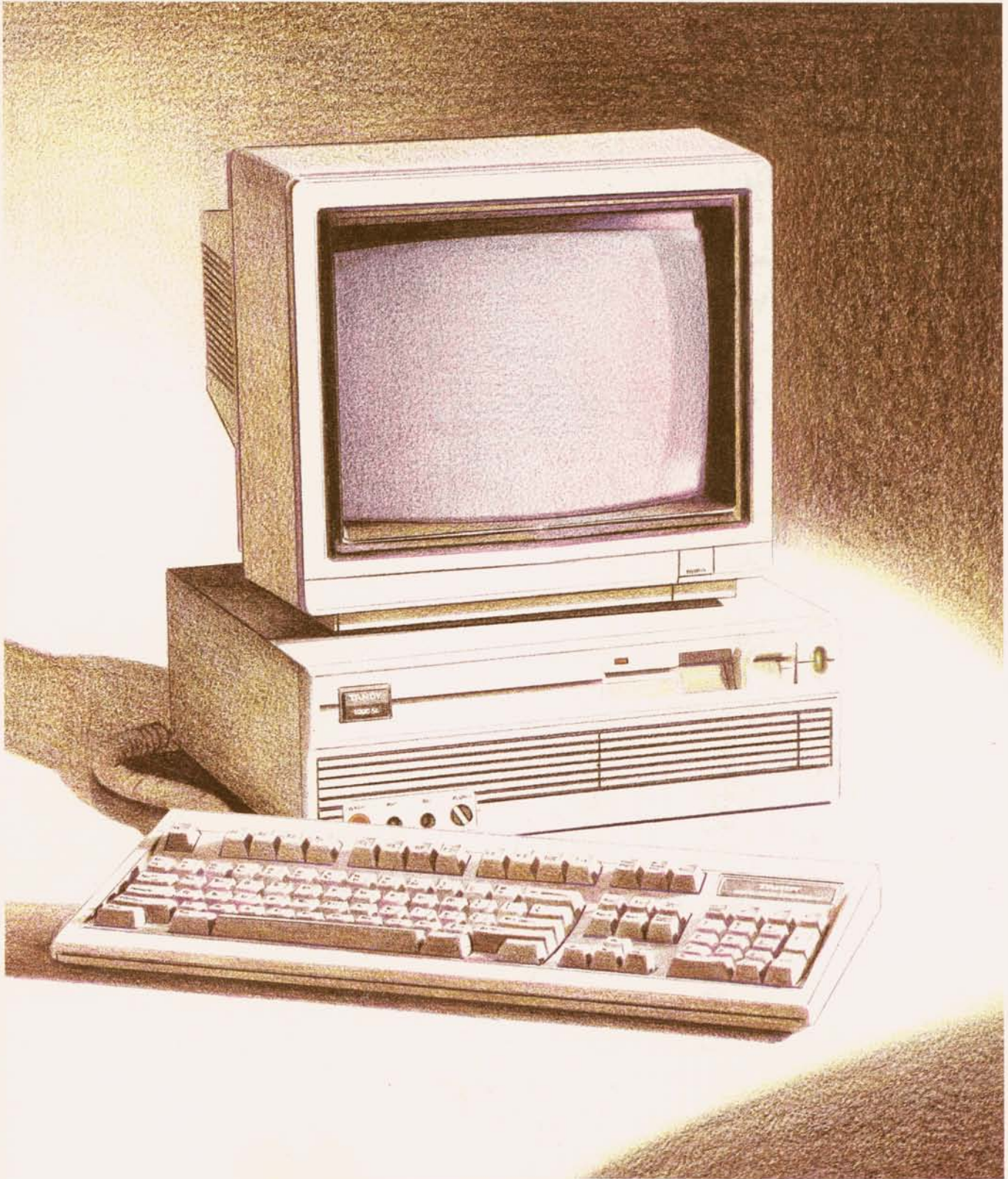


A Practical Guide to the **Tandy 1000 SL**



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.

12/88

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. The FCC certification of this computer requires the use of shielded cables when connecting peripheral devices to it.

12/88

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 SL**

Tandy 1000 SL 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 SL
©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.

Intel is a trademark of Intel Corporation.

Lotus is a trademark of Lotus Development Corporation.

CONTENTS

A Practical Guide to the Tandy 1000 SL

The Tandy 1000 SL 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 5 1/4-inch Diskettes	8
Write-Protecting 3 1/2-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 SL	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 SL Specifications	29
MS-DOS Quick Reference	31
MS-DOS Commands	33
Special Keys	50
Edlin Commands	52
Edlin Editing Keys	54
Debug Command Parameters	55
Debug Commands	56

GW-BASIC Quick Reference	59
 Loading GW-BASIC	61
 GW-BASIC Commands and Statements	63
GW-BASIC Function Key Settings	96
Typing Keywords Using the ALT Key	96
Exponential Notation and Numeric Precision Characters	96
Operator Precedence	97
Video Modes	98
Enhanced Graphics Color Selection	99
Error Codes and Messages	99

The Tandy 1000 SL 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 computer is quite special. It has a great deal of software built right in.

This chapter explains many of the features of your computer. 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	<p>Your computer uses the MS-DOS <i>operating system</i>. MS-DOS is software that controls the 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.</p> <p>Your computer, however, has the fundamental portion of MS-DOS in <i>read-only memory</i>. All you have to do is turn on your computer and it is ready to go. (Your computer retains the contents of ROM, even when you turn it off.) Access to many of the commonly used MS-DOS functions is available without having to use an MS-DOS diskette. (Even when turned off, your computer retains the contents of ROM.)</p>
DeskMate in ROM	<p>DeskMate is software that provides a collection of useful and fun application programs in a user-friendly environment.</p> <p>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.</p> <p>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.</p>
Speller in ROM	<p>Your computer also includes a spelling checker built into the ROM. Because it is built-in, the spelling checker is extremely fast. You can use the spelling checker for DeskMate applications as well as for other ASCII files.</p>

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

The *central processing unit* is the brain of your computer. This is where it processes information. The CPU's dual-speed processing lets you choose the appropriate speed (8 or 4 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.

**384K of RAM,
Expandable to 640K**

Random access memory is your computer's temporary memory. This is where programs, instructions, and information are kept when 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 640 kilobytes. (One kilobyte equals 1,024 bytes, or characters of information.)

Built-in Video Support

Your computer comes ready to connect to a color or monochrome monitor. You do not need to purchase a video adapter card. The color (CGA-compatible) video supports as many as 16 colors and provides resolutions up to 640 x 200 pixels. The monochrome video is compatible with the MDA text adapter and the Hercules video adapter, providing resolutions up to 720 x 348 pixels.

Special EEPROM Circuitry

A special *electronically erasable programmable read-only memory* gives your computer the ability to remember the way you want it to run.

The EEPROM stores various system configuration parameters, such as 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 360K, 5 1/4-inch
Diskette Drive**

Your computer uses diskette drives to read programs and information from diskettes and to store programs and information on diskettes. The 5 1/4-inch diskette drive lets you store 360 kilobytes of information on industry-standard 5 1/4-inch diskettes.

A Reset Button

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

Music and Sound

Your computer 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 recording, storing, and playing sophisticated 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.

Built-in Serial Port

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

Built-in Parallel Printer Port

Connect a 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, 8-bit, 10-inch Expansion Slots

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

MS-DOS/BASIC Diskettes

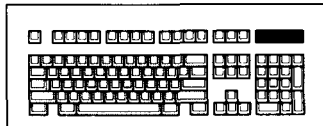
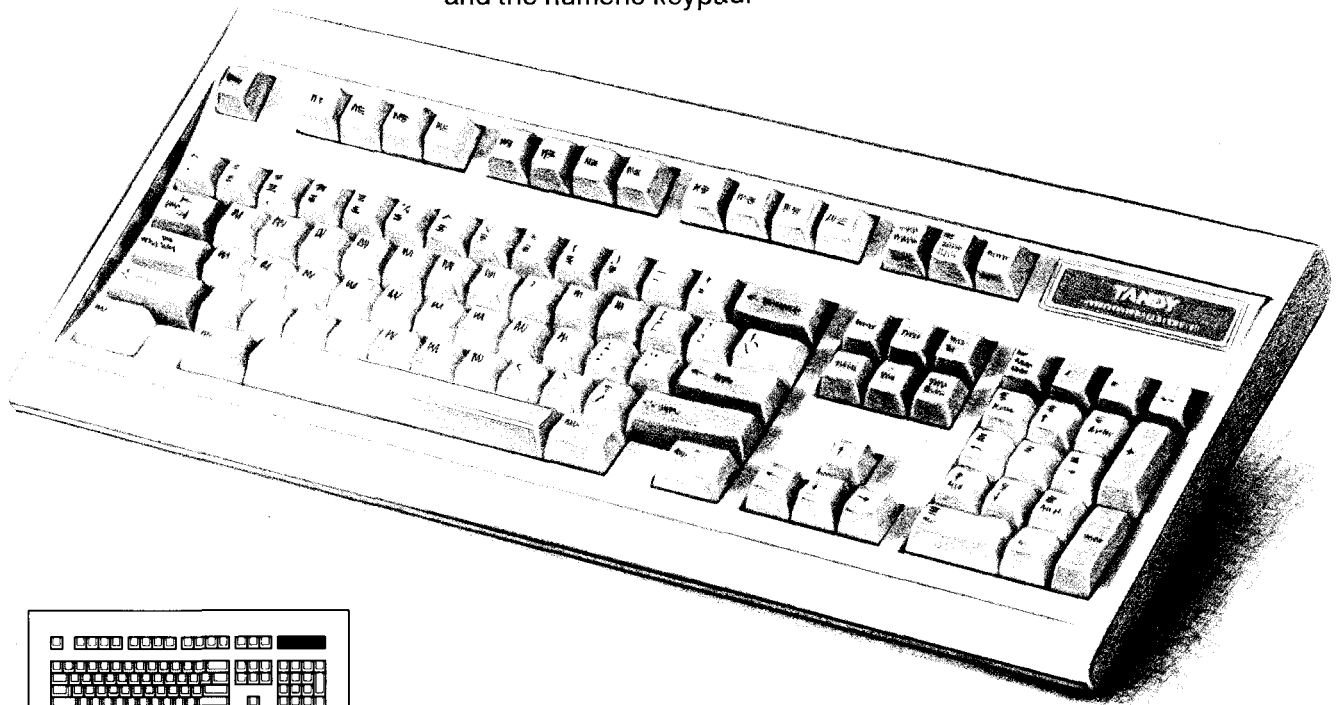
These diskettes contain the complete MS-DOS version 3.30 operating system and version 3.20 of the GW-BASIC programming language.

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.

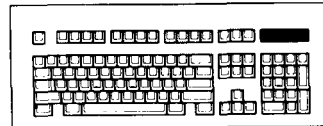
The Keyboard

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



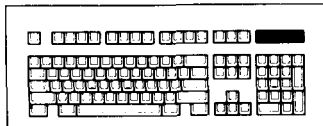
Typewriter Keys

These keys work in much the same manner as the keys on a standard typewriter. When you hold down a key, the keystroke repeats automatically.



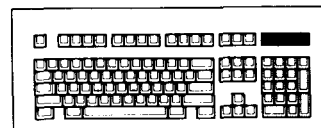
Function Keys

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



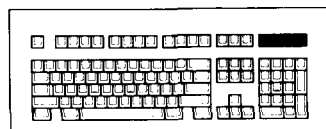
Cursor Keys

Many programs use the cursor keys to move 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 **NUMLOCK** to use the keypad for extensive number entry. When number lock is on, you can type numbers without pressing the **SHIFT** key.

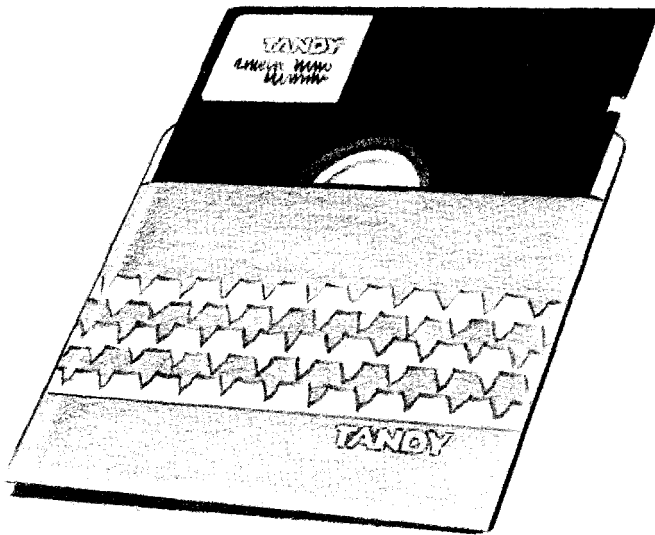


Special Function Keys

The 9 special-function keys are program-specific. Their functions depend on the program you are running.

Using Diskettes and the Diskette Drive

Care and Handling of Diskettes



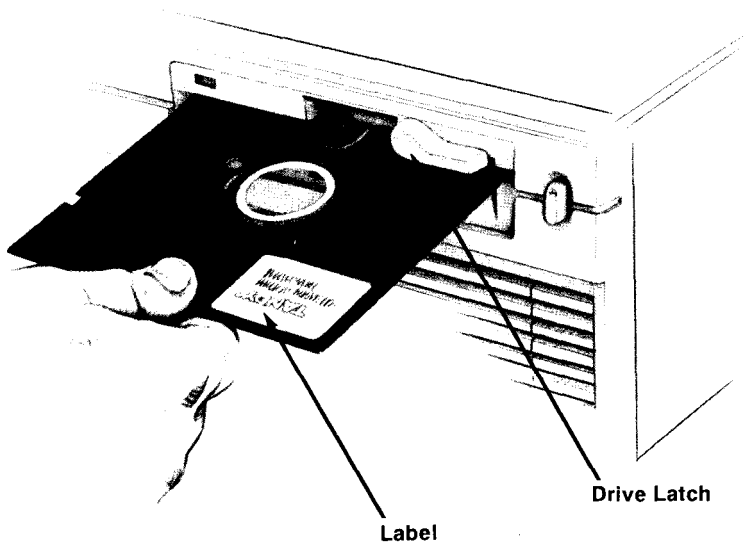
Diskettes store information, such as programs and the data you create, in *files*.

The diskette drive provided in your computer uses double-sided, 5 1/4-inch, 40-track diskettes (Cat. No. 26-411 and 26-412). These diskettes can store approximately 360 kilobytes (more than 368,000 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.
- Make several copies of your diskettes (*back-ups*) to use as working diskettes. Refer to "Making Copies" in this section.

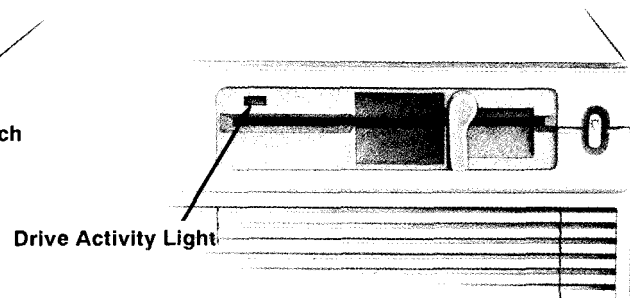
Inserting and Removing Diskettes



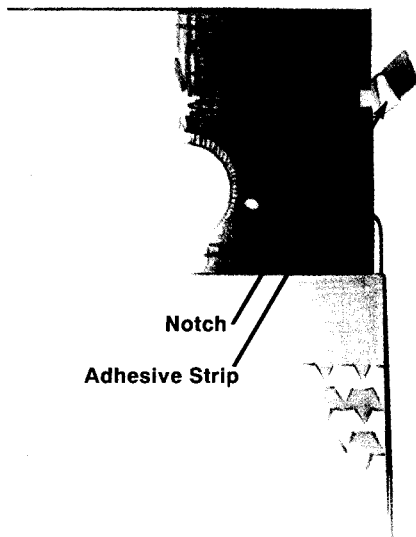
To insert a diskette into an empty drive, gently slide it, label side up, into the drive. Then, close the drive latch.

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, be sure the drive activity light is off. Then, release the drive latch and pull out the diskette.



Write-Protecting 5 1/4-inch Diskettes

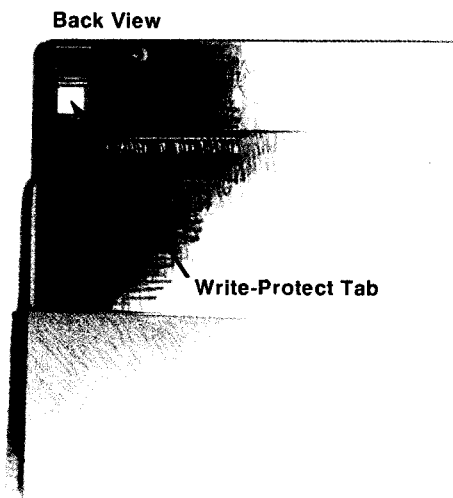


If you have important data on a diskette, you might want to make certain that your computer cannot erase or write over it. You can do this by *write-protecting* the diskette.

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 strip 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.

Write-Protecting 3 1/2-inch Diskettes



If you add a 3 1/2-inch diskette drive to your computer, you use a different method to write-protect diskettes for that drive.

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.

Note: Pressing F11 displays a menu from which you can restart the computer from either the top diskette drive (by pressing F1) or the bottom diskette drive (by pressing F2).

Using Program Diskettes

Your computer 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**.*

*If you want to use the built-in MS-DOS commands, type **C:** and press **ENTER** to change the current drive to the ROM.*

*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 much 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

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.

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*.

Entering MS-DOS Instructions

The MS-DOS instructions you give to the computer are called commands. You type commands at a *system prompt*, which indicates what drive you are using. You can type commands in either upper- or lowercase letters.

Changing Drives

Drive A (the top drive) is considered the current operating drive unless you specify otherwise. A second-disk drive is Drive B. The MS-DOS built into your computer's ROM is usually considered Drive C.

If you have more than one drive, you can easily change from one to another. For example, if you have two diskette drives, you can change the current drive to Drive B by typing `b:` and pressing **ENTER**. The system prompt changes from `A>` to `B>` and Drive B becomes the current drive.

Note: If you add a hard disk card, the hard disk is considered Drive C and the ROM then becomes Drive D. (Second and third partitions of a hard disk would be Drives E and F, respectively.)

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. Format is available in ROM or on the MS-DOS diskette.

Exit any application program you are using, and either change drives to ROM or insert the MS-DOS diskette into Drive A.

Type `format`, followed by a space. Type the letter of the drive that you will use for formatting, type a colon, and press **ENTER**. Insert a blank diskette into the drive you are using for formatting, and press **ENTER** again.

For example, if the ROM is the current drive and 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. Diskcopy is available in ROM or on the MS-DOS diskette. Be sure to either change drives to ROM or insert the MS-DOS diskette before using the Diskcopy command.

Using Diskcopy with One Disk Drive

At the system prompt, type:

```
diskcopy a: a:
```

Then, press ENTER.

Insert the diskette you want 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, 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

If you copy from a 3 1/2-inch diskette to a 5 1/4-inch diskette, the lower-capacity 5 1/4-inch diskette might not have room for all of the files. Use Xcopy to duplicate only selected files.

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.

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 SL

You can expand your computer'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 your computer 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.
A Serial Mouse	A mouse is a device that you roll on your desk 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 your computer's built-in serial port.
Joysticks	Joysticks can make many games and educational programs easier and more fun. The back panel of your computer 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	You can plug a microphone into your computer to record music and sound. 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 audio control satellite board from E1-E2 to E2-E3. Refer to the main logic board illustration later in this section for the location of this jumper.

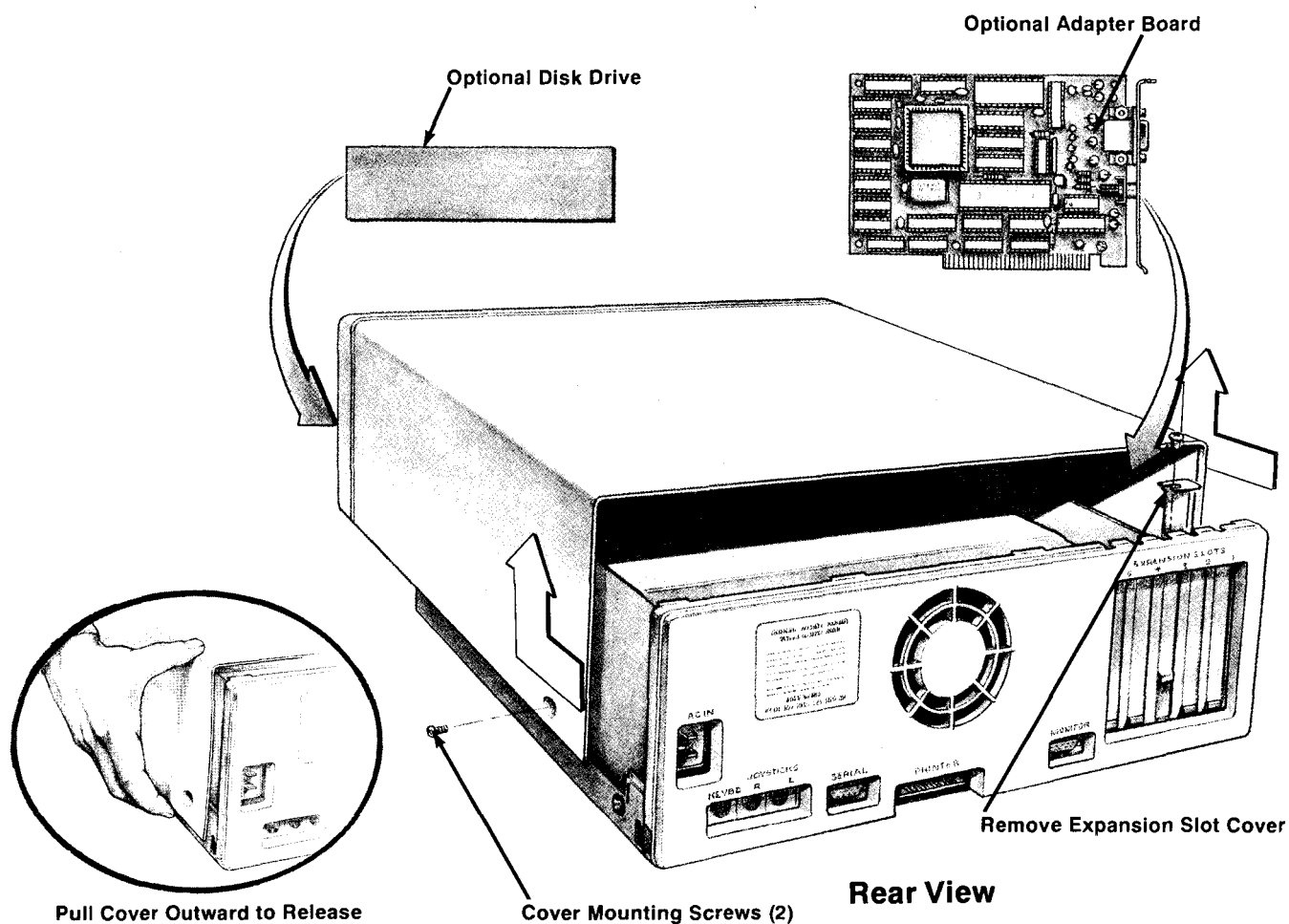
Adding Internal Options

If you want to handle the installations yourself, all of the add-on products for your computer 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.

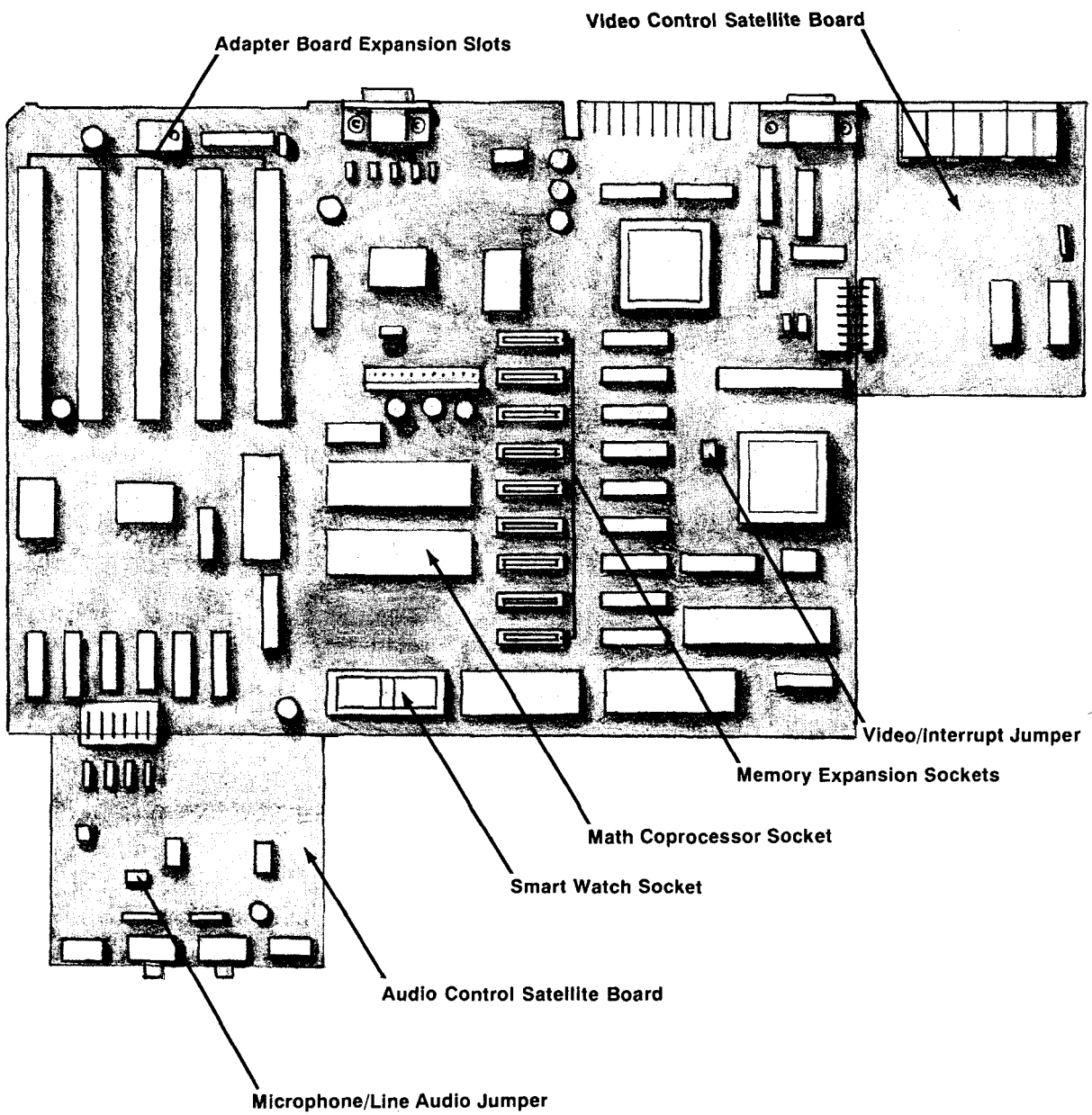
When adding internal options, do so in the following order:

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. Set any switches on the main logic board that need setting.
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:

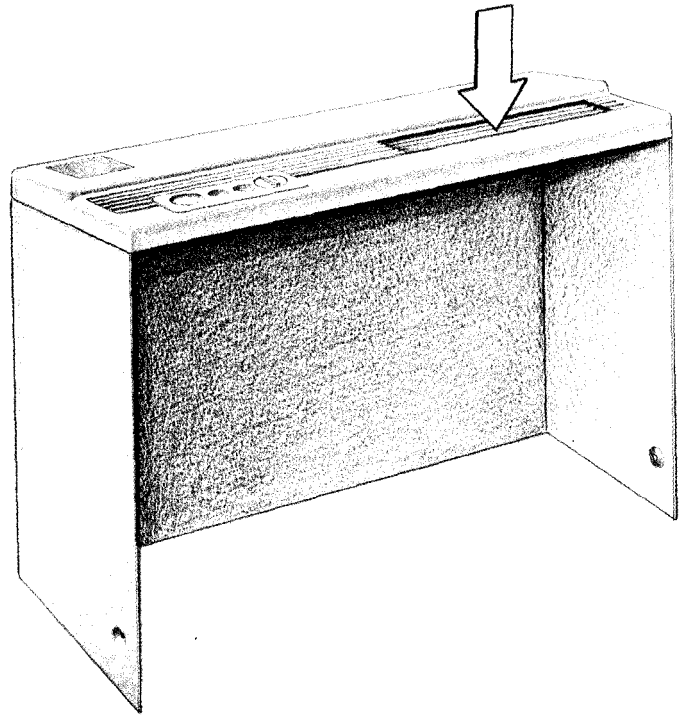


Main Logic Board



Adding a New Drive

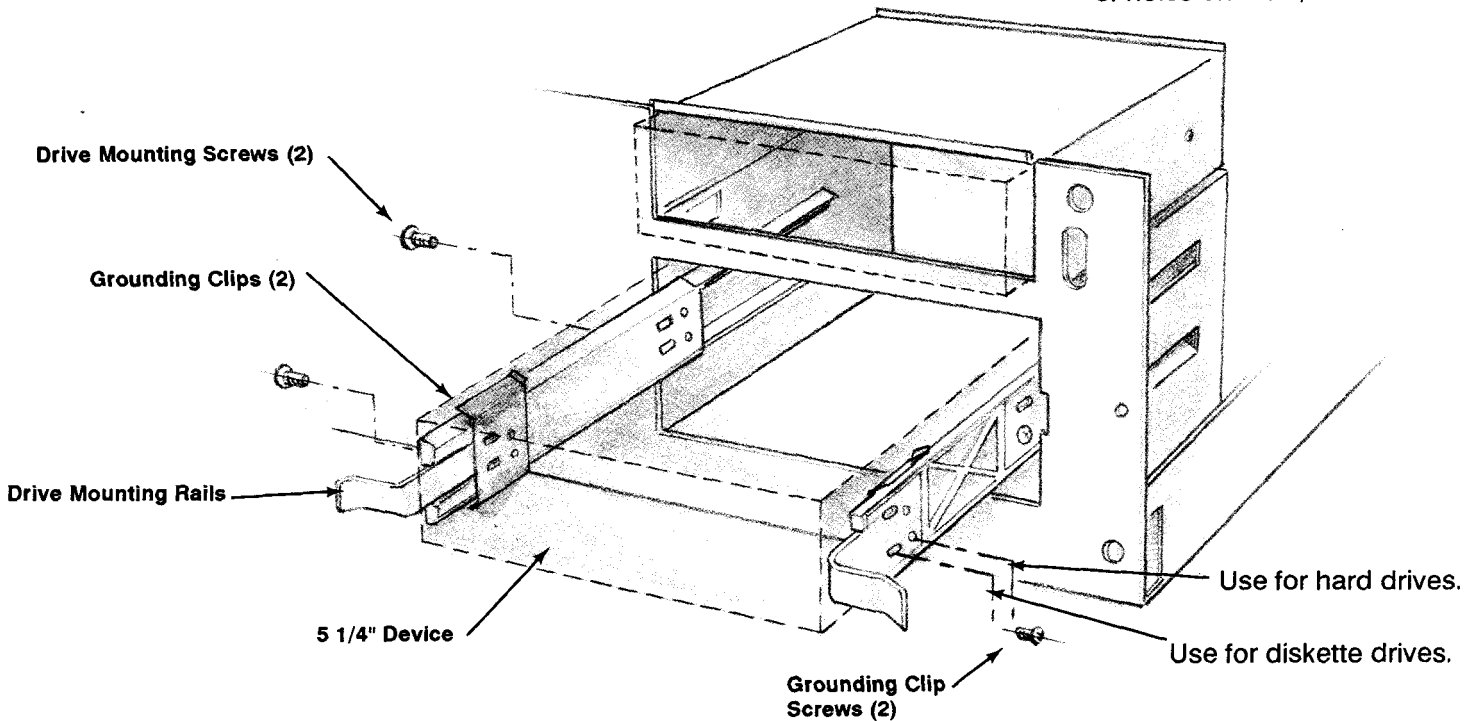
Push down on the lower panel to break it away.



Run Setupsl after you install a second diskette drive.

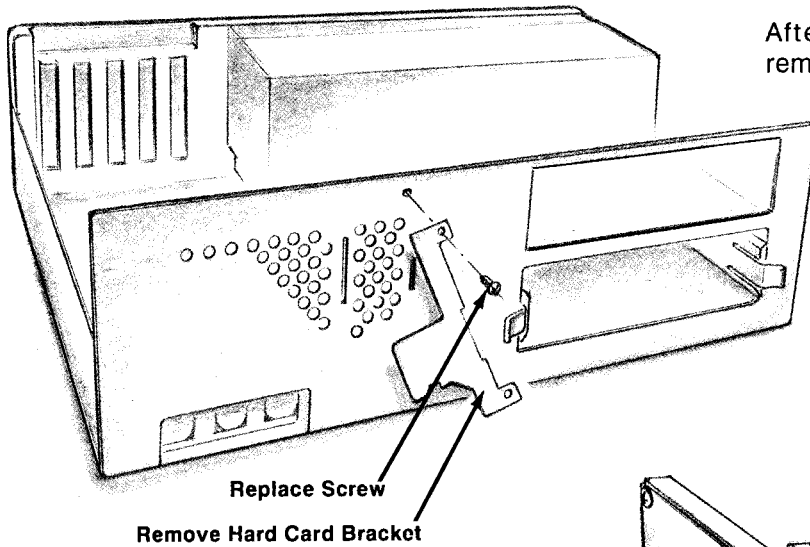
Installing a Secondary Disk Drive

Attach the rails to the lower set of holes on a 5 1/4" device.



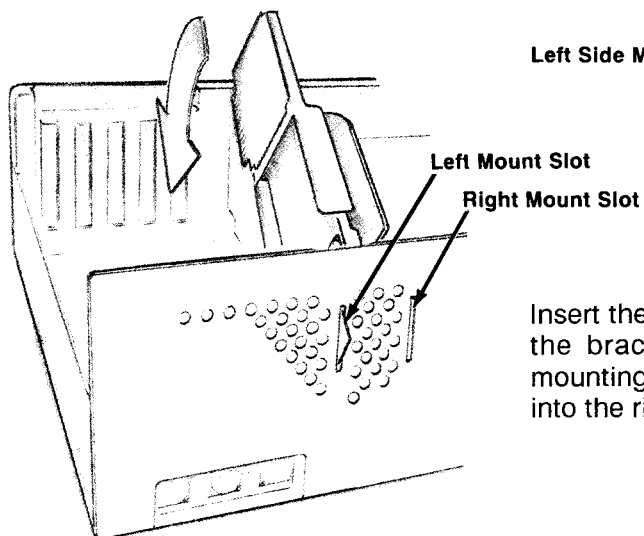
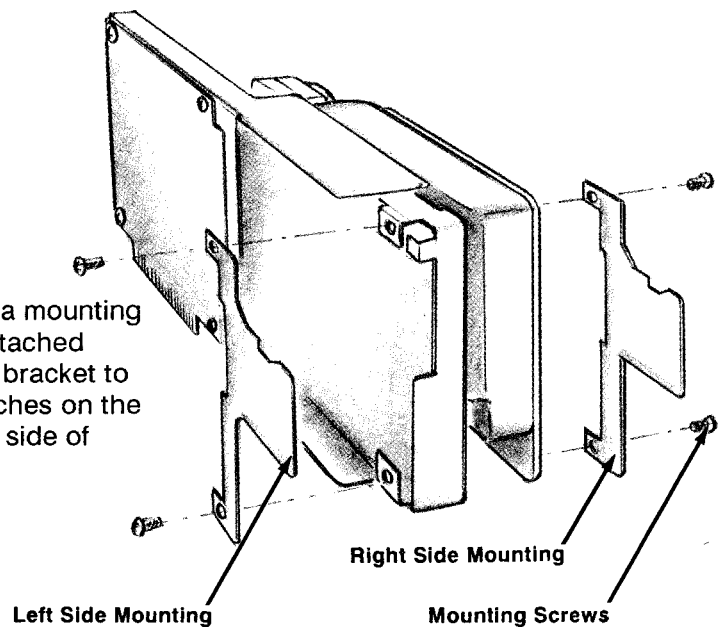
Hard Disk Card Installation

Front View



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

If the hard disk card is shipped with a mounting bracket attached to it, remove the attached bracket and attach the SL mounting 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	You can expand your computer to 640K <i>on-board</i> memory. Memory chips are plugged into sockets on the main logic board.
SmartWatch	SmartWatch is a perpetual clock and calendar on a chip. It has a built-in battery so that it can keep time even when your computer is turned off. The SmartWatch chip simply plugs into your computer's main logic board.
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 computer's main logic board.</p>
Diskette Drives	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. You can add either a 5 1/4-inch or a 3 1/2-inch internal drive to your computer.
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 of the 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. Refer to the illustrations earlier in this section.</p>
Expanded Memory	<p>Some programs are designed to use expanded memory in order to 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 computer lets you add as much as 2 megabytes of expanded memory.</p>
A Modem Board	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.

TandyLink

A 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.

Enhanced Video

With an enhanced video adapter board and a compatible monitor, your computer can generate high-resolution graphics. If the programs you run require color and higher resolution or enhanced graphics, you can install either an EGA or a VGA board in your computer.

There's More

This section discussed only a few of the options available for your computer. Other options you can add might include:

- A MIDI Interface Adapter for connecting musical synthesizers (keyboards) to your computer
- 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

Changing the Computer Defaults

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

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

You change the settings in the EEPROM using the Setupsl 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:
setupsl

Then, press **ENTER**.

In a moment the following Setupsl 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. 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 restart the computer in the correct video mode.

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

Select YES if you want the date and time prompt to appear when you restart (boot) the MS-DOS operating system. Select NO if your computer is equipped with SmartWatch. SmartWatch automatically sets and keeps the date and time when you start your computer. (see "The Tandy 1000 SL Features and Options").

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 start-up 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 uses 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 manuals 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, first be sure it is properly connected to a power source and that it is turned on. Then, be sure you have the cables properly connected to your computer. 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 the **CTRL-ALT-SHIFT-V** key sequence. Also, if you see horizontal wavy lines or if the screen rolls when you start the computer, use **CTRL-ALT-SHIFT-V** to restart the computer in the correct video mode.

If you attempt to run some game software designed for the 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 E3-E4. 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 of the conflicting device.

Also, ensure that 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 program'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.

Tandy 1000 SL Specifications

Processor: 8086, 8 or 4 megahertz

Size:

Length 337 mm (approx. 13.25 in.)

Width 394 mm (approx. 15.5 in.)

Height 140 mm (approx 5.5 in.)

Weight 7 kg (15.8 lb)

Power Requirements:

United States 120 VAC, 60 Hz

International Only 120 VAC / 240 VAC, 50Hz, 67 watts

Heat Output:

363 Btu/h

**Environment
Air Temperature:**

Operating 14°C - 30°C (55°F - 85°F)

Storage -40°C - 72°C (-40°F - 160°F)

Humidity:

Operating 20% to 80% (non-condensing)

Storage 10% to 80% (non-condensing)

Internal Disk Drive:

Media standard: 5 1/4-inch, double-sided, 40 track

Unformatted Capacity 500 kilobytes

Formatted Capacity 360 kilobytes

Number of Heads 2

Number of Cylinders 40

Average Access Time 93 ms (including settling time)
Track-to-track, 6 ms

Motor Starting Time 400 ms

Rotation Speed 300 RPM

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	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 the second key while holding down the first key.

Additionally, the commands in this section are labeled as either internal or external according to the following definitions:

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 access the disk and directory where these commands reside in order to 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. Omit all parameters to display 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.
;	sets the NUL data path; MS-DOS searches only the current 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 32 megabytes, 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.

<i>source pathname</i>	specifies the files to back up. It can be an entire drive, a directory name, or a filename.
<i>target drive:</i>	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 <i>filename</i> —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 the parameters.

```
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
/a

the size of the buffer in kilobytes. Default = 30K.
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 create the cache buffer, 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
```

MS-DOS Quick Reference

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 the **chkdsk** 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\jrrrrs
```

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 any of the following:

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.

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

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

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*.

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

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

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

(External) Compares the contents of two diskettes.

<i>drive1:</i>	the drive containing the source diskette.
<i>drive2:</i>	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.

```
diskcomp a: b:
```

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 (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" doc.txt document.txt
```

/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**.

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
```

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*.

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

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

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

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 the MS-DOS system files.

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
```

label is a character string.

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-PRINTSCRN**.

ptype

is one of these printer types:

cgp220 the Tandy CGP-220.
 dmp110 the Tandy DMP-110.
 pcmode a Tandy printer with a dip switch set for the PC mode or other PC-compatible printers.
 tmode a Tandy printer with the dip switch set for the Tandy mode.
 standard 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 background color as black (CGP-220 only).

/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-PRINTSCRN
```

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.

condition

can be one of the following:

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
not

the command to execute only if condition is met.
 executes the command only when *condition* is false.

```
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.

drive:
pathname

the drive you are joining.
 the empty path, including the drive, to which *drive:* is joined.

/d

turns off a previous **join** command.

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

keyb [xx [,yyy][,pathname]]

(External) Replaces the current keyboard BIOS with an international keyboard program. (Requires international drivers.) Displays the current keyboard code and its related code page as well as the current code page used by the console screen device (con) if you omit all parameters.

Press CTRL-ALT-F1 to return to the US keyboard. CTRL-ALT-F2 returns to the memory-resident keyboard program.

xx

can be one of the following:

us	United States	keyb us (default)
fr	France	keyb fr
gr	Germany	keyb gr
it	Italy	keyb it
sp	Spain	keyb sp
uk	United Kingdom	keyb uk
po	Portugal	keyb po
sg	Switzerland-German	keyb sg
sf	Switzerland-French	keyb sf
dk	Denmark	keyb dk
be	Belgium	keyb be
nl	Netherlands	keyb nl
no	Norway	keyb no
la	Latin America	keyb la
sv	Sweden	keyb sv
su	Finland	keyb su
cf	Canada-French	keyb cf

yyy

is the code page that defines the character set.

pathname

is the name of the keyboard definition file.

keyb gr

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.

n
chars
lines
p

specifies the printer number (1, 2, or 3).
specifies the characters per line (80 or 132).
specifies vertical spacing (6 or 8 lines per inch).
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 Lpdrv.sys device driver or execute the lf command.

mode lf off

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

(External) Sets RS232 communication parameters.

number
baud

parity

databits
stopbits
p

the RS232 serial port, either 1 or 2.
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.
can be n (no parity), o (odd parity), or e (even parity). The default is e.
can be either 7 or 8. The default is 7.
can be either 1 or 2. The default is 1.
tells the printer driver to continuously try to output on timeout errors.

mode com1:1200 N 8 1 p

mode speed

(External) Sets the CPU speed.

speed

can be one of the following:

slow	to reduce your computer's operating speed.
fast	to set your computer's speed to the default fast speed.

mode slow

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.

<i>device</i>	specifies the device to support code page switching. Valid device names are con, prn, lpt1, lpt2, and lpt3.
<i>yyy</i>	specifies a code page. Valid code pages are 437, 850, 860, 863, and 865.
<i>filename</i>	identifies the name of the code page information (.cpi) file MS-DOS should use to prepare a code page for the specified device.
<i>cpages</i>	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.

nlsfunc [[drive:][path][filename]]

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

<i>filename</i>	specifies the file containing country-specific information. The default value of <i>filename</i> 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.
-----------------	--

nlsfunc 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. Displays the current path setting if you omit *pathname*.

<i>pathname</i>	specifies a directory or an entire drive.
;	used without specifying a <i>pathname</i> , sets the path to "no path", which causes MS-DOS to search only the current directory.

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

pause [message]

(Internal) Suspends execution of the batch file.

<i>message</i>	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.

<i>pathname</i>	specifies the file(s) to print.
/d: <i>device</i>	specifies the print device. Lpt1 is the default.
/b: <i>size</i>	sets the size (in bytes) of the internal buffer. The default is 512 bytes.
/u: <i>value</i>	specifies the number of clock ticks print will wait for a printer. Default = 1.
/m: <i>value</i>	specifies the number of clock ticks (1-255) print can take to print a character.
/s: <i>timeslice</i>	specifies the interval of time to be used by the MS-DOS scheduler for the print command.
/q: <i>value</i>	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 <i>pathname</i> .)

```
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 current drive
g	>
l	<
b	
q	=
_ (underline)	a return-line feed
e	an escape code
h	a backspace

```
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.

<i>pathname1</i>	the pathname of the file to encrypt.
<i>pathname2</i>	the pathname of the file to receive the encrypted file. If you omit <i>pathname2</i> , 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.clic 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.

<i>source drive:</i>	specifies the drive that contains the backed up files.
<i>target pathname</i>	specifies the directory to which you want to restore the files.
<i>pathname</i>	specifies the disk directories and/or file you want to restore.
<i>/s</i>	restores the specified directory and its subdirectories.
<i>/p</i>	prompts for permission before restoring hidden or read-only files and before restoring any files changed since the last backup.
<i>/b:date</i>	restores only those files last modified on or before <i>date</i> (<i>mm/dd/yy</i>).
<i>/a:date</i>	restores only those files last modified on or after <i>date</i> (<i>mm/dd/yy</i>).
<i>/e:time</i>	restores only those files last modified at or before <i>time</i> (<i>hh:mm</i>).
<i>/l:time</i>	restores only those files last modified at or after <i>time</i> (<i>hh:mm</i>).
<i>/m</i>	restores only those files modified since the last backup.
<i>/n</i>	restores only those files that no longer exist on the target drive.

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

rmdir *pathname*

rd *pathname*

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

```
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. (Requires international drivers.)

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

```
select b: a: 049 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.

<i>string1</i>	any character string you want to replace.
<i>string2</i>	any replacement character string.

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

setupsl

(External) Initializes the system configuration.

```
setupsl
```

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 *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 subdirectories 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 file specified by *pathname*.

```
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 the parameters.

```
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.

source pathname
target pathname

specifies the drive, directories, and/or files to copy.
specifies the drive, directories, and/or file 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 waits before copying 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 <i>char</i>	Copy to Character. Copies all characters up to the specified character (<i>char</i>) 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 <i>char</i>	Delete to Character. Deletes all characters up to the specified character (<i>char</i>) 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 ENTER

Ends the Edlin program and saves the edited file.

e

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> template.	DEL	Deletes a character in the
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	voids 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)	Indicates the current line.
#		Indicates the end of a program.
+		References lines after the current line.
-		References lines before the current line.

Debug Command Parameters

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

<i>address</i>	<p>An alphabetic segment register and an offset, such as:</p> <p>cs:0100 A segment address and offset.</p> <p>04ba:0100 An offset only. (The default segment is cs for the go, load, trace, unassemble, and write commands, and ds for all other commands.)</p>																
<i>byte</i>	<p>A 2-digit hexadecimal value placed in or read from an address or a register.</p>																
<i>drive</i>	<p>A 1-digit value for the drive to be used for accessing or writing data, as follows:</p> <p>0 = Drive A 2 = Drive C 1 = Drive B 3 = Drive D</p>																
<i>pathname</i>	<p>A drive specification, filename, and extension. (The complete pathname is optional; however, you must specify at least the filename.)</p>																
<i>list</i>	<p>A series of strings or byte values. (<i>List</i> must be the last parameter.)</p>																
<i>range</i>	<p>An area of memory specified by:</p> <p><i>address1 address2</i> (<i>Address2</i> must be an offset.)</p> <p><i>address l value</i> (<i>Value</i> is the number of lines on which the command operates. The default is 80.)</p> <p>Do not use the <i>address l value</i> format if another hexadecimal value follows the <i>range</i> parameter.</p>																
<i>registername</i>	<p>One of the following registers:</p> <table><tr><td>ax</td><td>cs</td><td>bp</td><td>di</td></tr><tr><td>bx</td><td>ds</td><td>ip</td><td>si</td></tr><tr><td>cx</td><td>es</td><td>sp</td><td>pc</td></tr><tr><td>dx</td><td>ss</td><td>f</td><td></td></tr></table>	ax	cs	bp	di	bx	ds	ip	si	cx	es	sp	pc	dx	ss	f	
ax	cs	bp	di														
bx	ds	ip	si														
cx	es	sp	pc														
dx	ss	f															
<i>sector</i>	<p>A hexadecimal value (1-3 characters), indicating the relative sector number on the disk.</p>																
<i>sectorcount</i>	<p>A hexadecimal value (1-3 characters), indicating the number of disk sectors to write or load.</p>																

<i>string</i>	Any number of characters, enclosed in quotation marks (single or double).
<i>value</i>	A hexadecimal value (4 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** *value*Inputs and displays one item from the port specified by *value*.

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** *value byte*Sends the specified *byte* to the port specified by *value*.

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*][*/c*:*RS-232 receive buffer size*]
[*/m*:*highest memory location for BASIC, max. block 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 Config.sys file if you want a number other than the default. You must use the <i>/l</i> option when using the <i>/f</i> parameter.
<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>/m:location, block size</i>	loads GW-BASIC with the reserved memory specified by <i>block size</i> (block size x 16). GW-BASIC uses memory up to the memory <i>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>/s: record length</i>	sets the maximum length 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 GWBASIC 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:

bold	specifies a command, statement, or parameter that you must type exactly as it appears.
<i>lowercase italics</i>	represent variable names, letters, characters, or values.
[]	indicates that you can use optional parameters.
... (ellipsis)	indicates that you can repeat a parameter.
(vertical bar)	indicates an either/or situation.
SMALL CAPS BOLD	indicates a key combination, such as CTRL-C . Press the second key while holding down the first key.

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
```

BASIC Quick Reference

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**.

```
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.

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

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

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

bsave *pathname,offset,length*

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

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

offset 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.

length is the number of bytes to save (in the range 1-65535).

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

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

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

parameter list contains the variables that are passed to the external subroutine.

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.

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

clear [,*memory location*] [,*stack space*][,*video memory*]]

Frees memory for data without erasing the program in memory.

<i>memory location</i>	specifies the highest memory location available for GW-BASIC.
<i>stack space</i>	specifies the memory to set aside for temporarily storing internal data and addresses.
<i>video memory</i>	specifies the amount of video memory to set aside.

```
clear
clear ,45000
clear ,6100,300
clear ,,,32768
```

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.

<i>background</i>	specifies the background color.
<i>palette</i>	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.

<i>foreground</i>	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.
<i>background</i>	specifies the background color and can be colors 0 to 15 of the 16 Color Set.
<i>border</i>	specifies the border color. Border can be 0 to 15.

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

color [foreground][,background]

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 fnname [(argument list)] = expression

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

name is a valid variable name.
argument list is a list of dummy variables used in *expression*.
expression defines the operation to perform.

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

def seg[=address]

Assigns the specified address to be the current segment address.

address 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.

number is in the range 0-9. Default = 0.
offset is in the range 0-65535.

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

delete line1[-line2]

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

line1 defaults to the first line of the program.
line2 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 11%(8,25)
```

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

<code>u[n]</code>	Moves up <i>n</i> points.
<code>d[n]</code>	Moves down <i>n</i> points.
<code>l[n]</code>	Moves left <i>n</i> points.
<code>r[n]</code>	Moves right <i>n</i> points.
<code>e[n]</code>	Moves diagonally up and right <i>n</i> points.
<code>f[n]</code>	Moves diagonally down and right <i>n</i> points.
<code>g[n]</code>	Moves diagonally down and left <i>n</i> points.
<code>h[n]</code>	Moves diagonally up and left <i>n</i> points.
<code>mx,y</code>	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

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

```
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.

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.

```
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 one on the command line.

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.

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

eof(buffer)

Function. Detects the end of a file.

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 not. 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.

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

eof(buffer)

Communications. Detects an empty input queue for a communications 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.

```
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$
```

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
```

BASIC Quick Reference

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.

pathname 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 subroutines.

```
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 "a<>b"
```

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.

0-byte string = no key pressed
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*.

; immediately following **input**, echoes 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 #3346"
```

key on | off | list

Displays the function key assignment values.

on displays the key assignments on Line 25 of the screen.
off erases the key assignments.
list displays all 15 characters of the soft key assignments

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.

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*.

action can be on, off, or stop.
number 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).

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

len(string)

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

number is an integer in the range 1-255.

```
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: specifies the output device, scrn: (the screen) or lpt2: (the printer).

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

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

Lists program lines in memory to the printer. **llist** assumes a 132-character-wide printer. (You can change this using the **width** 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.

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

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

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)
```

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(5)<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"
```

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

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

start specifies the position of the first character you want to change.

length specifies the number of characters you want to replace.

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

BASIC Quick Reference

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

Returns a substring of *string*.

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.

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

mkdir pathname

Creates the directory specified by *pathname*.

pathname is a standard directory specification.

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

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 integer or single- or double-precision number.

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

name "old filename" as "new filename"

Renames *old filename* as *new filename*.

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

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

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 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.

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*.

buffer specifies the I/O buffer (1-16) 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 2-32767. 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.

```
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]][,data type] [,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.

data type can be bin to transmit binary data (default) or asc for ASCII data.

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 controls CTS (clear to send).

ds controls DSR (data set ready).

cd controls CD (carrier detect).

lf sends a line feed at each return.

pe enables parity checking.

seconds 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. 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.

In Screen Modes 7, 8, and 9, GW-BASIC uses 4 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 2 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.

(*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. 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.

```
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
```

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 4,2
```

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)
b = peek(23040)
```

play string

Plays musical notes specified by *string*.

string is a string expression consisting of one or more of the following music commands:

Music Commands

- a-g** plays a note (A-G) on 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. *n* can be a value in the range 1-64.
 - 1 specifies a whole note.
 - 2 specifies a half note.
 - 4 specifies a quarter note.
 - 8 specifies an eighth note.
 - 16 specifies a sixteenth note.
- on** sets the current octave (0-6). Octave 3 starts with middle C. Default = Octave 4.
- nn** plays a note. *n* can be in the range 0-84.
- pn** rests. *n* can be in the range 1-64.
- tn** sets the number of quarter notes in 1 minute. *n* can be in the range of 32-255. Default = 120 quarter notes in 1 minute.
- .** plays as a dotted note, increasing 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 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.
- vn** sets the volume. *n* can be in the range 0-15 (default = 8).
- xvar;** executes a substring represented by *var*. You can have one string execute another, which executes a third, and so on.

```
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 the voice channel (0, 1, or 2).

```
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.
- 1 returns the current physical y-coordinate.
- 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 and *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)
```

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.

format consists of one or more field specifier(s), or any alphanumeric character. Enclose *format* in quotation marks.
data can be a string, a numeric value, or both.

Specifiers for String Fields:

! prints only the first character in the string.
\spaces\ prints 2+n characters from the string. (n 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.
^ ^ ^ ^ 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.

Refer to **print using** for information about the *format* parameter and specifiers.

```
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.

```
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. If you omit the *number* parameter, GW-BASIC suspends program execution and prompts you for a number before executing.

```
randomize timer  
randomize 300
```

number can be an integer or a single- or double-precision number.

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**.

```
renum  
renum 600, 5000, 100
```

line is the program line at which renumbering starts. Default = the first line.

new line is the new number assigned to *line*. Default = 10.
increment numbers the successive lines in the specified increments. Default = 10.

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* | **next**]

Resumes program execution after an error handling routine.

line branches to the specified line number. Default = the statement in which the error occurred.

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"
```

rnd[(*number*)]

Returns a random number in the range 0-1.

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.

```
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*][,*r*]
run "*pathname*"[,*r*]

Executes the current program or loads and runs the specified program.

```
run
run 100
run "program.a"
```

line is the program line at which GW-BASIC begins execution. Default = the first line.
r leaves the current files open when the new program, specified by "*pathname*" is loaded into memory.

save "*pathname*" [,*a* | ,*p*]

Saves a program on disk with the specified name. If you do not specify a format, **save** uses the compressed format.

```
save "a:file.1.bas"
save "\educ\mathpak.txt",a
```

a saves the program in ASCII format.
p saves the program in a protected format.

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*][,*erase*]]]

Sets the screen mode and screen attributes to be used by all other graphics statements.

mode is an integer in the range 0-10 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.
erase indicates how much video memory to erase. *Erase* can be:

0	do not erase memory, even if the screen mode changes.
1	erase the union of the new and old pages if <i>mode</i> or <i>burst</i> changes (default).
2	erase all video memory if <i>mode</i> or <i>burst</i> changes.

```
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[,voice] | on | off

Generates a sound with the frequency and duration specified.

frequency is an integer in the range 1-1023, indicating the tone in hertz.
duration is a numeric expression in the range 1-65535, specifying the duration in clock ticks.
voice identifies the defined sound as voice 0, 1, or 2.
on enables the external speaker.
off disables the external speaker.

```
sound 37,2
```

space\$(number)

Returns a string of *number* spaces.

```
print "Cost" space$(4)
"Quantity" space$(9) "Total"
```

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)
```

BASIC Quick Reference

stop

Stops program execution.

```
stop
```

str\$(number)

Converts *number* to a string.

```
s$ = str$(x)
print str$(-234)
```

strig(number)

Returns the status of the 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.

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.

action can be on, off, or stop.

```
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.

```
usr("100")
```

BASIC Quick Reference

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$(a$)
```

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.

```
wait 32,2
```

number1 an integer in the range 0-255.
number2 an integer in the range 0-255.

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
```

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 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). 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-right 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 = 1
Video Page Size = 16384

Screen Mode 3

Graphics = Yes
Resolution = 160 x 200
Color Set = monochrome
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 = 20 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.
Colors = 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 (8 shades of 8 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	Number	Message
1	NEXT without FOR	29	WHILE without WEND
2	Syntax error	30	WEND without WHILE
3	RETURN without GOSUB	50	FIELD overflow
4	Out of DATA	51	Internal error
5	Illegal function call	52	Bad file number
6	Overflow	53	File not found
7	Out of memory	54	Bad file mode
8	Undefined line number	55	File already open
9	Subscript out of range	57	Device I/O error
10	Redimensioned array/duplicate definition	58	File already exists
11	Division by zero	61	Disk full
12	Illegal direct	62	Input past end
13	Type mismatch	63	Bad record number
14	Out of string space	64	Bad file name
15	String too long	66	Direct statement in file
16	String formula too complex	67	Too many files
17	Can't continue	68	Device unavailable
18	Undefined user function	69	Communication buffer overflow
19	No RESUME	70	Disk write protect
20	RESUME without error	71	Disk not ready
21	Unprintable error	72	Disk media error
22	Missing operand	73	Advanced feature
23	Line buffer overflow	74	Rename across disks
24	Device timeout	75	Path/File access error
25	Device fault	76	Path not found
26	FOR without NEXT	77	Deadlock
27	Out of paper	78	Unprintable error

RADIO SHACK
A Division of Tandy Corporation
Fort Worth, Texas 76102