

TopSpeed®

For IBM® Personal Computers and Compatibles

User's Guide

TopSpeed Corporation

Copyright© 1990-1991, by TopSpeed Corporation. All rights reserved.

TopSpeed® is a registered trademark of TopSpeed Corporation.

Other brand or product names are trademarks or registered trademarks of their respective holders.

Printed in the United Kingdom.

10 9 8 7 6 5 4 3 2 1

Contents

CHAPTER 1

INTRODUCTION **7**

Your New TopSpeed Package	7
The TopSpeed Editor	8
Environment Utilities	8
Using this Guide	8
Typographic Conventions	9
How this Manual is Organized	9

CHAPTER 2

GETTING STARTED **11**

What you will need	11
Installing TopSpeed	11
Installing for the first time	11
Upgrading a previous version of TopSpeed	12
Extending TopSpeed	13
Starting TopSpeed	13
Entering the TopSpeed Environment	13
Your First Program	14
Loading a File	14
Compiling and Running a Program	15
Leaving TopSpeed	15

CHAPTER 3

THE TOPSPEED ENVIRONMENT **16**

The TopSpeed Environment	16
Overview	16
Starting TopSpeed	18
Command Line Options for TS	18
The Help System	19
Help Line	20
The Menu System	20
Moving Between Menus	21

Shortcut Keys	22
Windows in the Environment	24
Zoom/Unzoom	25
User Dialog Windows	25
The Files Menu	27
The Editor	30
The Editor Menu	30
Pop-Up Menus	30
Loading a File	31
Viewing a File in Read Only Mode	31
Saving a File	32
Maximum File Sizes	32
Removing a File from an Editor Window	32
Moving Between Editor Windows	33
Editor Commands	33
Insertion and Deletion	33
Cursor Movement	35
Block Commands	36
Editor Options	37
Search and Replace	38
Other Editor Commands	39
Compiling and Running Programs	39
Compiling	40
Compilation Errors	40
The Project Name	41
Making a Program	41
Running Programs	42
Run-Time Errors	42
The Project System	42
Project Menu	43
The Options Menu	46
Run Options	47
Setup Options Menu	48
The Utilities Menu	50
C - Calculator	52
Base Conversion Keys	53
The Redirection File	56
Cut and Paste	58
Cutting Text	58

Pasting Text	59
Keyboard Macros	59
Recording and Playing Back Macros	60
Saving and Loading Keyboard Macros	61
Keyboard Macro File Format	61
Mouse Control	63
Mouse Actions	63
Customizing the TopSpeed Environment	64
Changing Windows	65
Repositioning Windows	65
Resizing Windows	65
Recoloring Windows	66

CHAPTER 4

VISUAL INTERACTIVE DEBUGGER	67
Introduction	67
What VID Will and Won't Do for You	68
Installing VID	69
Starting VID	70
Making Your Program with Debug Information	70
Running VID	70
Running VID from Within TopSpeed	71
Introduction to VID Keys, Menus and Windows	72
Giving Commands in VID	72
Expressions, Operators and Variables in VID	73
Operators	73
Variables	75
VID Command Line Options	76
The Source Window	78
Commands for Moving in the Text	78
Commands for Moving in the Running Program	79
Commands for Handling Windows and Screens	80
Commands for Exiting VID	81
Additional Commands for Moving in the Source File	82
Highlighting and Marking Lines	82
Assembler Display	83
VID's Menu Structure	84
VID's Main Menu	84
The Breakpoint Commands	86

The Breakpoint Menu	86
The Examine and Display Commands	89
Variable Type Modifiers	90
Objects in VID	91
Examine Menu	91
The Display Menu	94
The Go Commands	94
The Go Menu	94
The Trace Windows	96
The Trace Menu	97
Watch Variables	99
Commands for the Source Window	101
The Source Menu	101
User Options	103
The Options Menu	103
Default Options	105
Customizing VID	106

CHAPTER 1

INTRODUCTION

The TopSpeed Development Environment is designed to make it easy, quick and convenient for you to produce your own software applications on your personal computer. The TopSpeed Environment seamlessly integrates the full range of TopSpeed Languages, allowing you to write your program in any combination of languages. This true *Multi-Language Development* facility is unique to TopSpeed.

The TopSpeed Environment also contains many special functions and features especially designed to make your programming life easier and more productive.

Your New TopSpeed Package

The TopSpeed package is a complete development system that forms part of the TopSpeed Multi-language Development Environment. The package comprises of:

- A Language compiler
- A state-of-the-art optimizing code generator
- A multi-window programmer's editor and development environment
- A comprehensive Language run-time library
- The fully automatic TopSpeed Project System and Smart Linker
- TopSpeed VID: the Visual Interactive Debugger

All the languages supplied by Jensen & Partners International fit into the TopSpeed environment as overlays that are dynamically linked at runtime.

Furthermore, all the languages use the same optimizing code generator to produce fast, compact, executable code. TopSpeed automatically selects the appropriate compiler (or assembler), according to the type of the file you wish to compile. Since the system is designed to share the code generator and the environment, it is possible to add new languages to TopSpeed, simply by adding a new dynamic overlay.

Under TopSpeed, parts of an application can be written in several different languages allowing you to use the special features of any or all of the TopSpeed Languages and language libraries. All the hard work of matching up the various parts is done for you by TopSpeed. You can, for example, make full use of the extensive TopSpeed ANSI C libraries from within your *Modula-2*, *Pascal* or *C++* programs.

The TopSpeed Editor

The TopSpeed Editor allows you to edit up to ten files in different ‘windows’ on your screen simultaneously. Each window can contain a different file or another part of the same file.

The editor has numerous useful features to make life easier for you, including:

- Search and replace facilities.

- Program syntax checker for Modula-2 and Pascal.

- Comprehensive block functions.

- Cut and paste.

- Full context sensitive hypertext help system.

Environment Utilities

TopSpeed also provides a range of useful tools and utilities, including an ASCII table, a calculator, text cut and paste, multi-file search and keyboard macros.

In addition to the environment, your TopSpeed software contains an extensive set of libraries for the language(s) you have chosen. These libraries contain numerous ready made functions for you to use in your programs. These procedures make it easier for you to write useful, advanced programs in the quickest possible time.

Using this Guide

This guide is intended for two groups of people:

- People who have used TopSpeed before, in a previous version, and

- People who are completely new to TopSpeed.

The new TopSpeed environment for Version 3 is different in appearance, but similar in operation, to previous versions of TopSpeed. Previous users of

TopSpeed will find all of their favorite features still present, together with many new and improved features.

Typographic Conventions

The following typographic conventions are used throughout this manual:

<i>Italics</i>	are used for emphasis and to introduce new concepts
Typewriter	is used for program examples and other items which would appear on the screen or in a program
Boldface	is used to highlight key letters in menus, such as P in Project
Keyboard	is used to show keys (like F1 or Enter) which you press at the keyboard
<i>bold italics</i>	are used in syntax descriptions

When it is necessary to show a combination of keys the following convention is used:

`Alt-C`

This means hold down the `Alt` key and press `C`. When a sequence of keys is combined with `Alt` or `Ctrl`, the following convention is used:

`Ctrl-K D`

This convention means:

Hold down the `Ctrl` key and press the `K` key.

Release the `Ctrl` key and the `K` key.

Press the `D` key.

Where example screen outputs are shown, the illustration is boxed.

How this Manual is Organized

This guide is organized into a series of chapters, each dealing with one particular aspect of the new TopSpeed environment. This section describes each of these chapters, explaining the content and intended audience. Everyone should read Chapter before doing anything else.

Chapter 1

This chapter. This contains an overview of the scope, contents and intended audience of this guide. This chapter also acts as a

general introduction to your TopSpeed system and the other manuals which make up the document set.

Chapter 2

explains how to install TopSpeed onto your computer.

Chapter 3

introduces and explains how to use all the features of your TopSpeed environment and describes how to use TopSpeed to write, compile and run your own programs.

Chapter 4

describes how to install, configure and use the TopSpeed Visual Interactive Debugger (VID).

CHAPTER 2

GETTING STARTED

This chapter explains how to install your TopSpeed package onto your computer, and gives a very quick introduction to starting and using the standard TopSpeed environment.

What you will need

In order to install TopSpeed successfully you will need the following;

- Hard disk with 4MB free
- 640KB Internal RAM
- Any proprietary display type
- Mouse - optional (MS compatible)

Installing TopSpeed

This guide comes complete with a set of floppy discs, in either 5 $\frac{1}{4}$ " or 3 $\frac{1}{2}$ " format. These discs contain all the files necessary for you to install your TopSpeed Environment. The programming language(s) you have chosen are supplied on a separate set of discs, packaged with the documentation for that language.

The TopSpeed discs you have been given are in a specially compressed format. You *should not* attempt to copy files directly from these distribution disks onto your computer: Use the TopSpeed installation procedure described below.

Installing for the first time

TopSpeed must be installed using the following procedure:

1. Place the floppy marked *Installation Disk* (Disk 1) into drive A.
2. Log on to the A drive by typing A:

3. At the prompt, type `INSTALL` and press *Enter*.
4. You are then asked for a directory name where you wish to install TopSpeed. The installation program will place all files in sub-directories under the directory you specify. When you have specified the name of the directory to place TopSpeed you should press *Enter*.
5. A list of `Install` options is now displayed which specifies which file should be installed. You should go through these options selecting those components that you wish installed by using the cursor keys to move up and down, and the space bar to select options.
6. When you are satisfied with the setup, press *F2* to start copying the files to your hard disk. You will be told if there is not enough disk space to install the files you have requested, and allowed to quit installation.
7. When this operation is complete, you are prompted whether you have any other disk sets to install (Other Languages, SourceKits etc.). If you have, respond *Y* and place the Installation Disk (Disk 1) of the new set into drive A:. You then should repeat from step 5.
8. After installing all the sets of disks that you have and saying *N* to 'Any more disk sets?', the installation program runs an automatic system configuration. This ensures that the languages installed are correctly configured.
9. `INSTALL` may display further instructions when you finish the installation. Carefully read this information. You may be told to edit your `AUTOEXEC.BAT` and/or other system files in order to correctly support TopSpeed. Modify the files as indicated.
10. Reboot your computer if you have changed system files.

Upgrading a previous version of TopSpeed

You should follow the same procedure as described above, being sure to specify the same directory that you last installed TopSpeed. (Unless of course you wish to keep both versions installed, in which case you should specify a *different* directory from your previous version.)

During installation you may be prompted:

1. Whether you wish to overwrite the previous versions of TopSpeed files. You should answer *Y* to continue installation.
2. Whether you wish the `INSTALL` program to back-up any changed text files that would become overwritten by installation. These files are moved to the sub-directory `./SAVE`.

Extending TopSpeed

You can extend the TopSpeed system at any time simply by installing a new set of disks. In this way you can add new Languages whenever you wish. The TopSpeed INSTALL program automatically re-configures your system to include any new items installed.

When installing new files (for example, a new memory model), you must ensure that they are installed from both the language and environment disks.

Starting TopSpeed

You are now ready to start and use your TopSpeed environment for the first time.

First, start TopSpeed by entering the following command at the prompt:

TS

This is the command you will use to start to TopSpeed Environment.

If you get a message such as "file not found", check that your PATH is set up to include the TopSpeed system directory (/SYS for DOS and /OS2_SYS for OS/2. LIBPATH must also be set up for OS/2).

Entering the TopSpeed Environment

The TopSpeed start-up screen and welcome message are now displayed.

The screen displayed consists of three main areas:

- The first line of your screen is used to display the *Main Menu*.
- The center of your screen is dominated by the TopSpeed welcome banner. This describes the version of TopSpeed that you are using. The banner is removed as soon as you press a key.
- The bottom line of your screen displays a brief help summary of keys currently available.

You can perform various operations by pressing special keys, but for now the only ones you need to know about are:

Alt - X

This exits TopSpeed, returning you to your normal system prompt.

F1 This opens the TopSpeed Help System. This system contains information on all the operations and commands you can use in TopSpeed.

Your First Program

You have now installed and started TopSpeed. Let's go one step further and create and run an example program. Since the TopSpeed system is Multi-Language, you can create programmes in any of the languages you have purchased. Here we shall use Pascal as the example.

Loading a File

To load a file, you need to select the **File** option from the Menu.

You can select an option in a number of ways;

- by moving the menu bar with the cursor keys and pressing *Enter*,
- by pressing *Home* which moves the menu bar to the first item of the menu, and
- by pressing *End*, which moves it to the last
- by holding down the Alt key and pressing the relevant letter of the function you wish to use.

Alternatively, you can select an option by pressing the highlighted letter.

Choosing **File** displays the *File Menu*; select the **Load** option from this menu.

This 'pops-up' a window, listing all 9 editor windows. Normally, this list would show the names of the files most recently edited in those windows. However, since you have only just started using the system, all the names are shown as '*.*'.

We are going to load a file called HELLO.PAS into window number 1. To do this type:

```
HELLO.PAS
```

and press *Enter*. The first editor window opens with an empty window as the file did not previously exist.

Now you can type in the following program text :

```
Program Hello(output);  
Begin  
  WriteLn('Hello TopSpeed')  
End.
```

You have now have created the text of your program and you can save it to disk by pressing *F2*.

Compiling and Running a Program

Compiling and running this program is very easy; select the **R**un option from the menu, or press *Alt-R* (hold down the *Alt* key and press *R*).

This displays a box asking for a *project name*. The purpose of this is explained later in this guide. For now, type `HELLO` and press *Enter*. A window now appears showing the progress of the compilation. When the compilation is complete, the program runs automatically.

When the program ends, after displaying “Hello TopSpeed”, you are prompted to return to the environment by pressing *ESC*.

That completes your whirlwind tour of TopSpeed. The following chapter fully describes the TopSpeed Environment and all the functions it provides.

Leaving TopSpeed

To leave TopSpeed hold down the *Alt* key and press *X* at the same time.

CHAPTER 3

THE TOPSPEED ENVIRONMENT

The TopSpeed Environment

In this chapter, you will learn about the TopSpeed integrated development environment. You'll find brief summaries of the major environment features, the commands, and the options available in the TopSpeed Environment.

The first part of the chapter discusses general features of the TopSpeed Environment, including;

- how to get started,
- the help and menu systems,
- the use of windows in the environment.

The core of this chapter summarizes the commands available under the system menus. The last portion of the chapter describes how you can customize the TopSpeed Environment to suit your needs and tastes.

Overview

The TopSpeed Environment is a multi-windowed, integrated development system. Within the environment, you can enter and edit your program, compile the text into executable code, and then run that code on your computer.

You can do all of this without leaving the environment and without long or complicated command sequences. The environment will also detect errors when you compile or run your program and will take you to their exact position within the editor.

The environment is designed especially for developing programs that contain more than one *compilation unit*. Nine editor windows are provided, so that you can view and edit up to nine separate files simultaneously. Thus, you can enter the text of a *source file* using one editor window, while viewing relevant *header files* in other windows. In addition to the nine user windows,

the environment provides an extra *error editor window* in which compile or run-time errors are pointed out in the source text.

To ensure that each component module of a program is compiled and up-to-date, the environment provides an automatic Make facility. Thus, if any part of a program is changed, all modules affected by that change are automatically recompiled and a new executable file is generated.

The menu system and key setups are fully configurable to your own personal preferences by editing a simple text file (TSCFG.TXT). This allows you to change the text and structure of the menu tree, and the shortcut keys that invoke environment functions. You can even assign your own programs to menu and key commands.

The multi-windowed editor is also fully configurable; you can easily make it resemble your own favorite editor. It is initially configured to be WordstarΣ-compatible, familiar to users of Turbo PascalΓ or SideKickΣ, although with the ability to edit up to ten files simultaneously (including the *error editor window*) and with a much larger file capacity. As well as editing multiple files in multiple windows, the environment also allows you to edit the same file at different positions in the text. For example, you could view definitions at the top of the file in one window, while editing another part of the file in another.

Whenever you leave the environment, the current environment state is recorded, including which files are being edited and the position within each file. Also the window layout and all user options are saved. When you reenter the environment it will automatically start up from where you left it, allowing you to continue immediately. (If you do not want to continue your previous session, you can use the /N command line option when starting a TopSpeed session, as described below.)

The hierarchical directory systems of DOS and OS/2 can be very helpful in keeping your files in order. For example, you might put all your files with an .OBJ extension into a separate directory; you might also keep all your source files in a separate directory.

In the TopSpeed Environment, you can assign *search paths* specifying the directories in which various types of files can be found. With this useful feature, you can keep all files of a given kind (for example, .OBJ files or .LIB files) in a particular directory. In this way, you can organize your filing system tidily and consistently, without having cluttered work directories.

To set up the intended location of files, you need to edit the text file that defines file to directory redirection (TS.RED). See “*The Redirection File*,” page for more details.

Starting TopSpeed

To start up the environment, enter the command TS at your DOS or OS/2 prompt. The environment will start, first clearing the screen, then displaying the TopSpeed title banner, and then the main menu.

Files used by the environment include the following:

TS.EXE	TopSpeed Environment loader
TSMAN.DLL	Main TopSpeed Environment program
TSDATA.OVL	Environment configurable data
TSMAN.HLP	Environment help file

In addition, there are a number of .DLL and .HLP files associated with each compiler and assembler supplied with your system (e.g., for *TopSpeed C++*, TSCPP.DLL and TSCPP.HLP; for *TopSpeed Pascal*, TSPAS.DLL and TSPAS.HLP).

All of the above may be located in the start-up directory, i.e., the directory in which you keep TS.EXE.

In addition, the following files may also exist:

TS.RED	Redirection file (see “ <i>The Redirection File</i> , page).
TS.SES	Session file (system created, see below)
TS.CFG	Default configuration (see “ <i>Save Options/Windows</i> ,” page)
TS.MAC	Default keyboard macro file (see “ <i>Keyboard Macros</i> ,” page).

TS.SES contains details of your last session in the environment. If this file is present when you type TS, the environment resumes in the same state as when you left. This is true unless you use the /N option, in which case the system ignores the TS.SES file, and a new session starts instead.

Command Line Options for TS

The following options may be placed after TS when starting up the TopSpeed Environment:

/N Start a new session, for example:
 ts/n

file Load file(s) into editor(s) on entry. For example, to load

TEST.CPP into editor 1 and PWIN.ITS into editor 2,
enter:

```
ts test.cpp pwin.its
```

/1-9 Sets editor to load file into the editor indicated by the
number. For example, to load TSCFG.TXT into editor
9, you enter:

```
ts /9 tscfg.txt
```

/L Sets environment to large-screen mode (EGA 43-line
mode; VGA 50-line mode). For example:

```
ts/L
```

The Help System

Press F1 to get *context-sensitive* on-screen help at any time while within the TopSpeed Environment. You will get specific information related to your current location in the environment. You can also move around within the help system to look at other topics of interest.

If you are editing program text and the cursor is currently located on a library identifier (e.g., `printf`), then pressing F1 will display help for that function. Similarly, if you are in the error editor window on a line that contains an error, then F1 will display extra information on the error.

Within the help system, individual help topics are highlighted and you can select them by moving the inverse bar, known as the *help bar*, to the required topic (using cursor keys) and then pressing Enter. When More is displayed at the bottom of the window, there are further pages on the current topic. These can be displayed by pressing PgDn. To redisplay previously selected help pages, you can press PgUp, which will page through the last 60 pages viewed. See Table -1 for a summary of key functions when within the help window.

To reenter the help system at the position that you left it last, press the shortcut key Alt-H.

Command	Operation
Enter	Selects the topic. This will display the help page for the topic that is currently highlighted at the help bar.
¼ ; ~	Moves the help bar in the appropriate direction to the desired topic.
PgDn	Displays the next page of help information if More is displayed.
PgUp	Redisplays the previous page viewed.
F1	Displays the help index for the current help file.

Alt-K	Cuts text from the help screen (see “ <i>Cut and Paste</i> ,” page).
Esc	Closes the help window and exits the help system. Alt-H can be used to reenter at the point last exited.
shortcuts	Global environment shortcut keys (F10, Alt-1, etc.) will exit the help system while leaving the help window visible on the screen. This allows you, for example, to both edit and view help information at the same time.

Table -1: Keys in the Help System

Help Line

The help line is a single line displayed on the bottom line of the screen showing a summary of key functions for the current context. This information can help you use an unfamiliar function without resorting to the help system or user’s manual.

You can disable the display of the help line by setting **Options (Q)Help line** to off. See “*Options Menu*,” page for details.

The Menu System

You can use pop-up menus to invoke all the facilities of the TopSpeed Environment. This is one of the two major ways in which you can invoke TopSpeed commands; the other being through the use of shortcut key sequences. Generally, you need only a few keystrokes to find and activate any function through menus. The menu system is fully reconfigurable to your own requirements (see “*Customizing the Environment*,” page).

Although you may have several menus visible on your screen at any given time, only one of these menu windows will be active at a time. The *active menu* has a double frame and highlighted command characters.

Menu commands always act upon the active menu. You can select a command by moving to the command you want, and then pressing **Enter**. As you move, your current line will be in inverse video. This inverted line is known as the *menu bar* and is used to identify the command you wish to select. You can also select a command by typing the highlighted character in the command’s name. This form of the command will override the menu bar selection.

Moving Between Menus

The TopSpeed menu system is very detailed and comprehensive. The structure is organized as a tree, with the *main menu* at the top, and various submenus branching off from this.

Several keys are useful for moving through the menu structure (see Table -2).

Command	Operation
i	Moves the menu bar up one line.
-	Moves the menu bar down one line.
Enter	Opens a submenu if one exists, or activates the command at the menu bar if there is no submenu.
Esc	Closes the current menu.
Home	Moves the menu bar to the first line.
End	Moves the menu bar to the last line.
letter	Pressing a letter that is highlighted in the active menu moves the menu bar to the line containing that letter, then activates the associated function or submenu.
F10	Returns to the main menu from wherever you are in the menu system. This is also the key that activates the main menu from outside the menu system, for example, if you are in an editor window.

Table -2: Menu Movement Commands

In addition to the above, there are also *global shortcut* keys that you can use to invoke menus and commands. These are independent of the menu windows and can be used even with no menu active on the screen. For example, in the preceding list, F10 is the shortcut key that invokes the main menu. Shortcut keys are discussed in the next section of this chapter.

Thus, there are three ways to activate any environment function:

- Use the menu to select the required function and press Enter.
- Type the character that is highlighted in the line for the desired command.
- Press the appropriate shortcut key (if one exists for this command).

Shortcut Keys

Many frequently-used functions have predefined shortcut keys assigned to them. Some of these commands are function keys, and others consist of a key prefixed by *Alt*. You can change or add to these commands by editing the configuration file, TSCFG.TXT (See “*Customizing the Environment*,” page). See Table -3 for a list of the default shortcuts.

Command	Operation
Alt-X	Exits the TopSpeed Environment, returning to the DOS or OS/2 command line. This is equivalent to the F iles Q uit menu command. Note that before you exit, you will be given a chance to save any edited text. The state of the environment is also saved.
Alt-E	Invokes the editor. If you’re editing in more than one editor window, then Alt-E will take you to the editor window you last used. This is equivalent to invoking the E ditor command from the main menu (see “ <i>The Editor</i> ,” page for more details).
Alt-1 - Alt-9	Invokes editor windows 1-9.
Alt-0	Invokes the error editor window, provided it is active.
Alt-C	Compiles a single file.
Alt-M	Makes an up-to-date executable program or library, compiling and linking as necessary.
Alt-R	Makes an executable program, and then runs it.
Alt-L	Links together an executable program or library without compiling.

The above four commands are equivalent to the main menu commands **C**ompile, **M**ake, **R**un and **L**ink, which provide the interface to the TopSpeed Project system. (See “*Compiling and Running Programs*,” page for more details).

Command	Operation
Alt-D	Creates a DOS or OS/2 shell in which you can execute external commands and programs. On exit from the shell (by typing exit), the environment will be restored to its original state. This is equivalent to the F iles D OS shell command (see “ <i>DOS Shell</i> ,” page).
Alt-A	Invokes the ASCII table.
Alt-B	Invokes the calculator.
Alt-I	Displays the information window.

F2	Save the file in current window.
F3	Load file into current window.
Alt-F3	Load file from pick list.
F4	Invokes the file searching utility.
Alt-F4	Retrieves the search pick list generated by the previous file search.
F5	'Zooms/Unzooms' the current editor window. If the window is already at full size, pressing F5 returns the window to its original size, and vice versa.
Alt-F5	Reviews the DOS or OS/2 screen. This allows you to review the last output produced by a program or DOS shell. This is useful for examining the results of a program after you have returned to the environment. On entry to the environment, the review screen is loaded with the screen previously visible. Compile, Make and Link all automatically clear the review screen, ready for the next time you run a program.
F6	Cycles through the currently open editor windows, making each active in turn.

Table -3: Default Shortcut Keys

Menu Shortcut Keys

There are a number of menus that have shortcut keys assigned to them, allowing you to invoke them with a single keystroke. These are listed in Table -4 below.

Command	Operation
F10	Invokes the main menu.
Alt-F	Invokes the F iles menu (see page).
Alt-P	Invokes the P roject menu (see page).
Alt-O	Invokes the O ptions menu (see page).
Alt-S	Invokes the S etup menu (see page).
Alt-U	Invokes the U tilities menu (see page).

Table -4 Menu Shortcut Keys

Special Keys

A number of keys are reserved for special uses in the TopSpeed Environment. These enter or exit special modes which are not generally available from the menu system. These special keys are described in full detail in Table -5.

Command	Operation
Esc	Closes the current window, menu or command, terminating its operation.
Ctrl-Break	Aborts a compilation or make. Also terminates the playback of keyboard macros.
ScrollLock	Enters Window Control Mode (see “ <i>Changing Windows</i> ,” page).
Alt-W	Alternate method for entering Window Control Mode.
Alt =	Macro Definition Key. Starts the definition of a keyboard macro. (see “ <i>Keyboard Macros</i> ,” page).
Alt -	Macro Prefix Key. Key to prefix two keystroke macros.
Alt-K	Enters Window Text Cutting Mode, see “ <i>Cut and Paste</i> ,” for further details.
Alt-J	Pastes text into an editing window.

Table -5 Special Environment Keys

Windows in the Environment

All interaction between the user and the TopSpeed Environment occurs within overlapping windows. These windows can be dynamically positioned, sized and colored according to your taste and requirements.

Generally, each window is associated with a single task or function. For example, an editor window allows you to edit text, while the compiler window will contain messages from the compiler.

Each window is bounded by a frame. This, together with the coloring of the window, indicates where each window begins and ends, even when they overlap. The only exception is when the window has been zoomed using the F5 key (see below) to cover the whole screen without a frame.

Whenever you are in the environment, there is always a single active window. As mentioned above, this window contains the task currently being performed — whether it involves an editor, a compiler, a menu or any of the

other environment functions. You can always identify the active window by its double-lined frame, as opposed to the single frame of an inactive window.

The overlapping windows of each function are stacked on top of one another, according to the order in which the functions were invoked. The top window in the stack is always the active window. The active window is never obscured by other windows.

Throughout the environment, `ESC` always removes the active window from the screen and makes the next window in the stack active.

See “*Changing Windows*,” page , for details on how to reposition, resize, and recolor your windows.

Zoom/Unzoom

You can make each of the editor windows as large as the entire screen by pressing `F5`. This “zooms” the window in so that the entire screen is now filled. Zooming removes the right and left edges and the bottom frame edge (except in the `error editor` window where the bottom line is reserved for error messages). Zooming is convenient for full screen editing, allowing the maximum area possible for showing text.

Note: Even when an editing window is zoomed, the help line will still be visible unless `Options (Q)Help` line has been turned off. See “The Options Menu,” page for details.

You can “unzoom” by pressing `F5` again. This will restore the window to its original size. The zoom state of each window is saved in the configuration and session files.

User Dialog Windows

When the environment needs some input from you, it opens a dialog window. These windows have three forms:

- Single line input
- Multiple line input
- File selection

Single Line Input

Whenever it needs a one-line response, the environment opens a single line input window; for example, when prompting for a filename. This input window contains a description of the expected input (e.g.,

Filename:), followed by an editing field in inverse video. Type your response in this field, and then press Enter to indicate that you have finished your input. Table -6 shows the keys you can use when editing such a response.

Command	Operation
Ctrl-S	Character left
Ctrl-D	Character right
Ctrl- or Ctrl-A	Word left
Ctrl-¼ or Ctrl-F	Word right
Backspace or Ctrl-H	Delete character left
Del or Ctrl-G	Delete character right
Tab or Ctrl-I	History (previous fields entered)
~ or Ctrl-X	Pick file (on file selection prompts)
Ins or Ctrl-V	Toggle Insert/Overwrite
Home	Start of field
End	End of field
Ctrl-T	Delete word
Ctrl-Y	Clear field

Table 3-6: Editing Commands for Dialog Windows

When the editing field first appears, either the default text or the last text entered is already in the field. To accept this text, just press Enter. To edit the entry, use the cursor keys, and to clear the entry, just type over it.

The editing field may be larger than the input window. In this case, the field will scroll to the right or to the left during editing.

To abort input, press ESC. This closes the input window and aborts the function requiring input.

To retrieve a previously entered field, the Tab key cycles through the prompt *history*, i.e. the previous text entered within this field.

Multiple Line Input

Whenever the environment needs more than one line of text (for example, the Search command in the editor) or requires you to choose from several alternatives (for example, **Load file**), it opens a multiple line window with

several input fields. You can select these fields by using `Y` and `↓` (or `Ctrl-E` and `Ctrl-X`). You can then edit the fields, just as above. Again you can press either `Esc` to abort the function, or `Enter` to accept the input and continue.

File Selection Window

Whenever the environment prompts for a file to read (for example, in **Load file**), you can enter a wild filename. A wild filename is one that contains one or more of the characters `'?'` or `'*'`. The `'?'` character is used to match any single character, and `'*'` is used to match any sequence of characters. When a wild filename is entered, a file selection window is opened. Such a window has one of two formats, as shown in Figure -2. The wide format, shown at the top of the figure, contains just filenames and extensions, arranged with several entries on each line. The detailed format displays one file or directory entry per line and includes various types of information about the entry.

When being prompted for a single file, `↓` and `Y` can be used to cycle through *PickList* entries in order to select a previously used filename.

You can toggle between the two formats; just press the spacebar.

To select a file, move the menu bar to the required filename and then press `Enter`. To move the bar, you can use either cursor movement keys or press the first letter of the file required. Subdirectory entries and the parent `(. .)` directory are followed by `'\'` and highlighted. To move into a subdirectory (or the parent) you should move to the directory using the cursor keys or `'\'` and then press `Enter`. This will open a new directory window and allow you to select a file from here. You can press `Esc` to cancel the file selection.

The Files Menu

The **Files** menu contains commands to load and save files, to view directories, to change the current directory, to execute a DOS or OS/2 command or shell, and to exit from the TopSpeed Environment. In this section, we'll summarize the files menu commands.

Load File

Shortcut F3 loads a file into one of the nine editor windows and then opens that window for editing.

When **Load file** is selected, the load file window (as shown in Figure -3) is opened.

To load a file, first use the `Y` and `↓` keys to select the editor window you want. Then type the name of the file to load and press `Enter`.

If you used wildcards in your filename, a file selection window is opened (see “*File Selection Window*,” page). Simply move the cursor to the desired file and press `Enter`. Once a file is selected, it is loaded into the editor window and you can start editing. If the file requested does not already exist, a new file is automatically created.

Pick File

Shortcut `Alt-F3` allows you to pick the name of a file to load from a `pick list` of recently loaded files, as shown in Figure -4. The cursor is also restored to the point where you last edited the file.

To pick a file, move the selection bar to the file required and then press `Enter`. The file is loaded into the active editor ready for you to continue at the point you last edited the file.

The pick list is automatically saved in the session file and will be loaded when re-entering the environment. Use `Ctrl-Y` to delete an entry within the pick list.

Save File

Shortcut `F2` saves the file currently being edited to disk. This command is equivalent to the **Save file** command (`Ctrl-K D`) that can be issued from the editor (see “*Saving a File*,” page).

All Save

Saves all files being edited that have been changed since they were last saved. This command is a convenient way of ensuring that all changes you have made are saved to disk. This command is equivalent to the **All save** command (`Ctrl-K A`) that can be issued from the editor.

Change Dir

Changes the current directory. The command prompts for a new directory and displays the current directory name ready for you to edit. If you enter a blank directory name, a selection window opens, allowing you to select a new directory by moving the cursor.

Files Dir

Displays a directory listing of files after prompting for a file mask. All files in the specified directory that match the specified mask (e.g., `*.PAS`, `*.CPP`, `*.C`, `*.MOD`) will be listed. If you don't specify a path, then matching files from the current directory will be displayed.

There are two forms of the directory window: the **detailed directory**, which displays the size and last modification date of each

file, and the `short directory`, which only displays the filename. You can toggle between the two displays by pressing the spacebar.

To display other directories, you can move the selection bar to the required directory and then press `Enter`. The new directory is then displayed.

DOS Shell

Shortcut `Alt-D` enters a shell from which you can execute DOS or OS/2 commands and run programs. To exit the shell you must type `exit`. If running under DOS, you will then return to the environment at the point where you left. When running under OS/2, the DOS shell starts up a separate session that is independent of the environment. To switch back to the environment, you should use the OS/2 Session Manager.

If you would like all edited files to be automatically saved before entry to the DOS shell (for example, if you will need to look at these files in the shell), you should have the **Options Setup Auto save files** option set (see “*Auto Save Files*,” page).

Execute

Allows you to enter a single line DOS or OS/2 command, and then immediately return to the environment. This is a convenient way of quickly executing a program without leaving the environment.

As with the DOS shell, you can use the **Options Setup Auto save files** option to ensure that all edited files are saved.

Note: If you often need to execute a particular command or program, you may want to place the command directly in the environment menu. Then, you can run the program by menu selection or by shortcut key.

Quit Environment

Shortcut `Alt-X` leaves the TopSpeed Environment and returns to the DOS or OS/2 command line prompt.

If you have set the **Auto save files** option, your files are saved automatically. Otherwise, the environment will ask whether you wish to save edited files before exiting.

The environment also saves details of your session in the file `TS.SES`. The environment then uses this file to restart your work in exactly the same state the next time you invoke `TS`. The information saved includes the window layout and coloring, all options set, the files loaded in each editor window, the current project, and the contents of the pick list are read.

You can delete the session file if you want to ‘forget’ the previous session and start anew. You can also accomplish this by starting your current session using `TS/N`.

The Editor

The TopSpeed Editor (shortcut: `Alt-E`) is a multi-windowed editor in which you can edit up to nine text files simultaneously. The editing commands are initially configured to be WordstarΣ compatible, but you can customize them easily by altering the `TSCFG.TXT` file (see “*Customizing the Environment*,” page).

The following description of commands and facilities available within the editor assumes the supplied default configuration.

The Editor Menu

You can invoke editing functions through shortcut key sequences or through pop-up menus. Whenever you are in one of the editor windows, `F9` will invoke the **E**ditor menu.

From this menu, you can invoke editor functions in the same way as in the main menu. The editor menu tree is illustrated in Figure -5.

Pop-Up Menus

If you can only remember part of a longer editor command, just enter that part. After a short delay, editor submenus will automatically pop up. For example:

- Pressing `Ctrl-K` pops up the **B**loc**K** menu
- Pressing `Ctrl-Q` pops up the **Q**uick menu
- Pressing `Ctrl-O` pops up the **O**ptions menu

If you enter the complete key sequence (e.g., `Ctrl-K D`), the command will be executed immediately. The menu will not pop up in this case.

Loading a File

You can load a file in four ways:

- Using the **Load** file command on the **Files** menu, or the shortcut F3. This is described in “*Load File*,” page .
- Using the **Pick** file command on the **Files** menu, or using the shortcut Alt-F3. This is described in “*Pick File*,” page .
- Invoking the **Edit** command on the main menu, or using the shortcut Alt-E. This invokes the active editor window (initially 1).
- Entering one of the shortcut keys Alt-1 - Alt-9, to open editor windows 1 through 9, respectively (see “*Moving Between Editor Windows*,” page).

Whenever you invoke an ‘empty’ editor window (i.e., one that has not previously been used), you are asked for the name of the file to edit. If you enter the name of a file that does not already exist, the file is created.

You can load a new file into an editor window by pressing F3, or by selecting the **Load new file** command on the **Editor** menu (F9). If you were already editing a file in the window, you are asked whether you want to save changes to that file before the new file is loaded.

When prompted for a filename, you can use the ↓ key to cycle through the filenames that have previously been loaded. This allows you to select and reload or to edit these filenames.

The error editor window is loaded whenever you edit a file as a result of errors produced by compiling or running a program.

If the same file is loaded into two or more editor windows, then this file is shared between the windows. Any edit performed in one window will be reflected in all other windows sharing the same file. This powerful feature allows you to edit a file at more than one position simultaneously. The error editor window is also shared with any of the other windows that contain the file being edited.

Viewing a File in Read Only Mode

As well as loading a file to edit, you can also load a file just to view in `read only` mode by using the **Files View file** command (shortcut Shift-F3).

Once loaded, all editing functions are available, except those that change the file. Thus, you can view the file without the possibility of

inadvertently changing it. To indicate that an editing window is in read only mode, **Read Only** is displayed on the top line of the editing window in place of the **Insert/Overwrite** indicator. To toggle read only mode on/off, you can use the command **Options Read only** (shortcut **Ctrl-O R**) on the **Editor** menu.

Saving a File

You can save the file being edited to disk by doing any of the following:

- Pressing **F2**.
- Typing the command **Ctrl-K D**.
- Selecting the **Save file** command on the **Editor** menu.

You can also save to a file with a different name by selecting the **Write** to command from the **Editor** menu. TopSpeed always ensures that you never unwittingly lose changes made to a file by either prompting on exit from the environment or, if the **Auto save files** option is set, by automatically saving the file (see “*Auto Save Files*,” page).

Whenever you save a file to disk, a backup is created containing the previous contents of the file. If you don’t want to save such a backup version, set the **Number of backups** editor option to 0.

Maximum File Sizes

The editor is a disk-paged editor and can, therefore, edit files with total size larger than the available RAM. Individual files being edited can be up to 1000K characters long provided the total of all files loaded is less than the disk space available on drive that contains the `swap file` and the total size being edited does not exceed 5000K. The swap file used to hold temporary data is called `TS. $$$` and is created in the directory where you started `TS`. This file should not be deleted during an editing session, but will automatically be deleted on exit from the environment.

The environment will warn you if you approach either of the two limits. At that point, you will not be allowed to increase the file size.

Removing a File from an Editor Window

The command **Ctrl-K Q** will stop editing the file in the currently active editor window and will close that window. If the file has unsaved edits, you will be asked whether you wish to save these changes.

Moving Between Editor Windows

You can use the shortcut keys Alt-1 - Alt-9 to move to editor windows 1 - 9, respectively. In addition, you can move to the error editor window, if it is active; just press Alt-0.

You can also move between editor windows in sequence. Press F6 to cycle through any open editor windows.

Editor Commands

The TopSpeed Editor provides a rich variety of commands. Each command can be invoked by a key sequence or by a menu selection. As mentioned above, you can customize all keys and menus to your own preferences (see “*Customizing the TopSpeed Environment*,” page).

The initial setup, described below, resembles WordstarΣ, with some useful extensions to assist with program editing.

Insertion and Deletion

Table -7 summarizes the keys that enable you to insert or overwrite characters and to delete characters, words, and lines.

Command	Operation
Ins or Ctrl-V	Toggles Insert/Overwrite.
Enter or Ctrl-M	Inserts line below.
Tab or Ctrl-I	Inserts a number of spaces or a hard tab characters (see “ <i>Editor Options</i> ,” below).
Del or Ctrl-G	Deletes character to right.
Backspace or Ctrl-H	Deletes character to left.
Ctrl-T	Deletes word to right.
Ctrl-Y	Deletes line. The text of the deleted line is saved in an “undo” buffer and may be retrieved using Ctrl-K U (see “ <i>block commands</i> ” below).
Ctrl-Q Y	Deletes from cursor to end of line.
Ctrl-P	Prefixes a Control character, allowing it to be entered as text and displayed graphically.
Alt-J	Inserts (joins) a block of text at the cursor. The block

must have been previously ‘cut’ using Alt-K (see “*Cut and Paste*,” page).

Ctrl-N Splits the current line into two lines at the cursor.

Table 3-7: Insertion and Deletion Commands

Special Characters

The editor will allow you to insert most nonprintable characters by prefixing them with `Ctrl-P` and will display them as graphics characters. You can use the ASCII table to view the representation of individual characters.

Some characters, however, are reserved for special uses and should only be inserted in the text if their function is required. See Table -8 for a list of special control characters recognized by the editor.

Command	Operation
Ctrl-A	Reserved for environment use.
Ctrl-B	Highlights text either to the next <i>Ctrl-B</i> or the end of the line, whichever comes first.
Ctrl-C	Similar to <i>Ctrl-B</i> but inverts instead of highlights.
Ctrl-D	Similar to <i>Ctrl-C</i> but provides a different color combination.
Ctrl-J	Line feed, used after <i>Ctrl-M</i> to indicate end of line.
Ctrl-M	Carriage return used (with <i>Ctrl-J</i>) to indicate end of line. Note: <i>Ctrl-Ms</i> without <i>Ctrl-Js</i> may cause text not to be displayed by the editor giving unpredictable results while editing.
Ctrl-I	Tab character. These are inserted if Options (T) hard tabs is set on.

Table -8 Special Characters in the Editor

The characters *Ctrl-B*, *Ctrl-C* and *Ctrl-D* may be inserted in your files to create special effects when editing. The ‘colored’ text produced may be recolored using normal window recoloring (see “*Recoloring Windows*,” page). Care should be taken not to include unprintable control characters in source text other than within comments, as the compiler will not ignore these characters.

To view special characters, you can set **Options Control chars** to on. This displays all the above characters graphically. This can be useful for detecting spurious control characters within your files.

Cursor Movement

Table -9 summarizes the cursor movement keys.

Command	Operation
← or Ctrl-S	Moves cursor left.
→ or Ctrl-D	Moves cursor right.
↑ or Ctrl-E	Moves cursor up.
↓ or Ctrl-X	Moves cursor down.
Ctrl ← or Ctrl-A	Moves cursor to previous word.
Ctrl → or Ctrl-F	Moves cursor to next word.
Ctrl-W	Scrolls screen up.
Ctrl-Z	Scrolls screen down.
PgUp or Ctrl-R	Moves one page up.
PgDn or Ctrl-C	Moves one page down.
Home or Ctrl-Q S	Moves to start of line.
End or Ctrl-Q D	Moves to end of line.
Ctrl-Home or Ctrl-Q E	Moves to top of screen.
Ctrl-End or Ctrl-Q X	Moves to bottom of screen.
Ctrl-PgUp or Ctrl-Q R	Moves to top of file.
Ctrl-PgDn or Ctrl-Q C	Moves to bottom of file.
Ctrl-Q B	Moves to start of block.
Ctrl-Q K	Moves to end of block.
Ctrl-Q P	Moves to previous position.
Ctrl-Q G	Prompts for line number and moves to that line.

Table -9: Cursor Movement Commands

Block Commands

The `block` commands enable you to define an area of text that you can then copy, move, delete, write to a file, indent or import from another editor window. When a block is defined and on display, its text is highlighted. You can change the foreground and/or background of the block text. (See “*Recoloring Windows*,” page).

Command	Operation
Ctrl-K B	Marks the start of the block.
Ctrl-K K	Marks the end of the block.
Ctrl-K T	Marks a single word.
Ctrl-K L	Marks a single line.
Ctrl-K H	Toggles whether the block is displayed.
Ctrl-K C	Copies the block to the current cursor position.
Ctrl-K V	Moves the block to the current cursor position.
Ctrl-K Y	Deletes the block. If text deleted is less than 4K, it will automatically be placed in an “undo” buffer, which may be retrieved using Ctrl-K U (see below). If the deletion is larger than 4K, you are requested to confirm the deletion of the block.
Ctrl-K W	Writes the contents of the block to a file.
Ctrl-K R	Reads the contents of a file into a block.
Ctrl-K P	Prints the contents of the block to a standard printer device.
Ctrl-K G	Gets a block from a different editor window, copying it to the current cursor position. This is a convenient way of importing text from other editor windows.
Command	Operation
Ctrl-K I	Indents the text of the block. If the indent specified is positive, then the block will move to the right; if negative, the block moves to the left. This is a convenient way of fixing indentation in your program. Spaces are only removed when moving left.
Ctrl-K U	“Undoes” the last block delete (Ctrl-K Y) or line delete (Ctrl-Y), inserting the previously deleted text at the current cursor position.
Ctrl-K D	Saves the current file to disk. (see “ <i>Saving a file</i> ,”).
Ctrl-K A	Saves all changed files to disk. (see “ <i>Saving a file</i> ,”).

Ctrl-K Q	Quits file. This removes the file from the current active editor window and closes the window. You can save any changes before quitting, if you wish.
----------	---

Table -10: Block Commands

Editor Options

The `editor options` command let you change the overall behavior of the editor. These options, when set, apply to all the editor windows in the environment. For example, if you switch the control character display on (`Ctrl-O C`), then the control characters will be visible in every editor window.

Command	Operation
Ctrl-O V	Toggles between I nsert and O verwrite modes. (Same as <code>Ins</code> and <code>Ctrl-V</code>)
Ctrl-O I	Toggles A uto I ndent on and off. If A uto I ndent is set on then inserting a new line will automatically indent the new line to match the indent of the line above. Also <code>Tab</code> moves to match the start of words in the line above.
Ctrl-O T	Toggles whether spaces or a single tab character will be inserted, whenever <code>Tab</code> is pressed.
Ctrl-O W	Sets the width between tab stops.
Ctrl-O C	Displays control characters as graphics characters. This allows you to see the exact contents of your file.
Ctrl-O R	Toggles read only mode in the editor. If this option is set on, the editor ignores any operation which would change the file. This is useful to avoid unintentionally changing a file that is only being viewed.

Table -11: Editor mode commands

Search and Replace

Search lets you search through the file for a specific string, while Replace allows you to replace occurrences of one string with another.

Command	Operation
Ctrl-Q F	Prompts for a string to find, and then searches for this string, moving the cursor to its first occurrence, if found. Search options (see Table -12.) let you decide things like searching backwards, matching only whole words, and so on.
Ctrl-Q A	Prompts for a string to find and then for a replacement string, then searches for the string, replacing any occurrence found. See below for details of Replace options.
Ctrl-L	Repeats the last F ind or R eplace command, with the same strings and options.

When seeking or replacing text, you may specify one or more search options, which are described in Table -12:

Command	Operation
B	Searches or replaces backwards from the cursor towards the beginning of the file.
U	Ignores whether the string is in upper or lower case when matching.
W	Only matches the search string with whole words.
G	Replaces globally throughout the file.
L	Replaces locally within the currently marked block.
R	Replaces globally from the cursor to the end of the file, or, if the B option is set, from the cursor to the beginning of the file.
N	Replaces without prompting for confirmation.
number	Searches or replaces the specified number of times.

Table -12: Search and Replace Options

You can combine the above options. For example, the option GUN replaces globally without case sensitivity and without asking for confirmation. In a search string, the Ctrl-A character is wild and matches any character. To enter a wildcard character into the search string, type Ctrl-P Ctrl-A.

Other Editor Commands

Other commands available within the editor include upper-case word conversions and corrections, marker setting and movement.

The keys to invoke these commands are as follows:

Command	Operation
Ctrl-U	Converts the word at the current cursor position to upper case. This is useful for C upper case macro names, and <i>Modula-2</i> reserved words.
Shift-F7	Sets the alphabetic case of the word to be the same as the previous occurrence of the word in the file. This can be used to correct upper-case/lower-case errors within mixed-case identifiers.
Ctrl-K 1	Sets location marker 1 at the current cursor position.
Ctrl-K 2	Sets location marker 2 at the current cursor position.
Ctrl-Q 1	Goes to location marker 1.
Ctrl-Q 2	Goes to location marker 2.

The markers can be used to mark positions in the file for editing and moving between different positions in the file.

In the error editor window, which is invoked by compilation or run-time errors, you can move to the next and previous error positions as follows:

Command	Operation
F8	Move to the nearest error after the current cursor position.
F7	Move to nearest error before the current cursor position.

If, as a result of a `make` or compilation, there is more than one file containing errors, then F8 and F7 will move between the files, i.e., pressing F8 after the last error in the current file will take you to the first error in the next file. Similarly, pressing F7 before the first error takes you to the last error in the previous file.

Compiling and Running Programs

Within the TopSpeed Environment, you can instantly compile your program source into object files, link those objects into an executable program, and then run that program. You can do all these things with simple menu selections or by using shortcut keys.

Compiling

To start compiling a file, do either of the following:

- Select **Compile** from the main menu.
- Type the shortcut command, Alt-C.

If you are currently in an editor window, then the environment will try to determine the correct compiler to use from the extension of the file being edited.

The default extensions are:

- .MOD and .DEF for Modula-2
- .PAS and .ITF for Pascal
- .C and .H for C
- .CPP and .HPP for C++
- .A and .INC for TopSpeed Assembler

If a matching compiler is found, then the compilation (or assembly) of that file will start immediately. After the compiler has finished, the window in Figure -6 is displayed.

During compilation, the compiler window indicates the file being compiled, the current project name, any files included and the line currently being compiled. The thermometer-style indicator shows the proportion of the file that has already been compiled, and the disk activity indicator in the top righthand corner shows whenever the disk is accessed. At any stage, you can abort the compilation by pressing Ctrl-Break.

You can set various compiler options to generate debugging information and to modify the behavior of the compiler. These options are described in the chapter “*Pragmas*,” in the *TopSpeed Developer’s Guide*.

Compilation Errors

As compilation proceeds, the number of errors and warnings found is reported. When an error is reported, you can edit the file immediately by pressing Enter. This will start editing within the error editor window (window 0) — at the position of the first error, or, if there are no errors, at the first warning. The bottom line of the window displays any errors present in the current line. To move between multiple errors, use F8 to move to the

next and F7 to move to the previous error. Error positions will be adjusted as text is edited, allowing you to correct all the errors found. The compiler will keep up to about 500 error messages, aborting compilation after this limit.

After you have made corrections to the error file, you can continue compilation by either pressing `Esc` or `Alt-C`. Compilation will restart immediately.

The Project Name

Compile, **Make**, **Link** and **Run** each need to know the project name before commencing. This is in order to determine project-based information, such as the memory model used, type of target to be made and compiler and linker pragmas. This information is kept within the project (`.PR`) file (see “*The Project System*,” page). To set the project name, you can enter it on the **Project** menu (shortcut `Alt-P`), or you can enter it when you do a **Make** or **Run**. If the `.PR` file does not exist for the current project, then the file `UNNAMED.PR` is used instead.

If you enter a new project name on the project file, then a new project file will be created. You will be prompted for a project file to copy from as a template for the new file to be created. The default template is `TEMPLATE.PR`.

Note: Information in the project file is checked for consistency by the TopSpeed project system, so if you have previously compiled a file with a different project (or no project at all), then the **Make** may need to recompile that file even though its source has not changed.

Making a Program

TopSpeed has an automatic **Make** facility (shortcut `Alt-M`) within the project system. This calculates dependencies and automatically recompiles all out-of-date modules within your program. Then, if all compilations were successful, an executable (`.EXE`) file is created by linking the program objects together.

For example, if you change a particular definition file, then **Make** will automatically ensure that all modules importing the changed definition will be recompiled to make a consistent program.

To invoke the **Make** command, select **Make** from the main menu or press the shortcut key, `Alt-M`. After you enter the main module name, the make window in Figure -7 will be displayed, and the **Make** process will commence.

The make window shows all files being compiled. Then, if all files compile with no errors, the window shows the link taking place. If errors are found, the Make is aborted, and you can immediately correct the errors by pressing Enter. When you have finished your corrections, press either Esc or Alt-M to restart the Make process. You can abort the Make at any time by pressing Esc.

Once the Make has been completed with no errors, you can run the completed program O now in a .EXE file O either from inside or outside the environment.

Running Programs

To run your completed program within the environment, do either of the following:

- Select the **R**un command on the main menu.
- Use the shortcut command, Alt-R.

Provided the **Auto make** option is set on, this function will first perform an automatic **M**ake to handle any editing done since the last **M**ake. Then, a full-screen window is opened in which your program is executed. When your program has finished, you are prompted to press Esc to return to the environment. This environment will be in exactly the same state as it was when you left. You may move the prompt window using ScrollLock if you need to view the windows underneath (see “*Repositioning Windows*,” page).

To time the execution of your program exactly, you can set the **T**imed run option (see “*Timed Run*,” page).

Run-Time Errors

You can get TopSpeed to report run-time errors when running your program by setting the appropriate compiler directives. If you are running within the environment, you can be taken to the exact position in your source where the error occurred. This powerful feature allows you to quickly find and correct common run-time errors. You can also find the location of errors reported when running outside of the environment by running the **U**tility **(E)**Find run-time error utility command.

The Project System

The TopSpeed Project system provides a simple but powerful facility which gives you total control over the make system. The project system can be used

for all projects from the simplest single target (e.g. .EXE or .LIB files) project to the most complicated multiple DLL projects.

There are two ways of editing your project files:

- by the **P**roject Menu (see below), which provides a quick and easy menu interface to the most common project commands, and
- by editing the project script, which is written in a powerful, sequential command language. This language is described fully in the “*TopSpeed Developer’s Guide*”.

All project files must have the .PR extension (project include files, by convention, usually have the extension .PI). The format of the project file is simple text, so you can use any normal editor to change these files.

Project Menu

The **P**roject Menu (shortcut `Alt-P`) offers facilities to:

- Set the current project filename.
- Change the current project file using simple menu commands.
- Edit the current project file text.

Compile, **M**ake, **L**ink and **R**un all use the current project file to determine the options used. Changing options on the project menu automatically edits the top of the current project file. This ensures that project options are always retained between sessions.

N - New Project

This command sets the name of the current project. The contents of this project file (with the extension .PR) determines the settings of options displayed on the rest of the project menu and the default options used for **C**ompile and **M**akes.

If the project specified does not exist, then you are asked whether you wish to create a new file. If Y is pressed then you are prompted for a template project file name. The contents of this file is then copied to the new project file created. The default template file is `TEMPLATE.PR`, which may be edited to suit your own preferred defaults.

E - Edit Project File

This loads the current project file into editor number 9, allowing you to edit the project using a normal editor. Changes to project file are reflected immediately on the project menu and in subsequent **C**ompile and **M**akes.

S - System

This defines the target operating system. Currently `auto`, `dos`, `os2` and `win` (for Microsoft Windows) are supported. `auto` ensures that the target is the same as the operating system being used for development (e.g. `auto` means `dos` if you are running under DOS).

T - Target type

This defines the file type of the main result file that the current project is intended to make. Possible target types are: `exe` for creating programs, `lib` for creating libraries and `dll` for creating dynamically linked libraries (`.DLLs`).

M - Memory model

This defines the memory model used for the project. Memory models available are:

S	Small
C	Compact
M	Medium
L	Large
X	XLarge
T	MThread
O	Overlay
D	Dynalink

See the “*TopSpeed Developer’s Guide*,” for more details on the use and definition of the memory models.

Note: Only those which you installed are available on the menu.

V - VID

This option determines the level of debug information generated for use by VID (Visual Interactive Debugger). The settings are:

off	No debug information is generated
full	Full debug information is generated for VID.
min	Minimum debug information is generated, ensuring that the program to be debugged is not altered.

See chapter on the “*Visual Interactive Debugger*” for further details of using VID.

Note: If you wish to use `VID2CV` in order to use CodeView or CodeView compatible debuggers, then you should set this option to `full` or `min`. See the “TopSpeed Developer's Guide” for further details.

C - Runtime checks

The Runtime checks menu allows you to set appropriate runtime checks within your program. Runtime checks are turned off by default. See the “*TopSpeed Developer's Guide*,” for a full list of checks available.

O - Optimization options

The Optimization options menu allows you to set which optimizations are performed within your program. Usually all optimizations are turned on by default. See the “*TopSpeed Developer's Guide*,” for a full list of optimization options available.

P - Project options

The project options affect the **Make** and **Link** options set within the project. These options include the following:

J - Calling convention

This determines whether to use the default (`stdcall`) calling convention which passes parameters in registers, or to use a stack based convention (`stdcall`) for interfacing with stack based libraries of other vendors.

Note: If you use `stdcall`, you will need stack based libraries for the model you are using. See the “TopSpeed Developer's Guide,” for more details of this.

M - Map file

This determines whether links performed will generate a `.MAP` files. Map files contain all public symbols and corresponding addresses and are used by some debuggers for symbolic information. You will need to generate a map file if you are debugging in assembler using VID or you are using TSPROF to profile your code.

C - Case sensitive link

If on this makes the linker treat names as case-sensitive. Normally, the project system will determine automatically whether to turn it on, depending on the language(s) used.

W - Linker warnings

If on, this option disables the display of linker warning messages (although they still can be viewed in any MAP file generated).

R - Remake all

If on this will cause the next make to re-compile all files used within the project, even if they do not appear to require remaking. This option is reset to off by a successful **Make**

L - Language options

The **Language** options menu allows you to set various common compiler pragmas from the menu. See the “*TopSpeed Developer’s Guide*,” for a full description of compiler pragmas available.

The Options Menu

Your TopSpeed system is designed to be extremely flexible. It was designed to enable you to adapt it to your own particular needs and preferences wherever possible. Toward this aim, each of the major components has a set of options that you can change. These options are saved in the session and configuration files, allowing you to set options that will be remembered after leaving the environment.

You can invoke the **Options** menu (shortcut **Alt-O**), which contains two submenus: **Run Options** and **Setup**.

V - CGA Snow Check (DOS Only)

This should be set to on if your CGA monitor “snows” when displaying windows in the environment. However, be forewarned that the display of windows is considerably slower when this option is on. The default setting is off.

V - VIO Screen Output (OS/2 Only)

This should be set to on if you require the environment to only use VIO calls to update the screen. If set to off, then the environment will write to screen memory directly, and consequently may update the screen faster. If you are running the environment within a Presentation Manager window, then this option will automatically be turned on.

D - Delayed Screen Update

If on, this causes the editor, when paging through a file, to only update the screen if no keys are waiting. This will speed up paging if the display cannot keep up with the keyboard, as may be the case if option **V** (described above) is turned on or the display is slow.

H - High Background

This specifies whether the high background colors available with the IBM CGA should be enabled. (See “*Recoloring Windows*,” page .)

C - Solid Cursor

If this option is on, then the cursor will be a solid flashing block; if off, the cursor is the normal flashing underline.

M - Use Expanded Memory (DOS Only)

Normally the TopSpeed Environment will automatically use expanded memory if a LIM EMS driver is installed, and expanded memory is available on starting up the environment. The use of expanded memory speeds up all disk-based operations in the environment, compilers, and the Make system. Using expanded memory will not however, generally solve “Out of Memory” compiler messages, as these are usually due to internal limits in conventional memory being reached. If no expanded memory is available, this option is automatically turned off.

Q - Help Line

This option determines whether the help line on the bottom line of the screen is displayed. If on, the line is displayed; if off, the help line is removed, freeing the bottom line for use, for example, by ‘zoomed’ editing windows. The help line often contains useful information on the keys that can be pressed in a given situation, so it is not advisable to turn off the help line until you are reasonably familiar with operating the environment.

Run Options

The run options modify the operation of the **Run** command (see “*Running Programs*,” page). These options are described in this section.

C - Command Line

This option allows you to enter a DOS or OS/2 command line that is passed to the program when it is executed from within the environment. You will be prompted for the command line.

A - Auto Make

If this option is on, then an automatic make is performed before any program is run. This ensures that the program uses the most recent versions of all the modules.

S - Session Run (OS/2 Only)

If turned on, this causes your program to be independent of TopSpeed in a separate OS/2 session. To return to the environment, you should use the OS/2 Session Manager. This also allows you to exit TopSpeed while still leaving the program running.

T - Timed Run

If this option is on, the execution time of your program is displayed when the program completes. The format of the time displayed is *HH:MM:SS:HS* indicating Hours, Minutes, Seconds and 1/100ths of seconds. The value excludes the time taken to load the program from disk. (The precision with which these measurements are taken depends on your DOS or OS/2 implementation. Normally, the precision is about 0.06 seconds, since the clock is checked 18.2 times per second.)

Setup Options Menu

The **Setup Options** menu (shortcut **Alt-S**) allows you to modify the TopSpeed editor, described in the section on “*The Editor*,” page . You can also save and restore configuration files, and you can load a new redirection file if you wish.

A - Auto Save Files

When on, this option causes the editor to save all changed files whenever you run a program, enter a DOS shell, or exit TopSpeed. This ensures that no edits are lost, even if a program crashes. This option also enables programs that are run under the environment to read the latest version of files being edited. If the option is off, the environment asks whether to save edited files when exiting, but does not automatically save files when running programs.

F - Default Filenames

This option lets you set the default filenames supplied when you enter an editor window for the first time. The default filenames are initially set to **.** but may be changed to be different for each editor window, depending on your use of that window. For example, you may want to set the default filenames of windows 6 through 8 to **.H*, **.DEF*, **.ITF* or

* .HPP so that you can quickly select an interface file for editing within those windows.

E - Default Extensions

This allows you to specify the default extensions that are added to filenames entered without an extension. Again, you can have different extensions for each editor window, if you wish. The default extensions are all initially set blank.

N - Number Of Backups

This lets you specify the number of backup files that are retained by the editor whenever a file is saved. This value must be between 0 and 9, inclusive. 0 means that no backup file is created; 2 means that the two most recent previous versions of your file are kept. The first, most recent, backup file has the extension .BAK, the second .BK2, the third .BK3 and so on. This option is initially set to 1.

T - Top Scroll Zone

This sets the number of lines from the top at which the editor window begins to scroll when moving up. For example, if this is set to 2, then the screen will scroll if you move the cursor to the second line from the top. The initial default value is 0. Increasing it above this value allows you to ensure that there are always lines visible above the cursor.

B - Bottom Scroll Zone

This sets the number of lines from the bottom at which the editor window begins to scroll when moving down. The initial default value is 1. Increasing it above this value allows you to ensure that there are always lines visible below the cursor.

R - Load Redirection File

This loads a new redirection file, replacing all previous redirections. (See *"The Redirection File,"* page).

E - Edit Redirection File

This loads the current redirection file into editor number 9, allowing you to edit the path redirections. When exiting the redirection editor window (using Esc) or saving the file (using F2 or Ctrl-K D), the redirection file will automatically be reloaded causing any changes made to immediately come into effect.

J - Load Keyboard Macros

This loads a set of keyboard macros from the specified macro text file, replacing any macros currently defined. (See “*Keyboard Macros*,” page for further details.)

K - Save Keyboard Macros

This saves the current macros defined into a text file, which can be read using the (J)Load keyboard macros option. (See “*Keyboard Macros*,” page for further details.)

L - Load Options/Windows

This loads a .CFG file that has previously been saved using **Save options/windows** described below. The command loads each window’s color, position and size, and all option values. This lets you swap quickly between different configurations of the environment.

S - Save Options/Windows

This saves the options and window setup to a .CFG file, which can be reloaded later on using the **Load options/windows** command described above.

The Utilities Menu

The TopSpeed Environment comes with a number of utilities usually found in pop-up programs such as SideKickΣ. These utilities allow you to search and view files, calculate numbers, find ASCII characters and review the screen before entering the environment.

All the utilities lie on the **Utilities** menu (shortcut Alt-U) and have single-key shortcuts.

F - File search

File search (shortcut F4) allows you to find a piece of text, such as a structure or record definition, amongst the files on a disk. Once the search is complete, a search pick list contains the files that match the text. Pressing Enter on any of the filenames in the list will load an editor with the file and place the cursor at the first instance of the search text. Use the **Utilities Search list** command to load another file from the search list.

The menu comes with three prompts: the text to search for, the files to search and the options. The files to search can contain the standard wild cards (*) and (?). The search text can contain wild cards, but only when you specify

the * option. A search text will always match within a single line and won't cross a line break.

The following are the options that control the search:

- * When specified, the search text can contain the wild cards * and ?. The wild card ? matches any single character while the wildcard * matches any number of characters. Default is on.
- U When specified, the search string matches both lower case and upper case letters. This means that file search will find nigel, NIGEL or even nIGEL at the expense of some loss in speed. If not specified, you get a faster search but the search string must match exactly.
- 1-9 When you specify a number from 1 through 9, the matching file will loaded into that editor window rather than the last active editor.
- R When specified, the file search uses the redirection file to find the directories containing the file. When not specified, the search only uses the current directory. If you specify a leading path in the filename, for example C:\NIGEL*.H, then you override redirection and only search the specified directory.

If you don't specify any options, the search defaults to ' *UR ' - that is, it uses wild cards in the search text and for redirection, and is case insensitive.

S - Search list

This recalls the search pick list of the files found by the last **Utilities File** search or **Locate File** command with the cursor one below the last selection. The cursor movement allows you to use shortcut Alt-F4 to step through the files that contain the search text.

F - Locate File

Locate file will search for a file anywhere on the disk and will set-up the search list (see above) with all files found. The command prompts for the name to search for, which may contain wildcards, and the drives that should be searched.

E - Find Run-time Error

The **Find run-time error** command looks for the file `ERRORINF . $$$`, loads the source code file, and positions the cursor at the run-time error. The environment automatically invokes this command when a run-time error occurs because of the Run command.

The compiler must produce line number information and the link must generate a map to use this command.

You don't need this command when running a program from within the environment, since the error is located automatically in that case.

Note: Either VID information or a map file with line numbers is required to find the run-time error.

A - ASCII Table

The ASCII table (shortcut Alt-A) produces a table of ASCII characters with the equivalent decimal and hexadecimal numbers, and control character representations. At the end of the table, you will find a summary of the IBM CGA color attributes.

The table has five pages which contain:

Code	Description
ASCII 0-31	The control characters. The table shows these characters in two rows with hex and decimal values, PC graphic character, Control key value (prefixed by ^), and the ASCII mnemonic.
ASCII 32-95	Normal upper-case characters. The table shows these characters as four rows with the character and its value in both hexadecimal and decimal form.
ASCII 96-159	Lower case and special symbols.
ASCII 160-223	Graphics line and block drawing.
ASCII 224-255	Greek letters, mathematical symbols and display attributes. The 16-color display attributes allow you to vary the foreground and background color of a character.

C - Calculator

The calculator (shortcut Alt-B) is a simple, 32-bit, four-function device especially devised for programmers. The keys used by the calculator are given below.

Note: Calculator keys are case-sensitive - **A** is a hexadecimal digit, but **a** is the bitwise AND operator.

Basic Keys

Key	Function
0-9	Use for entering binary, decimal and hex numbers.
A-F	Use the upper-case letters A through F for entering hexadecimal numbers.

Bsp	Use the backspace key to delete the last number you entered.
c	Clear the contents of the last calculation.
Esc	Exits the calculator but with complete preservation of the calculator state.
Alt-K	Cuts the number in the display into the clipboard.
Alt-P	Pastes the contents of the clipboard into the required editor window.

Base Conversion Keys

Key	Function
b	Converts the displayed number to binary after completing the current calculation and makes all future numbers use the binary base. (The calculator can only display the final 16 digits of a 32 bit result though it uses all 32 bits in calculations.)
d	Converts the displayed number to decimal after completing the current calculation and makes all future numbers use the decimal base.
h	Converts the displayed number to hexadecimal after completing the current calculation and makes all future numbers use the hexadecimal base.

Keys for the Arithmetic Functions

Key	Function
•	Adds the previously entered number or result with the number in the display. The result is displayed.
-	Subtracts the number in the display from the previously entered number or result. The result is displayed.
*	Multiplies the number in the display by the previously entered number or result. The result is displayed.
/	Divides the number in the display with the previously entered number or result. The result is displayed.
s	Changes the sign or performs a two's complement of the displayed number after completing any pending calculations.
%	Calculates the modulus or remainder of the number in the display with the previously entered number or result. The result is displayed.

Keys for the Bitwise Functions

Key	Function
a	Bitwise ANDs the number in the display with the previously entered number or result. The result replaces the number in the display.
o	Bitwise ORs the number in the display with the previously entered number or result. The result replaces the number in the display.
x	Bitwise XORs the number in the display with the previously entered number or result. The result replaces the number in the display.

Keys for the Memory Functions

Key	Function
m=	Completes all pending calculations and stores the value displayed in the memory. The new value overwrites the previous contents of the memory.
m	m followed by any arithmetic operator such as \bullet , -, or a (for and) will cause the displayed value to be combined with the memory using the specified operation. The result goes into memory and is not shown on the display. For example, if the memory contains 3 and the display 2, then $m\bullet$ will place 5 into memory and keep 2 in the display.
mr	Recalls the value stored in the memory.
mc	Zeros the contents of the memory.

V - View File as Data

View file as data allows you to view a file in hexadecimal and ASCII. The window has three sections, from left to right: the absolute address within the file, the hexadecimal representation, and the ASCII representation.

H - Help Window

Shortcuts Alt-F1 or Alt-H will pop up the help window at the place you last left it. It does not perform a context-sensitive lookup of the help system, unlike the function *F1*.

R - Review DOS Screen

The **Review** DOS screen command (shortcut Alt-F5) allows you to look at the screen displayed after running a program. If you have not used **Run** or the **File** DOS shell commands, you view the screen as it was before you entered the TopSpeed Environment. You can use the **Cut** (Alt-K) and **Join**

commands (Alt-J) to save the contents of the screen for later use. Press any key to return to the environment.

I - System Info

System info (shortcut Alt-I) shows the current time, date, disk-free space, contents of the editor windows and the processor type.

P - Print File

The **Print** file command formats a single file and sends it to the printer. You are prompted for the name of the file. The format of the printed file is determined by the print settings described below.

Q - Print Settings

The **Print** settings command determines the format of the printed file. The following commands are supported:

Lines per page	The length of the page from top to bottom measured in lines. The default is 60.
Page width	The number of characters on each line. If more characters exist on a line in the file than the page width number, the line wraps or truncates, depending on the wrap option. The default is 76 characters.
Top margin	The number of blank lines between the heading or, if no heading, the top of page and the start of text. The default is 1.
Left margin	The number of characters from the left side of the page and the first printing character. The default is 3.
Page heading	<p>The text that prints at the top of each page. You can use the following macros in this string:</p> <p>%F to print the filename</p> <p>%D to print the date of the file or current date</p> <p>%T to print the current time</p> <p>%P to print the page number.</p> <p>The default header is</p> <p>File: %F Date: %D Time: %T Page: %P</p>
Init string	A string that Print File sends to the printer before printing the file. Prefix control characters with Ctrl-P, for example to enter Esc-4 you type Ctrl-P Esc 4. No initialization string is set by default.
Form feed string	The string sent to the printer at the completion of a page. The default is a form feed.

Final string	The string sent to the printer when the file has been printed. The default is a form feed.
Output file/device	This file or device prints the formatted file. The default device is PRN.

You can also vary the format by choosing one or more of the following settings from the options field.

H	When specified, the heading set by the page heading command prints at the top of each page.
W	When specified, characters greater than the page width will wrap onto the next line. If not specified, the line will be truncated and an exclamation mark (!) printed to indicate lost text.
F	When specified, the %D and %T macros use the date and time of the file rather than the current date and time.

K - Keyboard Codes

Keyboard codes allow you to determine the codes generated when pressing specific keys. When you press a key the utility displays:

ASCII code	The ASCII code of the key displayed in decimal and hexadecimal.
Extended code	The two-byte code for the non-ASCII keys shown in decimal and hexadecimal.
Key name	The name of the key as shown in the environment O for example Ctrl-PgUp displays as <CtrlPgUp>.

Press `ESC` `ESC` to exit the keyboard codes utility.

The Redirection File

Within the TopSpeed Environment, you can define a search path by which files are located in the DOS or OS/2 file system. This allows you to put different kinds of files in different subdirectories. The file that defines these search paths is a simple text file, `TS . RED`. This file contains one or more lines with the format:

```
MatchName = DirectoryPath { ; DirectoryPath }
```

`MatchName` is a filename specification that can contain wild ('*' and '?') characters, and `DirectoryPath` is the location for those files that match the `MatchName`. You can specify multiple directory paths by separating them with a semicolon (;). If there is more than one `Directory Path` specified, each path will be searched in sequence until the file is found.

This is best demonstrated by an example. Suppose the file `TS.RED` contains the following text:

```
*.cpp = . ; c:\nigel\cppsrc; c:\ts\src
*.pas = . ; c:\nigel\passrc; c:\ts\src
*.a   = . ; c:\nigel\asrc; c:\ts\src
*.hpp = . ; c:\nigel\inc; c:\ts\include
*.itf = . ; c:\nigel\inc; c:\ts\include
*.a   = . ; c:\nigel\asrc; c:\ts\lib\src
*.obj = c:\objs; c:\ts\objs
*.lib = c:\tc\lib
*.pr  = . ; c:\tc\pr
ts.ses = c:\nigel
```

This specifies that files that match `*.CPP` are first searched for in the current directory (specified by the `'.'`) and then, if not found, in `c:\nigel\cppsrc` and then, if again not found, in `c:\ts\src`.

Similarly, files that match `*.obj` are searched for, and stored, in the directory `c:\objs`; if not found in `c:\objs`, they are searched for in `c:\ts\objs`.

Whenever a new file is created, it is always created in the first directory that matches the filename. This means that in the above example, all `.OBJ` files will be created in the `C:\objs` and all `.PAS` files will be created in `'.'` (the current directory).

Note: Whenever you input a full pathname (i.e., a complete directory path) to the environment, redirection does not occur. This allows you to override redirection, if required.

The redirection file is always searched from the top until the file is found. This allows you to specify individual file redirection as well as general default cases. For example:

```
testdb.c = \tsc\testing
readdb.c = \tsc\testing
*.c      = . ; \tsc\work
```

tells the system to search certain directories for general files with a `.C` extension, but to search in different directories for two specific files with this extension. The current directory should always be specified as `'.'` since Null entries are ignored.

To quickly edit the current redirection file, you can use the **Options Setup Edit redirection file** command, shortcut `Alt-S E`. This will place the redirection file in editor 9 ready to be edited, and will reload the redirection file automatically on exit from this editor. (See “*Setup Options*,” page for more details on loading and editing the redirection file.)

Caution:

A word of warning about using file redirection; It can be rather confusing if there are several versions of a file in the search path. If you seem to be having a problem, check each directory to make sure that the files present are those you expect. One useful tip is to check where a file is being redirected from by loading it into an editor, which will then show the full path used.

Cut and Paste

The TopSpeed Environment has a built-in cut and paste facility that allows you to 'cut' text from the screen at any time and then subsequently 'paste' it into any editing window. This feature has many uses:

- Quick block copy between windows.
- Copy text from the help window.
- Paste numbers from the calculator display.
- Capture the output from running a program or a DOS shell.
- Copy characters from the ASCII table.
- Copy columns within an editor.

Cutting Text

Text can be cut from the screen quickly and easily with a few keystrokes. To do this, the following steps must be followed:

- Enter cutting mode by pressing Alt-K.
- Move the cursor to the start of where you want to cut text from, by using the cursor keys (← ↑ → ↓). Ctrl ← and Ctrl ↑ can be used to move quickly to the left and right.
- Press B to indicate the beginning of the area to cut.
- Use the cursor keys to define the area to cut. As the cursor moves, the area to be cut is displayed in inverse.
- When the area is correct, press Enter to save the block ready for pasting.

See Table -13 for a summary of keys available when in cutting mode.

You can cut text from anywhere in the TopSpeed Environment, even when reviewing the DOS or OS/2 screen using Alt-**F5**. If you want to cut text from the last line of the screen (which is obscured by the cut text help line),

the environment will remove this help line as the cursor enters the bottom line.

Command	Operation
$\frac{1}{4}$ $\frac{1}{4}$ -	Moves the cursor and extends or contracts the area defined if <i>B</i> has been pressed.
Ctrl $\frac{1}{4}$	Fast move left.
Ctrl	Fast move right.
B	Starts defining a block.
Enter	Saves the area defined and exits cutting mode.
Esc	Exits cutting mode without saving the defined block.

Table -13: Cut and Paste Commands

Pasting Text

Pasting text you have previously cut is extremely easy. First, move the cursor to the desired destination in the file that you are editing and press Alt-J (join text). The text last cut will then be inserted as a block at the current cursor position. You can repeat this as many times as required with the same cut text.

Keyboard Macros

Keyboard macros allow you to easily attach a frequently-used sequence of keys to an individual keystroke. This can be used to automate common command and editing sequences. Previously created macros can be saved to, and loaded from disk so you can set up macro libraries for every situation.

Recording and Playing Back Macros

There are two kinds of keyboard macros that you can define:

- The single keystroke macro, which can be attached to any key on the keyboard (except certain command keys, see below)
- The double keystroke macro, which consists of the macro key `Alt -` (that is, `Alt` pressed with the `'-'` key on the top row of the keyboard) followed by any other keystroke.

Single Keystroke Macro

These macros, once recorded, can be played back by simply pressing the key at any time within the environment. To record a single keystroke macro, simply press the macro definition key, `Alt =`, followed by the key you want to attach the macro to. Then, you can type the keys you want recorded, followed by another `Alt =` to terminate the recording.

For example, to define a macro that would type the string `'#pragma'` whenever `Alt-Z` was pressed, you would enter the following keys:

```
Alt= Alt-Z #pragma Alt=
```

Double Keystroke Macro

Because the environment provides a large number of shortcuts, it is often difficult to choose non-conflicting keys to use for macros. For this reason, double keystroke macros are provided. These consist of the macro key (`Alt -`) followed by another key. Because `Alt -` is reserved for macros, all the keys become readily available for use as the macro identifier, including the simple alphanumeric keys.

To start recording a double keystroke macro, you press `Alt =` followed by `Alt -` and the identifying key. Then, you can type the keys you want recorded, followed by another `Alt =` to terminate the recording.

To playback the recording, you just type `Alt -` followed by the identifying key.

For example, to record a macro that compiles the program `test.c` when `Alt - C` is pressed you could enter the following:

```
Alt= Alt- C F10 C test.c Enter Alt=
```

While recording macros, the help line continuously displays the current macro being recorded. This helps avoid your forgetting that you are currently recording a keyboard macro, which could result in very long macros being defined. On playback, the execution of a macro can be interrupted by pressing `Ctrl-Break`.

The following keys cannot be used as macro identifiers: Esc, Alt =, Alt - and Ctrl-Break. You also should be wary of using common keys (such as, A-Z, cursor keys, etc.) when defining single keystroke macros.

Note: Only keys that generate a keycode can be used to define macros; you cannot use Alt Use the Keyboard code utility (see page for details) to determine if the key you want generates a unique keycode.

Saving and Loading Keyboard Macros

Keyboard macros can be saved to, and reloaded from, disk using the **Options Setup** commands (**K**)Save keyboard macros (shortcut Alt-S K) and (**J**)Load keyboard macros (shortcut Alt-S J). These commands allow you to store and later retrieve all the currently defined macros.

The default macro file, TS.MAC is loaded whenever the TopSpeed Environment is invoked, so you can use this file to save all the macros that you want to retain between sessions. On exiting from the environment, you will be prompted as to whether you wish to save the current macro file, if there have been new macros defined.

Keyboard Macro File Format

Keyboard macro files (such as TS.MAC) saved using the **Save Keyboard macros** command (see above) are simple text files containing the macro definitions in an easily readable format. You may edit these files and reload them in order to edit macro definitions, create or delete macros, or add the special control commands described below.

The format of the file is a list of macro definitions, with all nonprintable keys represented by their name enclosed by '<' and '>'. Each macro definition is described exactly as it was recorded, including the initial and final Alt =. For example, the above examples would be represented thus:

```
<Alt=><AltZ>#pragma<Alt=>
<Alt=><Alt-><c><F10>C test.c<Enter><Alt=>
```

The names for the keys include the following:

```

<LeftArr>, <RightArr>, <UpArr>, <DownArr>,
<PageUp>, <PageDown>, <Home>, <End>, <Ins>, <Del>,
<ShiftTab>, <Enter>, <Esc>,
<F1>-<F12>,
<ShiftF1> - <ShiftF12>,
<CtrlF1> - <CtrlF12>,
<AltF1> - <AltF12>,
<CtrlLeftArr>, <CtrlRightArr>,
<CtrlUpArr>, <CtrlDownArr>,
<CtrlPgUp>, <CtrlPgDown>,
<CtrlHome>, <CtrlEnd>,
<CtrlIns>, <CtrlDel>,
<AltA>-<AltZ>, <Alt1>-<Alt9>.

```

To help minimize the size and improve the format of macro file, there are two additional rules:

- <Enter> is replaced by <|> and a new line is inserted in the file.
- Repeated keystrokes are represented as <name*nnn> where name is the name of the key and nnn is the number of times the key is repeated.

Thus, Enter, followed 8 cursor rights, is saved as:

```

<|>
<RightArr*8>

```

Special Commands

Often when recording a macro you want to pause the execution of the macro, either for a fixed time or until a certain key is pressed. To allow this, the environment provides a number of special commands that modify the behavior of a macro when played back. The format of these commands is '\$\$' followed by a command character and parameters if required. Table -14 lists the available special commands and their functions.

Key	Function
\$\$A key	Pauses macro execution until the specified key is pressed. While waiting for key, any other keys pressed are processed as normal by the environment. Note that the key waited for is not processed by the environment.
\$\$B key	Pauses macro execution until key is pressed. All keys up to and including key are consumed, i.e., are ignored by the environment.
\$\$C	Pauses the macro until any key is pressed. The key pressed is processed normally.
\$\$D delay	Waits for delay / 10 seconds before continuing. delay is specified as a decimal number, for example, \$\$D10 waits one second.
\$\$E	Waits for Esc to be pressed (this is equivalent to \$\$B<Esc>).

\$\$F	Waits for Enter to be pressed. The keys input up to Enter are processed normally (this is equivalent to \$\$A<Enter>). This is useful for getting input within a macro.
-------	---

Table -14: Special Macro Commands

Mouse Control

The TopSpeed User Environment is designed to be easiest to use and most efficient when being driven by the keyboard. However many pointing, moving and selecting operations can often be done faster using a ‘mouse’ pointing device. TopSpeed has a mouse interface which, without compromising the keyboard interface, allows you to perform many common actions using the mouse.

To work with TopSpeed, the mouse should be a Microsoft compatible mouse with an appropriate driver loaded. To enable the mouse within the TopSpeed Environment, you should set **Options Enable mouse** to on (shortcut **Alt - O N**). This option will be remembered (in TS . SES) for future sessions.

Mouse Actions

In this section the operations you can perform with the mouse are described. Throughout the following terms are used:

<i>Left</i>	means the left-most mouse button.
<i>Right</i>	means the right-most mouse button.
The center button (if present) is not used.	

<i>Click</i>	means click once on the specified mouse button.
<i>Double-click</i>	means click twice in quick succession on the specified mouse button.
<i>Drag</i>	means hold the specified button down and move the mouse cursor until the desired position is reached.

General Mouse Actions

<i>Left Click</i>	on an inactive window to select it (e.g. make it the active window).
<i>Left Click or Drag</i>	on a menu to move the menu bar to the mouse cursor.
<i>Left Double-click</i>	on a menu to select the menu option at the mouse cursor.

Left Click or Drag to move file selection, pick list or help cursor.

Left Double-click to select file or help topic at the mouse cursor.

Right Click or Double-click
to display the main menu.

Right Drag to move a window.

Mouse Actions while Editing

Right Click to display the editor menu.

Right Double-click
to display the main menu.

Left Click once to move the text cursor to the mouse cursor.

Left Double-click to mark the word at the mouse cursor.

Left Drag to mark a block at the mouse cursor.

Left Double-click on the either of the sidebars to the position the window within the file. The position moved to is the relative position in the file, e.g. **Click** above the position bar to move up and beneath to move down.

Left Double-click on the top bar of the window to zoom the current window.

Customizing the TopSpeed Environment

The TopSpeed Environment may be customized to your own particular taste and requirements. This can easily be done by editing the file TSCFG.TXT and then running the program TSCFG.EXE to compile this text into the environment data overlay TSDATA.OVL. Features within the environment that you are able to configure include the following:

- The environment and editor menu trees.
- Shortcut keys within both the Editor and the environment.
- Text displayed by the environment.
- Error messages output by TopSpeed compilers.
- The compilers installed in the system and the file extensions recognized.

As the contents of TSCFG.TXT are subject to change, the format is not described within this manual. See the documentation text file TSCFG.DOC for full details of TSCFG.TXT and how to run TSCFG.EXE.

Changing Windows

At any time, you can change the position, size, and color of the active window. You do this by pressing `ScrollLock` (or `Alt-W` - see below) which puts you into window control mode. You can then resize and reposition the window, and even recolor it by pressing `Enter`. To leave window control mode, press `ScrollLock` again. As you saw earlier, in addition to the window control mode, there is also instant window zoom/unzoom using the `F5` key.

`Alt-W` is provided for keyboards without a convenient `ScrollLock` key. It toggles the window in and out of window control mode in the same manner as `ScrollLock`.

Repositioning Windows

To reposition a window, use the cursor keys on the numeric keypad after pressing `ScrollLock` (or `Alt-W`). These move the window to the required position, uncovering any windows that may lie beneath. Of course full-sized, or ‘zoomed’, windows cannot be repositioned as they have no room to move. Provided that the environment doesn’t need to reposition a window for a specific context, the new window position will be saved in the configuration and session files. You can customize the layout of the editor windows by placing them, for example, side-by-side or on top of each other. The keys `Home`, `End`, `PgUp` and `PgDn` can be used to quickly reposition a window to the left, right, top and bottom edges of the screen.

Resizing Windows

You can resize each of the editor windows when in the window control mode. To do this, use `Shift` cursor keys (that is, `Shift ...`, `Shift f`, `Shift Y`, and `Shift ↓`).

When resizing a window, the top lefthand corner remains fixed, and the bottom and right edges move. In this way, `Shift Y` contracts and `Shift ↓` expands the height of the window. Similarly, `Shift ...` contracts and `Shift f` expands the width of the window. The size of each window is saved in the session file and restored when re-entering the environment.

Recoloring Windows

You can recolor the active window by pressing `ScrollLock`, then `Enter`. To recolor different areas:

- Select the area by pressing the `PgUp` or `PgDn` keys until the text in the required area flashes.
- Use `¼` and `|` to change the background and `;` and ``` to change the foreground colors.

When you find the desired color, you can either select a new area or you can exit window control mode by pressing `ScrollLock` again. If you want to abort recoloring, press `Ctrl-U` to restore the original colors.

The help line at the bottom of the screen (see “*Help Line*,” page) can be recolored by pressing `ScrollLock Enter *`.

The color of each window is saved in the configuration and session files.

If **Options High background** is on and you are using an IBM or compatible CGA display, then you can set the background to any of the 16 colors available including the bright backgrounds. Otherwise, only eight background colors are available on the CGA, together with 16 foreground colors.

Note: To make recoloring of the environment easier, windows having similar uses are grouped together into **window classes**. Recoloring any window of a window class will automatically recolor all other windows in that class.

Window classes include the following:

- Menu windows
- Error/Warning windows
- Prompt windows
- Input windows
- Directory/File selection windows
- Compiler/Make/Link windows
- Each editor window
- Utility windows

CHAPTER 4

VISUAL INTERACTIVE DEBUGGER

Introduction

If something is wrong in your program, you need:

- to go to the precise point in the program where things start to go wrong;
- to stop your program temporarily in mid-execution while you study what it has done;
- to look at your program's internal variables and see what has happened to them.

When you compile and run in the normal way it is impossible to do this.

TopSpeed's Visual Interactive Debugger (VID) lets you do all three of these things. It is a full source-level, symbolic debugger with an easy-to-use, but powerful, debugging environment. It lets you follow your program as it executes, keeping track of its variables and studying the changes it makes as they happen, step by step.

This is a powerful tool for finding errors, or studying the workings of a program or procedure.

VID is fully multilingual. You can debug programs written in any or several of the TopSpeed languages, provided they have been compiled by the TopSpeed compiler, switching from language to language as you proceed through the code. Functions or procedures compiled outside the TopSpeed system - for example library routines - can be skipped over, or stepped through in assembler. (We shall use the term `procedure` from now on to refer interchangeably to a function or to a Modula-2/Pascal procedure).

This chapter shows you how to install and use VID, and summarizes its commands. We'll assume that you're comfortable using the language that you are debugging, and that you've got the TopSpeed system installed, so that you can do the compilation and linking suggested in later sections.

The interface for VID is designed to be powerful yet straightforward. Debugging can now be fun as VID brings your source code to life!

What VID Will and Won't Do for You

VID won't find bugs by itself. It *will* give you a high-level view of your program's execution so you can get to the heart of the bug quickly. It offers three main facilities for doing this:

- A Source Window where your program is displayed just as you wrote it. VID shows you where program execution has reached, and where you can locate the part of the program you want to study and run until you reach it;
- A set of Examine, Display and Trace windows where you can display and modify variables and expressions;
- A powerful set of breakpoint commands which let you pause the program at the places you want to focus attention on.

Here are some of the things VID will do for you:

- Trace your program as it executes, either at high speed, or slowly to study it more carefully. You can execute any part of it step by step, either one statement at a time, or one procedure at a time.
- Pause the program at a selected statement while you study it. VID lets you set up to 100 or so such breakpoints.
- Set “one-shot” breakpoints, where the program will stop once, “sticky” breakpoints, where the program will stop each time it reaches the statement during execution, and expression breakpoints or watchpoints, where the program will stop when some condition is met, so that you can see, for example, when a variable is assigned an erroneous value.
- Study a particular variable or procedure, or automatically display all currently active variables using names from your program; you don't have to give an explicit address. You can view, trace or modify any variable.
- Watch variables and expressions so that their values are displayed continuously during program execution and updated when they change in the program.
- Evaluate expressions, using the standard operators as well as active variables from the program.
- Trace the sequence of procedure calls. You can see what procedure is executing, and display a list of all procedures currently suspended because of calls to other procedures.
- VID includes a powerful collection of commands for moving

around in your program. For example, you can go to particular statements, to procedures, to statements after returning from procedures, to breakpoints, and so forth. These commands make it easy to move to the part of your program which needs the most careful examination at any given time.

- The window containing this information also tells you the source line at which each of the procedures was suspended.
- You can trace your program at assembler code level and trace the contents of the machine registers. You can modify both high-level and low-level program elements including registers, memory, and the program and co-processor stack.
- You can disassemble the program and mix source and assembler display.
- You can cycle through the open windows. By making each window active in turn, you can track all aspects of the program's execution.
- Before running your program, you can initialize program memory to any byte value you specify. In this way, you can find errors that might occur because of uninitialized variables.
- You can view the contents of text files other than source modules. For example, you might need to consult the documentation or header files for a program you're debugging. VID makes this possible, even if the documentation file is larger than available memory.
- You can dump the entire screen or the active window to a disk file for later examination.

You'll find the debugger easy to use. Its command structure is very similar to TopSpeed's. You'll find pop-up menus, which generally can be activated by a single keystroke. However, you can also bypass these menus by using shortcut command keys generally one or two keystrokes. So let's get started!

Installing VID

If you have installed TopSpeed, VID will already be installed and you need take no further action. If you want to move VID to another directory, Just copy the VID.EXE and the VID.OVL files to wherever you wish to put them. Once installed, you can run VID from any directory, providing that the directory containing the EXE and OVL files is on your path.

Starting VID

VID needs a number of files from which it extracts the information used to run a debugging session. These files are:

- Source files for each module being debugged
- Debug (.DBD) information files for each module being debugged
- A map (.MAP) file for the program
- The program (.EXE) file itself

These files are all created by the TopSpeed system when you Make your project, provided the correct options have been set. The following section explains how to do this.

Making Your Program with Debug Information

Before running VID on a program, you need to ensure that you have remade your program with VID information enabled. This is done by setting **Project VID level** to `full` using the menu system within TopSpeed (shortcut `Alt-P V`), or set the compiler option `/V2` if you are using the batch command line.

VID will also use a MAP file, if present, for labels displayed in assembler. This can be generated by turning **Project Options Map file** on (shortcut `Alt-P P M`).

You can set or clear VID information for individual modules by using the `pragma debug(vid)` either in the source or in the project file. See the “*TopSpeed Developer’s Guide*” for further details of this pragma.

Running VID

After you have compiled and made your program, the following should exist and be accessible to VID.

modulename . * The source for each module that you want to debug ,

modulename . DBD

The debug information file for each module you want to debug,

`progname .MAP` The map file for the program and

`progname .EXE` The `.EXE` file for your program

VID can locate these files through the redirection file `TS .RED` (see “*The Redirection File*,” page).

You are now ready to invoke VID. From the DOS or OS/2 command line, the format of the command is as follows:

```
VID [/option] progname [progcommandline]
```

For example, the following are all valid command lines to invoke VID:

```
vid viddemo
vid /s3 grtest
vid /s0/f255 myprog myparams
```

You’ll learn what these options mean very soon.

When you are used to VID you can customize the default command line options and the window coloring using the menu-driven program `VIDCFG .EXE`. See “*Customizing VID*” at the end of this chapter.

Running VID from Within TopSpeed

To invoke VID from TopSpeed, you can type `Alt-V` to start the debugger. You’ll be prompted for the name of the program to be debugged, which may be followed with command parameters for the program.

When run from within TopSpeed, VID will prompt you to save any edited files; this will prevent you from losing data if your program crashes while being debugged. VID then proceeds to swap TopSpeed to disk in order to provide the maximum space possible for debugging. Because of this, it is advisable to ensure that you have plenty of free space on your disk before running VID. A minimum of 1MB free space is recommended.

On exiting from VID, TopSpeed is reloaded from disk, and you are returned to the point at which you invoked VID.

After it has started running as a program, VID reads the `EXE`, `MAP`, `DBD` and source files. From these, VID generates a temporary symbol file that contains (in compressed form) all the symbolic information contained in the `EXE`, `MAP` and `DBD` files. This file (with the name `progname .DB$`) is generated only when your program changes. This saves time when starting up VID. You can delete it at any time if you wish, as it can always be recreated - a process that takes only a second or so.

After VID has made the symbol file, it presents you with the normal VID view of the source of your program about to be executed. Also a small

“Welcome” banner is displayed which shows the program name and version number. This banner will disappear after the first key is pressed.

The program is now ready for you to debug using all of VID’s facilities.

Before the detailed description of VID’s commands, let’s look briefly at VID’s design and at some of its general features.

Introduction to VID Keys, Menus and Windows

VID can display a large amount of information about your program. If all this information was constantly available, the screen would look extremely cluttered. VID strikes the balance between information and clutter by giving you full control over what is displayed.

This is done with multiple overlapping windows, each containing a specific type of information (for example, the procedure call stack or a list of traced variables).

Giving Commands in VID

You can give commands to VID in one of two ways:

- You can use the pop-up menu system. To invoke the main menu for this, press *F10*. This menu system will be familiar from TopSpeed. Browse around the menu, and take a look at all the commands VID provides for you. More details about particular commands are provided later in this chapter.
- You can use “shortcut” key sequences. These have the same functionality as the menu commands but are quicker to type. You’ll probably use the menu system when you first start using VID. As you become more familiar with VID, you’ll probably find yourself using shortcuts more and more. As in TopSpeed, a menu will pop up if you type just part of a shortcut sequence.

Special Keys

In addition to the menu shortcut keys, certain other actions have been assigned single keys. These include actions (such as setting breakpoints) that are performed frequently when debugging. The bottom two lines of the VID screen contain a small *help window* that briefly lists some of these keys.

This help window will generally show which keys are applicable to the particular “mode” you are in. When you become more experienced with VID’s commands, you may find it useful to remove these help lines to give yourself more room for other information.

Command Help Key

To get a list of special keys and top-level menu commands, press the *F1* key. This puts up a “Key help” reference list, as shown in Figure -1.

You can press the *ESC* key to exit from this list (or from any other VID window), or you can press one of the keys listed to execute that command.

Expressions, Operators and Variables in VID

Before we start looking at the VID commands and options, it will be useful to discuss some points relating to expressions and how VID handles them.

Many of VID’s commands work with variables or *values*. You may often want to take actions depending on the value of an expression that uses variables from your program. You may, for example, want to do one thing if an expression is negative and another if it is positive.

Expressions play a major role in VID’s operation. VID lets you:

- Display the value of an expression.
- Watch expressions in a watch window as your program executes.
- Set breakpoints triggered by expressions becoming true.
- Execute your program until an expression becomes true with a **Go to Expression** command.
- Set some variable equal to the value of an expression.

The next two sections briefly describe how VID expressions are constructed.

Operators

VID’s operators are language independent, derived from a combination of C/C++ and Modula-2/Pascal operators. This means, for example, that pointer indirection can either be expressed using C’s prefixed ‘*’ or Modula-2/Pascal’s postfix ‘^’. This gives rise to no major clashes, the only substantive difference being the use of ‘~’ as exclusive OR when infix.

VID will accept expressions involving the most simple numeric and logical types and operators. Operator precedence is the same as that defined by *TopSpeed Modula-2* and *TopSpeed C*, and you may use parentheses to change ordering.

Types allowed in VID expressions include:

- 1 byte, 2 byte and 4 byte integers

- 1 byte, 2 byte and 4 byte unsigned integers (C/C++ unsigned types, Modula-2 CARDINAL types).
- 1 byte characters
- boolean
- enumerated types

Character, boolean and enumerated types evaluate the ordinal value of the variable's contents. For example, a character variable containing 'a' has the value 97; a boolean variable that is FALSE has the value 0. All types are converted to 4-byte integers to allow mixed types in expressions. For example, given i=-2, k=5 and c=0, the expression i+k+c evaluates to 51, irrespective of the sizes of i, k and c.

Operators allowed in expressions are listed in Table -1.

Key	Function
•	add
-	subtract
*	multiply
/ or DIV	divide
% or MOD	modulus (remainder)
<<	shift left
>>	shift right
	bitwise OR
&	bitwise AND
~	bitwise XOR (when infix)
~	bitwise NOT (when prefix)
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
== or =	equal
!= # or <>	not equal
! or NOT	logical NOT (prefix)
&& or AND	logical AND
or OR	logical OR
*	indirection (prefix)

^	indirection (postfix)
->	point at member
.	select member (record field)

Table -1: VID Operators

Logical results return TRUE (1) or FALSE (0) and may be combined using the logical operators to provide logical conditions. When supplied to *expression breakpoints*, such expressions allow you to set complex conditions for breaking the execution of your program.

Variables

Variables used in expression may be any of the basic types listed above; these include array elements, record fields and dereferenced pointers. For example, the following are valid VID expressions returning TRUE or FALSE:

```
(n<>0) && ((a[j]->tag<=0) || (i*2>j/n))
(n<>0) AND ((a[j]^tag<=0) OR (i*2>j DIV n))
```

Using the ‘Default Language’ option (see page) you can display variables and expressions in a form compatible with a specific language.

When using expressions in VID as the object of a **Watch** or a **Go to Expression** command, keep in mind that the variables names in an expression refer to the objects in scope when you entered the expression.

If any of them go out of scope, then the expression is deemed to have no value ie: will not appear in a *watch window*, and will not terminate a **Go to Expression** command. When the variables return into scope, however, the expression’s value will become valid again and will be redisplayed in the watch window. This applies even if a procedure has returned and has then been called again.

To specify variables in expressions that are not currently in scope (i.e., are not directly visible for the procedure currently executing), you can prefix the variable name with a procedure name and/or a module name.

To specify the module of a variable, you should prefix the variable name with the module name followed by ‘.’, for example, `testmod.level` specifies the global variable `level` in the module `testmod`.

To specify the procedure that a local (or static) variable belongs to, you can prefix the variable by the procedure name followed by ‘: :’, for example, `sortnames::index` refers to the variable `index` in the procedure

sortnames. You can refer to procedures in other modules by prefixing the procedure name with the module name, for example,
`testmod.sortnames::index.`

A local variable only has a value when the procedure it belongs to is ‘active’, i.e., it is currently on the calling chain. If a procedure is called recursively, its most recent incarnation will be referred to.

VID Command Line Options

You have several options that you can specify on the command line when starting VID:

Command	Option
/Sn	<i>Screen swap options</i> tell VID what kind of a screen swap you need O that is, what kind of screen your program is using. The default setting is /S1, but this can be changed by using the default command line option when you install VID.
/S0	No swap (/ refreshes the screen)
/S1	Page (hardware) swap; default for CGA text and compatibles (EGA/VGA).
/S2	Screen (software) swap; default for monochrome.
/S3	CGA graphics swap (slow).
/S4	General graphics swap (EGA/VGA) (slow).
/S5	Hercules graphics swap.
/S6	Two monitors/adaptors.
Note: /S3, /S4 and /S5 require more memory, which means there is less space available for VID to use.	
/BW	This option forces <i>monochrome mode</i> for the video. VID will automatically detect monochrome modes. You can use this mode to force monochrome operation if you use a monochrome monitor with CGA.
/K	This “snow check” option turns on snow suppression for CGA monitors that produce flickering.
/Ennn	This command causes VID to run partially in <i>expanded memory</i> . VID will also use expanded memory to buffer symbol information. This will add about 64K to the memory space available in which your program can execute. Also, debugging large programs will be faster as symbols will not need to be swapped in from disk.

The number *nnn* specifies the number of 16K EMS pages to allocate to VID. The minimum effective value is 4 (64K) and any value up to the capacity of the memory board may be specified (128 for 2MB). For example, /e64 says to use 1MB of EMS.

Note: To use this option, you must have an expanded memory board and a LIM EMS driver installed. If your configuration does not meet these criteria, the option will have no effect. If enough memory is not available, then the maximum amount available is used.

- /Fnnn This is used to specify the value with which to (byte) fill the program space. The default value is 0. Hexadecimal format is allowed; for example, /F255 = /FFFH. If no fill is required, then /F- turns off the fill.
- /R This forces a remake of the .DB\$ compressed map file. Normally, this file will be remade only when the map file changes.
- /A This starts up VID in assembler mode (i.e., with source mode turned off, see “*Option Source*”); this is useful when you want to debug through the assembler initialization code of your program; This is because instead of taking you to the first source line of the program, VID will start from the first assembler instructions to be executed.
- /P Starts VID in post-mortem mode. This reads a post-mortem dump file created by a TopSpeed program, which allows you to examine the data and execution state of that program at the point where it made the dump. See the TopSpeed “*Advanced Programmer’s Guide*” for details on creating post-mortem dumps from within your programs.

Note: Only examining commands may be issued in post-mortem mode, you cannot change memory, set breakpoints, or continue execution. The size of the post-mortem dump is not limited to available memory; hence you are able to examine the state of programs larger than would normally fit within VID.

When invoking post-mortem mode, the file `PM_DUMP.$$$` will be read and will automatically display the program that made the dump.

Note: The original .EXE, .MAP, source and .DBD files associated with the program that made the dump should all be available, as if you were debugging normally using VID. If mis-matching (i.e., not the original) files are supplied to VID, then VID will probably display incorrect information.

The Source Window

The *source window* is the main area of VID's screen used for displaying the source of the program being debugged. It contains two cursors which display and control what the program is doing.

The first, called the *execution cursor*, shows the source line about to be executed. This cursor is blue on a color screen or in inverse video on a monochrome monitor.

The second, called the *user cursor*, is the cursor you use to move around physically within the source. You also use this cursor to show VID where you want to put traps, etc. The user cursor is red on a color monitor and an underline on a monochrome monitor. When the two cursors coincide on the same line, the execution line takes precedence.

The source window has a header that shows the name of the file currently displayed and the line number within this file. The line number changes if you move either of the two cursors available in the window.

The source file text is displayed within the central area of the debugger's screen. You can use normal cursor keys to move around within the source text. When you move around in the text, you'll observe the two bar cursors highlighting lines of text.

This area is also where the monitor count is displayed when a "monitor" breakpoint is set.

You have a rich collection of commands available for moving around in your source program and for controlling its flow. In the next few pages, we'll summarize the commands available.

Commands for Moving in the Text

You can use the following keys to move the *user cursor* around in the text:

<i>j</i> <i>^</i>	Move the user cursor up or down a single line.
<i>¼</i> <i> </i>	Scroll the source window to the left or to the right.
<i>Home</i>	Return source window to the left edge.
<i>End</i>	Ensure that the right edge of the text is within the window.
<i>PgUp</i> , <i>PgDn</i>	Move the user cursor up or down a page through the source.
<i>Ctrl-PgUp</i>	Move the user cursor to the first line of current file displayed.

<i>Ctrl-PgUp</i>	Move the user cursor to the last line of current file displayed.
<i>Ctrl-Home</i>	Move the user cursor to the location of the execution cursor.
<i>Ctrl-G</i>	Prompts for line number to move to (or address in assembler mode).

The movement resulting from these commands is based on the physical location of lines. This is in contrast to movements based on the flow of the executing program, as in the next set of commands.

Commands for Moving in the Running Program

The following keys have an effect on program execution and effect the *execution cursor*:

<i>spacebar</i>	Pressing the spacebar will single step (that is, execute) a single action in your code and will then move the execution cursor accordingly. After the single step, the user cursor and the execution cursor will be on the same line. Note that the “size” of an action depends on whether or not the Trace Assembler window is open. If this window is open, an action will consist of a single line of assembler code; if the window is not open, an action will be a single line of source code.
<i>/</i>	<p>This command swaps to the output screen of the program being debugged. This allows you to see any output that the program has produced up to the current point in the program. You can invoke VID with a variety of different screen swap options that let you switch between various modes of display (for example, text & graphics) as described in “<i>Calling VID from the Command Line</i>,” page . To get back to VID’s screen when you have swapped to the program’s screen, just press any key.</p> <p>If screen swapping is disabled (that is, if option /S0 was used), then the / command will not swap screens but will refresh VID’s screen. This ensures that any information on the screen overwritten by the program is redisplayed.</p>
<i>P</i>	This key moves the execution cursor but unlike spacebar, it steps over procedure calls. If <i>P</i> is pressed with the execution cursor on a procedure call, the procedure will still be executed, but VID will not trace through the execution. Any breakpoints you had set will remain active, so they won’t be missed.

CtrlBreak

This command will pause the program being debugged and will return control to VID. You can use this command when the program tracing is continuous (see the section on “*Go Trace*”) or when the program is executing without VID tracing. Note that VID cannot break into DOS, so if your program is waiting for input you may have to press a key (or *Enter*) before the program is paused.

If the program stops in a module that is disabled or that is not being debugged, then the execution cursor will show the “nearest” calling point in an enabled module’s source. Together with the **Trace Procedure** calls window (described later in the chapter), this information will enable you to find the point at which the program was executing when *CtrlBreak* was pressed.

Commands for Handling Windows and Screens

VID is a multi-windowed system that can display many windows on the screen at the same time. These windows can overlap, so only the top window is guaranteed to be totally visible. The top window is called the *active window* and can be differentiated from the other windows by the fact that it has a double frame while all the others have single frames.

To close the active window, press *ESC*. This removes the window from the screen (uncovering anything beneath) and also terminates the function associated with the window tracing or input, for example. The next window “underneath” then becomes the new active window.

You can re-order trace windows on the screen, by using commands described in a later section. With the appropriate command, you can bring any window to the top. You can also “cycle” most windows (that is, bring the bottom window to the top) by pressing *F6*. Input windows (for example, menus and examine windows) cannot be cycled. They must remain the active window until they are closed by *ESC*.

In addition to opening and closing windows, you can move them, and can even save them to disk or send them to a printer. To move windows around on the screen in order to uncover information hidden beneath, you need to enter the *Window Control Mode*. Press the *ScrollLock* key to do this. In this mode the cursor keys will move the active window around.

When you have positioned the window in the correct place, press *ScrollLock* again to get out of Window Control Mode.

The *Window Control Mode* also provides facilities for dumping either an entire screen or the active window to a disk file or printer. This logging

capability allows you to easily append the results of debugging to a capture file for future reference. The following commands are available in this mode:

<code>↑ ↓ ← →</code>	These keys move the top window in the specified direction. In this way you can relocate the windows in your debugging environment.
<code>W</code>	This command writes the contents of the active window to the current dump file or to the printer.
<code>S</code>	This command saves the contents of the entire screen to the current dump file or to the printer.
<code>N</code>	This command lets you specify a new target name. Text saved by subsequent calls to <code>W</code> or <code>S</code> will be appended to the new dump file. Remember, you must be in window moving mode for this command.

If no dump file has been specified, you'll be asked to enter a file name. If you enter the filename `PRN`, VID will send all dumped text to the printer. If the dump file already exists (from a previous session) you will be asked whether you wish to **A**ppend to this file, **O**verwrite the file or **Q**uit the function.

Once you specify a dump file, subsequent window or screen dumps during the same session will go to the same target. For example, if you had specified a file named `DBGDMP` as the target, the contents for each `W` or `S` command in window moving mode would be appended to file `DBGDMP`.

If you want to send each dump to a different target, you need to specify a new target name between dumps using the `N` command.

Note: You do not have to close the dump file, so that even if your program should crash the machine, your logged data will remain intact.

Commands for opening various windows are discussed in the appropriate sections.

Commands for Exiting VID

You can terminate a VID session and return to the DOS or OS/2 command line or to TopSpeed at any time. To exit VID, you can press `Q` or `Alt-X`.

Additional Commands for Moving in the Source File

In addition to the commands available for moving on the screen, you can move between procedures in the source file using the *F7* and *F8* keys. *F7* takes you to the previous procedure called before the current cursor position; *F8* takes you to the next procedure called after the current cursor position. You can also specify these commands on the **Source** menu: type *S* for the Source menu, and then select *N* or *P*, respectively, to move to the next or previous procedure. In each case, the cursor is taken to the first procedure line that has executable code. (You can also get to the Source menu by means of the main menu, *F10*.)

You can use *F9* (or *L* on the **Source** menu) to find a specific procedure. You'll be asked for the name of the procedure to be found. You can enter a procedure name, and VID will move the cursor to the beginning of that procedure. Or you can press *Enter* to get a list of the procedures in the current file on display.

You can also specify *wild cards*. In that case, VID will open a selection window from which you can select the required procedure.

Note: '***' will match all procedures in the current file on display while
 '**. **' matches all procedures in all modules.

You can even find a particular string in the source text of the current module. This string may contain wild cards but must be on a single line. To do this, press *Ctrl-Q F* (or select *F* from the **Source** menu).

If you need to check the contents of another file while running VID, you can use the **Source View** file command to call up this file. The file can be any text file it need not be a source file. You can use the cursor keys to move through this file, but you can only read the file. When you're finished with the file, just press *ESC*.

Highlighting and Marking Lines

You've seen that the line corresponding to the current execution cursor position is highlighted. Although we've been taking it for granted that this is the case, the execution cursor is highlighted only because the **Options (C) Highlight code** is set to on.

When this option is on, any source line for which the compiler has generated code can be highlighted. What this means is that the execution cursor can be moved to any line for which there is code. This shows where traps may be placed in the code since you cannot place a trap where there is no code, as it will never be executed!

You may find sections of code that are not highlighted even though you would expect these sections to be executable. The code may not be highlighted because smart linking or smart compilation made it unnecessary ever to execute those sections of code. Checking the highlighting of your source often proves illuminating while debugging, since you can see exactly what code is and isn't used.

Note: VID only deals in whole lines. If you have multiple statements on a single line, then they will be treated as a single unit for the purpose of traps, etc. This was done to make the selection and display of lines much simpler for the user. If you have multiple statements that you need to differentiate, then you will have to adjust the source to ensure that each statement is on a separate line.

To the left of the source, two columns are reserved for *breakpoint indicators*. These show the position and type of breakpoints set on the line to the right. The following indicators are used:

- One shot breakpoint
- Sticky breakpoint
- n Delayed breakpoint with n to go ($n < 10$)
- * Delayed breakpoint with 10 or more to go
- n Delayed sticky breakpoint with n to go ($n < 10$)
- * Delayed sticky breakpoint with 10 or more to go
- ? Expression breakpoint
- + Monitor breakpoint

See the “**Breakpoints**” section below for more detail on the different types of breakpoints and how they are set.

Assembler Display

When you have **Option Source** turned off or you are single-stepping through code without source available, VID displays a disassembled listing within the source window. This allows you to page through the assembler and set breakpoints in the same manner as if you were debugging high-level source.

In this mode, the commands to move the execution and user cursors are generally the same as described above. You can use *Ctrl-G* to set the segment and offset of the user cursor and hence the location of the disassembly displayed.

VID's Menu Structure

VID has a two-level menu system. That is, the debugger has a main menu level, whose selections you can see by pressing *F10*. This menu is shown in Figure 4-2.

These selections represent the first menu level. All but the bottom two commands put you in a second menu. You can also reach a first-level menu directly (that is, without going through *F10*) by pressing the first letter of the menu's name. For example, press *B*. After a second or two, the **B**reakpoint menu pops up.

This menu contains the second-level selections. Again, you can select a command simply by pressing the letter associated with the command.

The delay before the pop-up menu appears makes it possible for you to speed up the process of selecting a second-level menu choice. For example, suppose you wanted to move the execution cursor to the current user cursor. Instead of pressing *G*, waiting for the Go menu to come up, and then pressing *C*, you can simply press *G* and *C* in quick succession. VID will bypass the menu presentation and will execute the command directly.

You can give any command using one or two keystrokes. Once you're familiar with the commands, you won't ever need to use the menus.

As you saw earlier, a number of common VID commands have been assigned to single keys. You can see a summary of these keys and their functions while running VID. Just press *F1*. This opens a "key help" window, which gives a short description of all the VID keys. After you press *F1*, your screen will look like the one in Figure -1.

Press any key to close the window. If you press one of the command keys, the help window will close and that command will be executed. If you just want to close the window without invoking a command, press *ESC*.

The next few sections provide a summary of the VID menus and the commands available in each of these menus. First, we'll look at VID's main menu.

VID's Main Menu

As you saw earlier, you can press *F10* if you want access to the VID's *Main menu* system. From here, you can reach all the submenus, although each is accessible by a more direct command. The main menu also contains the commands *Q* to exit VID and */* for screen swap, both of which were mentioned earlier.

To select an entry on the main menu, or on any other menu in VID, you can do either of the following:

- Move the bar cursor to the desired entry and press *Enter*.
- Type one of the characters that are displayed and highlighted. This will immediately invoke that selection. To leave a menu or submenu without making a selection, press *Esc*. This will close the menu window and return you to the window in which you were working before opening the menu.

The main menu lists six submenus and two other commands. The selections available from the main menu are:

B Breakpoint menu

D Display menu

E Examine menu

G Go menu

S Source menu

T Trace menu

O Options menu

/ Swap screen

Q Quit VID (*Alt-X*)

Each of the menus is described in detail in a later section, along with the commands available in each menu.

The */* command swaps between VID's screen and the program output screen.

The *Q* command will quit from VID and put you back at the DOS command prompt. If you are in the middle of a debugging session, you will be asked to verify that you want to exit from VID. This is to prevent you from unwittingly leaving VID by accidentally entering the wrong command.

You can also use *Alt-X* as a shortcut key to exit. This provides compatibility with TopSpeed.

Another way to leave the environment is to use **Go eXit** (*G X*). This command will take out all breakpoints but will continue running the program being debugged until its conclusion.

The Breakpoint Commands

Breakpoints are “software traps” that you can insert at any statement within your program provided the statement has executable code associated with it. A software trap will pause your program when it is either tracing or executing continuously and will then transfer control to VID. Depending on the type of breakpoint, VID will either allow you to examine, single step or restart the program (using one of the “Go” commands), or VID will perform some breakpoint counting operation.

You can set any of several types of breakpoints. These types are: *One-shot*, *Sticky*, *Delayed*, *Delayed/Sticky*, *Expression* and *Monitor breakpoints*.

You can have only one breakpoint on each source line even if you have multiple statements on that line. If you need to set independent traps on these statements, you must separate them onto different lines within the source.

When setting a breakpoint on a line, any previous trap present on that line is automatically removed before the new breakpoint is set. This makes it very easy to change the type of breakpoint.

To set (or clear) a breakpoint, you simply “point and shoot.” Move the cursor to the line on which you wish to place the breakpoint, and then give the appropriate breakpoint command.

You can use cursor/page movement and/or search commands (as described earlier) to move through the source. You may also need to change modules if you want to put the breakpoint in to a different module than the one on display. (To change modules, you can use **Source Change Module**, or *F3*, as described on page).

Once the user cursor is positioned on the target line, then the commands described below can be used to set (or with the *C* command, clear) any type of breakpoint. The three most common operations, which are setting a “one-shot” breakpoint, clearing a breakpoint and setting a “sticky” breakpoint, have been assigned single keys namely *+*, *-* and ***, respectively.


The Breakpoint Menu

The *Breakpoint menu* (accessible by pressing *B*) contains the commands shown in Figure -3, and enables you to set any of the types of breakpoints mentioned earlier:

Set Breakpoint (+ or B P)

This sets a “*one-shot*” *breakpoint* on the statement at the user cursor. This breakpoint will cause execution to stop when the statement is next executed. Once the statement executes, the breakpoint is automatically removed.

When the program “hits” a breakpoint, note that VID pauses the program *before* the statement is executed. To execute the action of the statement, you can single step through it by using the spacebar.


When a statement has a “one-shot” breakpoint set, the  symbol is displayed to the left of the statement text. This indicator remains attached to the statement while the breakpoint is in effect.

Clear Breakpoint (- or B C)

This command will remove a breakpoint set on the source line under the user cursor. You can remove any type of breakpoint with this command. Once removed the breakpoint indicator will disappear from the left columns, indicating that there is no longer a breakpoint on the line.




Set Sticky Breakpoint (* or B S)

This command sets a “*sticky*” *breakpoint* at the user cursor. This has the same properties as a one-shot trap, except that it is not removed when the statement has executed. This means the breakpoint continues to trap the line on which it is set until you explicitly remove it.

Sticky breakpoints are useful when, for example, you want to trap every time a certain procedure is entered. By making the breakpoint sticky, you save having to re-enter the breakpoint each time it is executed. A sticky breakpoint is represented by the  character in the indicator columns.

Set Delayed Breakpoint (B D)

This command sets a “*delayed*” *breakpoint*, which has the same effect as a one-shot breakpoint except that the delayed trap only “triggers” (pausing the program) after the statement has been executed a specified number of times. When you set a delayed breakpoint you are prompted for a “delay” number, which can be anything up to 65535. For example, if you specify 10, the trap will trigger after the statement has been executed 10 times.

The presence of a delayed breakpoint is indicated by the characters  or  where *n* is a whole number between 1 and 9, inclusive.  is used when there are nine or fewer occurrences left before the breakpoint is triggered. This counter will be continuously updated if you are single stepping or tracing the program.

When the breakpoint is finally triggered, after the nth execution, the breakpoint will be removed automatically. Such a delayed trap is very useful for debugging loops, to ensure that you get the right iteration without having to step through all the execution preceding the iteration in which you're interested.

Set Delayed/Sticky Breakpoint (B K)

This type of breakpoint is like a delayed breakpoint except that the “sticky” version is not removed after it has been triggered. Rather, the sticky breakpoint is reset to its original countdown value. This can save you the trouble of having to reenter the original breakpoint when carrying out the same function repeatedly.

As with an ordinary delayed breakpoint, you need to specify a delay value. Such breakpoints are indicated by * or by n, depending on whether the delay value (n) is greater than 9.

Set Expression Breakpoint (B E)

This command prompts for an expression and then sets a “sticky” conditional breakpoint at the user cursor position. This is like an ordinary sticky breakpoint except it will only be triggered, if the expression you entered is TRUE (non-zero) when the statement is executed. When the breakpoint is triggered the window containing the expression is displayed.

Since the breakpoint is a sticky breakpoint you must use the **Breakpoint Clear** breakpoint command to remove it. An *expression breakpoint* is represented by the ? characters in the indicator column.

This type of breakpoint is extremely useful for checking program assertions and also for stopping the program when a certain state is reached. Because the expression is only evaluated when the breakpoint is executed, the expression breakpoint should be used in preference to **Go until Expression** (see page) whenever possible.

Monitor Breakpoint (B M)

This breakpoint does not pause the program. Rather, a *monitor breakpoint* reports continuously (on the top status line of the source window) the number of times the breakpoint has been executed so far.

This is useful for checking whether program flow is occurring as expected. You can also use it in conjunction with delayed breakpoints in order to determine the delay number to set. The + symbol is used to indicate the presence of a monitor breakpoint.

Clear All Breakpoints (B A)

This command provides a quick way of clearing all breakpoints currently set in your program. After such an action, the program is free to continue without hindrance. Such a command can be used in conjunction with the **Go ReStart** command to restart your program with a “clean slate”.

The Examine and Display Commands

The *examine commands* allow you to look at your program’s data. You can even change this data, regardless of whether this is in variables, memory locations or processor registers.

The *display commands* provide exactly the same facilities as the examine commands except your programs data cannot be changed. This allows you to browse through variables and memory without the possibility of accidentally corrupting the data.

The data are presented in a “spreadsheet-like” window, which is opened when you give an examine or display command. Within this window there are a number of rows and columns with a single cell for each data element.

One of the cells is shown in inverse (possibly with a flashing cursor in it). This is called the *cursor cell*, and refers to the currently selected data element. If the cursor is flashing within this cursor cell, then you may edit the value displayed. If you accept the edit (by pressing *Enter* or moving to another cell), the new value will be written back into memory.

To move between cells (and pages of cells), use the usual cursor and page movement keys.

Values are displayed in symbolic form, and you can also enter them in this form.

Notice that structured types (such as arrays, structures, records and sets) and pointers may have sub-elements that are not immediately accessible or displayed. You can “follow” such variable types to get to the sub-elements.

To do this, first move to the variable cell and press *Enter*. The display changes to show the new selections’ contents. The value of the desired sub-element may be displayed. If not, it may be necessary to “follow” again.

You can use the cursor keys to move to the individual cells of arrays. The indicator at the top of the window tells you which cell you’re currently examining.

To close a variable display, press *ESC*. This takes you back to the “parent” display from which you selected the variable. If there is no parent display,

then *ESC* will close the Examine window. To exit from a deeply nested variable, you can press *Alt-Q* which will immediately close the examine window without having to completely retrace your steps to the first window opened.

Unions (variant records in Modula-2 and Pascal) have all alternative members (variants) displayed even if their storage locations overlap. This means that changing one field may have an effect on other fields.

For all memory display types, the address of the current cell in memory is displayed in the lower right corner of the window's frame. The title at the top of the display window shows what object or objects are currently being displayed, and, in the case of array elements, shows the current cell's array index. The leftmost column of the display also gives information on the contents of each row in the display.

ASCII strings are displayed (and input) as a contiguous line of characters. To change the characters, you can just type over the material contained in the cell.

If you need to enter a non-printing character, press *Alt-N*. A window will appear, prompting you for the ASCII code of the character you wish to write at the current cursor position in the string. The character you specify will replace the character in the variable. To enter the NULL character (ASCII 0), type *Alt-O*. When present in the string, this character is displayed as `|`.

For unsigned (Cardinal) and signed integer variables of 1, 2 or 4 bytes, you can enter expressions to calculate new values for the variables. See the section on expressions on page for more detail on the format valid expressions must have.

Variable Type Modifiers

When examining individual variables or components of variables, you can override the default variable type by using one of several *type modifiers*. These can be specified by prefixing the variable name by the sequence '*C*:' where *C* is one of the modifiers listed below.

- | | |
|----|---|
| A: | ASCII string (i.e., array of ASCII characters) |
| B: | Byte array (i.e., array of unsigned bytes) |
| I: | Integer array (i.e., array of signed 16 bit integers) |
| W: | Word array (i.e. array of unsigned 16 bit integers) |

For example, to display the variable *v* as an ASCII string, you would enter the string `a:v`. You can also apply these modifiers within the examine window by using the following keys:

<i>Alt-A</i>	Apply 'A:' ASCII array modifier.
<i>Alt-B</i>	Apply 'B:' byte array modifier.
<i>Alt-I</i>	Apply 'I:' integer array modifier.
<i>Alt-W</i>	Apply 'W:' word array modifier.
<i>Alt-C</i>	Clear any modifier currently set.

In addition you can use the keys *Alt-H* and *Alt-D* to change the default display base to hexadecimal and decimal respectively.

Objects in VID

When an *Object Variable* (i.e. an instance of a class) is displayed within VID, then the first field of the display window contains the class name (if known). This is the *static* name of the class, i.e. the class used to declare the object.

However if the object being displayed is indirect (i.e. is a de-referenced pointer) then it is common for the actual class of the object to be a different (derived) class. This class, called the *dynamic class*, can be displayed by pressing **Enter** in the first field, which will expand the window and display any extra fields defined in the dynamic class.

If pressing **Enter** has no effect, then the static and dynamic classes of the object displayed are the same.

Examine Menu

The commands accessible through the *Examine menu* are shown in the next figure.

Examine Variable(? or E V)

This command prompts for the name of a variable to display. You may specify any variable currently in scope or a variable prefixed with the module name (for example, `mod.variable`). By prefixing a variable name with a module name, you can access global variables belonging to modules other than the one currently on display.

When you specify the name and press *Enter*, the variable value(s) will be displayed, ready to be changed if necessary. The display window title indicates each variable's "owner." This will be either a procedure (for local variables) or a module (for global variables).

You can also enter VID expressions instead of a variable. In that case, the evaluated expression will be displayed. See "*Expressions*," page for further details.

Global Variables (Alt-F3 or E G)

This displays a list of all the *global variables* of the module currently on display in the source window. Structured variables may be selected as described above by opening a full display window. Other variables will have their values displayed in the list. These values can be changed by moving the bar cursor to them and editing the value.

Note: You can only access the global variables of modules that are enabled for debugging.

Procedure Parameters (Alt-F4 or E P)

This command is like the one for global variables except that *Alt-F4* displays a list of all the parameter variables of the procedure currently being executed.

If you wish to examine the parameters of the procedure one above in the “calling chain”, then you should select the first line of the display window (the line marked “. .”). After that selection, VID will display the parameters of the calling procedure. You can repeat this process to follow all of the calling procedures’ parameters.

Local Variables (Alt-F5 or E L)

This command displays a list of all the *local variables* of the procedure currently being executed. As was the case with procedure parameters, you can look at the local variables of calling procedures by selecting the “. .” entry.

Instance Variables (Alt-F6 or E I)

This command displays the fields of *SELF* (for *Pascal* or *Modula-2*) or the members **this* (for *C++*) for the instance of the class to which the active procedure belongs. This is a quick and easy way of displaying the current contents of the object being debugged.

If this command is invoked for a procedure that is not a method of a class, then an error message is displayed.

Disassemble Statement (E D)

This command produces a mixed source and assembler display window. The contents are based on the statement at the current position of the user cursor. You may page backwards and forwards through the display throughout the current module.

Evaluate Expression (E E)

This command displays the value of an expression entered. See the description of “*Expressions*,” page for more information about what you can enter, and the format this must have.

Stack Words (E S)

This displays the stack as hex words based from SS : SP.

Memory Bytes (E B)

This prompts for an address (or the name of a variable on which to base the address), and then displays the contents of memory as hexadecimal bytes.

Memory Words (E W)

This prompts for an address (or the name of a variable on which to base the address), and then displays the contents of memory as hexadecimal words.

Memory ASCII (E A)

This prompts for an address (or the name of a variable on which to base the address), and then displays the contents of memory as ASCII characters.

Registers (E R)

This allows you to display and change the *processor registers*. Registers are displayed as hexadecimal words. The following registers are displayed:

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

FL is the processor flags word. Note that CS, IP, SS and SP may not be changed as they are restored by VID when resuming the program.

In addition to examining registers, you can also get a permanent display of the registers by using the **Trace Registers** command, described later.

Floating Point Stack (E F)

This command displays the contents of the 80x87 floating point stack (or the 80x87 emulator stack if you have no 80x87 installed). These (eight) values are in 80x87 `TEMPREAL (10-byte)` format, and they show temporary values in use by the program currently executing.

The Display Menu

The display menu provides the following commands:

Key	Command
D V	Display Variable
D G	Display Global variables
D P	Display Procedure parameters
D L	Display Local variables
D I	Display Instance variables
D D	Disassemble statement
D E	Evaluate expression
D S	Display Stack words
D B	Display memory Bytes
D W	Display memory Words
D A	Display memory Ascii
D R	Display Registers
D F	Display 8087 Floating point stack

These are exactly the same as the examine commands described above *except* that you cannot alter values displayed. This prevents you from inadvertently changing the contents of memory or variables.

The Go Commands

The commands on the *Go menu* tell VID to resume the execution of the program being debugged. Each command either specifies a termination condition or the mode of execution (in the case of **Go Trace**).

The Go Menu

The following figure shows the menu when you specify G:

Go until Breakpoint (G B)

This command causes execution of the debugged program to continue until either an active breakpoint is hit or until you press *CtrlBreak*. When this happens the execution cursor in the source window shows the point you have

reached in your program (that is, the line containing the next statement to be executed).

The program executes at full speed until it reaches the breakpoint, as execution is neither being stepped nor traced.

Go Trace Source (G T)

This is the same as the **Go until Breakpoint** command except that VID displays each source line (by moving the execution cursor in the source window) as it executes on the way to the breakpoint.

The result is that the program's flow is visibly traced through the source executed. Because the display is updated quite rapidly, tracing at full speed may be too fast to see what is happening, so an option is provided to slow down the stepping speed. See **Options Debug** for details.

Go to User Cursor (G C)

This command is the same as **Go until Breakpoint** except that this command also installs a (temporary) breakpoint at the current user cursor position. This means that the program will stop when it gets to where you are "pointing" with the user cursor, provided the program doesn't stop elsewhere first for any other reason.

This command provides a quick way of moving around your program. Use cursor movement commands to move to the statement you want, then select **Go Cursor** (or type *G C*) to move to the execution cursor.

Go until Return from Current Procedure (G R)

This is also like **Go until Breakpoint** except that it places a temporary trap on the statement *following* the return from the current procedure (that is, on the statement after the place where the current procedure was called).

This means that when the procedure finishes (or earlier if another breakpoint was hit), control will be returned to VID. This is very useful if you have accidentally "stepped into" a procedure which you don't want to step through. For example, you might have pressed the spacebar instead of using *P* to skip the call. Type *G R* to return to the main thread that you were debugging.

Go until Procedure Call(G P)

This command moves the execution cursor to the start of the procedure you specify. You'll be asked for a procedure name. If you press *Enter*, you'll see a list of accessible procedures. You can use the cursor keys to move to your selection. Press *Enter* to confirm your selection.

You can also specify a name including any necessary qualifiers. If you specify a module name and a wildcard (for example, `MyModule.*`), VID will display a list of accessible procedures from the module you specified.

Go until Expression True (G E)

This command first prompts for an expression, as described on page . After you've entered an expression, this command will step through the program until that expression is true or non-zero (or until either a breakpoint or *CtrlBreak* occurs).

Control then returns to VID, and you are prompted with the expression and its result. Press *ESC* to continue debugging.

Although VID expressions are partially compiled and are evaluated quite rapidly, the *GE* command will still require a lot of recalculations. This may slow VID down considerably. It is always better to use statement breakpoints wherever possible.

ReStart the Program (G S)

The **Go ReStart** command will restart the program by reloading the program and resetting execution to the start of the program. All breakpoints are retained.

This command is very useful when you inadvertently execute a little too far in your program and you want to try again. It is also useful after you've learned more about what's going on in the program.

For example, after the **Monitor** breakpoint has shown how many times a statement is executed, you can restart the program and set a delayed breakpoint to go just "before" that number of times. To clear all breakpoints after restarting the program, you can use the **Breakpoint clear All** command.

Go Forever (eXit VID) (G X)

This command removes all breakpoints and resumes execution of the program. It is generally used when you have finished debugging and want to run the program to completion.

The Trace Windows

Trace windows are the windows through which you can view various aspects of your program as it executes. These windows are persistent in that they remain on the screen while you execute through your program. This is in contrast to windows such as the *Examine window*, which must be closed before you can continue.

The largest trace window is the *Source window*, which always remains open in the background. All the other trace windows can be opened and closed at any time. You can also move these windows around and can cycle through them, as described earlier.

To close the top window, press *ESC*. This stops the trace function active in the window and removes the window from the screen. The following trace windows are available, in addition to the source window:

- Procedure calls stack trace
- Active variables trace
- Registers trace
- Assembler execution trace
- Watch variable trace

Any combination of trace windows may be opened; these windows all function simultaneously.

The Trace Menu

The commands available through the Trace menu are shown in the following figure. As with the other menus, you can have a menu pop up by pressing *T* and waiting a second or two. Or you can activate the following commands by pressing *T*, followed immediately by the appropriate next character.

Procedure Calls (F4 or T P)

This command displays the current “call stack” (that is, the chain of procedure calls that led to the current position). Each line contains the name of the calling procedure and (except for the current procedure) the line number of the module source where the call to the next procedure was made.

The order of procedures is from “highest” to “lowest” (that is, the current procedure is last in the calling stack). If there is a deeper nesting of calls than the window allows (that is, more than 11 procedures), then only the final 11 calls will be displayed.

The following figure shows such a trace window:

Active Variables (F5 or T V)

This command displays the values of global and local variables as they are accessed. If the variable is part of an array or record or is referenced by a pointer, then VID will display the full variable name.

Alternate lines of the variable trace window have different background colors or, on monochrome adaptors, are alternately underlined. This striped effect makes it easier to associate the variable name with the correct variable value. You can change the display format (background and foreground colors) using the `VidInst` program. Figure -9 shows an example of such a window.

You will find the *variable trace window* extremely useful as you debug. This window can be opened quickly, to see the last few active variables without having to type their names. The variable trace window has a small “look-back” buffer so that, even when it is closed, opening the window will display the last few variables accessed. The following variable types are traced in this window:

- Unsigned integer - 1, 2 and 4 byte
- Signed integer - 1, 2 and 4 byte
- Floating-point - 4 and 8 byte
- CHAR and BOOLEAN
- Subrange types
- Enumerated types
- Pointer types

Furthermore, elements of arrays (up to 8 dimensions), fields/members of records/structures (up to 8 levels), and dereferenced pointers (up to 2 levels of indirection) will also be traced. In addition to tracing the dereferenced value of a pointer, the absolute address contained in the pointer is also displayed. If the current value is NIL (0), then this will be displayed.

Registers Trace (F 2 or T R)

This command displays the values currently contained in the *processor registers*, as shown in Figure -10. The values are displayed as hex words. Registers traced are:

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

and the processor flags. The flags are displayed in symbolic form if they are set, according to the following:

Symbol	Flag
C	Carry flag
P	Parity flag
A	Auxiliary carry flag
Z	Zero flag
S	Sign flag

T	Trap flag
I	Interrupts enable flag
D	Direction flag
O	Overflow flag

To *change* the values of registers, you should use the **Examine Registers** command.

Assembler Execution Trace (Alt-F2 or T A)

VID is designed primarily as a high-level debugger. Nevertheless, it is sometimes quite useful to be able to view the assembler code (generated by the compiler) as it executes. Following this code gives you a much better feel for what is actually happening as each statement executes.

The assembler trace function gives a scrolling display of the processor instruction as it executes. Procedure names are displayed symbolically and the values of registers used in the instruction are displayed in parentheses after each register. The address of the instruction is displayed on the left. After the assembler opcode and operands you'll sometimes find comments displayed. These identify procedure entry points and 80x87 emulated code.

Watch Variables

You can trace the values of variables and expressions in the *watch window*. This window can hold up to 9 separate expressions. To set a watch variable, use the **Trace Set watch variable** command (*T S*), as described below.

To delete a watch variable, use the **Trace Delete watch variable** command (*T D*).

The **Trace Watch** command (*T W*) will open (that is, display) the watch variables window. All these commands are described below.

You can watch all “simple” variable types:

Unsigned integer - 1, 2 and 4 byte

Signed integer - 1, 2 and 4 byte

Floating point - 4 and 8 byte

CHAR and BOOLEAN

Enumerated types

Pointer addresses

Array of char (the first 20 chars may be watched)

This means, of course, that you can watch record fields, array elements and dereferenced pointers provided they have one of the above types.

It is very important to note that the variables used in a watch variable or a watch expression must be in scope when the watch is set. These variables are then “bound” to the watch variable/expression. This means that, even if a new variable with the same name as one of the bound variables comes into scope, the original variable’s value will still be the one displayed.

When any variable component of a watch expression is not “active,” then the watch value will be blanked out as it has an indeterminate value. This can happen, for example, with a local variable belonging to a procedure not currently on the call stack.

Watch Variables Trace (T W)

This command opens the watch variables window and makes this the active window. If there are no watch variables already set, then you will be prompted to enter one.

To the left of the window, the watch number (1 - 9) is displayed. This serves to identify the watch variables when deleting them. After the watch number, you’ll find the variable/expression being watched. Finally, the value of the variable value is at the right, provided the variable is active (see above). Figure -12 shows such a window.

The depth of the watch window is determined by the maximum watch number currently set. This is so the watch window takes up as little space as possible on the VID display. For easier reading, the window is “striped” in a manner similar to the active variables trace window (see above).

Set Watch Variable (T S)

This opens the watch window and prompts you to set a new watch variable. This variable will be added to the watch trace window in the first available (free) slot. The size of the window will be extended, if necessary. The maximum number of watch variables allowed is 9.

Delete Watch Variable (T D)

This command prompts for the number of the watch variable to delete. The watch variable with the specified number is deleted, and that slot is freed for reuse. The watch window will “shrink” if the deleted watch number was the highest in the window.

Commands for the Source Window

VID is a multi-module debugger. To aid with the debugging of multiple modules, VID provides the ability to dynamically enable and disable the modules being debugged. A disabled module will play no part in execution tracing or with respect to breakpoints; such a module will not “get in the way” of debugging other modules. Initially, all modules that have been prepared for debugging are enabled. But during execution, you can disable or re-enable any module by using a module selection window, which can also be used to change the module on display.

The Source Menu

The *Source menu* also contains a number of source movement commands which are shown in Figure -13 (see also “*Additional Commands for Moving in the Source*,” page).

Change Module (F 3 or S C)

This command displays a list of all modules from which you can select the new module to display in the source window. This is used when you want to set breakpoints in another module, or you just want to look around the module.

Enable Module (S E)

This displays a list of disabled modules from which you can select a module to re-enable. This will bring the module back into the group being debugged.

Disable Module (S D)

This displays a list of enabled modules from which you can select a module to disable. The selected module is removed from debugging actions. The debugger will not trace through the module, and any breakpoints set in it will be inactive.

You will not be able to set new breakpoints in the disabled module until it has been re-enabled. If you are currently tracing through the module being disabled, that trace will continue. However, once you have left the module, you will not be allowed to reenter.

Previous Procedure (F7 or S P)

This moves the user cursor in the source window to the start of the previous procedure in the module, where “previous” refers to physical proximity and not to the procedure executed previously. This command is useful for rapidly scanning through a module’s source.

Next Procedure (F8 or S N)

This command moves the user cursor in the source window to the start of the next procedure in the module, where “next” refers to physical proximity and not next in execution sequence. This command is useful for rapidly scanning through a module’s source.

Locate Procedure (F9 or S L)

This prompts for the name of a procedure to locate, then moves the user to the start of the procedure specified. If you simply press *Enter*, without specifying a name, you’ll get a window containing the accessible procedures. You can use the cursor movement keys to select the procedure to locate from this list. If you specify the name, you must qualify it with the module name, unless the procedure can be found in the current module on display.

You may use the *wild cards* * and ?. In that case, a selection window will open, and a list of matching procedure names will be displayed. If no module name is specified, then the current module on display is assumed. You may use wild cards for module names as well. Thus * matches all procedures in the current module, name . * matches all procedures in module name, and * . * matches all procedures.

If the procedure you select is from another module, then that module is loaded into the source window.

Find String (Ctrl-Q F or S F)

This command lets you search for a string that you specify. The command prompts for a string, then searches for the next occurrence of that string in the currently displayed module.

The string must be within a single line. You can specify the wild cards * and ? as part of the search string. You can repeat the search as often as necessary by pressing *Ctrl-L*.

Goto Line (S G)

This command moves the user cursor to the line number you specify. This can be helpful in conjunction with the Procedure Calls trace window. The Procedure Calls window tells you the line number at which each suspended procedure called the next procedure. With the *S G* command, you can quickly look at those lines to make sure the procedures are being called correctly, etc.

View File (S V)

The view file command is provided to enable you to view files that are not amongst the module source being debugged. For example, you may want to look at header or documentation files for a module.

If you select this command, you'll be asked for the name of the file to display. This file is loaded and is displayed in a full screen window. You can use the ordinary cursor keys to move around the file and examine the file's contents.

Files are not limited to 64K or to memory size. This means, for example, that you can view the contents of a large map file. While you are viewing a file, you can search for a string by typing either *F* or *Ctrl-Q F*. This search will be done in the same manner as described above (in relation to Find String).

Pressing *ESC* will close the file and will return you to the normal VID screen.

Note: You may not use wild cards when specifying the name of the file to view.

User Options

There are a number of options designed to make VID easier and more convenient to use. You can set these options at any time while running VID. You can use the **Options Save** command to keep these settings in the file `VID.CFG`. Or, you can set the **Auto save options** so that such settings are saved automatically at the end of the session.

The Options Menu

The settings available through the *Options menu* are shown in the following figure.

Source mode (O S)

This option toggles between source and assembler modes. When turned off or when single-stepping through code without source available, then VID will display a disassembled listing within the source window.

Hex Output (O H)

This option determines whether the default output format is decimal or hexadecimal for values displayed by VID. Some values are displayed in hexadecimal format by default (for example, register and memory values), and such variables are not affected by this option.

This option toggles between decimal and hexadecimal.

Screen Swap (O R)

This toggles the mode for screen swapping between two options:

- on This is the normal screen swap mode as specified by the /S command line option.
- off This setting disables screen swapping. When this option is set, / will then refresh the VID screen instead of swapping.

Default Language (O J)

This determines the form in which variables are displayed. Normally (in Auto mode), the form is determined by the language in which the variable is defined. This may be overridden to force all variables to be displayed in a form compatible with a specific language. Modes currently supported are Auto, C, Pascal, C++ and Modula-2.

Highlight Code (O C)

This enables (if on) or disables (if off) the VID feature that source lines with associated code are highlighted when the execution cursor reaches that line. You may wish to disable this if the differing colors in the source window become distracting.

Status Line (O L)

This enables (if on) or disables (if off) the two-line help window displayed at the base of the VID screen. If the help window is disabled, then the source window is extended by two lines giving you a larger view into the source file. You will probably want to disable the help lines once you have become familiar enough with VID's keys and menus.

Single Process (O P)

When on, this option restricts tracing and breakpoints to a single process (e.g stack segment value). Thus, if you have multiple processes running, other processes will be ignored.

Trace Assembler (O T)

When on, single stepping (or continuous tracing) with the assembler window open will "follow" calls into the Assembler, disabled modules or into modules not being debugged. This means that you can follow the execution wherever it may lead. This option is normally set to off, as you are usually only interested in tracing the enabled source modules.

Trace Delay (O D)

This prompts for the time (in hundredths of a second) to delay each single step when continuously tracing. This is useful in order to view the execution of the program at a more leisurely pace, giving yourself a chance to see exactly what is occurring.

Save Options (O S)

This command saves to disk all currently set options in the file VID . CFG. This also saves the positions of all the windows, along with information as to which trace windows are currently open. When VID next executes, it will reload all the options and window positions from the configuration file before starting. VID can find the VID . CFG file using the TS . RED redirection file.

Auto Save Options (O A)

When set to on, this causes options to be saved automatically whenever you leave VID (that is, the **Options Save** command is carried out automatically). This ensures that when you restart VID, you will resume the session with the same options and windows configurations you had when you ended the last session.

Default Options

If you ever need to reset VID to its default options, simply delete the VID . CFG file. The default options are as follows:

Default Option	Setting
Source Mode	ON
Hex Output	OFF
Screen Swap	ON
Default Language	AUTO
Highlight Code	ON
Status Line	ON
Single Process	OFF
Trace Assembler	OFF
Trace Delay	0
Save Options	(no default)
Auto Save Options	OFF

Customizing VID

The VIDCFG program, which is included in your TopSpeed package, lets you customize your VID working environment by installing window colors and specifying various other settings and options.

To start the program, type

```
VIDCFG
```

The main menu for VIDCFG will then be displayed. This allows you to change various features of your working environment, including the background and foreground colors of the various VID windows, the settings related to screen swaps, and so forth.

To select an entry from this menu, use the cursor keys to highlight the entry you want, and then press `Enter`. For example, to change the color setting for a monochrome monitor, select the second entry.

You'll get a list of windows from which you can modify the settings. The selection process from this menu is the same as for the main menu: use the cursor keys to highlight the window you want, then press `Enter`. For example, after selecting the Disassembly window, your screen will display an example of a disassembly window to the right of the screen.

Use the *Up* and *Down* cursor keys to cycle through the available foreground colors, and use the left and right cursor keys to change the background. When the window looks the way you want it to, press `ESC` to return to the menu.

The screen swap defaults refer to the way in which VID will handle the process of switching between the debugging screen and the screen as it looks when the program is executing.

The *BW Override* feature lets you force VID to use monochrome mode. The *Snow Check* lets you turn on snow suppression on CGA monitors that flicker. The *Fill Value* lets you specify the initial value to be stored in memory before the program being debugged starts to execute. Finally, the *EMS* value allows you to specify whether you wish to use expanded memory for VID, and how many 16K pages should be allocated. The minimum useful value for this is 4 (e.g 64K).

When you've finished setting all the features and colors you want, select `I` on the main VIDCFG to save the settings. The values you specified will be placed in the `VID.OVL` overlay file, and will be used each time you run VID. You can also specify all but the window color settings on the command line when you invoke VID.

INDEX

A

Active window 24
ASCII Table 52

B

Binary calculations 53
Bitwise functions
 calculator 54
Block commands 36
Bottom scroll zone 49

C

Calculator
 bitwise functions 54
 memory functions 54
Command line 47
Compilation errors 40
Compiling 40
 environment 39
Configuration
 load 50
 options 48
 save 50
Cursor movement 35
Cut 54, 58
Cutting keys 59

D

Default
 extensions 49
 filenames 48
 shortcut keys 22
Dialog window 25
 editing commands 26
Directory
 change dir 28
 displays 28
 files dir 28
Directory window 28
DOS
 execute program 29
 shell 29

E

Edit
 project file 43
Editor 30
 block commands 36
 commands 33, 39
 correct case 39
 cursor movement 35
 insertion and deletion 33
 keys 33
 load file 27, 31
 menu 30
 options 37
 pick file 28
 read only mode 31
 save file 28, 32
 search and replace 38
Editor menu 30
editor windows 16
Environment
 changing windows 65
 command-line options 18
 compiling 40
 customizing 30, 64
 cut and paste 58
 editor 30
 files menu 27
 help 19
 help line 47
 input 25
 introduction 16
 keyboard macros 59
 keys 22, 33
 make 41
 menus 20
 project name 41
 quit 29
 running programs 42
 special commands 62
 special keys 24
 starting up 18
 system files 18
 using expanded memory 47
 utilities 50
 windows 24
error editor window 16
Errors
 compilation 40
 run-time 42
Execute program 29
Expanded memory 47

F

File

- auto save 48
- backup 49
- configuration 50
- load 27, 31
- maximum size 32
- pick 28
- redirection 49, 56
- save 28, 32
- selection window 27
- swap 32
- view 31

File options 50

File search 50

File selection window 27

Filename

- wild 27

Filenames

- default 48

Files

- menu 27

Find run-time error 51

H

Help 54

Help line 20, 47

Help system 19

Hex calculations 53

Hex dump of a file 54

High background 47

I

Input

- single line 25

Installation disks 11

Installing TopSpeed 11

K

Keyboard codes 56

Keyboard macros 59

- Load 50

Keys 22, 33

- cutting 59
- insertion and deletion 33
- special 24

L

Last help screen 54

Load file 31

Load file window 27

Locate File 51

M

Macros

- keyboard 50, 59

Main menu 21

Make 41

Make window 41

Memory functions

- calculator 54

Memory models

- project menu option 44

Menu

- editor 30
- files 27
- keys 21
- main 21
- options 46
- pop-up 30
- project 41, 43
- Setup 48
- shortcut keys 23
- system 20
- utilities 50

Mouse 63

Multi-language development 7

N

New project 43

O

Objects in VID 91

Options

- editor 37
- file 50
- load 50
- menu 46
- run 47
- save 50
- setup 48

Options menu 46

P

Paste 54, 58, 59

- Pick file 28
- Pop-up menus 30
- Print file 55
- Print settings 55
- Program
 - compiling 40
 - making 41
 - running 42
- Project
 - file 41
 - menu 41, 43
 - name 41
 - options menu 45
- Project system 42

R

- Recoloring
 - help line 66
 - windows 66
- Redirection file 49, 56
- Review DOS Screen 54
- Run options 47
- Run-time errors 42

S

- Save file 32
- Scroll zone 49
- Search and replace 38
- Search files list 51
- Setup
 - menu 48
- Shortcut keys 22
 - default 22
 - menu 23
- Single line input 25
- Snow check 46
- Solid cursor 47
- Special commands 62
- Swap file 32
- System
 - project menu option 44
- System info 55

T

- Target type
 - project menu option 44
- Top scroll zone 49
- Typographic conventions
 - general 9

U

- Utilites
 - Help Window 54
- Utilities 50
 - ASCII Table 52
 - File search 50
 - Find run-time error 51
 - keyboard codes 56
 - Locate file 51
 - print file 55
 - print settings 55
 - Review DOS Screen 54
 - Search list 51
 - system info 55
 - View file data 54
- Utilities menu 50

V

- VID
 - assembler debugging 83
 - breakpoints 86
 - command line options 76
 - customizing 106
 - display data 89
 - examine data 89
 - expressions 73
 - go commands 94
 - Introduction 67
 - keys, menus and windows 72
 - menus 84
 - Objects 91
 - options 103
 - project menu option 44
 - running 70
 - source menu 101
 - source window 78
 - starting 70
 - trace windows 96
 - vidcfg.exe 70
- View file, read only 31
- Viewing files 54

W

- Wild filenames 27
- Window
 - active 24
 - changing 65
 - classes 66
 - dialog 25

- file selection 27
- make 41
- recoloring 66
- repositioning 65
- resizing 65
- zoom 25

Windows

- Moving between 33

Z

- Zoom 25