [/a] [/b] [/v]

(Internal) Combines any number of source files into a new file, specified by *target pathname*. If you omit /a and /b, Copy uses /a.

- /a Source file: treats the file as an ASCII file (text or data file). Target file: adds an EOF character to the end of the file.
- /b Source file: treats the file as a binary (program) file. Target file: does not add an EOF character to the end of the file.
- /v verifies the sectors written to disk.

```
copy b:memos.txt + b:letters.txt b:corr.txt
```

ctty device

(Internal) Changes the I/O device to the device specified.

device can be:

aux	to specify RS232 Serial Port 1
com1	to specify RS232 Serial Port 1
com2	to specify RS232 Serial Port 2
con	to specify the console

```
ctty aux
```

date [mm/dd/yyyy]

(Internal) Enters or changes the system date. Displays the current date if you omit the date parameter.

mm/dd/yyyy specifies the month, day, and year to set as the date.

```
date 11/15/1987
date 11/15/87
```

dc [/c]/[d]/[q][pathname]

(External) Compresses the specified file or returns it to its original state.

```
dc /c myfile
dc /d a:\region1\sales
```

/c compresses the specified file.
/d decompresses the specified file.
/q displays a message indicating whether or not the specified file is compressed.

del

(Internal) See Erase.

dir [pathname]/[p]/[w]

(Internal) Displays information about: (1) files in the current directory, (2) files in the directory specified by *pathname*, or (3) the one file specified by *pathname*.

```
dir b:
dir \user\*.bat /p
```

/p selects the "page" mode.
/w selects a wide display.

diskcomp [drive1:][drive2:]/[1]/[8]

(External) Compares the contents of two diskettes.

```
diskcomp a: b:
```

drive1: the drive containing the source diskette.
drive2: the drive containing the target diskette.
/1 compares only the first side of two double-sided diskettes. If you omit /1, Diskcomp compares both sides.
/8 compares only the first 8 sectors of each track. If you omit /8, Diskcomp automatically compares either 9 or 15 sectors, according to the format of the two diskettes.

diskcopy [source drive:][target drive:]

(External) Copies the contents of the diskette in the source drive to the diskette in the target drive. The target diskette must be of the same density as the source diskette. If the target is unformatted or is formatted differently than the source diskette, Diskcopy formats it with the same format as the source diskette. Use Diskcomp to verify the accuracy of the duplication.

```
diskcopy
diskcopy a: b:
```

diskopt

(External) Optimizes the file storage on the current disk by rearranging its files into a contiguous format. Do not use Diskopt with copy protected programs.

```
diskopt
```

disktype [drive:]

(External) Displays information about the size and capacity of the indicated disk. For a floppy diskette, Disktype displays the number of sides, tracks, and sectors per track. For a hard disk, it displays the number of heads, cylinders, and sectors per track. Omit *drive*: to display information about the current drive.

```
disktype c:
```

echo [on][off][message]

(Internal) Turns the batch Echo feature on or off, displays the specified message, or—if you omit all parameters—displays the current setting of Echo .

```
echo off
echo Insert disk
```

erase [pathname]**del [pathname]**

(Internal) Erases (deletes) one or more files from the current or specified directory. Omitting *pathname* erases all files in the specified directory.

```
erase \bin\user\jd\jd.txt
del b:\sales\joe
```

exit

(Internal) Exits the command processor and returns to a previous level, if one exists.

```
exit
```

fastopen [drive:[=nnn][...]]

(External) Decreases the amount of time needed to open frequently used hard disk files and directories. Each time you open a file or directory, Fastopen records its name and location. Then, if you reopen the file or directory, your system immediately knows where to find it.

nnn the number of files on the specified hard disk that you want Fastopen to track. *nnn* can be in the range 10-999. The default is 10.

```
fastopen c:=100
fastopen c:
```

fdisk

(External) Creates, changes, deletes, or displays hard disk partitions.

```
fdisk
```

find [/v]/[c]/[n] "string" [pathname...]

(External) Searches for the specified string of text in one or more files, specified by *pathname(s)*. Searches for *string* among the lines from the current console input device if you omit *pathname*.

```
find /n "mispell" *.txt
```

for %c in (set) do command (regular) for %%c in (set) do command (batch file)

(Internal) Executes the specified command for each item in the set.

```
for %f in (taxfile autofile homefile) do  
del %f
```

format [drive:]/[1]/[8]/[b]/[v]/[s]

(External) Prepares the disk in the specified drive. When formatting a hard disk, first use Hsct and Fdisk to initialize it.

```
format  
format b: /s/v
```

goto label

(Internal) Used in a batch file to transfer execution to the line following the line that contains *:label*.

```
:g  
rem looping...  
goto g
```

/v displays all lines that do not contain *string*.
/c displays only the number of lines in each file that contain *string*.
/n displays each line's relative line number in that file; do not use with /c.

set a list of items, separated by spaces, or one wild card item.
c can be any one-character variable except 0-9. If you include %c or %%c at the end of the command, MS-DOS sequentially substitutes each member of set in *command*. If you do not include it, MS-DOS executes *command* the appropriate number of times but does not substitute the members of set.

drive : the drive containing the disk to format.
/1 formats a diskette for single-sided use (The default is double-sided.)
/8 formats a diskette for 8 sectors per track.
/v prompts for a volume label.
/s copies the system files to the disk.
/b leaves space on the disk to install MS-DOS's system files.

label is a character string.

graftabl [xxx]

(External) Loads character definitions for ASCII characters 128-255. If you have a color or graphics adapter, this table lets you display foreign language characters when in graphics mode.

xxx a code page identification number.

```
graftabl
```

graphics *ptype* [/r]/[b]/[cr]/[lf]

(External) Enables you to reproduce a graphics screen in color on the Tandy CGP-220 printer or in shades of gray on other printers. To reproduce the screen, press **SHIFT-PRTS**.

ptype is one of these printer types:

- cgp220*. specifies the Tandy CGP-220.
- dmp110*. specifies the Tandy DMP-110.
- pcmode* specifies a Tandy printer with a dip switch set for the PC mode or other PC-compatible printers.
- tmode* specifies a Tandy printer with the dip switch set for the Tandy mode.
- standard* specifies any other Tandy printer.
- /r* prints black as black and white as white. The default is black as white and white as black. (Do not use with a CGP-220 printer.)
- /b* prints the background color as black. (Use only with a CGP-220.)
- /cr* causes the end-of-line character to be a carriage return.
- /lf* causes the end-of-line character to be a line feed only.

```
graphics standard /r
SHIFT-PRTS
```

hsect

(External) Formats track and sector information on a hard disk. The Hsect menu prompts for the drive you want to format.

```
hsect
```

if [not] *condition* command

(Internal) Allows conditional execution of commands in batch file processing.

Not executes the command only when the condition is false.

conditions are:

- error level number* executes the command only if the program previously executed by Command.com has an exit code of *number* or higher.
- string1 = string2* executes the command only if *string1* and *string2* are identical after parameter substitution.
- exist filename* executes the command only if *filename* exists.
- command* the command to execute only if *condition* is met.

```
if exist memo.txt goto g
```

join [drive:][pathname][d]

(External) Links the root directory of *drive:* to the *pathname* specified. Displays the current Join status if you omit all parameters.

```
join d: c:\memos
```

keybxx [/us]

(External) Replaces the current keyboard BIOS with an international keyboard program. (Requires international drivers.)

xx can be one of the following:

us	United States	keybus (default)
fr	France	keybfr
gr	Germany	keybgr
it	Italy	keybit
sp	Spain	keybsp
uk	United Kingdom	keybuk
po	Portugal	keybpo
sg	Swiss-Germany	keybsg
sf	Swiss-French	keybsf
dk	Denmark	keybdk
be	Belgium	keybbe
nl	Netherlands	keybnl
no	Norway	keybno
la	Latin America	keybla
sv	Sweden	keybsv
su	Finland	keybsu
cf	Canada-French	keybcf

Press **CTRL-ALT-F1** to return to the US keyboard.

```
keybgr
keybuk /us
```

/us converts character scan codes to US scan codes.

label [drive:][label]

(External) Creates, changes, or deletes a volume label. Omit the label to delete the existing label.

drive: the disk that has the label you want to modify. Be sure to include the colon in the *drive:* specification, and do not put a space between the drive and the label. If you omit *drive:*, Label uses the current drive.

label the new volume label.

```
label a:mydisk
```

lf

(External) Suppresses the line feed after a carriage return in printer output.

```
lf
```

mkdir pathname**md pathname**

(Internal) Creates a directory.

pathname tells MS-DOS the directory under which to create the new directory and specifies the name to give the new directory.

```
mkdir \user
md b:\letters
```

mode [display]

(External) Sets the video display mode.

display the color and line width in characters (40 or 80). *display* can be:

40	for 40-character display
80	for 80-character display
bw40	for monochrome 40-character display
bw80	for monochrome 80-character display
co40	for color 40-character display
co80	for color 80-character display
mono	for monochrome display adapter, 80-character

```
mode co80
```

mode lptn[:][chars][,lines][,p]

(External) Sets the printer parameters.

<i>n</i>	specifies the printer number (1, 2, or 3).
<i>chars</i>	specifies the characters per line (80 or 132).
<i>lines</i>	specifies vertical spacing (6 or 8 lines per inch).
<i>p</i>	specifies continuous retry if a printer timeout error occurs.

```
mode lpt1: 80
```

mode lfoff | lfon

(External) Turns printer linefeed off or on. Before using this command, you must load the Lpdvrv.sys device driver or execute the Lf command.

```
mode lf off
```

mode 200
mode 225 } Set number of scan lines on the display. (200 gives a rolling screen).

mode comnumber: [*baud*][*parity*][*databits*][*stopbits*][*p*]

(External) Sets RS232 communication parameters.

number the RS232 serial port, either 1 or 2.
baud can be 110, 150, 300, 600, 1200, 2400, 4800, 9600, or 1200/75. Setting the rate to 1200/75 initializes the international parallel/serial adapter.
parity can be n (no parity), o (odd parity), or e (even parity). The default is e.
databits can be either 7 or 8. The default is 7.
stopbits can be either 1 or 2. The default is 1.
p tells the printer driver to continuously try to output on timeouts errors.

mode com1:1200 N 8 1 p

mode speed

(External) Sets the CPU speed.

Speed can be:

slow to reduce your computer's operating speed.
fast to set your computer's speed to the default fast speed.
auto to set your computer to automatically change speeds as required by the software you are running.

mode auto

mode device codepage prepare = ((*cpages*)[*drive:[path]*] *filename*)
mode device codepage select=*yyy*
mode device codepage refresh
mode device codepage [/status]

(External) Sets display code pages for parallel printer or console screen devices.

device specifies the device to support code page switching. Valid device names are con, prn, lpt1, lpt2, and lpt3.
yyy specifies a code page. Valid code pages are 437, 850, 860, 863, and 865.
filename identifies the name of the code page information (.cpi) file MS-DOS should use to prepare a code page for the specified device.
cpages specifies a list of valid code pages. (See *yyy*.)
prepare tells MS-DOS to prepare code pages for a given device.
select specifies the code page you want to use with a device.
refresh reinstates code pages that are lost due to an error.
status displays the current code pages prepared and/or selected for a device.

mode con codepage

more

(External) Reads from standard input and displays one screen of information at a time, with the message --MORE-- at the bottom. Press the space bar to see the next screen.

nlstunc [[drive:][path][filename]]

(External) Loads country-specific information. (Requires international drivers.)

filename specifies the file containing country-specific information. The default value of *filename* is defined by the Country command in your Config.sys file. If there is no Country command in your Config.sys file, MS-DOS uses the Country.sys file in your root directory.

```
nlstunc newcdpg.sys
```

path [[;][pathname][;pathname...]]

(Internal) Sets a command path, which tells MS-DOS the directories or drive in which to search for external commands. The command Path; sets the path to "No PATH," which causes MS-DOS to search only the current directory. Displays the current path setting if you omit *pathname*.

pathname specifies a directory or an entire drive.

```
path \bin\user\joe;b:\bin\user\joe
```

pause [message]

(Internal) Suspends execution of the batch file.

message a message to be displayed when execution pauses.

```
pause Insert diskette.
```

print [pathname [/d:device][/b:size][/u:value][/m:value][/s:timeslice][/q:value][/c][/p]...][/t]

(External) Puts files in the print queue for background printing.

pathname specifies the file(s) to print.
/d:device specifies the print device. Lpt1 is the default.
/b:size sets the size (in bytes) of the internal buffer. The default is 512 bytes.
/u:value specifies the number of clock ticks Print will wait for a printer. Default = 1.
/m:value specifies the number of clock ticks (1-255) print can take to print a character.
/s:timeslice specifies the interval of time to be used by the MS-DOS scheduler for the Print command.
/q:value selects the number of files (4-32) allowed in the print queue. The default is 10.
/c deletes (cancels) from the print queue the file that immediately precedes and all files that follow /c in the command line.
/p turns on the print mode and adds the preceding and all following filenames to the print queue.
/t deletes (terminates) all files in the print queue. (Do not use /t with a pathname.)

```
print /t  
print temp1.tst /c temp2.tst /p temp3.tst
```

prompt [*text*]

(Internal) Changes the system prompt to *text*. Sets the prompt to the current drive specification if you omit *text*.

text a string of characters that generates the prompt. Precede special characters with a dollar sign. Special characters are:

Specify this:	To set this:
\$	\$
t	the current time
d	the current date
p	the current directory
v	the version number
n	the default drive
g	>
l	<
b	
-	a return-line feed
s	a space
e	an escape code

```
prompt $p$g
```

rcrypt *pathname1* [*pathname2*]

(External) Changes the format of a file so that the contents are meaningless without the encryption key. The encryption key is 0-8 characters that Rcrypt uses to alter the file contents.

pathname1 the pathname of the file to encrypt.
pathname2 the pathname of the file to receive the encrypted file. If you omit *pathname2*, Rcrypt uses the standard output device.

```
rcrypt myfile b:\secret.fil
```

recover [*drive* | *pathname*]

(External) Recovers the file (specified by *pathname*) that contains bad sectors, or recovers all files on a disk (specified by *drive*;) that contains bad sectors.

```
recover oldbook.txt  
recover B:
```

rem [*remark*]

(Internal) Includes the specified remark in a batch file.

```
rem This file is called Billfile.bat.
```

ren *pathname filename*

(Internal) Changes the name of the file specified by *pathname* to *filename*.

```
ren b:\sales\region1 region2
```

replace *source pathname* [*target pathname*][*/a*][*/d*][*/p*][*/r*] [*/s*][*/w*]

(External) Updates previous versions of files.

source pathname

the drive or directory that contains the replacement files. It can also be a single file or a wildcard filename.

target pathname

the drive or directory that contains the files you want to replace.

- /a* adds files that exist in the source directory, but not in the target directory, to the target directory. Do not use */a* with */d*.
- /d* replaces files in the target directory with source files only if the source files are newer than the corresponding target files. Do not use */d* with */a*.
- /p* prompts before replacing a target file or adding a source file.
- /r* replaces read-only files as well as unprotected files.
- /s* searches all subdirectories of the target directory while replacing matching files. Do not use */s* with */a*.
- /w* causes Replace to wait for you to press any key before it replaces files.

```
replace a:\phones.cli c:\ /s
```

restore *source drive: target pathname*
[*pathname*][*/s*][*/p*][*/b:date*][*/a:date*][*/e:time*]
[*/l:time*][*/m*] [*/n*]

(External) Restores one or more files previously backed up using the Backup command. You can restore files from one type of disk to another, such as from a diskette to a hard disk or from one type of diskette to another.

source drive:

the drive that contains the backed up files.

target pathname

specifies the directory to which you want to restore the files.

pathname

specifies the disk directories and/or file you want to restore.

- /s* restores the specified directory and its subdirectories.
- /p* prompts for permission before restoring hidden or read-only files and before restoring any files changed since the last backup.
- /b:date* restores only those files last modified on or before *date* (*mm/dd/yy*).
- /a:date* restores only those files last modified on or after *date* (*mm/dd/yy*).
- /e:time* restores only those files last modified at or before *time* (*hh:mm*).
- /l:time* restores only those files last modified at or after *time* (*hh:mm*).
- /m* restores only those files modified since the last backup.
- /n* restores only those files that no longer exist on the target drive.

```
restore a: c:\*.dat /n$
```

rmdir *pathname*
rd *pathname*

(Internal) Removes the subdirectory specified by *pathname* from the specified disk.

```
rmdir \bin\user\jim
```

select [[drive1:][drive2:][path]][yyy][xxx]

(External) Installs MS-DOS on a new diskette with the specified country information and keyboard layout. Select creates the necessary Config.sys and Autoexec.bat files on the new disk.

drive1: the source drive.
drive2: the target drive.
path: the directory name on the target drive in which to copy system files.
yyy: specifies the country code.
xxx: specifies the keyboard code.

```
select b: a: o49 gr
```

set [[string1]=[string2]]

(Internal) Sets *string1* equal to *string2* in the environment for use in later programs and batch files. Displays the Set values if you omit all parameters. Including the *string1* parameter without the *string2* parameter removes the *string1* name from the environment.

string1: any character string you want to replace.
string2: any replacement character string.

```
set drive=b:  
set pathname=c:\sales
```

setuptl

(External) Initializes the system configuration.

Option /a permits more setup.

```
setuptl
```

share [/f:space][/l:locks]

(External) Installs file sharing and locking for active networking.

/f:space: allocates file space (in bytes) to record file sharing information.
/l:locks: allocates the number of locks allowed.

```
share
```

shift

(Internal) Lets you use more than the usual ten replaceable parameters (%0-%9). Each parameter definition shifts up one place.

```
shift
```

sort [/r][/+n][<input pathname>][>output pathname]

(External) Reads input from the keyboard or the file specified by *input pathname*, sorts the data, and writes it to the screen or to the file specified by *output pathname*.

/r: reverses the sort (sorts from Z to A).
/+n: begins the sort at Column *n*. The default is Column 1.

```
sort /r <unsort.txt >sort.txt
```

subst [drive:][pathname][/d]

(External) Substitutes a virtual drive name for a pathname.

drive: the virtual drive name. The highest available drive letter is the one specified in Config.sys with the Lastdrive command. The default is Drive E.

pathname the pathname you want to replace.

/d deletes the association between a virtual drive and pathname.

```
subst d: b:\sales\region1
```

sys drive:

(External) Transfers the MS-DOS system files from the current disk to the disk in the specified drive.

```
sys b:
```

time [hh:mm[:ss[.cc]]]

(Internal) Displays or sets the time.

hh:mm:ss.cc specifies the time in hours, minutes, seconds, and hundredths of a second. Omit the time to display the time.

```
time 14:30
```

tree [drive:][/f]

(External) Displays all directories and sub-directories on the specified drive.

/f causes Tree to also display all files on the drive.

```
tree b: /f
```

type pathname

(Internal) Displays the contents of the specified file.

```
type b:testfile
```

ver

(Internal) Displays the version number of your MS-DOS operating system.

```
ver
```

verify [on][off]

(Internal) Enables or disables disk write verify. Displays the current Verify setting if you omit on and off.

```
verify on
```

vol [drive:]

(Internal) Displays the volume label of the disk in the current or specified drive.

```
vol b:
vol
```

xcopy source pathname [target pathname][/a][/d:mm/dd/yy][/e] [/m][/p][/s][/v][/w]

(External) Copies files and directories, including subdirectories. You can use Xcopy to back up between different drive or media types.

<i>source pathname</i>	specifies the drive, directories, and/or files you want to copy.
<i>target pathname</i>	specifies the drive, directories, and/or file you want to copy to. If you omit this parameter, Xcopy copies to the current directory. The default filename is *.*.
/a	copies only those files that have the archive bit set, without modifying the archive bit.
/d:mm/dd/yy	copies files modified on or after the specified date.
/e	copies any subdirectories, even if they are empty.
/m	copies only those files that have the archive bit set, and modifies the source files by turning off the archive bit.
/p	prompts you with Y/N? before copying each source file.
/s	copies directories and subdirectories, unless they are empty. If you omit /s, Xcopy works within a single directory.
/v	verifies each target file as it is written to be sure it is identical to the source file.
/w	causes Xcopy to wait before it copies the files. At the message, press any key to continue, or press CTRL-C to cancel Xcopy.

```
xcopy a: b: /s /e
```

Special Keys

→	Display Template. Displays the contents of the keyboard template (a storage place for the last line you typed) one character at a time. For more information on the template, see "Edlin" in the <i>Tandy 1000 MS-DOS Reference Manual</i> .
CTRL	Execute Special Commands. Lets you press only two or three keys to generate complex commands. Hold CTRL while pressing another key.
CTRL-C	Terminate Process. Stops the execution of a program if the program uses MS-DOS functions. Otherwise, this key sequence is not recognized. The computer might take a few moments before it recognizes the key sequence.
CTRL-H	Correct Typing Error. Moves the cursor left one character. Erases the character beneath the cursor.
CTRL-J	End Line. Causes the current line to end and performs a carriage return but does not cause the line to process. You can continue typing the command line.
PAUSE or CTRL-S	Stop Scroll. Halts scrolling and lets you view the display. Press the space bar to continue scrolling.
CTRL-P	Print Output. Causes all computer output to print on your printer if it is connected and ready. Press this sequence again to stop the function.
CTRL-ALT-DEL	Reset (reboot) your computer.
DEL	Erase Character. Removes the character under the cursor. If the cursor is at the first character of the line, this key does nothing.
ENTER	Process Command: Generate Carriage Return. Starts the processing a command line. ENTER also causes a carriage return; the cursor drops one line and returns to the left margin.
ESC	Terminate Line. Ends the current line but does not process it. ESC clears the line buffer, outputs a backslash, carriage return, and line feed but does not affect the keyboard template. The system prompt is not displayed, but the system is ready for a command.
F1	Display Character. Displays the next character in the keyboard template until you reach the end of the line.
F2 (char)	Copy to Character. Copies all characters up to the specified character (char) and displays them.
F3	Display Template. Redisplays the last typed command line. Execute the line again by pressing ENTER or edit the line before re-executing it.
F4 (char)	Delete to Character. Deletes all characters up to the specified character (char) from the keyboard template. They are skipped and not copied to the command line.
F5	Replace Template. Makes the line you type the new template but does not execute the command.
F6 or CTRL-Z	End-of-File. Puts an end-of-file character in the keyboard template.
INS	Insert Text. Lets you insert characters in a command line. To begin the insertion, press INS. Press INS again to end the insertion. Use the arrow keys to position the cursor before you begin inserting .

space bar	Add a Space. Moves the cursor one space to the right, and adds a space to a line.
PRINTSCRN	Print Screen. Causes the current screen display to print if your printer is connected and ready.

Edlin Commands

append lines

[*number*]**a**

Adds the specified number of lines from disk to memory. If you omit *number*, Edlin appends lines until available memory is 75% full.

100a

copy lines

[*line1*][*line2*][*line3*][*count*]**c**

Copies all lines in the range *line1* to *line2*, and places them immediately ahead of *line3* for the number of times specified by *count*.

3,9,12c
,20,35c

delete lines

[*line1*][*line2*]**d**

Deletes all lines in the range *line1* to *line2*. Deletes the current line if you omit *line1* and *line2*.

5,25d
4d
,4d

edit line

line

Displays the specified line for editing.

4

end edit

e

Ends the Edlin program and saves the edited file.

e ENTER

insert

[*line*]**i**

Inserts lines of text immediately before the specified line, or enters lines into a new file. Omit *line* or include a period to use the current line. Include a number sign (#) to append the lines to the end of the file.

3i
.i
#i

list`[line1][line2]l`Displays all lines in the range *line1* to *line2*.`2,5l
26l
,10l`**move lines**`[line1][line2][line3]m`Moves all lines in the range *line1* to *line2* to the line immediately preceding *line3*.`23,30,100m`**page**`[line1][line2]p`

Pages through a file 23 lines at a time, or lists the specified block of lines.

`10,15p
20p`**quit**`q`

Quits the editing session without saving the file.

`q`**replace string**`[line1][,line2][?]r[string1] CTRL-Z [string2]`Replaces all occurrences of *string1* with *string2* in the lines between *line1* and *line2*. The question mark (?) causes Edlin to let you verify each modification.`2,7?rhi CTRL-Z bye`**search text**`[line1][,line2][?]s[string]`Searches all lines in the range *line1* to *line2* for each occurrence of the text string. The question mark (?) prompt appears at each occurrence of string.`1,10Sand`**transfer lines**`[line]t[drive:]filename`Inserts the contents of the file specified by *filename* immediately ahead of the specified line or the current line in the file being edited.`10tb:myfile`**write lines**`[number]w`

Writes a specified number of edited lines from memory to disk, beginning with Line 1. If you omit number, Edlin writes until 25% of memory is freed.

`100w`

Edlin Editing Keys

Function	Key(s)	Description
Copy <i>char</i>	→	Copies one character to the new line.
Copy to <i>char</i>	F2 (<i>char</i>)	Copies all characters up to the specified character to the new line.
Copy all	F3	Copies all remaining characters in the template to the new line.
Delete <i>char</i>	DEL	Deletes a character in the template.
Delete to <i>char</i>	F4 (<i>char</i>)	Deletes a number of characters up to, but not including, the specified character (<i>char</i>).
Void line	ESC	Void or terminates the current line entry.
Insert	INS	Enters or exits the insert mode.
Replace template	F5	Replaces the template with the characters displayed to allow further editing.
Void line	F6	Cancels any changes made to the current line. The template is unchanged and you can continue making changes.
Enter line	ENTER	Makes the new line the new template and sends it to the requesting program.

You can use the following symbols in Edlin:

.	(period)	to indicate the current line.
#	..	to indicate the end of a program.
+	.	to reference lines after the current line.
-	.	to reference lines before the current line.

Debug Command Parameters

The following parameters are available for the debug commands listed in the next section.

address an alphabetic segment register and offset

cs:0100	A segment address and offset.
04ba:0100	An offset only. (The default segment is cs for the g, l, t, u, and w commands, ds for all other commands.)

byte

A 2-digit character hexadecimal value placed in or read from an address or a register.

drive

A 1-digit value for the drive to be used for accessing or writing data, as follows:

0 = Drive A 2 = Drive C
1 = Drive B 3 = Drive D

pathname

A drive specification, filename, and extension. (The complete pathname is optional; however, you must specify at least the drive or filename.)

list

A series of strings or byte values.

portaddress

A hexadecimal value (four characters maximum) that specifies a port number.

range

An area of memory specified by:

address1 address2 (address2 must be an offset.) address1 value (value = the number of bytes to operate on. The default is 80.) Do not use the address1 value format if another hexadecimal value follows the range parameter.

registername

One of the following registers:

AX	CS	BP	DI
BX	DS	IP	SI
CX	ES	SP	PC
DX	SS	F	

sector

A hexadecimal value (1-3 characters), indicating the relative sector number on the disk.

sectorcount

A hexadecimal value (1-3 characters), indicating the number of disk sectors to write or load.

string

Any number of characters, enclosed in quotation marks (" or ').

value

A hexadecimal value (four characters maximum).

Debug Commands

assemble

a [address] Assembles statements directly into memory, starting at address.

a cs:0100

compare

c range address Compares the portion of memory specified by range to a portion of the same size beginning at the specified address, and displays all differences.

c 100,1ff 300

c 100 1 100 300

dump

d [address]
d [range] Displays the contents of the specified memory address or range.

d cs:100 109

enter

e address [list] Enters byte values into memory at the specified address; replaces the contents of memory, beginning at address, with the list of values.

e ds:100 45 a1 "abc" 0f

e cs:1004

fill

f range list Fills the memory locations in the specified range with the values in the list.

f 04ba:100 1 100 42 45 52 54 41

go

g [=address1][address2...]

Executes the program currently in memory, beginning at address1 and stopping at each breakpoint (specified by address2...).

g cs:7550

hex

h value1 value2

Displays the results of value1+value2 and value1-value2 (hexadecimal arithmetic).

h 19f 10a

input

i portaddress

Inputs and displays one item from the specified port.

i 2f8

load

l [address[drive: sector sectorcount]]

Loads the specified file from the drive: (0-3) into memory, beginning at the specified address. Loads absolute sectors from the drive:, beginning at sector and continuing until the number of sectors specified by sectorcount is loaded.

l 04ba:100 2 0f 6d

move

m range address

Moves the block of memory specified by range to the location beginning at address.

m cs:100 110 cs:500

name

n pathname1 [pathname2...]

Assigns program names for later load or write commands and assigns pathname parameters for the file being debugged.

n file1.exe
n file2.dat file3.dat

output

o portaddress byte

Sends the specified byte to the specified portaddress.

o 2fb 4f

proceed

p [=address][value]

Beginning at address, proceed executes the number of instructions specified by value. The purpose of this is to execute all the instructions associated with call, int, or loop—or to repeat string instructions—and then stop execution at the next instruction. After it executes each instruction, proceed displays the register contents, flags, and next instruction. If you omit all parameters, proceed causes the execution of the instruction pointed to by CS:IP.

p
p=011a 10

quit

q

Quits debug without saving the file.

q

register

r [registername]

Displays the contents of all registers and flags, or displays the one register or the flags specified by registername, and lets you change the setting.

r
rax
rf

search

s range list

Searches the location in the specified range for the list of bytes.

s cs:100 110 41

trace

t [=address][value]

Executes one or more instructions (specified by value), beginning at address. The trace command displays the register contents, flags, and the next instruction after each instruction executes.

t
t=011a 10

unassemble

u [address]
u [range]

Disassembles instructions beginning at the specified address (or for the specified range), and displays their addresses, their hexadecimal values, and the source statements that correspond to them.

u 04ba:100 1 10

write

w [address[drive: sector sectorcount]]

Writes the data being debugged to a disk file on the specified drive (0-3), beginning at the specified address. Writes absolute sectors to the specified drive, beginning at sector, and continuing until the number of sectors specified by sectorcount is written.

w cs:100 1 37 2b

GW-BASIC

Quick Reference

Loading GW-BASIC

Use the following syntax to load GW-BASIC
at the MS-DOS system prompt:

basic | **basica** [*pathname*] [*<input file*][*>*][*>*]*output-file*] [*/f*:*max. # of files open at same time*]
[*/m*:*highest memory location for BASIC, max. block size*][*/c*:*RS-232 receive buffer size*]
[*/s*:*max. record length for direct access files*][*/d*][*/i*]

<i>pathname</i>	loads and executes the specified GW-BASIC program file.
<i><input file</i>	inputs data from the specified file instead of from the keyboard.
<i>output file</i>	outputs data to the specified file instead of to the video display. Use <i>></i> to overwrite the existing output file or <i>>></i> to append to it.
<i>/f:files</i>	specifies the maximum number of data files GW-BASIC can open at one time, including the files it reserves for internal use. The maximum value you can specify for files is 15. The default is 3. Use a files command in your Con-fig.sys file if you want a number other than the default. You must use the <i>/f</i> option when using the <i>/f</i> parameter.
<i>/m:memory location, block size</i>	loads GW-BASIC with the reserved memory specified by block size (block size x 16). GW-BASIC uses memory up to <i>memory location</i> , and memory above is reserved for machine language routines. The default is 64K bytes for GW-BASIC (4096 blocks of 16 bytes each).
<i>/c:buffer size</i>	sets the size of the RS232 receive buffer. The default is 256 bytes. (The RS232 transmit buffer is always set to 128 bytes.)
<i>/s:max. record length for direct access files</i>	sets the maximum of any record in a file created for direct access. The default is 128 bytes.
<i>/d</i>	loads the double-precision transcendental math package.
<i>/i</i>	causes GW-BASIC not to dynamically allocate space during file operations. This switch is always invoked if you use BASICA .

GW-BASIC Commands and Statements

This chapter contains an alphabetical reference to all GW-BASIC statements and functions. At the beginning of each statement or function description is a syntax line that gives the format for typing the statement or function. Each syntax line observes the following notations.

Notations

bold

specifies a command, statement, or parameter that you must type exactly as it appears.

lowercase italics

represent variable names, letters, characters, or values. You supply the names, letters, characters, or values.

[]

indicates that you can use optional parameters. Include these optional parameters if you want the functions they provide.

... (ellipsis)

shows that you can repeat a parameter.

SMALL CAPS BOLD

indicates a key combination, such as **CTRL-C**. Press and hold the first key, and press the second while holding down the first.

abs(number)

Computes the absolute value of *number*.

```
print abs(-44)
x=abs(y)
```

asc(string)

Returns the ASCII code (a decimal number) for the first character of *string*.

```
print asc("a")
n=asc(b$)
```

atn(number)

Computes the arctangent of *number* in radians.

```
print atn(7)           x=atn(y/3)*57.29578
```

auto [*line*][*increment*]

Automatically generates a line number when you press **ENTER**. If *line* exists in memory, GW-BASIC displays an asterisk after the number. To turn off **auto**, press **BREAK**.

<i>line</i>	is the starting line number. Default = Line 10.
<i>increment</i>	is the increment to use when generating line numbers. Default = 10.

```
auto
auto 100,50
```

beep

Produces a sound at 800 Hz for 1/4 second.

```
beep
```

bload *pathname* [,*offset*]

Loads a memory image file, specified by *pathname*, into memory.

<i>offset</i>	is the location (the number of bytes from the beginning of the current segment) where GW-BASIC loads the image. <i>offset</i> must be in the range 0-65535. Default = the value set by bsave .
---------------	--

```
bload "prog1.txt"
bload "prog2.txt",0
```

bsave *pathname*,*offset*,*length*

Saves the contents of an area of memory into a disk file (*pathname*).

<i>offset</i>	is the location (the number of bytes from the beginning of the current segment) where GW-BASIC starts saving. It must be in the range 0-65535.
<i>length</i>	is the number of bytes to save (in the range 1-65535).

```
bsave "prog1.sav",0,500
```

call *variable* [(*parameter list*)]

Transfers program control to an assembly-language subroutine stored at *variable*.

<i>parameter list</i>	contains the variables that are passed to the external subroutine.
-----------------------	--

```
call c
call c (a$,z,x)
```

cdbl(*number*)

Converts *number* to double precision.

```
print cdbl(465.342)
z=cdbl(a)
```

chain [merge]pathname[,line][,all][,delete line-line]

Lets the current program load and execute another program (specified by *pathname*). You must have first saved *pathname* in ASCII format. Commas in the syntax line are significant and must be entered even if you omit the option.

line is the line number at which execution begins in the chained program. Default = the first line of the chained program.

all passes every variable in the current program to the chained program. If you omit *all*, the current program must contain a **common** statement to pass variables to the chained program.

merge overlays the lines of the chained program with the current program.

delete deletes lines in the overlay that you can merge in a new overlay.

```
chain "prog2"
chain "subprog.bas",,all
```

chdir pathname

Changes the directory to the directory specified by *pathname*.

```
chdir "b:\accts\recvble"
chdir ".."
```

chr\$(code)

Returns the character corresponding to any ASCII or control code.

```
print chr$(35) c$=chr$(32)
```

cint(number)

Converts *number* to an integer representation by rounding the fraction portion of the number.

number must be in the range -32768 to 32767.

```
print cint(1.6)
z=cint(-1.67)
```

circle [step] (x,y),radius[,color[,start[,end[,aspect]]]]

Graphics. Draws an ellipse, the center of which is (x, y), on the screen.

step designates (x, y) as relative coordinates.

radius is the major axis of the ellipse.

color designates the color of the ellipse. *Color* must be a valid number in the current color set.

start and end are the beginning and ending angles, in radians (in the range -6.283186 to 6.283186).

aspect is the ratio of the x-radius coordinate to the y-radius coordinate. See "Text and Graphics Modes" for the range for *color*.

```
circle (150,100),50
```

clear [,memory location] [,stack space]

Frees memory for data without erasing the program in memory.

memory location

specifies the highest memory location available for GW-BASIC.

stack space

specifies the memory to set aside for temporarily storing internal data and addresses.

```
clear
clear ,45000
clear 6100,300
```

close [buffer,...]

Closes access to a disk file or communications channel.

buffer

buffer number of file to close. Default = Close all files.

```
close
close 1,2,8
```

cls

Clears the screen (or active viewport) and returns the cursor to the home position (the upper left corner of the screen or viewport).

```
cls
```

color [background] [,palette]

Graphics. Selects the background color and the palette for Screen Mode 1.

background

specifies the background color.

palette

specifies the palette to use for the foreground colors.

```
color 0,7
color 0,1
```

color [foreground] [[,background] [,border]]

Text Mode Only. Selects the display colors for the foreground, background, and border for Screen Mode 0. See "Text and Graphics Modes."

foreground

specifies the color of the foreground characters. Foreground can be colors 0 to 15 of the Color Set. Specify color +16 to get blinking characters.

background

specifies the background color and can be colors 0 to 15 of the 16 Color Set.

border

specifies the border color. Border can be 0 to 15.

```
color 0,7
color 1,0
```

color [*foreground*][,*background*][,*border*]

Selects the foreground and background palette color numbers for Screen Modes 7, 8, 9, and 10.

foreground In Screen Modes 7, 8, and 9, *foreground* is a palette color integer in the range 0-15 (or 0-63 with an EGA adapter with more than 64K). In Screen Mode 10, *foreground* is a palette color integer in the range 1-3.

background In Screen Modes 7, 8, and 9, *background* is an integer in the range 0-15 (or 0-63 with an EGA adapter with more than 64K). In Screen Mode 10, *background* is an integer in the range 0-8. Possible background colors are:

- 0 Off
- 1 Blinking off to on
- 2 Blinking off to high intensity
- 3 Blinking on to off
- 4 On
- 5 Blinking on to high-intensity
- 6 Blinking high-intensity to off
- 7 Blinking high-intensity to on
- 8 High-intensity

```
color 7,0
```

com(*channel*) *action*

Turns on, turns off, or temporarily halts the trapping of communications channel activity.

channel can be 1 or 2.
action can be on, off, or stop.

```
com(1)on  
com(2)stop
```

common *variable* [,*variable*...]

Reserves space for variables so you can pass them to a chained program. Be sure both programs in the chain contain **common** statements.

```
common a, b$, d()
```

cont

Resumes execution when either CTRL-BREAK or the execution of a **stop** or an **end** statement halts a program.

```
cont
```

cos(*number*)

Computes the cosine of *number*.

```
print cos(5.8)  
y=cos(x*.00174533)
```

csng(number)

Converts *number* to single precision. GW-BASIC rounds the number when converting it to single-precision.

```
print csng(.145388509)
z=csng(a#)
```

csrln

Returns the current row position of the cursor.

```
print csrln
a=csrln
```

cvd(8-byte string) cvi(2-byte string) cvs(4-byte string)

Converts a string to a double-precision (**cvd**), an integer (**cvi**), or a single-precision (**cvs**) number. Use these functions to restore data to numeric form after reading from the disk.

```
a# = cvd(grosspay$)
d = cvs(total$)
```

data constant [,constant...]

Stores numeric and string constants to be accessed by a READ statement. Enclose string constants containing delimiters (leading or trailing blanks, colons, commas) in quotation marks.

```
data Edmonton, New York, Chicago, Brooks
```

date\$[=string]

Sets the date to the specified string, or retrieves the current date if you omit *string*.

```
date$="04/17/85"
today$=date$
```

defdbl letter[,letter...]defint letter[,letter...]defsng letter[,letter...]defstr letter[,letter...]

Defines variables as double precision (**dbl**), integer (**int**), single precision (**sng**), or string (**str**).

```
defdbl a
defint a-g
defstr h-m
```

def ffname [(argument list)] = expression

Defines *name* as a function according to the expression.

<i>name</i>	is a valid variable name.
<i>argument list</i>	is a list of dummy variables used in <i>expression</i> .
<i>expression</i>	defines the operation to perform.

```
def fnr = rnd(1)*89+10
```

def seg[=address]

Assigns the specified address to be the current segment address.

<i>address</i>	is in the range 0-65535. Default = GW-BASIC's Data Segment.
----------------	---

```
def seg
def seg = &H8800
```

def usr[number]=offset

Defines the user number and segment offset of a subroutine to be called by the **usr** function.

<i>number</i>	is in the range 0-9. Default = 0.
<i>offset</i>	is in the range 0-65535.

```
def usr = 0
def usr3 = &h0020
```

delete line1 [-line2]

Deletes *line1* through *line2* of the program in memory.

<i>line1</i>	defaults to the first line of the program.
<i>line2</i>	defaults to the last line of the program.

```
delete 70
delete -110
```

dim array (dimension)[,array(dimension)...]

Sets aside storage for arrays with the dimensions you specify.

```
dim ar(100)
dim ll%(8,25)
```


BASIC Quick Reference

draw string

Graphics. Draws an image on the screen.

string specifies one or more of the movement commands. Prefix commands can precede the movement commands.

Movement Commands

<i>u[n]</i>	Moves up <i>n</i> points.
<i>d[n]</i>	Moves down <i>n</i> points.
<i>l[n]</i>	Moves left <i>n</i> points.
<i>r[n]</i>	Moves right <i>n</i> points.
<i>e[n]</i>	Moves diagonally up and right <i>n</i> points.
<i>f[n]</i>	Moves diagonally down and right <i>n</i> points.
<i>g[n]</i>	Moves diagonally down and left <i>n</i> points.
<i>h[n]</i>	Moves diagonally up and left <i>n</i> points.
<i>mx,y</i>	Moves to point <i>x,y</i> . If you precede <i>x</i> with a plus (+) or minus (-) sign, draw assumes it is a relative position. Otherwise, it is an absolute position.

Prefix Commands

<i>b</i>	Plots no points after the move.
<i>n</i>	Returns to the original position when the move is complete.
<i>aangle</i>	Sets the angle of the move.
<i>ccolor</i>	Sets the color.
<i>pcolor,border</i>	Paints, using <i>color</i> and <i>border</i> .
<i>sfactor</i>	Sets the scale factor.
<i>tangle</i>	Moves at the specified angle.
<i>xvariable</i>	Executes a substring.

See "Text and Graphics Modes."

```
draw"u30;"+"d30;"+"l40;"+"r40;"
```

edit line

Enter the Edit mode. GW-BASIC displays *line* for editing.

```
edit 100  
edit .
```

end

Ends program execution, and closes all files.

```
end
```

environ "parameter id = text"
[;"parameter id = text"...]

Lets you modify GW-BASIC's Environment String Table to change the **path** parameter for a child process or to pass parameters to a child process.

```
environ "PATH=A:\"
environ "SALES=MYSALES"
environ "PATH=A:\";"SALES=MYSALES"
```

environ\$ ["parameter id"][(number)]

Returns the specified environment string from GW-BASIC's Environment String Table. *number* and *parameter id* are mutually exclusive. You can specify only on the command line.

```
print environ$("PATH")
print environ$(3)
```

eof(buffer)

Function. Detects the end of a file.

```
if eof(1) then goto 1540
```

eof(buffer)

Communications. Detects an empty input queue for a communications file.

```
if eof(1) then return
```

erase array [,array...]

Erases one or more arrays from memory. Lets you either redimension arrays or use their previously allocated space in memory for other purposes.

```
erase c
erase g,h,i,z$
```

parameter id

is the name of the parameter. You must enclose it in quotation marks and type it in uppercase characters.

text

is the new parameter text. You must use an equal (=) sign to separate it from *parameter id*. If you omit *text*, or specify a null string or a semicolon, GW-BASIC removes the parameter from the Environment String Table and compresses the table.

parameter id

is the parameter for which to search. It must be uppercase and enclosed in quotation marks.

number

specifies which parameter to return by its position within the table.

buffer

is the number assigned to the file when you opened it. In sequential access files, **eof** returns 0 (false) if end-of-file has not been read or -1 (true) if it has. In direct access files, **eof** returns -1 (true) if the last executed **get** statement was unable to read an entire record because of an attempt to read beyond the physical end of the file.

buffer

is the number assigned to the file when you opened it. In ASCII mode, **eof** returns -1 (true) if CTRL-Z is received. In binary mode, **eof** returns -1 (true) when the input queue is empty.

erdev

Returns the value of a device error within MS-DOS as set by the Interrupt 24 handler. The lower 8 bits of **erdev** contain the Interrupt 24 error code.

```
erdev
```

erdev\$

Returns the name of the device (as set by the Interrupt 24 handler) when a device error occurs. A character device error returns a 8-byte device name. A block device error returns a 2-byte device name.

```
erdev$
```

erl

Returns the number of the line in which an error has occurred. If no error has occurred, **erl** returns 0. If the error occurs while you are entering something at the prompt, **erl** returns 65535 (the largest number than can be represented in two bytes).

```
print erl  
e=erl
```

err

Returns the error code if an error has occurred.

```
if err = 7 then 1000 else 2000
```

error code

Simulates a specified error.

code is one of GW-BASIC's error codes.

```
error 1
```

exp(number)

Returns the natural exponent of *number*, that is e (base of natural logarithms) to the power of *number*.

number must be less than or equal to 88.02968.

```
print exp (-2)  
a=exp(-6)
```

exterr (number)

Returns extended error information.

number is an integer in the range 0-3 as follows:

- 0 = Extended error code
- 1 = Extended error class
- 2 = Extended error suggested action
- 3 = Extended error location

```
exterr(0)
print exterr(3)
```

field buffer, length as variable [,length as variable...]

Divides a direct access buffer into fields so you can send data from memory to disk and from disk to memory. You identify each field by a string variable and you specify its length.

length must be an integer in the range 1-255.

```
field 3, 128 as a$, 128 as b$
```

files[pathname]

Displays the names of the files and directories on a disk.

If you specify *pathname*, GW-BASIC lists all files that match *pathname*. If you omit the filename from *pathname*, GW-BASIC lists all files and directories in the specified directory.

```
files
files"\books\"
```

fix(number)

Returns the truncated integer of *number*.

```
print fix(2.6)
z=fix(b)
```

for variable = initial value to final value [step increment] next [variable]

Establishes a program loop that allows a series of program statements to execute a specified number of times.

increment is the number added to *initial value* each time the loop executes. Default=1

variable must be either integer or single precision.

```
for x =1 to 5 : print x : next
```

fre(dummy argument)

Returns the number of bytes in memory not being used by GW-BASIC. Specify a string argument if you want GW-BASIC to compress the data before returning the amount of memory available.

```
print fre(44)
print fre("44")
```

get[#]buffer[,record]

Reads a record from a direct access disk file, and places it in the specified buffer.

record is an integer in the range 0-16,777,215. If you omit *record*, GW-BASIC gets the next sequential record.

```
get 1
get 1,25
```

get[#]buffer,number

Communications. Transfers data from the communications line to the communications buffer.

number specifies the number of bytes to transfer.

```
get 1,8
```

get(x1,y1)-(x2,y2),array

Graphics. Transfers points from an area on the display to an array.

(x1,y1) and (x2,y2) are the coordinates at which the image begins and ends.
array is a numeric array to hold the image.

```
get (0,0) - (100,100),z
```

gosub line

Branches to the subroutine, beginning at *line*. Every subroutine must end with a **return** statement.

line is the starting line number in your program of the sub-routines.

```
gosub 1000
```

goto line

Branches to the specified line.

line is the line number in your program to branch to.

```
goto 100
if r = 14 then goto 80
```

hex\$(number)

Calculates the hexadecimal value of *number*.

number specifies the decimal value of the range -32768 to +65535 to convert into a hexadecimal string.

```
print hex$(30)
y$=hex$(x/16)
```

if expression(s) then statement(s) [else statement(s)]

Tests a conditional *expression*, and makes a decision regarding program flow. If *expression* is true, GW-BASIC executes the **then** statement. If *expression* is false, GW-BASIC executes the matching **else** statement or the next program line.

expression is any numeric or string expression, usually making logical or relational comparisons.
statement(s) can be one or more valid BASIC statements.

```
if a=b then print "a=b" else print "ab"
```

inkey\$

Returns a 0-, 1-, or 2-byte string from the keyboard without requiring that you press ENTER. **inkey\$** does not echo the character you press to the display.

0-byte string = no key press.

1-byte string = the ASCII value of the key.

2-byte string = 00 in Byte 1, extended ASCII value of the key in Byte 2.

```
10 a$ = inkey$:if a$ = "" then 10
```

inp(port)

Returns the byte read from *port*.

port can be any integer in the range 0-65535.

```
a = inp(255)
```

input[;]["prompt";]variable[,variable...]

Inputs data from the keyboard into one or more variables. GW-BASIC stops execution, displays the message specified by *prompt*, followed by a question mark, and then waits for input. If you do not want GW-BASIC to display the prompt, type a comma instead of a semicolon after *prompt*.

If you immediately follow **input** with a semicolon, GW-BASIC doesn't echo ENTER when you press it as part of a response.

```
input y%
input "Enter your name and age";n$, a
```

input # buffer, variable[,variable...]

Inputs data from a sequential device or file, and stores it in a program variable.

buffer is the number BASIC assigned to the file when opening it.

```
input #1,a,b
input #4,a$,b$,c$
```

input\$(number[,[#]buffer])

Inputs a string of characters from either the keyboard or a sequential access file.

number specifies the number of characters to input. It can be in the range 1-255.

buffer tells GW-BASIC to input the string from a sequential access file. If you omit *buffer*, GW-BASIC inputs the string from the keyboard.

```
a$ = input$(5)
a$ = input$(11, 3)
```

instr([number,]string1,string2)

Searches for the first occurrence of *string2* in *string1*, and returns the position at which the match is found.

number specifies the position in *string1* at which to begin searching. Default = the first character in *string1*.

```
instr(3, "1232123", "12")
a$ = "Lincoln":instr(a$, "inc")
```

int(*number*)

Converts *number* to the largest integer that is less than or equal to *number*.

number is not limited to the integer range.

```
print int(79.89)
print int (-12.11)
```

ioctl [#]*buffer*,*string*

Sends a control data string to a device driver.

string is a string expression containing a series of commands. The commands are separated by semicolons. *string* can be a maximum of 255 bytes.

```
ioctl #1,"p156"
```

ioctl\$([#]*buffer*)

Returns the control data string from a device driver that you have previously opened.

buffer is the number assigned to the driver when you opened it. The number sign (#) is optional.

```
if ioctl$(1) = "nr" then print "Printer
not ready"
```

key *number*,*string*

Assigns function key values.

number specifies a function key (F1-F10) or a user key (15-20). See **key (*number*) action**.

string is the string expression assigned to the key. It can contain a maximum of 15 characters.

```
key 1,"John Seymore #3"
```

key on

Displays the function key assignment values on Line 25 of the screen.

key off

Erases the soft key assignments from Line 25 of the display screen.

key list

Displays all 15 characters of all ten soft key assignments on the screen.

key(number) action

Turns on, turns off, or temporarily halts key trapping for a specified key. GW-BASIC numbers the cursor direction keys 11-14, and the user-defined keys 15-20.

<i>action</i>	can be on, off, or stop.
<i>number</i>	can be a number in the range 1-20, indicating the number of the key to trap. Function keys use their corresponding function key number (1-10).

Use the following syntax to define your own user keys:

```
key number,CHR$(key) + CHR$(scan)
```

kill pathname

Kills (deletes) *pathname* from disk.

```
kill "file.bas"  
kill "a:\report\data"
```

lcopy

Copies all text data on the screen to the printer.

```
lcopy
```

left\$(string,number)

Returns the specified number of characters from the left portion of *string*.

number is an integer in the range 1-255.

```
print left$("battleships",6)
```

len(string)

Returns the number of characters (including blanks) in *string*.

```
x = len(sentence$)  
print len("dog") + len("terrier")
```

[let]variable=expression

Assigns the value of *expression* to *variable*.

```
let a$ = "a rose is a rose"  
b1 = 1.23
```


line [(x1,y1)]-[step](x2,y2),[color][,b [f]][,style]

Graphics. Draws a line or a box on the video display.

- (x1, y1) is the point at which the line begins. Default = last specified screen point.
- (x2, y2) is the point at which the line ends.
- color specifies the color of the line. See "Text and Graphics Modes."
- b, f causes GW-BASIC to draw a box. The points that you specify are opposite corners. If you specify both the b and f options, GW-BASIC draws a box and fills the box in with color.
- style is a 16-bit integer that lets you select the line-style used when drawing normal lines and unfilled boxes. If bit = 1, GW-BASIC draws the point. If bit = 0, it does not.

```
line (0,0)-(319,199)
line -(319,199)
```

line input[:][*"prompt"*];string variable

Inputs an entire line (up to 254 characters) from the keyboard including delimiters (commas, quotation marks, etc). You terminate string input by pressing ENTER. However, if **line input** is immediately followed by a semi-colon, pressing ENTER does not echo a carriage return to the display.

- prompt* is a string literal that is displayed on the screen.
- string variable* accepts all input characters from the keyboard.

```
line input a$
line input "Last name, first name?";n$
```

line input# *buffer*, *variable*

Inputs an entire line of data from a sequential file including delimiters (commas, quotation marks, etc).

- buffer* is the number assigned to the file when it was opened.
- variable* is the string variable name in which you want the data stored.

```
line input#1, a$
```

list [*startline*][*-[endline]*][*,device:*]

Lists a program in memory to the display or specified device.

- startline* specifies the first line to edit. Default = first program line.
- endline* specifies the last line to list. Default=last program line.
- device:* can be either scrn: (the screen) or lpt2: (printer). Default=scrn:.

```
list
list 50-100,"lpt1:"
```

llist [*startline*][*-[endline]*]

Lists program lines in memory to the printer. **llist** assumes a132-character-wide printer. (You can change this using the **width** statement.) For descriptions of *startline* and *endline*, see the **list** statement.

- startline* specifies the first line to be listed.
- endline* specifies the last line to be listed.

```
llist          llist 68-90
```

load *pathname* [,*r*]

Loads a GW-BASIC program from disk into memory. The *r* option tells GW-BASIC to run the program.

```
load "a:prog1.bas"
load "prog1.bas",r
```

pathname is the filename specification of the file you want to load.

loc(*buffer*)

Returns the current record position within a file.

In direct access files, **loc** returns the record number accessed by the last **get** or **put** statement. In sequential access files, **loc** returns the number of 128-byte records that you have read or written.

```
a=loc(2)
if loc(1)=55 then end
```

loc(*buffer*)

Communications. Returns the number of characters in the input queue. If more than 255 characters are in the input queue, **loc** returns 255. If there are fewer, **loc** returns the actual number of characters waiting to be read.

buffer is the number assigned to the file when you opened it.

```
if loc(x)>0 then 1000
```

locate [*row*],[*column*],[*cursor*],[*start*],[*stop*]]

Positions the cursor on the screen at *row* and *column*.

cursor sets the cursor. A value of 1 makes the cursor visible. A value of 0 makes the cursor invisible.

start is a numeric expression in the range 0-31 that specifies the first scan line of the cursor.

stop specifies the last scan line of the cursor, and it can be in the same range as *start*.

```
locate 10,20,1,
locate 25,1,1,3
```

lock [#]*buffer* [,*record*] **unlock [#]*buffer* [,*record*]**

Controls access by other processes to all or part of an opened file, specified by *buffer*. **Lock** and **unlock** are for use in conjunction with the MS-DOS SHARE command.

record is the record or the range of records to lock or unlock.

```
lock 1,1 to 4
unlock 1,1 to 4
```

lof(*buffer*)

Returns the length of the file in bytes.

buffer is the number BASIC assigned to the file when opening it.

```
y = lof(5)
```

BASIC Quick Reference

lof(buffer)

Communications. Returns the free space in the input queue. You can use **lof** to determine when an input queue is getting full.

buffer is the number assigned to the file when it was opened.

```
if lof(s)<20 goto 1000
```

log(number)

Computes the natural logarithm of *number*.

number must be greater than zero.

```
print log(3.14159)
z = 10 * log(p5/p1)
```

lpos(number)

Returns the logical position of the print head within the printer's buffer.

number can be 0 or 1 to indicate LPT1.

```
100 if lpos(x)>60 then lprint
```

lprint [using format;] data[,data...]

Prints data on the printer. **Lprint** assumes a print width of 132 characters. You can change the width using **width**.

format is the one or more field specifiers enclosed in quotation marks

data is a string or numeric value.

See **print** and **print using** for more information on formatting the output.

```
lprint (a * 2)/3
lprint using "#####.##";2.17
```

lset field name = data

Moves data to the direct access buffer, and places it in *field name* in preparation for a **put** statement. You must convert to a string any numeric value placed in a direct access file buffer with an **lset** statement. See **mks\$**, **mkd\$**, and **mki\$**.

field name is a string variable defined in a **field** statement. **Lset** left-justifies the data. You must use **field** to set buffer fields before using **lset**.

```
lset ad$ = "2000 East Pecan St."
lset td$=e$
```

merge pathname

Loads a program and merges it with the program in memory. The file must be in ASCII format (saved with the **a** option).

pathname is the filename specification of the file you want to load.

```
merge "prog2.txt"
```

noise source, volume, duration

volume, duration same as sound (q.v.)
source is type of noise: 0-3 are beeps (0 highest freq.)
4-7 are white noise (7 is clicks)

mid\$(oldstring,start[,length])=newstring

Replaces a portion of *oldstring* with *newstring*.

```
mid$ (a$,3,4) = "12345":print a$
```

start specifies the position of the first character you want to change.
length specifies the number of characters you want to replace.

mid\$(string,start[,length])

Returns a substring of *string*.

```
a$ = "Weatherford":print mid$(A$,3,2)
```

start specifies the position in the string from which to get the substring.
length is the number of characters in the substring. It must be in the range 1-255.

mkdir pathname

Creates the directory specified by *pathname*.

```
mkdir "a:\accts\payable"  
mkdir "\address"
```

pathname is a standard directory specification.

mkd\$(double-precision expression)

mki\$(integer expression)

mks\$(single-precision expression)

Converts numeric values to string values. **Mkd\$** returns an 8-byte string; **mki\$**, a 2-byte string; and **mks\$**, a 4-byte string. These are the inverse functions of **cvd**, **cvi**, and **cvs**.

expression is an iteger or single or double precision number.

```
1set ytd$ = mks$(564.33)  
1set tot$ = mks$(tot)
```

name "old filename" as "new filename"

Renames *old filename* as *new filename*.

```
name "file.bas" as "file.old"
```

old filename is the name of the file to be renamed.
new filename is the new name of the file.

new

Deletes the program in memory, and clears all variables.

```
new
```

oct\$(number)

Returns a string that represents the octal value of a decimal number.

number specifies the decimal value to convert into an octal string.

```
print oct$(30) s$=oct$(90)
```

on com(channel)gosub line

Transfers program control to a subroutine beginning at *line* when activity occurs on the specified communications channel.

channel specifies communications Channel 1 or 2.
line is the subroutine line at which execution begins when activity occurs on the communications channel. Specifying Line 0 turns off communications trapping.

```
on com (1) gosub 1000
```

on error goto line

Transfers control to *line* if an error occurs. You must execute an **on error goto** before the error occurs. Specifying Line 0 turns off error trapping.

```
on error goto 1500
```

on number gosub list

Causes a branch to the line specified by the value of *number*.

list is a list of one or more line numbers separated by commas. Therefore, if *number* equals 1, GW-BASIC branches to the first line in *list*. If *number* equals 2, GW-BASIC branches to the second line in *list*.
number must be in the range 0-255.

```
on y gosub 1000, 2000, 3000
```

on number goto list

Goes to the line specified by the value of *number*.

list is a list of one or more line numbers separated by commas. Therefore, if *number* equals 1, GW-BASIC branches to the first line in *list*. If *number* equals 2, GW-BASIC branches to the second line in *list*. *number* must be in the range 0-255.

```
on mi goto 150, 160, 170, 150, 180
```

on key(number)gosub line

Transfers program control to a subroutine beginning at *line* when you press the specified key.

number specifies the number of the key to trap. GW-BASIC numbers function keys 1-10, the cursor direction keys 11-14, and the user keys 15-20. You define the user keys with the **key** statement.

```
on key(13) gosub 500
```

on pen gosub line

Transfers program control to the subroutine at *line* when you activate the light pen.

```
on pen gosub 1000
```

on play(*number*)gosub *line*

Transfers program control to the subroutine beginning at *line* when the number of notes in the background music buffer goes from *number* to *number* minus 1.

number must be in the range 1-32.

```
on play(30) gosub 200
```

on strig(*number*)gosub *line*

Transfers program control to the subroutine at *line* when you press one of the joystick's buttons.

number specifies the button pressed and is one of the following. (Specifying Line 0 turns off joystick trapping.)

- 0 left joystick, button 1
- 2 right joystick, button 1
- 4 left joystick, button 2
- 6 right joystick, button 2

```
on strig(0) gosub 1000
```

on timer(*number*)gosub *line*

Transfers program control to the subroutine at *line* when the specified time has elapsed. *number* specifies the number of seconds. It can be a value in the range 1-86400 (86400 seconds = 24 hours).

```
on timer(3600) gosub 500
```

**open *mode*,*buffer*,[*pathname*][*device*;] [,*record length*]
open[*pathname*][*device*;][*for mode*][*access*] as *buffer*[*len=record length*]**

Establishes an input/output path for a file or device.

buffer specifies the I/O buffer (1-255) in memory to use when accessing the file. If you do not specify *pathname*, you must specify *device*;

record length sets the record length for direct access files. It can be in the range 2-32768. Default = 128 bytes.

mode specifies any of the following:

o (open)	sequential output mode
i (input)	sequential input mode
a (append)	sequential extension of an existing file
r (random)	direct input/output mode

access controls the processes that can access the file and the degree to which they do so.

access can be shared, lock read, lock write, or lock read write.

In the first form of the syntax, you must use the abbreviated form of *mode* and must enclose it in quotation marks.

In the second form of the syntax, you must specify the complete word for *mode*. You cannot specify random. If you want to use direct access in the second form of the syntax, omit *mode*.

```
open "r",2,"test.dat"
open "lpt1:"for output as 2
```

open "com channel: [speed] [,parity] [,data][,stop] [,rs] [,cs[seconds]][,ds[seconds]][,cd[seconds]][,mode][,pe][,lf]" [**for mode**] **as** [#][buffer] [**len** = number]

Communications. Opens a file and assigns a buffer for RS-232C (Asynchronous Communications Adapter) communication.

channel can be 1 or 2 to select the communications channel to open.

speed specifies the baud rate. It can be 75, 110, 150, 300, 600, 1200, 2400, 4800, or 9600. Default = 300.

parity can be e for even, o for odd, m for mark, s for space, or n for no. Default = e.

data specifies the number of bits. It can be 5, 6, 7, or 8. Default = 7.

stop can be either 1 or 2 to specify the number of stop bits. Default = 2 for baud rates of 75 and 100, and 1 for all other baud rates.

mode is either output or input for sequential access. Default = random input/output.

buffer specifies the buffer that accesses the file. It can be in the range 1-15.

number specifies the maximum number of bytes that the **get** and **put** statements can access in the communications buffer. Default = 128 bytes.

rs suppresses RTS (request to send)

cs[n] controls CTS (clear to send)

ds[n] controls DSR (data set ready)

cd[n] controls CD (carrier detect)

lf sends a line feed at each return

pe enables parity checking

n is the number of milliseconds to wait (0-65535) for that signal before a device timeout error occurs

```
open "com1:" as 1
open "com1:9600,n,8,1,bin" as 2
```

option base value

Sets *value* as the minimum value for an array subscript. This statement must precede the **dim** statement.

value can be 1 or 0. Default = 0.

```
option base 1
```

out port, data byte

Sends a data byte to a machine output port.

port is an integer in the range 0-65535.

data byte is an integer in the range 0-255.

```
out 32,100
```

paint (x,y) [color[,border][,background]]

Graphics. Fills a screen area with a selected color or pattern.

(x, y) are the coordinates at which painting begins.
color can be a number or a string. If *color* is a number, it specifies a color number available in the current screen mode. (See "Text and Graphics Modes.") If *color* is a string, it specifies the mask to use for tiling in the form:
 chr\$(&hnn)+chr\$(&hnn)=chr\$(&hnn)...

border specifies an object's border color. Default = *color*.
background is the color to skip when checking for borders.

paint (x, y) [,color[,border][,background]]

Enhanced screen modes use a different method to create tile patterns. Rather than interpret *color* string elements sequentially, GW-BASIC stores the string as a stack of 8-bit units, called bit planes.

(x, y) are the coordinates at which painting begins.
color can be a number or a string. If *color* is a number, it specifies a color number available in the current screen mode. (See "Text and Graphics Modes.") If *color* is a string, it specifies the mask to use for tiling in the form:
 chr\$(&hnn)+chr\$(&hnn)=chr\$(&hnn)...

border specifies an object's border color. Default = *color*.
background is the color to skip when checking for borders.

In Screen Modes 7, 8, and 9, GW-BASIC uses four bit-planes to define one tile byte by reading each 4-bit column in the stack as a single value. This value represents a palette slot number.

In Screen Mode 10, GW-BASIC uses two bit-planes to define the attributes of eight pixels. The attributes are: 0=black, 1=normal intensity, 2=blinking intensity to off, 3=high intensity.

```
10 cls:screen 7:color 7,0
20 t1$=chr$(&h55)+chr$(&h0)+chr$(&h0)
   +chr$(&h55)
30 t2$=chr$(&h0)+chr$(&h55)+chr$(&h55)
   +chr$(&h0)
40 t3$=chr$(&h55)+chr$(&h55)+chr$(&h0)
   +chr$(&h0)
50 paint (160,90),t1$+t2$+t3$
```

palette [color,display,color]

Graphics. Changes the color associated with a particular color number in the current palette.

color specifies the color in the current palette you want to change.
display color specifies the new color you want GW-BASIC to display when *color* is specified.

```
palette 3,7
```


BASIC Quick Reference

palette [*palette color number*][*,color number*]

Defines one of 64 colors to store in the specified slot of the 16-color palette.

palette color number specifies the palette position to change. In Screen Modes 0, 7, 8, and 9, *palette color number* is an integer in the range 0-15. In Screen Mode 1, *palette color number* can be 0, 1, 2, or 3. In Screen Mode 2, *palette color number* is either 0 (background) or 1 (foreground). In Screen Mode 10, *palette color number* can be 0, 1, 2, or 3.

color number is the color to put into the palette position specified by *palette color number*. In Screen Modes 0 and 9, *color number* can be any of 64 colors. In Screen Modes 1, 2, 7, and 8, *color number* can be any of the first 16 colors. In Screen Mode 10, *color number* is an integer in the range 0-8, as defined in the following table:

0	Off
1	Blinking off to on
2	Blinking off to high-intensity
3	Blinking on to off
4	On
5	Blinking on to high-intensity
6	Blinking high-intensity to off
7	Blinking high-intensity to on
8	High-intensity

palette using *array(subscript)*

Graphics. Changes the colors associated with more than 1 of the color numbers in the current palette.

array is the name of an integer array in which you can define the order of colors to be put in the current palette.

subscript is the position in that array that contains the value of the first position for the palette.

```
palette using a(0)
palette using a(2)
```

pcopy *source page, destination page*

Copies the *source* video page to the *destination* video page.

```
pcopy 3,5
pcopy 6,4
```

peek(*memory location*)

Returns a byte from *memory location*.

memory location must be in the range -32768 to 65535. The value returned is an integer in the range 0-255.

```
a = peek(&h5a00)
```

pen(number)

Returns the light pen's coordinates.

number is a number in the range 0-9 that tells GW-BASIC what to return.

- 0 Returns -1 if the pen button was pressed since the last poll. Returns 0 if it was not.
- 1 Returns the x-coordinate (horizontal) at which the pen was last activated.
- 2 Returns the y-coordinate (vertical) at which the pen was last activated.
- 3 Returns -1 if the pen button is pressed, or 0 if it is up.
- 4 Returns the last known valid x-coordinate (horizontal).
- 5 Returns the last known valid y-coordinate (vertical).
- 6 Returns the character row position where the pen was last activated.
- 7 Returns the character column position at which the pen was last activated.
- 8 Returns the last known character row position.
- 9 Returns the last known character column position.

a = pen(1)

pen action

Turns on, turns off, or temporarily halts light pen event trapping.

action can be on, off, or stop.

```
pen on
pen off
pen stop
```

play string

Plays musical notes specified by *string*.

string is a string expression consisting of one or more single-character music commands.

Single-Character Music Commands

a-g	plays notes A through G of one musical scale. Include a number sign (#) or plus sign (+) to specify a sharp note. Include a minus sign (-) to specify a flat note.
ln	sets the duration of the notes that follow. <i>n</i> can be a value in the range 1-64. <div><div>1 specifies a whole note.</div><div>2 specifies a half note.</div><div>4 specifies a quarter note.</div><div>8 specifies an eighth note.</div><div>16 specifies a sixteenth note.</div></div>
on	sets the current octave. There are 7 octaves. 0-6. Octave 3 starts with middle C. Default = Octave 4.
nn	plays a note. <i>n</i> can be in the range 0-84.
pn	rests. <i>n</i> can be in the range 1-64.
tn	sets the number of quarter notes in 1 minute. <i>n</i> can be in the range of 32-255. Default = 120 quarter notes in 1 minute.
.	plays as a dotted note. GW-BASIC increases the value of the note by one-half.
mf	plays the music in the foreground. Default = mb.
mb	plays the music in the background. A maximum of 32 notes and/or rests can play in background at a time. Default = mb.
mn	sets "music normal." Each note plays 7/8 of the duration as set by the l option. Default = mn.
ml	sets "music legato." Each note plays the full duration as set by the l option. Default = mn.
ms	sets "music staccato." Each note plays 3/4 of the duration as set by the l option. Default = mn.
xvar;	executes a substring represented by <i>var</i> . You can have one string execute another, which executes a third, and so on.
Vn	sets the volume. <i>n</i> must be in the range 0 to 15. You must execute a SOUND ON statement to use this option. Default = 8.

```
play "c4f.c8f8.c16f8.g16a2f2"
```

play(number)

Returns the number of notes in the background music queue. The maximum value play can return is 32 (the maximum buffer size).

number is a dummy argument.

```
x=play(0)    x=play(2)
```

play action

Turns on, turns off, or temporarily halts background music event trapping.

action can be on, off, or stop.

```
play on
play off
play stop
```

pmap(coordinate,action)

Returns the physical or world coordinate for the specified coordinate.

coordinate is any x- or y-coordinate.
action is one of the following:

- 0 returns the physical x-coordinate for the specified world coordinate.
- 1 returns the physical y-coordinate for the specified world coordinate.
- 2 returns the world x-coordinate for the specified physical coordinate.
- 3 returns the world y-coordinate for the specified physical coordinate.

```
x=pmap(200,0)
z=pmap(50,0)
```

point(x, y) point(action)

Graphics. Returns the color number of a point on the screen, or returns the current physical or world coordinates.

(*x*, *y*) are the coordinates of the point.
action is one of the following:

- 0 returns the current physical x-coordinate (horizontal).
- 1 returns the current physical y-coordinate (vertical).
- 2 returns the world x-coordinate if **window** is active. Otherwise, returns the physical x-coordinate.
- 3 returns the world y-coordinate if **window** is active. Otherwise, returns the physical y-coordinate.

```
10 if point(1,1)=0 then preset(1,1) else
   pset(1,1)
20 x=point(0)
```

poke memory location, data byte

Writes *data byte* into *memory location*. Both *memory location* and *data byte* must be integers.

memory location must be in the range -32768 to 65535.

```
10 poke &h5A00, &hff
```

pos(number)

Returns the current column position of the cursor.

number is a dummy argument.

```
if pos(X)>70 then if a$ = chr$(32)
then a$ = chr$(13)
```

pset [step](x, y)[,color] preset [step](x, y)[,color]

Graphics. Draws a point on the display.

(x, y)	are the coordinates of the point.
step	appoints (x, y) as relative coordinates.
color	specifies the color of the point. If you use pset, <i>color</i> defaults to the foreground color. If you use preset, <i>color</i> defaults to the background color.

See "Text and Graphics Modes."

```
pset (1,1)
preset step (1,1),0
```

print data[,data,...]

Prints numeric or string data on the display. If you use commas, the cursor automatically advances to the next tab position before printing the next item. If you use semicolons or spaces to separate the data items, **print** prints the items without spaces between them.

```
print "Do"; "not"; "leave"; "spaces";
"between"; "these"; "words"
print "The total is", ttl
```

print using format; data[,data...]

Prints *data* using a format you specify.

<i>format</i>	consists of one or more field specifier(s), or any alphanumeric character. Enclose <i>format</i> in quotation marks.
<i>data</i>	can be a string, a numeric value, or both.

Specifiers for String Fields:

!	prints only the first character in the string.
\spaces\	prints 2+ <i>n</i> characters from the string. (<i>n</i> is the number of spaces between the slashes.)
&	prints the string without modifications.

Specifiers for Numeric Fields:

#	prints the same number of digit positions as number signs (#).
+	prints the sign of the number.
-	prints a negative sign after negative numbers (and a space after positive numbers).
**	fills leading spaces with asterisks.
\$\$	prints a dollar sign immediately before the number.
**\$	fills leading spaces with asterisks, and prints a dollar sign immediately before a number.
,	prints a comma before every third digit to the left of the decimal point. put [#]buffer[,record]
^ ^ ^ ^	prints in exponential format.
--	prints the next character as a literal character.

```
print using "####^"; 888888
print using "$$###.##"; 1234.5
print using "###.-"; -768.660
print using "###.##"; 876.567
```

print# buffer,[using format] data[,data...]

Writes *data* items to a sequential access disk file. **print#** does not compress data before writing it to disk. It writes an ASCII-coded image of the data.

See **print using** for information about the *format* parameter.

```
print#1,a
print#1,b$,t$
print#1,using "###.##";a(t)
```

pset [step](x, y)[,color] preset [step](x, y)[,color]

Graphics. Draws a point on the display.

(x, y) are the coordinates of the point.
step appoints (x, y) as relative coordinates.
color specifies the color of the point. If you use **pset**, *color* defaults to the foreground color. If you use **preset**, *color* defaults to the background color.

See "Text and Graphics Modes."

```
pset (1,1)
preset step (1,1),0
```

put [#]buffer[,record]

Puts a record in a direct access disk file.

record is the record number to be written to the file and must be in the range 1-16,777,215. Default = the current record number.

```
put 1
put 1,25
```

BASIC Quick Reference

put [#]buffer, number

Communications. Transfers data from the communications buffer to the communications line.

```
put 2,80
```

number is the number of bytes to transfer.

put (x, y),array[,action]

Graphics. Transfers an image stored in an array onto the screen.

```
put (200,100),a
```

(x, y) are the coordinates at which the image begins (the upper left corner of the image). Default = the last point referenced.

array is the array variable name that holds the image.

action sets the type of interaction between the transferred image and the image already on the screen. *action* can be pset, preset, and, or, or xor. Default = pset.

randomize[number]

Reseeds the random number generator.

```
randomize timer  
randomize 300
```

number can be an integer or a single- or double-precision number. If you omit *number*, GW-BASIC suspends program execution and prompts you for a number before executing.

read variable[,variable,...]

Reads values from a **data** statement, and assigns them to variables.

```
read t  
read n$, d$, t
```

rem

Inserts a remark line in a program. You can use an apostrophe (') as an abbreviation for rem.

```
rem average velocity  
'totals
```

renum [new line][,line][,increment]

Renums the program in memory including line numbers appearing after **goto**, **gosub**, **then**, **on/goto**, **on/gosub**, **on error goto**, **resume**, and **eri**.

line is the program line at which GW-BASIC starts renumbering. Default = the first line.

new line is the new number assigned to *line*. Default = 10.

increment tells GW-BASIC how to number the successive lines. Default = 10.

```
renum            renum 600, 5000, 100
```

reset

Closes all open files on all drives.

```
reset
```

restore [*line*]

Restores a program's access to previously read DATA statements.

line specifies the statement to access at the next read statement. Default = the first **data** statement.

```
restore
```

resume [*line*]

resume next

Resumes program execution after an error handling routine.

resume *line*

branches to the specified line number. Default = the statement in which the error occurred.

resume next

branches to the statement following the point at which the error occurred.

```
resume
resume 10
resume next
```

return [*line*]

Returns control from a subroutine executed by a **gosub** to the specified line. Default = the line immediately following the GOSUB.

```
return          return 40
```

right\$(*string,number*)

Returns the specified number of characters from the far right portion of *string*.

number is an integer in the range 1-255.

```
print right$("watermelon",5)
print right$("puppylove",4)
```

rmdir *pathname*

Removes (deletes) the directory specified by *pathname*. This directory must be empty except for the "." and ".." symbols. See the MS-DOS **copy** and **kill** commands.

```
rmdir "names"
rmdir "a:\accts\payable"
```


BASIC Quick Reference

rnd[(*number*)]

Returns a random number in the range 0-1.

```
print rnd(1)
a = rnd(0)
```

rset *field name* = *data*

Sets *data* in a direct access buffer field *name*, in preparation for a put statement, and right-justifies it.

```
rset a$ = cvi(z)
```

run [*line*]

run "*pathname*"[,*r*]

Executes the current program or loads and runs the specified program.

```
run
run 100
run "program.a"
```

number is an integer in the range -32767 to 32768. If *number* is negative, ***rnd*** starts the sequence of random numbers at the beginning. If *number* is 0, ***rnd*** repeats the last number generated. If *number* is a positive number, ***rnd*** returns the next number in the sequence.

line is the program line at which GW-BASIC begins execution. Default = the first line.

r uses GW-BASIC to leave the current files open when it loads the new program into memory. If you omit the *R* option, BASIC closes all open files before loading the program.

save "*pathname*" [,*a*]***save*** "*pathname*" [,*p*]

Saves a program on disk with the specified name. The *a* option saves the program in ASCII format. The *p* option saves the program in a protected format. Default = the compressed format.

```
save "a:file.1.bas"
save "\educ\mathpak.txt",a
```

screen (*row*,*column* [,*number*])

Returns the ASCII code for the character at the specified *row* and *column*. If *number* is specified and is non-zero, GW-BASIC returns the color attribute.

```
a = screen(20,20)
print screen(10,10,1)
```

screen [mode][,[burst][,[active page][,display page]]]

Sets the screen mode and screen attributes to be used by all other graphics statements.

mode is an integer in the range 0-2 or 7-10 that specifies the screen mode. Screen Modes 7, 8, 9, and 10 are only valid with an EGA video adapter.

burst activates or de-activates *color* in Screen Mode 0, 1, 7, 8 or 9. *burst* has no effect on Screen Modes 2 and 10. In Mode 0, use 1 to activate, 0 to de-activate. In Mode 1, use 0 to activate and 1 to de-activate.

active page selects the video page to which GW-BASIC is to write. Default = the current active page.

display page is an integer that selects the video page GW-BASIC is to display. Default = the current active page.

```
screen 0,0
screen 2
screen 8,1
```

sgn(number)

Determines *number*'s sign.

If *number* is negative, sgn returns -1. If *number* is positive, sgn returns 1. If *number* is 0, sgn returns 0.

```
y = sgn(a * b)
```

shell(command)

Loads and executes another program (a program with either a .exe or .com extension) or an internal command as a child process to the original program.

command is a string expression containing the name of the program you want to run.

```
shell "format b:"
```

sin(number)

Returns the sine of *number*.

number must be in radians.

```
print sin(7.96)
d=sin(t)
```

sound (frequency,duration)

Generates a sound with the frequency and duration specified.

frequency is an integer in the range 37-32767, indicating the frequency in Hertz.

duration is a numeric expression in the range 1-65535, specifying the duration in clock ticks.

```
sound 37,2
```

Takes up to 4 params: freq, dur, volume (0-15), sound channel (0-2)

space\$(number)

Returns a string of *number* spaces.

```
print "Cost" space$(4)
"Quantity" space$(9) "Total"
```

BASIC Quick Reference

spc(number)

Skips *number* spaces in a **print** statement.

```
print "Hello" spc(15) "there"
```

sqr(number)

Returns the square root of *number*.

number must be greater than zero.

```
print sqr(155.7)
```

stick (action)

Returns the coordinates of the joysticks.

action can be 0 or 1 for the horizontal and vertical coordinates of the left joystick, respectively. It can be 2 or 3 for the horizontal and vertical coordinates of the right joystick, respectively. You must read 0 before you can read 1, 2 and 3.

```
a=stick(0)
```

stop

Stops program execution.

```
stop
```

str\$(number)

Converts *number* to a string.

```
s$ = str$(x)  
print str$(-234)
```

strig(number)

Returns the status of joystick buttons. You must execute **strig on** before using this function.

number is a number in the range 0-7 to test the status of the joystick buttons. Even numbers test to see if the joystick button has been pressed and released. Odd numbers test to see if the button is being pressed.

0,1 Left joystick, Trigger 1.
2,3 Right joystick, Trigger 1.
4,5 Left joystick, Trigger 2.
6,7 Right joystick, Trigger 2.

```
if strig(0) then beep
```

strig(number) action

Turns on, turns off, or temporarily halts joystick trapping. *action* is on, off, or stop.

number specifies the joystick and trigger. It can be:

- 0 Left joystick, Trigger 1.
- 2 Right joystick, Trigger 1.
- 4 Left joystick, Trigger 2.
- 6 Right joystick, Trigger 2.

```
strig(0) on
strig(6) off
```

string\$(number,character)

Returns a string containing the specified number of characters.

number must be in the range 0-255.
character is a string or an ASCII code.

```
b$ = string$(25, "x")
print string$(50, 10)
```

swap(variable1,variable2)

Exchanges the values of two variables of the same type.

```
swap f1#,f2#
```

system

Returns you to the MS-DOS command level.

```
system
```

tab(number)

Spaces to position *number* on the display.

number must be in the range 1-255.

```
print "Name" tab(25) "Amount":print
```

tan(number)

Returns the tangent of *number*.

number must be in radians.

```
print tan(7.96)
s = tan(x)
```

time\$[=string]

Sets the time to the specified string or retrieves the current time if you omit *string*. GW-BASIC uses a 24-hour clock.

string is a literal, enclosed in quotation marks.

```
time$ = "14:15"
print time$
```

timer

Returns the *number* of seconds since midnight or since the last system reset.

```
a = timer
print timer
```

timer action

Turns on, turns off, or temporarily halts timer event trapping.

action can be on, off, or stop.

```
timer on
timer off
timer stop
```

troff

tron

Turns on (tron) or turns off (troff) the program flow tracer.

```
tron
troff
```

usr[number](argument)

Calls the user's assembly-language subroutine identified with *number*, and passes *argument* to that subroutine.

number specifies a number in the range 0-9 which identifies the subroutine being called as defined with the DEF USR statement.

The number you specify must be the same as the corresponding **def usr** statement for that routine. Default = 0.

argument is a numeric or string expression passed to the subroutine.

val(string)

Calculates the numerical value of *string*.

string is a numeric string.

```
print val("100")
print val(num$)
```

varptr(variable)

varptr([#]buffer)

Returns the offset into GW-BASIC's data segment of a variable of the file control block.

When used with *variable*, **varptr** returns the address of the first byte of data identified with *variable*.

variable is the name of the variable for which you want the address.

When used with *buffer*, it returns the address of the file's control block.

buffer is the number assigned to the file when it was opened.

```
a = varptr(a$)
print varptr(3)
```

varptr\$(variable)

Returns a 3-byte string representing the memory address of *variable*. Byte 0 is the type, Byte 1 is the low byte of address, and Byte 2 is the high byte of address. Type 2 is for integer variables, 3 for string variables, 4 for single-precision variables, and 8 for double-precision variables.

variable is the name of the variable for which you want the offset.

```
play "x" + varptr$(as)
```

view [screen] [(x1, y1)-(x2, y2)[, [color][, [border]]]]

Graphics. Creates a viewport that redefines the screen parameters.

(x1, y1) specifies the upper-left coordinates for the rectangular viewport.
(x2, y2) specifies the lower-right coordinates for the rectangular viewport.
color the color with which to fill the viewport.
border an integer specifying the color for the boundary line around the viewport.
screen specifies that all coordinates used in drawing are absolute to point 0,0 on the screen. If you omit *screen*, all coordinates specified are relative to the viewport coordinates.

```
view (10,10)-(100,100)
view screen (20,25)-(100,150)
```

view print top line to bottom line

Creates a text viewport that redefines the text screen parameters.

top line specifies the first line of the text viewport. It can be in the range 1-24, but must be less than *bottom line*. Default = Line 1.
bottom line specifies the last line of the text viewport. It can be in the range 1-24, but must be greater than *top line*. Default = Line 24.

```
view 1 to 15
```

wait port, number1 [, number2]

Suspends program execution until the specified machine input port develops a specified bit pattern.

number1 and *number2* are integers in the range 0-255.

```
wait 32,2
```

while expression wend

Executes a series of statements in a loop as long as a given condition is true.

expression is a logical expression that returns either true or false. If *expression* is true, GW-BASIC executes the statements after the **while** statement until it encounters a **wend** statement. If *expression* is still true, GW-BASIC repeats the process. If it is not true, execution resumes with the statement following the **wend** statement.

```
while a
print "Calculating..."
wend
```

BASIC Quick Reference

width [lprint] size width buffer, size width device, size

Sets the line width in number of characters for the display, printer, or communications channel.

size can be an integer in the range 0-255 that specifies the number of characters in a line. For the screen, *size* can be only 40 or 80. Default = 255 (communications channel).

buffer is the number assigned to the file in the OPEN statement.

device is a valid device, enclosed in quotation marks, that specifies the device for which you are setting the width. It can be `scrn:`, `lpt2:`, `com1:`, or `com2:`.

```
width 40
width lprint 100
width "scrn:", 40
```

window [screen] [(x1, y1)-(x2, y2)]

Lets you change the physical coordinates of the screen (or current viewport). **window** lets you plot points outside the normal screen coordinate limits by setting new world coordinates to the screen.

(x1, y1) are the world coordinates for the upper-left corner of the screen.

(x2, y2) are the world coordinates for the lower-left corner of the screen.

screen tells GW-BASIC to set the coordinates like the screen display. If you omit **screen**, GW-BASIC inverts the y-coordinates to show a true Cartesian coordinate system.

```
window (1984,100000)-(1987,300000)
```

write data[,data...]

Outputs *data* to the screen.

```
10 A=80:B=90:C$="That's All"
20 WRITE A,B,C$
```

write# buffer, data[,data...]

Writes *data* to a sequential access disk file.

```
write#1,A$,B$
```

GW-BASIC Function Key Settings

F1	list"	F6	,"lpt1:" ENTER
F2	run ENTER	F7	tron ENTER
F3	load"	F8	troff ENTER
F4	save"	F9	key
F5	CONT ENTER	F10	screen 0,0,0 ENTER

Typing Keywords Using the ALT Key

ALT-A	auto	ALT-N	next
ALT-B	bsave	ALT-O	open
ALT-C	color	ALT-P	print
ALT-D	delete	ALT-Q	(none)
ALT-E	else	ALT-R	run
ALT-F	for	ALT-S	screen
ALT-G	goto	ALT-T	then
ALT-H	hex\$	ALT-U	using
ALT-I	input	ALT-V	val
ALT-J	(none)	ALT-W	width
ALT-K	key	ALT-X	xor
ALT-L	locate	ALT-Y	(none)
ALT-M	motor†	ALT-Z	(none)

†**Motor** is a reserved word, but it is not recognized in this implementation of GW-BASIC.

Exponential Notation and Numeric Precision Characters

D	Used in double-precision exponential notation.
E	Used in single-precision exponential notation.
%	Makes the variable preceding it integer precision.
!	Makes the variable preceding it single precision.
#	Makes the variable preceding it double precision.
\$	Makes the variable preceding it a string.

Operator Precedence

Each operator or group of operators takes precedence over the group below it.

()	Parentheses
^	Exponentiation
+ -	Unary positive, negative
* /	Multiplication, division
\	Integer division
MOD	Modulus arithmetic
+-	Addition, subtraction
< > <=	Relational tests
>= <>	
NOT	
AND	
OR	
XOR	
EQV	
IMP	

Video Modes

Screen Mode 0

Graphics = No
Resolution = N.A.
Colors = 16
Text Width = 40/80
Maximum Pages = 8 at Width 40, 4 at Width 80
Video Page Size = 2048 at Width 40, 4096 at Width 80

Screen Mode 1

Graphics = Yes
Resolution = 320 x 200.
Colors = 4 colors 2 palettes
Text Width = 40
Maximum Pages = 8
Video Page Size = 16384

Screen Mode 2

Graphics = Yes
Resolution = 640 x 200
Colors = monochrome
Text Width = 80
Maximum Pages = 8
Video Page Size = 16384

Screen Mode 3

Graphics = Yes
Resolution = 160 x 200
Color Set = 16 colors
Text width = 20 columns
Maximum Pages = 8
Video Page Size = 16384

Screen Mode 4

Graphics = Yes
Resolution = 320 x 200
Color Set = 4 colors
Text width = 40 columns
Maximum Pages = 8
Video Page Size = 16384

Screen Mode 5

Graphics = Yes
Resolution = 320 x 200
Color Set = 16 colors
Text width = 40 columns
Maximum Pages = 4
Video Page Size = 32768

Screen Mode 6

Graphics = Yes
Resolution = 640 x 200
Color Set = 4 colors
Text width = 80 columns
Maximum Pages = 4
Video Page Size = 32768

Screen Mode 7 (EGA only)

Graphics = Yes
Resolution = 320 x 200
Colors = 16
Text Width = 40
Maximum Pages = 8
Video Page Size = 32768

Screen Mode 8 (EGA only)

Graphics = Yes
Resolution = 640 x 200
Colors = 16
Text Width = 80
Maximum Pages = 4
Video Page Size = 65536

Screen Mode 9 (EGA only)

Graphics = Yes
Resolution = 640 x 350
Colors = 16 of 64
Text Width = 80
Maximum Pages = 2
Video Page Size = 131072

Screen Mode 10 (EGA only)

Graphics = Yes
Resolution = 640 x 350
Colors = monochrome
Text Width = 80
Maximum Pages = 2
Video Page Size = 131072

Enhanced Graphics Color Selection

With an EGA/EGM combination in the enhanced mode, the number of available colors increases to 64 (eight shades of eight colors). The following chart shows how the colors are distributed:

Color	Shades							
Black	0	8	16	24	32	40	48	56
Blue	1	9	17	25	33	41	49	57
Green	2	10	18	26	34	42	50	58
Cyan	3	11	19	27	35	43	51	59
Red	4	12	20	28	36	44	52	60
Magenta	5	13	21	29	37	45	53	61
Yellow	6	14	22	30	38	46	54	62
White	7	15	23	31	39	47	55	63

Error Codes and Messages

Number	Message
1	NEXT without FOR
2	Syntax error
3	RETURN without GOSUB
4	Out of DATA
5	Illegal function call
6	Overflow
7	Out of memory
8	Undefined line number
9	Subscript out of range
10	Redimensioned array/duplicate Definition
11	Division by zero
12	Illegal direct
13	Type mismatch
14	Out of string space
15	String too long
16	String formula too complex
17	Can't continue
18	Undefined user function
19	No RESUME
20	RESUME without error
21	Unprintable error
22	Missing operand
23	Line buffer overflow
24	Device timeout
25	Device fault
26	FOR without NEXT
27	Out of paper
29	WHILE without WEND
30	WEND without WHILE
50	FIELD overflow
51	Internal error
52	Bad file number
53	File not found
54	Bad file mode
55	File already open
57	Device I/O error
58	File already exists
61	Disk full
62	Input past end
63	Bad record number
64	Bad file name
66	Direct statement in file
67	Too many files
68	Device unavailable
69	Communication buffer overflow
70	Disk write protect
71	Disk not ready
72	Disk media error
73	Advanced feature
74	Rename across disks
75	Path/File access error
76	Path not found
77	Deadlock
78	Unprintable error

Index

Index

A

abs (GW-BASIC command/statement), 63
adapter board, illustration, 16
adding options. *See* external options; internal options.
ALT key, for typing GW-BASIC keywords, 101
append (MS-DOS command), 33
append lines (Edlin command), 52
application software, running, 9
asc (GW-BASIC command/statement), 63
assemble (Debug command), 56
assign (MS-DOS command), 33
atn (GW-BASIC command/statement), 63
attrib (MS-DOS command), 34
auto (GW-BASIC command/statement), 64
AUTOEXEC.BAT file, checking for, at start-up, 26
autofmt (MS-DOS command), 34

B

backup (MS-DOS command), 34
batch files
 echo command and, 39
 for...in...do command and, 40
 goto command and, 40
 if command and, 41
 pause command and, 45
 rem command and, 46
 shift command and, 48
beep (GW-BASIC command/statement), 64
bload (GW-BASIC command/statement), 64
break (MS-DOS command), 34
bsave (GW-BASIC command/statement), 64

C

cache (MS-DOS command), 35
call (GW-BASIC command/statement), 64
cd (MS-DOS command), 36
cdb (GW-BASIC command/statement), 64
chain (GW-BASIC command/statement), 65
chcp (MS-DOS command), 35
chdir
 (GW-BASIC command/statement), 65
 (MS-DOS command), 36
chkdsk (MS-DOS command), 36
chr\$ (GW-BASIC command/statement), 65
cint (GW-BASIC command/statement), 65
circle (GW-BASIC command/statement), 65
clear (GW-BASIC command/statement), 66
close (GW-BASIC command/statement), 66

cls

 (GW-BASIC command/statement), 66
 (MS-DOS command), 36
code pages, chcp command and, 35
color
 (GW-BASIC command/statement), 66-67
 selection, for EGA/EGM, in GW-BASIC, 103
com (GW-BASIC command/statement), 67
command processor
 exiting, 39
 starting new, 36
commands. *See* Debug commands; Edlin commands; GW-BASIC; MS-DOS commands; names of specific commands.
common (GW-BASIC command/statement), 67
comp (MS-DOS command), 36
compare (Debug command), 56
comparing files, with comp command, 36
compressing files, with dc command, 38
computer speed, default setting for, 25
CONFIG.SYS, checking for at start-up, 26
configuration. *See* setting up the computer.
cont (GW-BASIC command/statement), 67
coprocessor, math, 20
copy (MS-DOS command), 37
copy lines (Edlin command), 52
copying diskettes. *See* diskettes.
cos (GW-BASIC command/statement), 67
country-specific information
 nlsfunc command and, 45
 select command and, 48
csng (GW-BASIC command/statement), 68
csrlin (GW-BASIC command/statement), 68
CTRL-ALT-DEL keys, 51
CTRL-C, 51
 break command and, 34
CTRL key, and CTRL key combinations, 51
ctty (MS-DOS command), 37
cursor keys, 5
cvt (GW-BASIC command/statement), 68

D

data (GW-BASIC command/statement), 68
date
 changing, with date command, 37
 prompt for, setting, 25
date\$ (GW-BASIC command/statement), 68
dc (MS-DOS command), 38
Debug commands. *See also* names of specific commands.
 alphabetical list of, 56-58
 parameters for, 54-56

def fn (GW-BASIC command/statement), 69
def seg (GW-BASIC command/statement), 69
def usr (GW-BASIC command/statement), 69
defaults, changing. *See* setting up the computer.
defdbl, defint, defsgn, defstr (GW-BASIC command/statement), 68
del (MS-DOS command), 38
delete (GW-BASIC command/statement), 69
delete lines (Edlin command), 52
DeskMate
 exiting, 1, 11
 starting from MS-DOS, 11
 stored in ROM, 1
devices, mode command and, 43-44
dim (GW-BASIC command/statement), 69
dir (MS-DOS command), 38
directories
 changing, with chdir, 36
 copying, with xcopy command, 13, 50
 creating, with mkdir command, 43
 displaying, with tree command, 49
 fastopen command and, 39
 removing, with rmdir command, 47
disk buffers, default setting for, 25
diskcomp (MS-DOS command), 38
diskcopy command, 12-13, 38
diskette drives
 changing drives, 12
 designating as drive A, 26
 installing additional drives, 18, 20
 join command and, 42
 reassigning, with assign command, 33
 substituting, with subst command, 49
diskettes
 backup command and, 34
 care and handling of, 7
 chkdsk command and, 36
 comparing, with diskcomp command, 38
 copying, 9
 disktype command and, 39
 displaying label, with vol command, 50
 duplicating
 with diskcopy, 12-13, 38
 with xcopy, 13
 formatting, 12, 40
 inserting and removing, 7
 label command and, 42
 optimizing, with diskopt, 38
 program diskettes, 9
 select command and, 48
 write-protecting, 8
diskopt (MS-DOS command), 38
disktype (MS-DOS command), 39
DOS. *See* MS-DOS.
draw (GW-BASIC command/statement), 70
drives. *See* diskette drives.

dump (Debug command), 56
duplicating diskettes. *See* diskettes.

E

echo (MS-DOS command), 39
edit (GW-BASIC command/statement), 70
edit line (Edlin command), 52
editing keys, Edlin, 54
Edlin commands. *See also* names of specific commands.
 alphabetical list of, 52-54
Edlin editing keys, 54
encrypting files, with recrypt command, 46
end (GW-BASIC command/statement), 70
end edit (Edlin command), 52
enhanced graphics color selection, in GW-BASIC, 103
enhanced video adapter boards, adding, 21
enter (Debug command), 56
ENTER key, 51
environ and environ\$ (GW-BASIC command/statement), 71
environment, set command and, 48
eof (GW-BASIC command/statement), 71
erase
 (GW-BASIC command/statement), 71
 (MS-DOS command), 39
erdev and erdev\$ (GW-BASIC command/statement), 72
erl (GW-BASIC command/statement), 72
err (GW-BASIC command/statement), 72
error codes and messages, in GW-BASIC, 104
error (GW-BASIC command/statement), 72
ESC key, 51
exit (MS-DOS command), 39
exp (GW-BASIC command/statement), 72
exponential notation, in GW-BASIC, 101
external options
 joysticks, 15
 microphone, 15
 modem, 15
 mouse, 15
 printer, 15
exterr (GW-BASIC command/statement), 73

F

fastopen (MS-DOS command), 39
fdisk (MS-DOS command), 40
features of the Tandy 1000 TL, 1-3
field (GW-BASIC command/statement), 73
files
 attrib command and, 34
 backup command and, 34

files (*Continued*)

- comparing, with comp command, 36
 - compressing, with dc command, 38
 - copying
 - with copy command, 37
 - with xcopy command, 13, 50
 - deleting
 - with del command, 38
 - with erase command, 39
 - displaying, with dir command, 38
 - displaying contents of, with type command, 49
 - encrypting, with recrypt command, 46
 - fastopen command and, 39
 - maximum number open, setting, 26
 - optimizing, with diskopt, 38
 - print command and, 45
 - recover command and, 46
 - renaming, with ren command, 46
 - restore command and, 47
 - searching for text in, with find command, 40
 - share command and, 48
 - updating, with replace command, 47
 - verify command and, 49
- files (GW-BASIC command/statement), 73
- fill (Debug command), 56
- find (MS-DOS command), 40
- fix (GW-BASIC command/statement), 73
- for (GW-BASIC command/statement), 73
- foreign language characters. *See also* code pages; country-specific information.
- grftabl command and, 41
 - keybxx command and, 42
- for...in...do (MS-DOS command), 40
- format (MS-DOS command), 12, 40
- fre (GW-BASIC command/statement), 73
- function keys
 - location of, 5
 - settings, in GW-BASIC, 100
 - uses for, 51

G

- get (GW-BASIC command/statement), 74
- go (Debug command), 57
- gosub (GW-BASIC command/statement), 74
- goto
 - (GW-BASIC command/statement), 74
 - (MS-DOS command), 40
- grftabl (MS-DOS command), 41
- graphics (MS-DOS command), 41
- GW-BASIC
 - alphabetical list of commands and statements, 63-100. *See also* names of specific commands and statements.
 - enhanced graphics color selection, 103
 - error codes and messages, 104

GW-BASIC (*Continued*)

- exponential notation and numeric precision characters, 101
- function key settings, 100
- keywords, typing with the ALT key, 101
- loading into memory, 61
- operator precedence, 101
- video modes, 102

H

- hard disk card, 20
 - installing, 19
- hard disk drives
 - autofmt command for, 34
 - chkdsk command and, 36
 - disktype command and, 39
 - displaying label, with vol command, 50
 - fastopen command and, 39
 - format command and, 40
 - hsect command and, 41
 - label command and, 42
 - optimizing, with diskopt, 38
 - partitioning, with fdisk, 40
- hardware. *See also* external options; internal options.
 - definition, 1
 - keyboard, 5
 - maintenance of, 28
 - mode command and, 43-44
- hex (Debug command), 57
- hex\$ (GW-BASIC command/statement), 74
- hsect (MS-DOS command), 41

I

- if
 - (GW-BASIC command/statement), 74
 - (MS-DOS command), 41
- inkey\$ (GW-BASIC command/statement), 75
- inp (GW-BASIC command/statement), 75
- input and input\$ (GW-BASIC command/statement), 75
- input (Debug command), 57
- input/output devices, specifying, with cty command, 37
- INS key, 51
- insert (Edlin command), 52
- instr (GW-BASIC command/statement), 75
- int (GW-BASIC command/statement), 76
- internal options
 - adapter board, illustration, 16
 - diskette drives, 20
 - installing, 18
 - enhanced video adapter boards, 21

internal options (*Continued*)

- expanded memory, 20
- hard disk card, 20
- installing, 19
- main logic board, 17
- math coprocessor, 20
- memory kits, 20
- modem board, 20
- other options, 21
- procedure for adding, 16
- rear view of computer, 16
- removing the cover, 16
- TandyLink board, 20

iocti and iocti\$ (GW-BASIC command/statement), 76

J

join (MS-DOS command), 42

joysticks, adding to your system, 15

K

key (GW-BASIC command/statement), 76, 77

key list (GW-BASIC command/statement), 76

key off (GW-BASIC command/statement), 76

key on (GW-BASIC command/statement), 76

keyboard

- displaying template for, 51
- keybxx command and, 42
- sections of, 5

keybxx (MS-DOS command), 42

keys

- DOS keys, 51-52
- Edlin editing keys, 54

keywords, in GW-BASIC, 101

kill (GW-BASIC command/statement), 77

L

label (MS-DOS command), 42

lcopy (GW-BASIC command/statement), 77

left\$ (GW-BASIC command/statement), 77

len (GW-BASIC command/statement), 77

let (GW-BASIC command/statement), 77

If (MS-DOS command), 42

line (GW-BASIC command/statement), 78

line feed, suppressing, with If command, 42

line input and line input# (GW-BASIC command/statement), 78

list

- (Edlin command), 53
- (GW-BASIC command/statement), 78

l!ist (GW-BASIC command/statement), 78

load

- (Debug command), 57
- (GW-BASIC command/statement), 79

loc (GW-BASIC command/statement), 79

locate (GW-BASIC command/statement), 79

lock (GW-BASIC command/statement), 79

lof (GW-BASIC command/statement), 79-80

log (GW-BASIC command/statement), 80

logic board, illustration, 17

lpos (GW-BASIC command/statement), 80

lprint (GW-BASIC command/statement), 80

lset (GW-BASIC command/statement), 80

M

main logic board, illustration, 17

math coprocessor, 20

md (MS-DOS command), 43

memory

- cache command and, 35
- expansion boards for, 20
- kits for expanding, 20
- memory diagnostics, on start-up option, 25

merge (GW-BASIC command/statement), 80

microphone, adding to your system, 15

mid\$ (GW-BASIC command/statement), 81

mkd\$ (GW-BASIC command/statement), 81

mkdir

- (GW-BASIC command/statement), 81
- (MS-DOS command), 43

mki\$ (GW-BASIC command/statement), 81

mks\$ (GW-BASIC command/statement), 81

mode (MS-DOS command), 43-44

modem

- adding to your system, 15
- internal modem, adding, 20

more (MS-DOS command), 44

mouse, adding to your system, 15

move (Debug command), 57

move lines (Edlin command), 53

MS-DOS. *See also* MS-DOS commands.

- definition, 11
- displaying version of, with ver command, 49
- entering commands, 11
- prompt for, 11
- starting DeskMate from, 11
- stored in ROM, 1
- transferring, with sys command, 49

MS-DOS commands. *See also* names of specific commands.

alphabetical list of, 33-50

N

name
 (Debug command), 57
 (GW-BASIC command/statement), 81
 networks, share command and, 48
 new (GW-BASIC command/statement), 81
 nlsfunc (MS-DOS command), 45
 numeric keypad, 5
 numeric precision characters, in GW-BASIC, 101

O

oct\$ (GW-BASIC command/statement), 81
 on ... gosub (GW-BASIC command/statement), 82
 on ... goto (GW-BASIC command/statement), 82
 on com ... gosub (GW-BASIC command/statement), 82
 on error goto (GW-BASIC command/statement), 82
 on key ... gosub (GW-BASIC command/statement), 82
 on pen gosub (GW-BASIC command/statement), 82
 on play ... gosub (GW-BASIC command/statement), 83
 on strig ... gosub (GW-BASIC command/statement), 83
 on timer ... gosub (GW-BASIC command/statement), 83
 open (GW-BASIC command/statement), 83-84
 operating system. *See* MS-DOS.
 operator precedence, in GW-BASIC, 101
 option base (GW-BASIC command/statement), 84
 options, adding. *See* external options;
 internal options.
 out (GW-BASIC command/statement), 84
 output (Debug command), 57

P

page (Edlin command), 53
 paint (GW-BASIC command/statement), 85
 palette (GW-BASIC command/statement), 85-86
 palette using (GW-BASIC command/statement), 86
 parameters for Debug commands, 54-56
 path, setting
 with append command, 33
 with path command, 45
 subst command and, 49
 pause (MS-DOS command), 45
 pause key, 51
 pcopy (GW-BASIC command/statement), 86
 peek (GW-BASIC command/statement), 86
 pen (GW-BASIC command/statement), 87
 play (GW-BASIC command/statement), 88-89

pmap (GW-BASIC command/statement), 89
 point ... point (GW-BASIC command/statement), 89
 poke (GW-BASIC command/statement), 89
 pos (GW-BASIC command/statement), 89
 precision characters, numeric, in GW-BASIC, 101
 primary start-up device, default setting for, 25
 print
 (GW-BASIC command/statement), 90
 (MS-DOS command), 45
 print# (GW-BASIC command/statement), 91
 print output, redirecting, with CTRL-P, 51
 print using (GW-BASIC command/statement), 90-91
 printers
 adding to your system, 15
 troubleshooting, 27
 printing the screen
 graphics command and, 41
 SHIFT-PRTSC keys for, 41, 52
 problem-solving. *See* troubleshooting.
 proceed (Debug command), 58
 processor speed, default setting for, 25
 program diskettes, 9
 prompt, system, 11
 changing, with prompt command, 46
 pset (GW-BASIC command/statement), 90, 91
 put (GW-BASIC command/statement), 91-92

Q

quit
 (Debug command), 58
 (Edlin command), 53

R

RAM buffer, cache command and, 35
 randomize (GW-BASIC command/statement), 92
 rd (MS-DOS command), 47
 read (GW-BASIC command/statement), 92
 recover (MS-DOS command), 46
 recrypt (MS-DOS command), 46
 register (Debug command), 58
 rem
 (GW-BASIC command/statement), 92
 (MS-DOS command), 46
 ren (MS-DOS command), 46
 renum (GW-BASIC command/statement), 92
 replace (MS-DOS command), 47
 replace string (Edlin command), 53
 reset (GW-BASIC command/statement), 93
 resetting the computer, with CTRL-ALT-DEL, 51
 restore
 (GW-BASIC command/statement), 93
 (MS-DOS command), 47

resume (GW-BASIC command/statement), 93
return (GW-BASIC command/statement), 93
right\$ (GW-BASIC command/statement), 93
rmdir
 (GW-BASIC command/statement), 93
 (MS-DOS command), 47
rnd (GW-BASIC command/statement), 94
rset (GW-BASIC command/statement), 94
run (GW-BASIC command/statement), 94

S

save (GW-BASIC command/statement), 94
screen (GW-BASIC command/statement), 94-95
screen, printing
 graphics command and, 41
 SHIFT-PRTSC keys for, 41, 52
search (Debug command), 58
search text (Edlin command), 53
select (MS-DOS command), 48
set (MS-DOS command), 48
setting up the computer
 default settings
 AUTOEXEC.BAT, checking for, 26
 computer speed, 25
 CONFIG.SYS, checking for, 26
 disk buffers, number of, 25
 disk drive, designating, 26
 files, maximum number open, 26
 initial start-up program, 25
 memory diagnostics on start-up, 25
 primary start-up device, 25
 time and date prompt, 25
 video display, 24
 screen for Setup program, 24
 setup command and, 48
 starting the Setup program, 23
 storage of settings in the EEPROM, 23
sgn (GW-BASIC command/statement), 95
share (MS-DOS command), 48
shell (GW-BASIC command/statement), 95
shift (MS-DOS command), 48
SHIFT-PRTSC keys, 41, 52
sin (GW-BASIC command/statement), 95
software
 definition, 1
 running, 9
solving problems. *See* troubleshooting.
sort (MS-DOS command), 48
sound (GW-BASIC command/statement), 95
space\$ (GW-BASIC command/statement), 95
spc (GW-BASIC command/statement), 96
sqr (GW-BASIC command/statement), 96
start-up device, primary, default setting for, 25
start-up program, initial, default setting for, 25
stick (GW-BASIC command/statement), 96

stop (GW-BASIC command/statement), 96
str\$ (GW-BASIC command/statement), 96
strig (GW-BASIC command/statement), 96-97
subst (MS-DOS command), 49
swap (GW-BASIC command/statement), 97
sys (MS-DOS command), 49
system (GW-BASIC command/statement), 97
system prompt, 11

T

tab (GW-BASIC command/statement), 97
tan (GW-BASIC command/statement), 97
Tandy 1000 TL
 features of, 1-3
 specifications for, 29
TandyLink board, adding, 20
time
 prompt, default setting for, 25
 setting, with time command, 49
time\$ (GW-BASIC command/statement), 97
timer (GW-BASIC command/statement), 98
trace (Debug command), 58
transfer lines (Edlin command), 53
tree (MS-DOS command), 49
troff tron (GW-BASIC command/statement), 98
troubleshooting
 equipment maintenance, 28
 printer problems, 27
 video problems, 27
type (MS-DOS command), 49
typewriter keys, 5

U

unassemble (Debug command), 58
unlock (GW-BASIC command/statement), 79
upgrades. *See* external options; internal options.
usr (GW-BASIC command/statement), 98

V

val (GW-BASIC command/statement), 98
varptr and varptr\$ (GW-BASIC command/
 statement), 98-99
ver (MS-DOS command), 49
verify (MS-DOS command), 49
video adapter boards, 21
video display
 default setting for, 24
 troubleshooting, 27
video modes, in GW-BASIC, 102
view (GW-BASIC command/statement), 99
vol (MS-DOS command), 50

W

wait (GW-BASIC command/statement), 99
while ... wend (GW-BASIC command/
statement), 99
width (GW-BASIC command/statement), 100
window (GW-BASIC command/statement), 100
workgroups, connecting to, with TandyLink, 20
write
 (Debug command), 58
 (GW-BASIC command/statement), 100
write# (GW-BASIC command/statement), 100
write lines (Edlin command), 53

X

xcopy command, 13, 50

RADIO SHACK
A Division of Tandy Corporation
Fort Worth, Texas 76102