Periscope®

Professional Software and Hardware-Assisted Debuggers

Periscope Manual

Periscope Manual Version M54

License Agreement

Should you have questions concerning the Program or this License Agreement, please contact:

The Periscope (Computing) Co., Inc., 1475 Peachtree St., Suite 100, Atlanta, GA 30309 USA
Phone: 404/888-5335 **FAX:** 404/888-5520 **Sales:** 800/722-7006 (US & Canada)
Support: 404/888-5550 (1-5 pm EST, Mon-Fri, except holidays) **BBS:** 404/888-5522

Table of Contents

List of Tables and Figures

C H A P T E R O N E

Introduction 1

- Registration, Updates, and Upgrades
- Warranties
- Money-Back Guarantee
- Using the Manual
- Products Covered in this Manual
- System Requirements
- Getting Started

Thank you for choosing Periscope. If you're anxious to get started on a bug hunt, go immediately to Section 1.7, Getting Started. Just be sure to read the other sections in this chapter as soon as you can. The first three sections will familiarize you with important benefits of owning a Periscope and how to take advantage of those benefits. The next three sections will familiarize you with this manual, the Periscope products covered in this manual, and the system requirements for running Periscope.

1.1 REGISTRATION, UPDATES, AND UPGRADES

Registration. Please complete and return the registration card included with your Periscope package. When we receive your card, we will register your Periscope so that you will receive free Technical Support when you call our support line, leave a message on our BBS, or send us a fax. (Please see the back of the title page for numbers and hours of operation). We will send you update notices, special offers on new products, newsletters, and other pertinent information as well.

If you misplace your registration card, call us with your registration number (sticker on inside front cover of binder). If you do not have your registration number, you will have to provide us with proof of purchase.

Updates and Upgrades. Registered users of Periscope may update or upgrade at any time. Please call Sales (see back of title page for phone number) for current prices and full details.

1.2 WARRANTIES

Software. Please see the License Agreement on the back of the title page for our warranty on software, disks, and manuals.

Hardware. We (The Periscope Company, Inc.) warrant all hardware in the Periscope package to be in good working order for a period of one year from the date of purchase as a new (or factory-refurbished) product. This warranty covers all hardware, i.e., boards, cables, switches, adapters, clips, and other hardware accessories included in the package.

Should any of these items fail to perform properly any time within the stated warranty period, we will, at our option, repair or replace it at no cost except as set forth in this warranty. We will furnish replacement parts or products on an exchange basis only. Replaced parts and/or products become our property.

We do not express or imply any warranty for damage caused by accident, abuse, misuse, natural or personal disaster, or unauthorized modification, nor do we warrant that the hardware will meet your requirements or that its operation will be uninterrupted or error-free.

To obtain warranty service, please follow this procedure:

- Call Technical Support (see back of title page for phone number) for a return authorization number.
- Write this number clearly on the outside of the package you are returning.
- Include proof of your purchase date and your registration number (or registration card) in the package if you did not purchase your Periscope directly from us.
- Insure the package (or accept all liability for loss or damage) and prepay all shipping charges, duties, and taxes.

1.3 MONEY-BACK GUARANTEE

If you purchased Periscope directly from us (The Periscope Company, Inc.) and the product does not perform to your satisfaction, you may return it to us within 30 days of our invoice date for a refund of your purchase price (excluding shipping charges) less, on Model IV products only, a restocking fee of 5% of your purchase price.

Please follow this procedure to return a Periscope package under this guarantee:

- Make sure we receive the package within 30 days of our invoice date. We cannot extend the guarantee beyond this 30-day period.
- Call for a Return Authorization number prior to sending the package to us. (See the back of the title page for phone numbers.)
- Write this number clearly on the outside of the package you are returning.
- Make sure all items are in the package, including the registration card (unless you have already sent it to us). We will deduct the price of any missing items from your refund.
- Make sure that the package is well-packed so that items are not damaged in shipping. We will deduct the price of any damaged items (whether damaged during or prior to shipping) from your refund.

1.4 USING THE MANUAL

Typography. The normal font used in the body of the manual is a proportional font, meaning different spacing is used for different characters to make the text appear evenly-spaced and more readable. This can make it difficult for you to determine the spacing of items you might enter into the computer. Therefore, when we specify syntax, a command or options that you might enter, messages that appear on the screen, etc., we use a fixed-space font, which uses exactly the same amount of space for all characters.

This sentence is formatted in the normal proportional font used for most body copy.

```
This sentence is formatted in the fixed-
space font that is used for "entry"
items, screen messages, etc.
```

We point out Periscope model differences, warnings, exceptions, and special instructions with a note like this.

A ♦ appears at the end of each chapter to indicate the end of the chapter.

Other. This manual covers both the Periscope/EM software and its derivative, Periscope/32. We use the generic term "Periscope" to refer to both, unless we're discussing something specific to one that does not apply to the other. See Section 1.5 below and Appendix C for details.

The Model I Board and the Model IV Plus Board are one and the same board. Wherever we use the term "Model I Board", you can assume we also mean the Plus board, unless we specify otherwise.

1.5 PRODUCTS COVERED IN THIS MANUAL

This manual documents all Periscope capabilities except those requiring the Model IV hardware (see the Model IV manual for Model IV specifics). Use this manual with these Periscope products:

Periscope/EM: Periscope/EM is the "regular" or "main" Periscope debugger software. It is sold with a Break-out Switch and this manual as "Periscope/EM". It is also included with the hardware-assisted Periscope Model I and Periscope Model IV products (see below).

Periscope/32: Periscope/32 is a 32-bit-aware version of (and is derived from) the Periscope/EM software. It can run standalone as a full-function debugger in a single system, like Periscope/EM, or as the host debugger in a remote debugging environment, as it does in the Periscope/32 for Windows and Periscope/32 for OS/2 products. Appendix C provides an overview of how Periscope/32 differs from Periscope/EM. Specific command and other syntactical differences are noted in the appropriate locations throughout the manual. The information you need to install Periscope/32 as the host debugger for target Windows or OS/2 environments is included in the addenda included with the Periscope/32 for Windows and Periscope/32 for OS/2 products.

The Periscope 32-bit Toolkit, combined with Periscope/32, can help you create a system-level debugger for a proprietary target operating environment. Please call Sales or Technical Support for more information.

Periscope Model I: Periscope Model I includes the Periscope/EM software, a Break-out Switch, the Model I Rev 3 Board (with 512K of write-protected RAM), this manual, and an addendum that covers hardware installation and running the hardware diagnostics program, PSTEST.

Periscope Model IV: Periscope Model IV includes the Periscope/EM software, the PopUp Periscope software, a Break-out Switch, a Model IV (Rev 1 or Rev 2) Board, this manual, the PopUp Periscope manual, and the Model IV manual. (A CPU-specific Pod, sold separately, is also required to run Model IV.)

If you're using the Periscope/EM software with your Model IV, use this manual as a general reference and the Model IV manual as a Model IV-specific reference.

If you're using the PopUp Periscope software, you will not need this manual. Use the PopUp Periscope and Model IV manuals for reference.

You'll find a list of discontinued products in the NOTES.TXT file on the Periscope distribution disks. In most cases, we currently produce a product that covers the functionality of the discontinued product. For instance, Periscope/EM covers all the capabilities—and more—of discontinued Models II and II-X.

You'll also find descriptions of the files on the distribution disks in the NOTES.TXT file.

1.6 SYSTEM REQUIREMENTS

The Periscope software requires:

- an IBM PC, XT, AT, 80386, 80486, PS/2 or compatible personal computer
- PC/MS-DOS 3.10 or later
- one disk drive
- an 80-column monitor
- 75K (minimum) to 150K RAM in the lower 640K ***or*** for 386 and higher machines with a supporting memory manager (QEMM, 386MAX, BlueMAX, or NETROOM) installed, zeroK RAM in the lower 640K *and* 32K (36K for QEMM and NETROOM) RAM between 640K and one megabyte *and* ~300K RAM beyond one megabyte

The Periscope hardware requires:

- The **Periscope Model I Board** requires an IBM PC-compatible (ISA or EISA) bus, one full-length slot, and 32K of address space above the lower 640K but in the first megabyte of system memory.
- The **Periscope Model IV Boards and Pods** require a machine with an 80286, 80386DX, 80386SX, 80486DX, or 80486DX2 CPU running up to 33MHz (external speed) with zero or more wait states. The boards require an ISA or EISA bus and one full-length slot. The optional Plus Board is the Periscope Model I Board (see requirements above).

Please call Technical Support if you have compatibility questions or problems.

1.7 GETTING STARTED

Here are the steps you need to follow to start debugging with Periscope/EM or Periscope Model I. These steps direct you to additional details should you have questions or need help. If you're using Model IV, please see the Model IV and PopUp Periscope manuals.

Step 1—Install the Model I Board or Periscope Break-out Switch.

To install the Model I Board, see the installation addendum. To install the Break-out Switch without a board, see Chapter 4.

Step 2—Configure Periscope.

Insert the distribution disk in Drive A and enter `a:setup`. Follow the on-screen instructions. See Chapter 3 for details.

Step 3—Install the software.

From the Periscope directory (usually `c:\peri`), enter `ps`. For options, enter `ps ?` or see Chapter 5.

Step 4—Begin debugging with Periscope!

For a quick tour of Periscope, see the tutorials in Chapter 2.

Step 5—For reference, see:

Chapter 6 for usage tips.

Chapter 7 for information on using Periscope utilities.

Chapters 8 and 9 for information on menus, commands, command parameters, the command editor, shortcut keys, function keys, keyboard assignments, and aliases.

Appendices A and B for error messages and troubleshooting tips.

Appendix C for the differences between Periscope/EM and Periscope/32.⌖

C H A P T E R T W O

Tutorials

2

- **C Tutorial**
- **Assembly Tutorial**

We intend that the following tutorials help you get familiar enough with Periscope that you can begin using it to debug your own programs right away. (If you're using Model IV, please see the Model IV manual for a Model IV-specific tutorial.) The C tutorial focuses on symbolic and source-level debugging for high-level languages. The assembly tutorial is more general. We recommend you take both tutorials regardless of the programming language you use. Because the tutorials do not cover all commands, we also recommend that you supplement what you learn in the tutorials with information contained elsewhere in this manual.

Please don't skip, add, or change any steps in the tutorials.

2.1 C TUTORIAL

This tutorial takes you through a guided tour of Periscope from the high-level language perspective. It uses the program FTOC, with these files:

FTOC.C is the source code for FTOC. Periscope uses this file to display source code.

FTOC.DEF is the optional FTOC definition file, which contains two alias definitions and one record definition. Periscope uses it to read alias and record definitions into memory. You create a .DEF file with an editor as a standard ASCII text file. See the description of the utility program RS in Section 7.5 for more information about .DEF files.

FTOC.EXE is the executable program.

FTOC.MAP is the .MAP file produced by the linker. Unless you use the TS program to create symbols from another source, Periscope uses the information in the .MAP file to replace memory addresses with symbols from the program you're debugging. You create a .MAP file at link time by specifying a .MAP file and the link options `/LI` and `/M`. This file contains public and line-number symbols.

In this tutorial we'll use symbols stored at the end of the FTOC.EXE file by the link option `/CO` instead of those in the .MAP file, as explained in Step 3 below.

Step 1—Configure Periscope for your system.

Place the distribution disk in Drive A and enter **`A:SETUP`**.

SETUP copies the files from the distribution disk to your working directory. CONFIG generates the executable Periscope software to support the hardware you have installed, if any. For purposes of this tutorial, enable Periscope's menu system and select the CodeView function-key emulation. You can run CONFIG again later to change these options if you wish. See Chapter 3 for details.

You can configure Periscope as software only even if you have hardware. If you do configure for one of the board models, you must install the hardware before you can take the tutorials.

Step 2—Install the Periscope software.

Enter `CD\PERI`, then `PS /H /T:4`.

This makes the Periscope directory the default directory, loads Periscope into memory, loads the on-line help and interrupt comment files into memory, and allocates 4KB for symbol tables.

If you have installed a Model I Board in your system or if you are using EM memory (with 386MAX, QEMM, or NETROOM), just enter `PS`. You do not need the `/H` and `/T` options. If you're using the Model I Board and changed the DIP switches on the board, however, do enter the memory (`/M`) and/or port (`/P`) options. See Section 5.2 for details.

Step 3—Create a symbol file with local symbols.

Enter `TS FTOC /E`.

This creates a Periscope symbol file from FTOC.EXE. (To use the .MAP file for symbols, you would enter `TS FTOC`.)

We used Microsoft C to compile FTOC. If we had used the Borland compiler and linker, you'd enter `TS FTOC /B`.

Microsoft LINK does not place local symbol information in the .MAP file. To provide you with access to local symbols as well as public and line-number symbols while you're debugging, Periscope must get symbol information from the .EXE file.

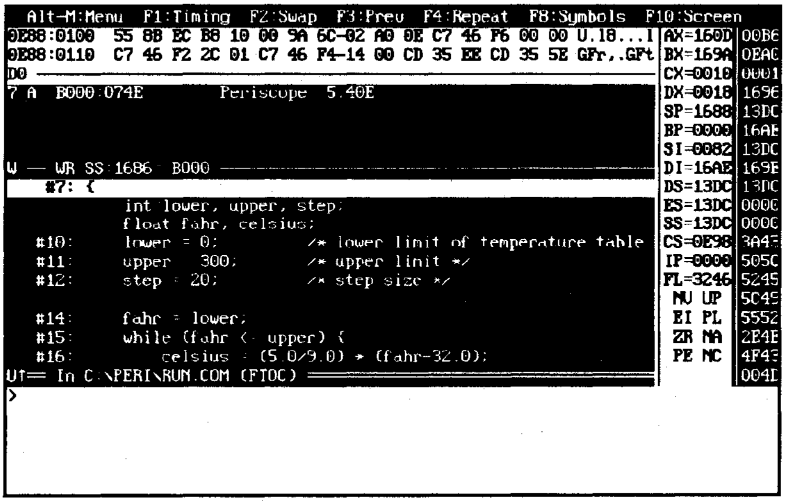
Step 4—Use the program loader to run FTOC.

Enter **RUN FTOC**.

RUN displays informational messages and the address of the PSP, then switches to Periscope's screen. Normally, Periscope starts at the very beginning of the program and displays assembly code, but we've included a special alias in the .DEF file, **\X0=G _MAIN**, that causes Periscope to immediately go to _MAIN, which is the beginning of the C source code.

Step 5—Display Periscope's windows.

Since we didn't specify any windows when we installed Periscope in Step 2, you see Periscope's default windows, as shown in Figure 2-1.



```
 Alt-M:Menu  F1:Timing  F2:Swap  F3:Prev  F4:Repeat  F8:Symbols  F10:Screen
0E88:0100  55 8B EC B8 10 00 9A 6C-02 A0 0E C7 46 F6 00 00 U.18...I AX=160D 00B6
0E88:0110  C7 46 F2 2C 01 C7 46 F4-14 00 CD 35 EE CD 35 5E GFr,.GFt BX=169A 0EAC
D0 ------------------------------------------------------------------ CX=0010 0001
7 A  B000:074E          Periscope  5.40E                              DX=0018 1696
                                                                      SP=1688 13DC
                                                                      BP=0000 16AE
                                                                      SI=0082 13DC
W -- WR SS:1686  B000 ------------------------------------------------ DI=16AE 169E
   #7: {                                                              DS=13DC 13DC
          int lower, upper, step;                                     ES=13DC 0000
          float fahr, celsius;                                        SS=13DC 0000
  #10:    lower = 0;        /* lower limit of temperature table       CS=0E98 3A43
  #11:    upper   300;      /* upper limit */                         IP=0000 505C
  #12:    step = 20;        /* step size */                           FL=3246 5245
                                                                       NU UP  5C45
  #14:    fahr = lower;                                                EI PL  5552
  #15:    while (fahr <- upper) {                                      ZR NA  2E4E
  #16:        celsius = (5.0/9.0) * (fahr-32.0);                       PE NC  4F43
Ut== In C:\PERI\RUN.COM (FTOC) ===========================================    004D
>
```

Figure 2-1. Periscope/EM's Default Screen

- The first line of the screen contains the menu bar, which now shows the function key assignments for CodeView emulation. If you don't see this, start over at Step 1!
- The next two lines of the screen show a Data window which you can use to display memory in various formats.
- The next window is the Watch window, which you can use to monitor various memory and I/O port locations. The only watch item currently set shows the Periscope software version with a suffix that indicates the model of

Periscope you're using.

- The next window is the Disassembly window. You use this window to display the program being debugged in source, assembly, or mixed mode.
- On the right-hand side of the screen, there are two vertical windows for the system registers and the stack.

The current stack pointer is at the top of the Stack window. (A chevron points to the value of the BP register when BP is in the range shown by the stack.)

Press **Alt-R** several times to toggle the Register window.

Press **Alt-S** several times to toggle the Stack window.

You can specify your own windows using either the `/W` installation option or the `/W` command.

When we refer to an installation option, we mean an option that you specify when you install Periscope. (See Chapter 5 for details.) When we refer to a command, we mean a command that you enter while Periscope's screen is displayed. (See Chapter 9 for details.)

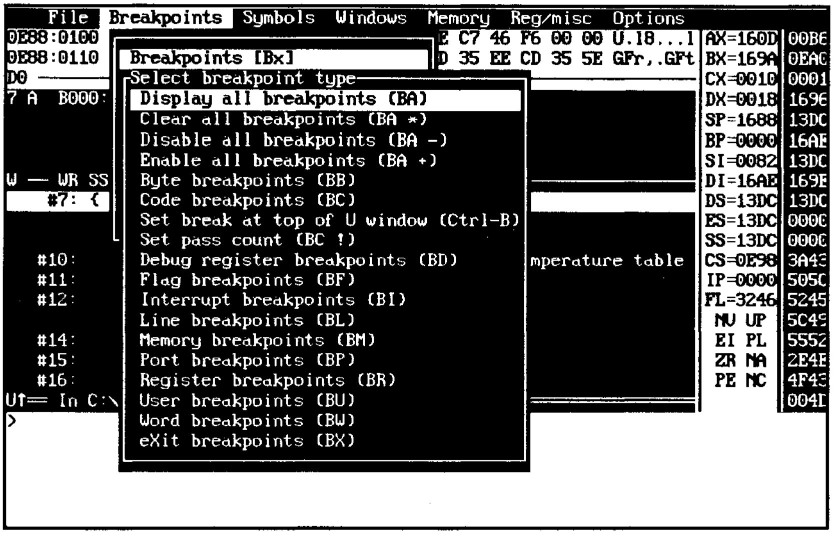
Step 6—Activate the menu system.

Press **Alt-M** to activate the menu system. Notice how the menu bar changes from showing the key prompts to showing the various menus available.

Use the right arrow key to browse through the various menus. See Section 8.1 for information on using menus.

The menus are a convenient way to execute infrequently-used or complicated commands, but because of the extra keystrokes required, you'll want to either enter the most frequently used commands directly or set them up on function keys as has been done with the CodeView function key emulation.

Press **Esc** to return to the command interface.



```
    File  Breakpoints  Symbols  Windows  Memory  Reg/misc  Options
0E88:0100                          E C7 46 F6 00 00 U.18...1  AX=160D  00B6
0E88:0110   Breakpoints [Bx]       D 35 EE CD 35 5E GFr,.GFt  BX=169A  0EA0
D0 ——      Select breakpoint type                             CX=0010  0001
7 A  B000:  Display all breakpoints (BA)                      DX=0018  1696
            Clear all breakpoints (BA *)                      SP=1688  13DC
            Disable all breakpoints (BA -)                    BP=0000  16AE
            Enable all breakpoints (BA +)                     SI=0082  13DC
W — WR SS   Byte breakpoints (BB)                             DI=16AE  169E
   #7: {    Code breakpoints (BC)                             DS=13DC  13DC
            Set break at top of U window (Ctrl-B)             ES=13DC  000C
            Set pass count (BC !)                             SS=13DC  000C
  #10:      Debug register breakpoints (BD)  mperature table  CS=0E98  3A43
  #11:      Flag breakpoints (BF)                             IP=0000  505C
  #12:      Interrupt breakpoints (BI)                        FL=3246  5245
            Line breakpoints (BL)                              NU UP   5C49
  #14:      Memory breakpoints (BM)                            EI PL   5552
  #15:      Port breakpoints (BP)                              ZR NA   2E4E
  #16:      Register breakpoints (BR)                          PE NC   4F43
U↑= In C:\  User breakpoints (BU)                                      004D
>           Word breakpoints (BW)
            eXit breakpoints (BX)
```

Figure 2-2. Periscope/EM's Menus

Step 7—Display the three disassembly modes.

Enter `UA`, then `US`, then `UB`.

You can see FTOC in the Disassembly window in three modes : `UA` shows Assembly only; `UB` shows Both source and assembly; and `US` shows Source-only. (The source-only display shows assembly code until the first source line reference is found, then displays source-only.) Periscope defaults to `US` mode, unless it finds no line numbers in the symbol file.

Step 8—Use Periscope's help feature.

To see a summary of all commands, enter `?`. To see help on the Display command, enter `? D`.

Press **Enter** to remove the help display.

Step 9—Set up to local variables to watch.

To go to line 10 in FTOC, enter `G #10`.

To display STEP in integer format, enter `W I STEP`.

To display FAHR in short-real format, enter `W S FAHR`.

Neither variable is initialized yet, so their values mean nothing.

Step 10—Trace at source then assembly level.

Press **F10**, or enter `JL`. This steps to the next source line in the program. The Jump Line command uses the Jump command to step through your program until the 'current' instruction is the beginning of a source line.

The Jump command steps to the next sequential assembly level instruction. The Trace command is the same as the Jump command, except that it traces into called routines and interrupts, instead of staying at the instruction level.

Enter `J` and `T` a few times. Note how the Disassembly window changes.

Step 11—Go to the next call of _PRINTF.

Enter `G _PRINTF`.

Since _PRINTF is a library procedure, you're away from FTOC's source code, so you don't see any source code. Also, you don't see the names of the local variables in the Watch window, since the procedure in which they are defined is not currently executing.

Step 12—Analyze the stack for return points.

Enter `SR`.

`FTOC#17+0020` indicates the instruction after the call to **`_PRINTF`**.

Enter `GR` to quickly return to the mainline code.

Then to go to line 18, press **F10** or enter `JL`.

Step 13—Go to the next source line <u>very slowly</u>.

Enter `G _PRINTF`, then press **F8** or enter `TL`. The `TL` command traces instructions until the next source line, line 18, is executed. Check the value of FAHR in the Watch window versus the value just displayed by FTOC.

Press **Alt-O** or **F4** to switch back to the program's display.

Step 14—Watch the local variable FAHR.

Press **F7** a few times or enter `G FTOC#18`.

This shows the next several executions of line 18. Notice how the value of FAHR in the Watch window changes each time. You can use the short form of the line number, `G #18`, when you are already 'in' the module.

Step 15—Use a record definition to display local symbols.

To turn the windows off temporarily, enter `/W`.

Then type `//` and press **Alt-I** to display the local symbols.

To call a keyboard macro that executes the command `DR SS:BP-E VARS`, press **Ctrl-F1**. This command uses a Periscope record definition to display the values of the local variables. You would normally use this method only with compilers that provide no local symbol support.

To restore Periscope's windows, press **Ctrl-F9**.

Step 16—End the debugging session.

To exit Periscope, enter `QC` or `G`.

2.2 ASSEMBLY TUTORIAL

This tutorial takes you through a guided tour of Periscope using a simple Assembly language program. It demonstrates some of Periscope's commonly-used debugging commands. For more detailed information see Chapter 9.

This tutorial uses the program SAMPLE, which displays the total and available memory in the system, and these files:

PS.DEF is the optional Periscope definition file. It contains two alias definitions and three record definitions. Periscope uses this file to read alias and record definitions into memory. You create a .DEF file with an editor as a standard ASCII text file. Record definitions can be used to display any area of memory in an easy-to-read format. The utility program RS verifies and calculates the memory a .DEF file requires. See the description of RS in Section 7.5 for more information.

SAMPLE.ASM is the source code for SAMPLE. Periscope uses this file to display SAMPLE's source code.

SAMPLE.COM is the executable program.

SAMPLE.MAP is the .MAP file produced by the linker. See the description of FTOC.MAP at the start of the C tutorial above.

If you just completed the C tutorial, begin at Step 3 below.

Step 1—Configure Periscope for your system.

See Step 1 of the C tutorial above.

Step 2—Install the Periscope software.

See Step 2 of the C tutorial above.

Step 3—Use the program loader to run sample.

Enter `RUN SAMPLE`. RUN displays informational messages and the address of the PSP, then switches to Periscope's screen.

Step 4—Display Periscope's windows.

To toggle the Register window, press **Alt-R**.

To toggle the Stack window, press **Alt-S**.

To display the 386 registers on an 80386 or later system, press **Alt-3**.

Since we didn't specify any windows when we installed Periscope in Step 2, you see Periscope's default windows: a Data window, a Watch window, a Disassembly window, and the vertical Register and Stack windows. The current stack pointer is at the top of the Stack window. (See Figure 2-1.)

Step 5—Display the three disassembly modes.

Enter `UA`, then `US`, then `UB`.

You'll see SAMPLE in the Disassembly window in three modes : `UA` shows Assembly only; `UB` shows Both source and assembly; and `US` shows Source-only. (The source-only display shows assembly code until the first source line reference is found, then displays source-only.) Periscope defaults to `US` mode, unless it finds no line numbers in the symbol file. Note that `UB` mode appears to show each line twice: once as source and once as assembly.

The register pair BX:CX shows the size of the program. Registers DS, ES, SS, and CS all show the PSP segment since this is a .COM program. The segment value varies, depending on the version of DOS and any memory-resident programs you're using.

Step 6—Use Periscope's help feature.

To see a summary of all commands, enter `?`. To see help on the Display command, enter `? D`.

Step 7—Display the PSP in the Data window.

To see the first 32 bytes of the PSP in Byte format, enter `DB CS:0`.

To turn Periscope's windows off, enter `/W`.

To see the PSP displayed in Record definition format, enter `DR CS:0 PSP`. This makes it much easier to see what's what in the PSP. You can add record definitions as you need them by editing the .DEF file. (See the description of RS in Section 7.5).

To see the record definitions available, press **Alt-E** before entering anything after the Periscope prompt.

To restore Periscope's windows, press **Ctrl-F9**.

Step 8—Set up variables in the Watch window.

To clear the Watch window, enter `W *`.

Enter `W I TOTMEM`, then enter `W I FREMEM`.

You can monitor the values of TOTMEM and FREMEM in integer format.

Step 9—Set and execute breakpoints.

To set a sticky code breakpoint at the end of the program (DOSRET), enter `BC DOSRET`.

Press **Alt-I** to see the available symbols.

To set a word breakpoint on the value of the variable TOTMEM changing from zero to any other value, enter `BW TOTMEM NE 0`.

To display the current breakpoints, enter `BA ?`. You'll see `BC DOSRET` and `BW TOTMEM NE 0000`.

To execute SAMPLE with both of the breakpoints activated, enter `GT`. Execution will stop at the instruction after the one that moves register AX into TOTMEM.

Press **PadMinus** twice to move the Disassembly window back two lines.

To see the instruction that caused the breakpoint, enter `TB`. Periscope's traceback command shows previously-executed instructions in a full-screen mode using the software trace buffer. The last entry is the breakpoint event. Use **PgUp** and **PgDn** to scroll through the buffer, then press **Esc** to display Periscope's prompt.

To clear the word breakpoint, enter `BW *`.

To display the breakpoints, enter `BA or BA ?`.

Only one breakpoint is still set, **BC DOSRET**.

Step 10—Convert hex to decimal.

To display the value of TOTMEM in the Watch window in hex, enter **W W TOTMEM**.

To convert the value back to decimal, enter **X nnnn**. **nnnn** is the hex value of TOTMEM.

The decimal value is displayed as the second field. Since the value of TOTMEM is still in register AX, **X AX** gives the same result.

Step 11—Trace at the assembly level.

To step through the next few instructions, enter **J**, then press **Alt-D** several times.

When you get to the RET instruction, enter **J** again. This will take you back to the mainline code.

Step 12—Disassemble and verify the number conversion routine.

To disassemble the number conversion routine, enter **U CONVERT**.

To restore the disassembly to the current instruction, enter **R**.

To go to the instruction after the first call to CONVERT, enter **G #35**.

To display the converted value, enter **DA TOTAL** or **DA TMEMORY**. The result should agree with the value of TOTMEM in the Watch window.

Step 13—Return to DOS.

Enter **G** twice.

The first **G** takes you to DOSRET, since the sticky code breakpoint for DOSRET is still in effect. The second **G** returns you to DOS. If the sticky breakpoint did not exist, you could just enter **G** once.✦

CHAPTER THREE

Configuring Periscope

3

■ **Running SETUP and CONFIG**

This chapter describes the procedure for configuring Periscope for your computer system and for the Periscope hardware you're using, if any.

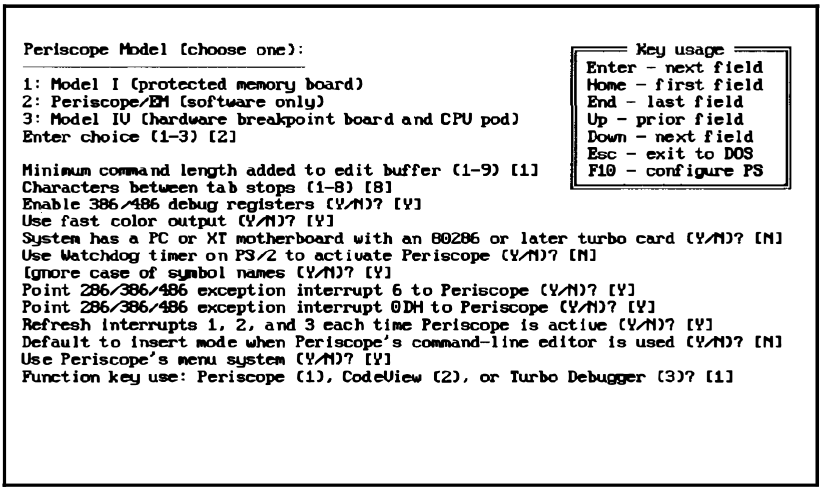
3.1 RUNNING SETUP AND CONFIG

Step 1—Place the Periscope disk in drive A, and enter `A: SETUP`.

SETUP will prompt you for the name of the Periscope directory. The default is `C:\PERI`. SETUP then copies the Periscope files from the floppy disk to the directory. After the files have been copied, SETUP executes the CONFIG program, which configures Periscope for your system.

For best results, edit AUTOEXEC.BAT to contain the line `SET PS=C:\PERI`, substituting the directory you entered, if any, for `PERI`.

Step 2 - Answer the prompts on the full-screen display shown in Figure 3-1 (explanations follow), then press F10.



```
Periscope Model (choose one):                                 ═══ Key usage ═══
                                                              Enter - next field
1: Model I (protected memory board)                           Home - first field
2: Periscope/EM (software only)                               End - last field
3: Model IV (hardware breakpoint board and CPU pod)           Up - prior field
Enter choice (1-3) [2]                                        Down - next field
                                                              Esc - exit to DOS
Minimum command length added to edit buffer (1-9) [1]         F10 - configure PS
Characters between tab stops (1-8) [8]
Enable 386/486 debug registers (Y/N)? [Y]
Use fast color output (Y/N)? [Y]
System has a PC or XT motherboard with an 80286 or later turbo card (Y/N)? [N]
Use Watchdog timer on PS/2 to activate Periscope (Y/N)? [N]
Ignore case of symbol names (Y/N)? [Y]
Point 286/386/486 exception interrupt 6 to Periscope (Y/N)? [Y]
Point 286/386/486 exception interrupt 0DH to Periscope (Y/N)? [Y]
Refresh interrupts 1, 2, and 3 each time Periscope is active (Y/N)? [Y]
Default to insert mode when Periscope's command-line editor is used (Y/N)? [N]
Use Periscope's menu system (Y/N)? [Y]
Function key use: Periscope (1), CodeView (2), or Turbo Debugger (3)? [1]
```

Figure 3-1. Periscope/EM Configuration Screen

The configuration screen shows the current settings (in PS1.COM). If you've never run CONFIG before (you'll see `[?]` after `Enter choice (1-3)`), the screen displays Periscope's default settings. Otherwise it displays the settings as of the last time you ran CONFIG, as shown in Figure 3-1.

```
Periscope Model (choose one):
Enter choice (1-3)
```

Enter the number corresponding to your model of Periscope.

If you enter `1` (Periscope Model I), the Periscope software will load into the Model I Board's memory when you install it *unless* (1) you install the software with the `/N` option or (2) the board is not present when you install the software. In either of these cases, Periscope will load into conventional DOS memory (lower 640K).

If you enter `2` (Periscope/EM) or `3` (Periscope Model IV), the Periscope software will load into extended memory ('EM' memory) when you install it *unless* (1) you install the software with the `/N` option or (2) you install the software with the `/NE` option or memory manager support is not available. See Section 6-1 for details.

In the first case, the Periscope software will load into conventional DOS memory (lower 640K). In the second case, the software will load into the Model I Board's memory if a board is present and into conventional DOS memory (lower 640K) if a board is not present.

To force Periscope to run from the Plus Board's memory when you're using Model IV with a Plus Board and you have a supporting memory manager installed, use the `/NE` installation option.

Please see Section 5.2 for options when installing the Periscope software. See Section 6.1 for additional information on using supporting memory managers to load Periscope into extended memory.

```
Minimum command length added to edit
buffer (1-9)
```

The default value `1` adds all commands to the buffer. To add only commands longer than two characters, enter `3`.

```
Characters between tab stops (1-8)
```

Enter the tab column width, from `1` to `8` characters.

`Enable 386/486 debug registers (Y/N)?`

Enter `N` if you're using a memory manager such as Compaq's CEMM that does not support real-mode access to the 80386 debug registers. Enter `Y` if you're using QEMM or 386MAX or no 386 control program.

`Use fast color output (Y/N)?`

Enter `N` if you're using a 'snowy' color card.

`System has a PC or XT motherboard with an 80286 or later turbo card (Y/N)?`

Enter `Y` if you're using a system with a PC or XT motherboard and an 80286 or later turbo card.

`Use Watchdog timer on PS/2 to activate Periscope (Y/N)?`

Enter `Y` if you're using an IBM PS/2 and you want Periscope to automatically wake up after two seconds of 'missed' timer ticks. When the watchdog kicks in, Periscope displays the message `Grrr!`.

➪ If you enter `Y`, do not use **Ctrl-Alt-Del** to reboot or your system will hang. Instead use Periscope's `QB`, `QS`, or `QL` commands to reboot your system, or install PSKEY, which disarms the watchdog.

`Ignore case of symbol names (Y/N)?`

Enter `N` if you want Periscope to support case sensitivity. If you enter `N`, Periscope will display `Periscope` in mixed case in the Watch window. Enter `Y` if you want Periscope to ignore the case of symbols. If you enter `Y`, Periscope will display `PERISCOPE` in all caps in the Watch window.

`Point 286/386/486 exception interrupt 6 to Periscope (Y/N)?`

Enter **N** if you're using software that uses this interrupt to intercept an illegal instruction. If possible, allow Periscope to handle this interrupt. It will help you capture runaway programs quickly.

Point 286/386/486 exception interrupt 0DH to Periscope (Y/N)?

Enter **N** if you're using software or hardware that uses this interrupt (IRQ 5). If you must use a mouse on this IRQ line, install the mouse before you install Periscope. If possible, allow Periscope to handle this interrupt.

Refresh interrupts 1, 2, and 3 each time Periscope is active (Y/N)?

Enter **N** if you don't want Periscope to ensure that these vectors point to it. This is useful if you want your program to filter INT 1, 2, or 3 before Periscope gets control.

Default to insert mode when Periscope's command-line editor is used (Y/N)?

Enter **Y** if you want Periscope to default to insert mode when you enter commands.

Use Periscope's menu system (Y/N)?

Enter **N** if you want to turn off Periscope's menu system. When the menu system is enabled with **Y**, CONFIG also loads the on-line help and interrupt comments, consuming approximately 60KB. If you need to minimize Periscope's use of DOS memory, you may want to turn the menu system off.

Function key use: Periscope (1), CodeView (2), or Turbo Debugger (3)?

Periscope's function key usage is configurable. You can choose the standard Periscope function keys, the CodeView or Turbo Debugger emulation keys, or configure your own keys. For details, see Section 8.5 and the NOTES.TXT file. ♦

Installing the Break-out Switch

■ Installing the Break-Out Switch

You'll find the information you need to install your Periscope Break-out Switch *without a board* in this chapter. If you have Model I or Model IV, you'll find installation instructions for your Break-out Switch, which plugs into your board, included in the hardware installation instructions for your board.

4.1 INSTALLING THE BREAK-OUT SWITCH

Before you install the Break-out Switch, turn the power off, remove the power cord from your PC, and obtain a small screwdriver and a flashlight. Figure 4-1 shows the switch installed in a PC.

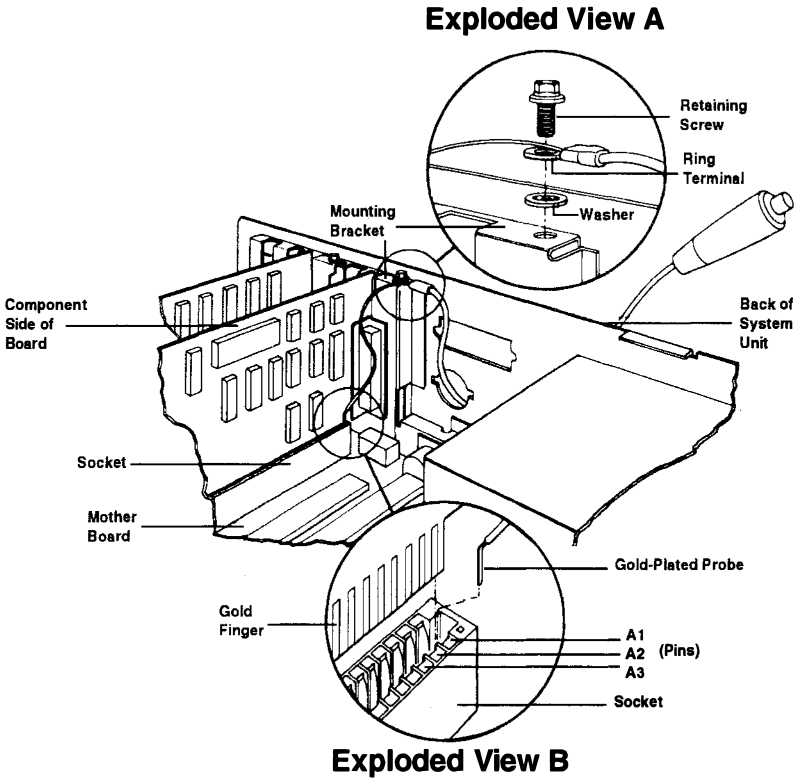


Figure 4-1. Installation of the Break-out Switch

Step 1—Open your system.

Remove the cover mounting screws on the rear of the system unit. Slide the cover of the system unit forward as far as possible without removing it from the system unit. **Be sure to ground yourself. While working inside the system unit,**

touch the chassis or other ground frequently to avoid the build up of static electricity.

Step 2—Route the connector end of the cable assembly into your system from the back of the unit.

The cable assembly has a push-button switch, a five-foot length of cable and two connectors. One of the connectors is a ring terminal (see Exploded View A in Figure 4-1) and the other is a gold-plated probe (see Exploded View B).

There are several possible ways of routing the cable into the PC, either through a knock-out panel or in the space between the keyboard connector and the expansion slots. Do NOT install the cable so that it is lying on top of the back panel. When the cover is installed, the cable may be damaged.

Step 3—Select an in-use slot, then install the ring terminal for ground and strain relief.

Remove the retaining screw on the expansion slot mounting bracket nearest to the power supply. This slot is usually, but not always, occupied by a disk controller card. If the slot nearest to the power supply is not in use, use the in-use slot that gives you the best accessibility to the component (chip) side of the board.

Insert the retaining screw through the ring terminal and then place the washer on the retaining screw. Re-install the retaining screw as shown in Figure 4-1. Align the cable so that it is parallel with the back panel of the system unit. Be sure it has a minimum of twists and turns between the ring terminal and the point where the cable comes into the system unit. The ring terminal provides both an electrical ground and strain relief. Be sure that it is securely installed.

Step 4—Install the gold-plated probe.

Using the flashlight or other bright light source, install the gold-plated probe into pin A1 of the expansion slot used in Step 3. The slot must be in use for secure attachment.

Pin A1 is the pin on the component (chip) side of the expansion board that is closest to the board's mounting bracket. This pin is used to generate a Non-Maskable Interrupt (NMI).

To install the probe, hold the probe so that it is pointing downward and the cable is angled away from the board. Push the probe down firmly into pin A1 between the gold finger on the board and the connector in the socket as shown in Figure 4-1. (Not all boards have a gold finger at pin A1. Look for the first socket connector to positively identify the pin.) Push the probe in as far as possible to ensure a good connection and to keep the non-insulated part of the probe from contacting anything other than the desired pin.

If you can't get the probe into the socket, try removing the board and sliding the board and the probe in together. On systems without enough room to get your hands near the socket, you may need to use a pair of needle-nose pliers.

Step 5—Double check the placement of the probe.
It should be in the socket on the component (chip) side of the expansion board nearest the board's mounting bracket. The probe must be between the board and the connector pin in the socket. It must not be between the connector pin and the outer edge of the socket. Some sockets have a dummy hole at the end of the socket. Do not insert the probe in this hole.

Step 6—Double check the placement of the ring terminal.
It should be firmly held by the retaining screw that holds the expansion board in place. For the best electrical contact, be sure that the supplied washer has been installed between the ring terminal and the board's mounting bracket.

Step 7—Slide the cover of the system unit back over your machine and install the cover mounting screws.

Step 8—Re-connect all peripherals; replace the power cord.

Step 9—Install the Periscope software.
Enter `PS` to install Periscope, press the Break-out Switch to display the Periscope screen, then enter `G`. Do not press the Break-out Switch before you install the Periscope software or you'll get a parity error.◆

C H A P T E R F I V E

Installing the Software

5

- **Installing Periscope**
- **Installation Options**
- **Alternate Start-up Methods**

This chapter describes how to install the Periscope software.

5.1 INSTALLING PERISCOPE/EM

To load Periscope, enter `PS` from the DOS prompt, followed by the appropriate installation options (details follow below). You can install Periscope multiple times per DOS session, but each install allocates more memory if you're loading it into DOS memory. Specify the `/Y` installation option to remove the previous copy from memory.

Press **Alt-Ctrl** while PS.COM is loading to terminate Periscope. Use this procedure to keep Periscope from loading from AUTOEXEC.BAT (or any other batch file) without having to modify the batch file.

5.2 INSTALLATION OPTIONS

All installation options contain a forward slash (`/`) and a one- to three-character mnemonic, except as shown. Some options include a number. Numbers are always preceded by a colon (`:`) and are in hexadecimal notation. The installation options are:

`?`—Display help information about Periscope's installation options.

`/2`:px—Run Periscope in active remote mode.
Use this option when you're running remote Periscope software, such as Periscope/Remote for DOS, Periscope/Remote for OS/2, or Periscope/Remote for Windows, on a target system. See your remote documentation for more information.

`/25`—Run Periscope in split-screen mode on a VGA.
Use this option to debug text-based applications on a VGA display. It sets the display to 50-line mode and uses the top 25 lines for your program and the bottom 25 lines for Periscope. You cannot use this option on an EGA or with graphics applications. For best results, use ANSI.SYS from prior to DOS 5.0 when using this option.

`/286`—Run Periscope Model IV in passive remote mode

where the target system contains an 80286 CPU.

To use this option, you must have a Model IV Pod installed in a target 286 system. See the Model IV manual for more information.

`/386`**—Use Periscope Model IV in passive remote mode where the target system contains an 80386 CPU.**

To use this option, you must have a Model IV Pod installed in a target 386 (DX or SX) system. See the Model IV manual for more information.

`/486`**—Use Periscope Model IV in passive remote mode where the target system contains an 80486 CPU.**

To use this option, you must have a Model IV Pod installed in a target 486 (DX or DX/2) system. See the Model IV manual for more information.

`/43` **or** `/50`**—Use the EGA 43-line mode or the VGA 50-line mode.**

When you specify one of these options, Periscope supports a single monitor in the specified mode, presuming that the display is already in that mode when the resident portion of Periscope is activated. These options reserve 8KB of memory for Periscope's screen and 8KB for the application's screen. Dual-monitor support is not currently available for 43- or 50-line mode.

If you set Periscope to run in 43- or 50-line mode with screen swap off and the program's screen is set to 25-line mode, Periscope will revert to 25-line mode and turn screen swap back on.

`/A`**—Use an alternate monitor.**

Use this option to indicate that you have both a monochrome and a color monitor attached to the system via separate display adapters. Periscope uses the monitor that is not currently active when you press the Break-out Switch or when you load a program with RUN. If you specify this option, Periscope ignores the `/S` option and reserves no memory for the program's or Periscope's screen buffer. Periscope ignores this option if it is unable to initialize the second monitor.

Periscope/32 users: This option is not available in remote mode.

`/AD [:nnn]`—Use a Dual-VGA as an alternate monitor.
See the NOTES.TXT file for details.

`/AK [:px]`—Use an alternate PC for Periscope's keyboard.
Use this option to specify that you are using an alternate PC for Periscope's keyboard. See the description of `/AV` below for more information.

Periscope/32 users: This option is not available in remote mode.

`/AV [:px]`—Use an alternate PC for Periscope's display.
Use this option to specify that you are using an alternate PC for Periscope's display.

The `/AK` and `/AV` options have two requirements:

- You must start the program PSTERM.COM on the alternate PC before you load PS.COM on the (local) debugging system. Please see the description of PSTERM in Section 7.4 for details.
- You must connect the two systems with a null-modem cable that has Receive and Transmit crossed and has CTS and RTS crossed. See the specifications for the null-modem cable in the file NOTES.TXT.

The `/AK` and `/AV` options default to using COM port 1 at the fast (115,200 baud) speed. To use another port or a different speed, specify `/AK:px` or `/AV:px`, where `p` is the port number (`1` through `8`) and `x` is the speed (`S`, `M`, or `F` for Slow (9,600), Medium (38,400), or Fast (115,200), respectively). Be sure that the speed used by PS.COM matches the speed used by PSTERM.COM on the alternate PC! For best results, use the Fast speed. If you encounter time-outs or other problems, try using Medium speed.

When you use an alternate video, the screen appears exactly as it does when Periscope runs on the same system. When you use an alternate keyboard and press any key on the alternate system, Periscope "wakes up" on the local system. Should you ever get a timeout message on the alternate system, avoid giving it the `Fail` reply since this can cause a system hang. For more information, see the description of PSTERM in Section 7.4.

Periscope/32 users: This option is not available in remote mode.

`/B:nn`—Set the size of the software trace buffer to something other than 1KB.

Use this option when you want more than the 32 trace entries available from the default buffer size. The one- or two-digit hexadecimal number `nn` is the size of the trace buffer in KB. The number may be from `0` to `3F`H, allowing a maximum of 2,016 entries. Remember that the input is in hex! When you use the Model I Board or EM memory, this buffer defaults to 8K.

The real-time hardware trace buffer available with Periscope Model IV is separate from this software buffer. See the Model IV manual for details.

`/C:nn`—Set the color attribute for Periscope's display.

Use this option to change the foreground and background colors of Periscope's display. The two digit number `nn` is from `1` to `FF`H.

To calculate the number for the colors you want, see the file NOTES.TXT or enter `/C` at the Periscope prompt. For example, if you want white on blue, enter `/C:1F`.

There are some illegal color combinations that Periscope won't allow. These include 0, 80H, and 88H, which are all variants of black on black, and other similar situations where the foreground and background colors are the same, such as 77H and F7H.

`/D`—Restore the original INT 13H vector before a short boot.

Use this option with certain RAM disk software to re-point the diskette interrupt vector to BIOS before you issue a short boot. You need to use this option only if the RAM disk software you use modifies the original interrupt 13H to point to memory corrupted by a short boot.

`/E:n`—Set the size of the source file buffer to something other than 2KB.

Use this option to improve performance when you use the Unassemble Source or View file commands. The one-digit hexadecimal number `n` is the size of the buffer in KB. The number may be from `0` to `8`.

You should not use this option unless you're debugging large files. Then make the buffer size one thirty-second of the size of the largest file.

When you're using the Model I Board or EM memory, the source buffer defaults to 8KB.

`/H`—Install the on-line help and interrupt comments.

Specify this option to load the files PSHELP.TXT and PSINT.DAT into RAM so they are available from Periscope. This option defaults to 'on' if you're using the menu system, EM memory, or a Model I Board.

The help and interrupt comments files must be in the current directory or Periscope path. (To set the Periscope path, enter **`SET PS=XXX`** at the DOS prompt, where **`XXX`** is the path.) The amount of memory required is the same as the size of the file.

The file PSHELP.TXT is a normal ASCII text file. You can edit it to add or remove help information. Be sure to leave the back-slashes and commands on a separate line and to end the file with a single back-slash.

The file PSINT.TXT is a normal ASCII file that contains interrupt and I/O port comments. It is compiled to the file PSINT.DAT when you run CONFIG. You can edit the .TXT file to add or remove interrupt or I/O port information. Each line in the file contains one of the following three formats:

1) An interrupt number, a space, a 2-byte function number (value of register AH), a space, and a description of up to 26 characters.
2) An interrupt number, a space, a 4-byte number (value of register AX), a space, and a description of up to 26 characters.
3) Two asterisks, a space, a 1- to 4-digit port number, a space, and a description of up to 26 characters.

Each line must end with a carriage return and line feed. The numbers are all in hex. If a function number is '`..`', Periscope displays the associated description for any value of AH or when the interrupt is not the current instruction. See the file PSINT.TXT for examples of each of the three formats.

Each line in the file PSINT.TXT uses 32 bytes in the file PSINT.DAT. The maximum size of the file PSINT.DAT is 64K, so the maximum number of lines in the file PSINT.TXT is 2048.

To load only one of the files PSHELP.TXT or PSINT.TXT, rename the other file.

`/I:nn`—Allow access to user-written code from Periscope.
The user-written program must be a memory-resident routine that is already installed using an interrupt from 60H to FFH, specified with `nn`. It must meet the specifications defined for the program USEREXIT in Section 7.10.

`/K:nn [nn]`—Mask hardware interrupt request (IRQ) lines when Periscope's screen is displayed.
Use this option to protect yourself from a hardware interrupt while you're in Periscope.

The value of `nn` may be from `0` to `FF`H. The first value corresponds to the first 8259. The optional second value corresponds to the second 8259 on AT-class machines. The default value is `0`, meaning mask no IRQ lines. A one bit in `nn` masks the corresponding IRQ line.

For example, to mask the timer (IRQ 0), use `/K:1`. To mask the keyboard (IRQ 1), use `/K:2`. (Don't do this unless you're using an alternate keyboard!) To mask IRQ 1 and IRQ 2, use `/K:6` (bits 1 and 2 on).

When you use a Trace, Go, or Jump command, interrupts may occur unless the DI (disable interrupts) flag is set.

`/L:nnnn`—Load Periscope starting at the specified segment.

If you're debugging non-DOS programs, you can use this option to place Periscope in an area of memory that is not corrupted by a short boot. The four-digit hexadecimal number is the segment where the tables should start. It must be greater than the current PSP plus 10H paragraphs and less than the top of memory minus 2000H paragraphs. For example, if the PSP is C00H and the top of memory is A000H, the limits for this option would be C10H through 8000H.

When you're using EM memory or the Model I Board's memory, Periscope ignores this option.

`/LCD`—Set menu colors for use with an LCD or Plasma display.

This option tells Periscope to adjust its default menu colors so they'll be readable on an LCD screen.

`/M:nnnn`—Set the Model I Board's paged memory segment to something other than D000H.

The Model I Board uses 32KB of system memory above the lower 640K but within the first megabyte. The default starting segment of this 32K is D000H. You can specify use of another segment with the four-digit hexadecimal number **nnnn**. Be sure that the segment you specify does not conflict with other memory installed in your system and that the memory settings on the board correspond to the value you use here. See the hardware installation instructions for the board.

`/N`—Run the Periscope software from conventional DOS memory.

If you want to run the Periscope software from conventional DOS memory (lower 640K), use this option.

`/NE`—Do not run the Periscope software from EM memory.

If you have a supporting memory manager installed but do not want to run from EM memory, use this option.

`/P:nnn`—Set the starting port to something other than 300H.

The Periscope Model I Board uses two consecutive ports and the Periscope Model IV Board uses eight consecutive ports. In both cases the default starting port is 300H.

Use this option if you need to change the starting port from the default. Be sure that you've also changed the DIP switches on your board(s) to the specified port (see the hardware installation instructions for the boards).

The architecture of the 8086 family supports 64K I/O ports (0 through FFFFH), but the IBM PC and compatibles support only the first 1024 (3FFH) of these ports, many of which are reserved. The high 6 bits are ignored, with the result that port 1200H is really 200H, etc.

`/Q`—Activate Periscope at ROM-scan time.

Use this option, which requires the Model I Board, to activate Periscope at ROM-scan time so that you can debug add-in ROM. See Section 6.10 for details.

`/R:nn`—Set the size of the record definition table to something other than 1KB.

Use this option to debug programs with large .DEF files and large resultant record tables. The one or two-digit hexadecimal number `nn` is the table size in KB. The number may be from `0` to `20`H. Remember that the input is in hex! See the description of RS, which determines the required size and verifies the .DEF file, in Section 7.5.

The record definition table size defaults to 8KB instead of 1KB if you're using EM memory or the Model I Board's

5

memory.

`/S:nn`—Set the size of the program's screen buffer to something other than 4KB.

Use this option only for single-monitor systems with a CGA, EGA, or VGA where a 4KB screen buffer is too small. The one or two-digit hexadecimal number `nn` is the buffer size in KB, from `0` to `40H`. If you need 16KB, enter **`/S:10`**. Remember that the input is in hex!

The maximum size allowable is 64KB, or **`/S:40`** in hex. Keep the number as small as possible, since each Trace or Go command has to copy this buffer twice. If you're using graphics modes that need more than 16KB on an EGA or VGA, we strongly recommend you use a dual-monitor system for speed.

Periscope ignores this option if you use the **`/A`**, **`/AD`**, or **`/AV`** options.

`/SX`—A 386SX chip is in use.

You must specify this option if you're using Model IV in a system with an 80386SX CPU.

`/T:nnn`—Set the size of the symbol table to something other than 1KB.

Use this option to debug programs with large symbol tables. The one- to three-digit hexadecimal number **`nnn`** is the size in KB. The number may be from `0` to **`1FFH`** (511). Remember that the input is in hex! See Section 7.9 for information on TS, which determines the symbol table size required and generates the .PSS file.

You can override the **`/T`** installation option when you use RUN to load a program. You can temporarily increase the symbol space Periscope uses with the appropriate RUN option. See the description of RUN in Section 7.6 for details.

You can specify a second **`/T`** option to set up two symbol tables. When you do this, the **`/1`** and **`/2`** commands enable you to switch between the two symbol tables while you're debugging.

When you use EM memory or the Model I Board's memory, the default symbol table size is 80H (128) KB.

`/V:nn`—Do not reset this BIOS interrupt vector while Periscope is active.

Normally, each time Periscope displays its screen, it first saves your program's interrupt vectors, then temporarily resets the interrupt vectors it uses to their power-on default values. (It does this so it will be as dependable as possible.) When Periscope returns control to the interrupted program, it restores the interrupt vectors to their prior values.

Assume your program changes INT 9. If you don't specify `/V:9`, Periscope saves your program's vector, swaps to a known good value (in BIOS) for its use, and then restores your program's vector on exit from Periscope. If you specify `/V:9`, Periscope uses your program's vector. If your program crashes, it may take Periscope with it.

If you have a situation where you want Periscope to use your modified interrupt vectors while it is running, specify the `/V:nn` option, where `nn` is a one- or two-digit hexadecimal interrupt number. The possible numbers are `8` (timer), `9` (keyboard), `10H` (video), `15H` (scheduler), `16H` (keyboard I/O), `17H` (printer), and `1CH` (timer control). Note that you must enter each vector separately. For example, to leave vectors 10H and 17H alone, use `/V:10` `/V:17`. Periscope temporarily changes the Ctrl-Break vector (1BH), the DOS Ctrl-Break exit address (23H), and the DOS Fatal error vector (24H), but you cannot override these changes.

386MAX and other 386 control programs must have interrupt vector 15H left alone while Periscope is active. To automate this process on a 386 system, Periscope looks for a 386 control program. Periscope automatically generates a `/V:15` installation option if it finds one.

Here are some specific situations where you should use the `/V` option:

- If you're using one of the keyboard translation programs (KEYBxx.COM), use `/V:9` to keep the keyboard handler available from within Periscope.
- If you're using Hercules graphics, run the public domain program HERC and use `/V:9` to be able to switch screen modes from within Periscope.
- If you're using a serial printer, run MODE and use `/V:17` to be able to access your printer from within Periscope.

`/W`—Set Periscope's windows.

Please see the Option W command in Chapter 9 for the syntax and a complete description. (You can set up Periscope's windows with this installation option or with the Option W command at the Periscope prompt.)

`/Y`—Remove the current copy of Periscope from memory.

Use this option to release any DOS memory used by Periscope and restore any vectors Periscope is using to their original values.

You should use this option alone. You cannot use it if you specified the `/L` option when you loaded Periscope.

➪ When using the Model I Board or EM memory, you do not need this option. You can run Periscope as many times as you desire. Each copy of Periscope will overlay the previous copy in the Model I Board's memory or extended memory, and will never consume any DOS memory.

➪ If you're running Periscope from conventional DOS memory and you've installed another resident program after Periscope, this option can leave a black hole in the middle of memory.

Examples.

You can enter the installation options in any combination of upper and lower case. No spaces are required between entries, except after numbers in the window specification. Some examples follow:

`PS/A/WD:8 RU:8` — Use two monitors and establish windows showing eight lines of data, two lines (default) of register information, and eight lines of disassembly.

`PS/M:A000/P:31C/S:10` — Use memory in the screen buffer area (starting at A000:0000) for the protected memory and use ports starting at 31C. Reserve 16KB to save the program's screen on a single-monitor system.

`PS/T:20/V:10/H` — Reserve 32KB for the symbol table, preserve the current INT 10H vector when Periscope is active, and load the on-line help file.

The cumulative size of the external tables can be from 0KB to over 511KB. These buffers are located in normal RAM using the terminate-and-stay-resident function of DOS when you run Periscope from DOS memory. No external buffers are used when you run from EM memory or the Model I Board's memory.

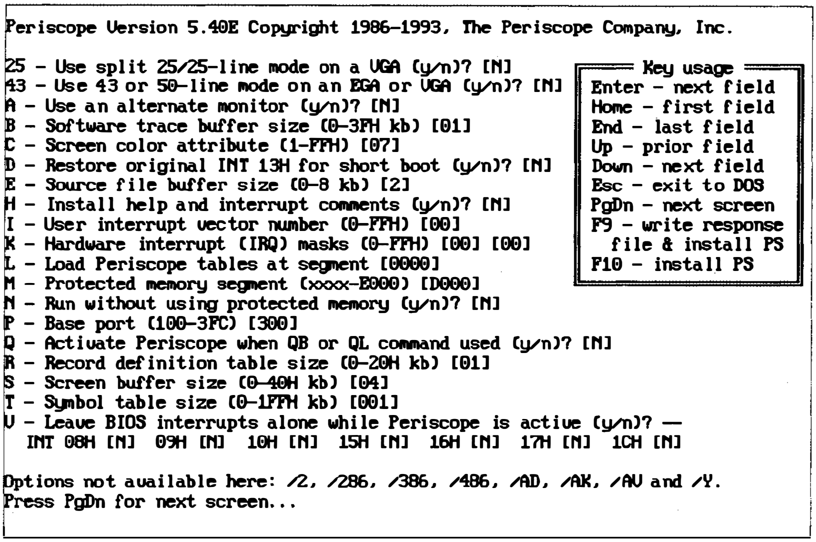
If Periscope encounters any errors during the initialization process, it displays an error message and terminates. See Appendix A for an explanation of the error messages. It is possible to hang the system by specifying an invalid port or memory address or by setting the DIP switches incorrectly. If you are not sure whether Periscope is installed, do not press the Break-out Switch. Execute RUN. If Periscope is not installed, RUN will display an error message.

5.3 ALTERNATE START-UP METHODS

Full-Screen Install. You can install Periscope via a full-screen display of the installation options. To use this method, enter `PS *` at the DOS prompt. Periscope displays the screen shown in Figure 5-1. A second screen, not shown, helps you customize the Periscope windows if you do not want to use the default windows. Periscope does not display information messages when you use the installation screen to install Periscope. It displays only error messages.

Once you have entered your responses, press **F10** to install Periscope. Note that you can also create a response file (see below) named PS by pressing **F9** instead.

The **/2**, **/286**, **/386**, **/486**, **/AD**, **/AK**, **/AV**, **/LCD**, **/NE**, **/SX** and **/Y** installation options are not available from Periscope's full-screen display.



```
Periscope Version 5.40E Copyright 1986-1993, The Periscope Company, Inc.

25 - Use split 25/25-line mode on a VGA (y/n)? [N]             ═══ Key usage ═══
43 - Use 43 or 50-line mode on an EGA or VGA (y/n)? [N]    Enter - next field
A - Use an alternate monitor (y/n)? [N]                     Home - first field
B - Software trace buffer size (0-3FH kb) [01]              End - last field
C - Screen color attribute (1-FFH) [07]                     Up - prior field
D - Restore original INT 13H for short boot (y/n)? [N]     Down - next field
E - Source file buffer size (0-8 kb) [2]                    Esc - exit to DOS
H - Install help and interrupt comments (y/n)? [N]          PgDn - next screen
I - User interrupt vector number (0-FFH) [00]               F9 - write response
K - Hardware interrupt (IRQ) masks (0-FFH) [00] [00]          file & install PS
L - Load Periscope tables at segment [0000]                 F10 - install PS
M - Protected memory segment (xxxx-E000) [D000]
N - Run without using protected memory (y/n)? [N]
P - Base port (100-3FC) [300]
Q - Activate Periscope when QB or QL command used (y/n)? [N]
R - Record definition table size (0-20H kb) [01]
S - Screen buffer size (0-40H kb) [04]
T - Symbol table size (0-1FFH kb) [001]
V - Leave BIOS interrupts alone while Periscope is active (y/n)? —
  INT 08H [N]  09H [N]  10H [N]  15H [N]  16H [N]  17H [N]  1CH [N]

Options not available here: /2, /286, /386, /486, /AD, /AK, /AV and /Y.
Press PgDn for next screen...
```

Figure 5-1. Periscope Installation Screen

Response Files. Normally, we recommend that you invoke Periscope from an AUTOEXEC.BAT file to ensure its presence each time you boot your system. If, however, you sometimes need different Periscope options for debugging different types of programs, the response file is for you.

The response file is an ASCII text file that contains Periscope installation options. For example, if you create a file named C:STD that contains **/B:4 /V:10 /A**, you can then enter **PS @C:STD** to load Periscope using the options in the file C:STD. Entering **PS /C:17 @C:STD** sets the color attribute and then retrieves additional options from the file C:STD. Periscope ignores any options you enter after the response file name. For example, **PS @C:STD /C:17** does not set the color attribute. You can use any legal file name for the response file.✦

Using Periscope

This chapter provides information for debugging in various environments supported by Periscope.

6.1 RUNNING FROM EM MEMORY

Periscope works with the three most popular 386 memory managers to give you the option of running the debugger software from extended memory rather than from conventional DOS memory (lower 640K) or from the Model I Board's memory. Two of the supporting memory managers write-protect the extended memory where Periscope resides, so no runaway program can corrupt the debugger software.

The functionality you get by running from EM memory is basically that you'd get by using the Model I Board, i.e., Periscope doesn't need any conventional DOS memory and (usually) is protected from being overwritten. The two features not available using EM memory that are available with the Model I Board are the ability to debug at ROM-scan time and support for pre-80386 machines.

To run from extended memory, you'll need a 386 or later PC, approximately 300K of extended memory, and 32K or 36K of memory between 640K and one megabyte. You cannot run Periscope from extended memory in a DOS box under Windows 3.x. Specifics for running with each of the supporting memory managers follow.

For all of the memory managers, don't let the optimization process load Periscope into high DOS memory. It needs to start in low DOS memory to have enough transient workspace to load.

386MAX or BlueMAX by Qualitas. You'll need 386MAX or BlueMAX version 5.11 or later and Periscope version 5.10 or later to run from EM memory. These memory managers write-protect the extended memory they provide for Periscope.

Place the option `PSMEM=nnn,mmmm` in your MAX profile where `nnn` is the decimal amount of memory in KB to assign and the optional `mmmm` is the hex memory segment you'd like MAX to use for the EM memory. The memory size should be a multiple of four. For example, the option `PSMEM=512` reserves 512KB of memory for use by Periscope at an address selected by MAX. The option `PSMEM=384,B000` reserves 384KB of memory for use by

Periscope at the monochrome screen address of B000H. If you specify a memory address, be sure that it is not being used by another device!

If you're not sure how much memory to allocate, try starting at 512KB and then adjust the amount upward or downward depending on the memory Periscope actually uses. Periscope will not load if insufficient memory is available, so err on the high side when starting out. If you get `Error 41-Not enough memory above 640K`, increase the memory specified with the PSMEM option, or change the Periscope installation options to reduce Periscope's memory requirements.

Periscope needs 32K of memory between 640K and one megabyte for paging when you're using MAX to provide EM memory.

We suggest that you place the `DEBUG=INV` statement in your MAX profile so that illegal instructions will be passed on to Periscope.

The OEM versions of MAX do not contain support for PSMEM. You'll need the full production version of MAX to get EM support for Periscope.

QEMM by Quarterdeck. You'll need QEMM version 6.0 or later and Periscope 5.40 or later to run from EM memory.

You do not need to make any changes to your QEMM386.SYS invocation. Just make sure that the `device=qemm386.sys` line contains the statement `ram` to make XMS memory available. At load time, Periscope will need at least 512K of extended memory. It will release any surplus when it completes loading.

Periscope uses a 36K block of XMS memory (the extra 4K assures starting on a 4K boundary) with QEMM.

Unlike the other memory managers, QEMM does not write-protect the EM memory it provides. For best results, use version 7.0 or later of QEMM. Version 6.x may cause a system reboot when you press the Break-out Switch.

NETROOM by Helix Software. You'll need NETROOM version 2.2 or later and Periscope version 5.31 or later to run from EM memory.

You do not need to make any changes to your RM386.SYS invocation. Just make sure that you have EMS and XMS memory available. At load time, Periscope will need at least 512K of EMS memory. It will release any surplus when it completes loading. Note that Periscope's use of EMS will not conflict with any EMS memory your program needs to use, nor will it inhibit you from debugging programs that use EMS memory.

Periscope uses a 36K block of XMS memory (the extra 4K assures starting on a 4K boundary) with NETROOM. If no UMB space is available, Periscope displays a warning message and loads into low DOS memory.

If you use NETROOM's BIOSHMA:FULL directive, be sure to use all possible **`/V:xx`** installation options when you load Periscope.

If you use the DOS=UMB directive in CONFIG.SYS, Periscope will use a 96-byte area in low DOS memory as an anchor for the memory it uses in the UMB region. To avoid this low memory usage, do not use the DOS 5.0 DOS=UMB directive.

6.2 SYMBOLS AND SOURCE SUPPORT

Symbols are the names you give to variables and procedures in your program. When you compile and link your program, the names are matched to their run-time memory locations. If you use a linker that outputs this name-and-location information to its .MAP file or .EXE file, Periscope is able to substitute the symbol names for the memory addresses, and can display source code while you're debugging. This symbolic capability makes debugging much easier and faster.

As an example, assume that a program you're debugging calls a subroutine named PRINTLINE, and you want to go to the first call of this subroutine in your program. You enter **`G PRINTLINE`**. (**`G`** is the mnemonic for the Periscope

Go command.) If symbols were not available, you'd have to know the address of the subroutine in order to get to it. When you disassemble this same program, the disassembly displays `PRINTLINE` wherever it is referenced in the program.

There are three major categories of symbols: public symbols, line-number symbols, and local symbols. **Public symbols** are global symbols that may reference code or data. **Line-number symbols** reference line numbers in your source program. Line-number symbols are the hook that makes source-level debugging possible. Both public and line-number symbols (but not local symbols) are available via the .MAP file for most languages.

Local symbols include local code, local data, and stack data. Local code and local data symbols are names that are valid only within the scope of a particular module, i.e., the current code segment matches that of the module and the instruction pointer is within the range of the module. The locations corresponding to these symbols are fixed and may be accessed even when the module is not active, although the symbol name cannot be used when the module is not active. Stack data symbols have the same scope restrictions, plus even further restrictions on the range of the instruction pointer, since the stack must be set up before these values can be accessed.

For some languages, such as the Microsoft and Borland products, local symbols are available in the .EXE file, and Periscope provides local symbol support via the .EXE file. Periscope does not currently support typing, structures, or register variables, however.

Debugging at the source level. To debug at the source level, you'll need to get line numbers into your symbol table by using the necessary compile and link options (see Section 6.3 below). You can verify that line numbers are in your symbol table by using the `/V` switch when you run TS.COM (see Section 7.9). This will display the count of various symbols, including line numbers. If no line symbols are indicated, recheck your compile and/or link options.

We suggest that you specify the full path name of the source file to the compiler. That way, the full path name will be passed on to the linker and to Periscope so that Periscope can find your source file no matter what directory you debug your program from. If you are debugging on a system whose file structure is different than the system the program was compiled on, you can use the DOS APPEND program to pull together the source code from one or more directories.

When using RUN to load a high-level language program, you'll start at the very beginning of the program, which is usually an assembly prolog. To get to your source code, assuming a C program, enter **`G _MAIN`**.

If you have problems displaying source code, check the following:

- Use the `/L` command to confirm the presence of line symbols, the availability of DOS, and the presence of a source file buffer. See the `/L` command in Chapter 9.
- You must be disassembling memory in an area where source line symbols exist (you should see some symbols of the form module name plus # plus the decimal line number).

Notes on Periscope's Symbolic and Source-level Support. A single source statement that spans two or more lines may generate only one line-number reference, causing Periscope to show only one line when displaying the source code.

Line numbers are formed by the module name, a #, and the line number. For example, line number 12 in the module FTOC is called **`FTOC#12`**. When in the module, the short form **`#12`** may be used.

For best results, turn code optimization off when using source-level debugging, since the line sequence can become quite scrambled when optimization is on.

Symbol names may be up to 32 characters. Any characters beyond the 32-character limit are discarded by Periscope.

To force Periscope to interpret a name as a symbol, use a period before the name. To force Periscope to interpret an item as not being a symbol, use a tilde (~) before the name.

To display the current symbols, press **Alt-I**. Unloaded symbols are shown in parentheses. These unloaded symbols can only be referenced by a `BC` or `G` command. To see all symbols for an address, enter `/address` and press **Alt-I**.

6.3 SUPPORTED COMPILERS AND LINKERS

Periscope works with programs written in any language. However, the amount of symbol and source-level support Periscope can provide to you varies depending on the compiler and linker you use. This is because Periscope normally gets symbol information from the .MAP or .EXE file output by the linker, and the linker gets symbol information from the .OBJ files generated by the compiler.

If your compiler and/or linker are not discussed below, please call Technical Support. We'll be happy to help you figure out how to get the maximum symbol support for your programming environment.

Borland's Assembler. Use the `/Zi` option.

Borland's Compilers. Use the `-v` compiler option to output public, line-number, and local symbols to the .OBJ file.

Microsoft's Assembler and Compilers. Use the `/Zi` compiler option to output public, line-number, and local symbols to the .OBJ file.

SLR's OPTASM. Use the `/Zi` option.

Other Compilers. Here's how you can find out what symbol information your compiler provides:

- Compile a test program.
- Run the IBM or Microsoft linker, specifying a .MAP file and the `/LI` and `/M` options.
- Look at the .MAP file. Any of the names in the 'Publics by Value' list or any of the line numbers at the end of the .MAP file can be used as symbols.

External references to other modules or subroutines are always included in the .MAP file. Line numbers indicate the beginning of a source line in memory and are required for Periscope source-level support.

Borland's TLINK. Use the `/M` and `/LI` options to output public and line-number symbols to the .MAP file. Use the `/V` option to output public, line-number, and local symbols to the .EXE file.

Microsoft's LINK . Use the `/M` and `/LI` options to output public and line-number symbols to the .MAP file. Use the `/CO` option to output public, line-number, and local symbols to the .EXE file.

SLR's OPTLINK. Use the same options as with Microsoft's LINK.

PocketSoft's .RTLink. Specify a .MAP file and use the `S`, `A`, and `L` options to output public and line-number symbols. Periscope does not support local symbols or the VML feature of .RTLink.

PLINK. Use SYMTABLE as a linker directive to output all possible symbols to the .EXE file. PLINK does not provide local symbol information, so Periscope cannot provide local symbol support for PLINK programs.

6.4 DEVICE DRIVERS

You can use the utility program SYSLOAD.SYS to load Periscope at CONFIG.SYS time. Then Periscope is available to debug your device driver. See the description of SYSLOAD in Section 7.8.

6.5 PLINK APPLICATIONS

Using Periscope, you can set breakpoints in PLINK overlays. Be sure to set the breakpoints on program symbols only, however, since other locations are subject to being relocated without communication back to Periscope. The short form of the line number cannot be used to set breakpoints when PLINK overlays are used. Do not set a breakpoint at the symbol $LDEX$, since Periscope uses this location.

When loading a program generated with PLINK, RUN always uses the DOS EXEC function to load the .EXE file.

At link time, specify the `SYMTABLE` directive. When you run TS, use the `/Q`, `/RP`, and `/RX` options. Also, set up the **MP** and **MX** aliases using RS. (See Sections 7.5 and 7.9 for more information on RS and TS.)You may also interactively enter the aliases using the `EA` command. (See Chapter 9.)

6.6 .RTLINK APPLICATIONS

You can set breakpoints in .RTLink overlays and you can get access to global and line-number symbols (no locals) for standard or overlaid programs. You'll need version 2.05 or later of .RTLink and you'll need to use the `/P` option when you run TS.COM. When you use RUN.COM to load your program, it will use the DOS EXEC function. If you ever debug an .RTLink .EXE without a symbol table, be sure to use either the `/T` or `/X` option with RUN.COM.

When setting breakpoints in overlays, be sure to use symbol names instead of addresses, since Periscope needs to know the overlay to correctly set the breakpoint. Be careful to not execute across an overlay reload using the Jump command, since this can cause Periscope to lose control. When you debug a program that uses reloads, set code breakpoints at the .RTLink vectors (see the .VEC file) to avoid problems using the Jump command.

If you're debugging programs that use RTLs, execute the following Periscope commands immediately after loading your program:

```
j;j;j;ls ds+10 xxxx;g $$main
```

replacing `xxxx` with your program's name. This sequence must be used to execute through the loading of the run-time library before the symbol table is loaded.

6.7 MICROSOFT WINDOWS AND IBM OS/2

Use Periscope/32 for Windows for debugging system-level software running under enhanced-mode Windows 3.x. Use Periscope/32 for OS/2 for debugging system-level software running under OS/2 2.x. For details, see Section 1.5, Appendix C, and the addendum included with the Periscope/32 product you have.

You can use Periscope to debug DOS applications running in the DOS box under Windows 3.x. While in a Windows DOS box, neither debug register support or EM memory is available.

We do not recommend that you use Periscope in the DOS box under OS/2 2.0 due to incompatibilities between the OS/2 DOS box and "real" DOS. The problems are caused mostly by OS/2's lack of full instruction emulation for the DOS box. We have addressed the problems with IBM and hope to see them corrected in future versions of OS/2.

6.8 NON-DOS AND PRE-DOS PROGRAMS

For non-DOS or pre-DOS programs, install Periscope normally, then press the Break-out Switch to get into the debugger. Then enter `QS` to perform a short boot. This technique can be used to cross-boot into another operating system, a non-DOS environment such as a self-contained program, or back into DOS. The short boot performs an INT 19H, but destroys INT 2 when you use DOS 3.20 or later (see the NOTES.TXT file). If you are debugging non-DOS or pre-DOS programs, you can use the Break-out Switch after a short boot to get back into Periscope. If the timing is critical, embed an INT 2 or INT 3 in the code itself.

When performing a short boot, be aware that any DOS memory used by Periscope is no longer an extension of DOS and may be used by another program or garbled during the boot process. For best results, if you're using the `/N` installation option to run Periscope from conventional memory, also use the `/L` installation option to place Periscope's code in the middle of conventional memory. If you're using the Model I Board's memory, all of Periscope's code and data are on the board and won't be corrupted by the short boot.

6.9 DEBUGGING AT ROM-SCAN TIME

If you have a Model I Board, you can have Periscope become active at ROM-scan time. Here's how:

Boot the system normally and install Periscope with the `/Q` installation option (see Chapter 5). Then enter the `QL` or `QB` command to perform a long or normal boot. When the ROM scan of the Model I Board's protected memory occurs, Periscope displays its screen. You can then trace through the remainder of the boot process. There are three caveats:

- The Model I Board's protected memory must be in the ROM-scan region (usually C800 to E000)
- Recent versions of DOS zap INT 1 and INT 3 while DOS is loading, so be careful not to set a code breakpoint across this section of code.
- After you enter `QB` or `QL`, you cannot use DOS until you load another copy of Periscope.

➪ If your system uses shadow RAM, the video interrupt vector (INT 10) may point to memory that is not valid at ROM-scan time. For example, on Compaq 386 systems, the VGA ROM at C000 may be remapped to faster RAM at E000. If this is the case, either disable the RAM shadowing, repoint the interrupt vector to C000 before you use the QL or QB command to boot the system, or use the /V:10 installation option when you install Periscope.

This method is better than the method described at the beginning of this chapter, using the `QS` command, since that command does not work in some systems.

6.10 HARDWARE INTERRUPTS

To debug hardware interrupts most easily, the code should be in RAM so that you can set code breakpoints in the interrupt service code. Periscope's `T`, `GA`, or `GT` commands will not trace into hardware interrupts such as the Timer tick (IRQ 0) and the Keyboard (IRQ 1), so you must set a code breakpoint to be able to stop in the interrupt code. Once stopped, you can trace through the code as desired.

For serious hardware interrupt work, you need Periscope Model IV to be able to set real-time breakpoints and to see just what happens when your code runs at full speed. See the Model IV manual for more information.

If Periscope must issue a non-specific End of Interrupt (EOI) to clear interrupts, it displays a message of the form `EOI issued for IRQ x`, where `x` is zero for the timer and one for the keyboard, respectively. The EOI is one of the few things that Periscope cannot undo when it returns control to your program, so you may want to use a semaphore mechanism to keep your code from being re-entered while you're debugging it. On AT-class machines, Periscope can issue an EOI for IRQ 0 and 1 only, but on a PC-class machine, Periscope may issue an EOI for any IRQ level.

If you're debugging a keyboard or video handler, you may want to use an alternate PC for the keyboard or video. (See Section 6.12.)

To prevent hardware interrupts from being executed while Periscope is active, use the `/K` installation option when you install Periscope.

6.11 MEMORY-RESIDENT PROGRAMS

To debug a memory-resident program, use RUN to load the program and its symbol table. The program will load in the same location it would if you were to run it directly from the DOS prompt. Enter `G` to install the program and return to the DOS prompt. Until you use RUN to load another program, the symbol table will remain available, ready for you at a push of the Break-out Switch! Source code will be available only when DOS is not busy, however. To keep DOS from being busy, use the WAITING.COM program described in Section 7.11.

6.12 USING AN ALTERNATE PC

You can use an alternate PC for either an alternate screen or keyboard or both. Use the `/AV` installation option to select alternate video support, and the `/AK` option to select alternate keyboard support. See the descriptions of the `/AK` and `/AV` installation options in Chapter 5 for more information.

6.13 USING AN EGA OR A VGA

Periscope supports the various EGA and VGA video modes. For single-monitor systems, a maximum of 64K of the screen buffer is saved and restored by Periscope. Generally, for graphics work, you'll want a dual-monitor system for best results.

Periscope has limited support of the EGA's 43-line mode and the VGA's 50-line mode. See the description of the `/25`, `/43` and `/50` installation options in Chapter 5.

6.14 PS/2 MACHINES

Periscope supports dual VGAs on PS/2 systems using the Dual-VGA board made by Colorgraphic Communications. See the NOTES.TXT file for more details. You can also use another PC for an alternate keyboard, video, or both.

Periscope supports the PS/2 watchdog timer. See the description of CONFIG in Chapter 3 for details.

6.15 DEBUGGING SPAWNED (CHILD) PROCESSES

If you want to debug a program that must be loaded by another program, you won't be able to use Periscope's program loader RUN.COM to get your symbols loaded. You can still use Periscope to debug your program with one minor change — simply embed an INT 2 (software NMI) or an INT 3 (code breakpoint) in your program, possibly activated by a command-line switch.

When your program executes the interrupt, it will activate

Periscope. To load the symbols for the program, enter `LS $ <file>`, where `<file>` is the name of your program. This syntax assumes you're debugging an .EXE file. If you are debugging a .COM file, enter `LS CS <file>`.

From here, you can set breakpoints in the spawned program and debug it. Also, for debugging multiple processes, be aware of the dual symbol table support provided by Periscope. Please see the description of the `/1` and `/2` commands in Chapter 9.

To automate the above process, you may wish to use the SYMLOAD utility. Please see Section 7.7 for more information on this program. ◆

C H A P T E R S E V E N

7

Periscope Utilities

- **Clearing NMI (CLEARNMI)**
- **Your Program's Interrupts (INT)**
- **Setting up Hot Keys (PSKEY)**
- **Using an Alternate PC (PSTERM)**
- **Record and Alias Definitions (RS)**
- **Periscope's Program Loader (RUN)**
- **Loading Symbol Tables (SYMLOAD)**
- **Debugging Device Drivers (SYSLOAD)**
- **Generating Symbol Tables (TS)**
- **Customizing Periscope (USEREXIT)**
- **When DOS is Busy (WAITING)**

This chapter describes the capabilities of Periscope's utility programs and how and when to use them.

7.1 CLEARING NMI (CLEARNMI)

Despite the name, the non-maskable interrupt (NMI) used by the Break-out Switch can be masked out. Since the NMI signal travels through two ports going from the expansion bus to the CPU, these ports can disable the Break-out Switch. The memory-resident utility program CLEARNMI attaches to the user timer interrupt (1CH) and clears these two ports once a second.

To use it, load CLEARNMI anytime, preferably from your AUTOEXEC.BAT file. There are three options:

`/Q`—Use only if you configured Periscope to run on a PC- or XT-based machine with a 286 or later turbo card installed.

`/R`—Use to refresh the NMI vector to point to Periscope once every 18 timer ticks (the program 'learns' Periscope's address and then refreshes INT 2 as needed).

`/Y`—Use to remove the current copy of CLEARNMI from memory.

7.2 YOUR PROGRAM'S INTERRUPTS (INT)

Use this program to display, save, or compare interrupt vectors. The three usage modes are:

`INT <file> /W`—save the current interrupt vectors to a file.

`INT [<file>] [/D <lowint> <hiint>]`—display the previously-saved interrupt vectors from a file (if no filename is present, INT displays the current vectors). The optional numbers `lowint` and `hiint` indicate the range of vectors to be displayed.

`INT <file> /C`—compare the interrupt vectors in a previously-saved file with the current interrupt vectors.

To see the interrupt vectors used by a resident program, save the current vectors using the `/W` option, load the program in question, and then compare the current vectors with the saved vectors using the `/C` option.

For example, to see the interrupts used by CLEARNMI, enter `INT CLEAR/W` to save the current interrupt vectors. Load CLEARNMI from DOS. Enter `INT CLEAR/C` to compare the current interrupt vectors with the values saved in the file CLEAR. You can also display the saved vectors using `INT CLEAR/D`.

7.3 SETTING UP HOT KEYS (PSKEY)

PSKEY is a memory-resident utility program that enables you to select your own hot-key combination to activate Periscope. This program can use interrupts 2 or 3, 5, or 15H to activate Periscope, depending on the command-line options. It always uses interrupt 9.

The command-line options available are:

`3` — Use INT 3 instead of INT 2 to activate Periscope
`A` — Use **Alt** (combined with other shift keys)
`C` — Use **Ctrl** (combined with other shift keys)
`I` — Use **Insert** (combined with other shift keys; must be first key pressed)
`L` — Use left **Shift** (combined with other shift keys)
`P` — Use **Shift-PrtSc** to activate Periscope (via INT 5)
`R` — Use right **Shift** (combined with other shift keys)
`S` — Use **SysReq** to activate Periscope (via INT 15H)
`/Y` — Remove the current copy of PSKEY from memory

For example, `PSKEY LR` would activate Periscope when the left and right **Shift** keys are simultaneously pressed.

Avoid using the `P` or `S` options since Periscope comes up in BIOS, far away from your code. Since the **Shift** key combinations 'back-end' the keyboard interrupt, a single Trace command puts you back to the code that was interrupted by the key press. You can define the **Shift** key combinations as needed to avoid conflicts with other memory-resident programs.

If PSKEY cannot find Periscope via interrupts 2 or 3 when you press the hot keys, it displays the message `Error`

`150 - Int x does not point to Periscope!` Since PSKEY loads into normal memory and is dependent on hardware interrupts being enabled, use the Break-out Switch for maximum dependability.

7.4 USING AN ALTERNATE PC (PSTERM)

To use Periscope's alternate PC support, you'll need another PC with an available COM port and a null-modem cable similar to the ones provided with Brooklyn Bridge and Laplink. See the file NOTES.TXT for a description of the cable required.

There are four options:

`/AK` and `/AV`—Use to enable the keyboard and video *only* when you must restart PSTERM on the alternate PC without restarting Periscope on the main system.

`/2:px`—Use to specify the com port, `p` (`1-8`), and the speed, `x` (`S`=Slow, `M`=Medium, or `F`=Fast), when you're using other than com port 1 at the fast speed. Use the fast (115,200 baud) speed if at all possible.

`/T`—Use *only* to test the serial ports and null-modem cable on both the main and alternate systems. You can run `PSTERM /T` on either system first. Be sure to also specify the `/2:xy` option if you're using other than com port 1 at the fast speed. If PSTERM displays `No errors detected` on both systems, the serial ports and cable are okay for use with Periscope. If PSTERM displays `Waiting for slave` on both systems and does not appear to proceed on either system, either the serial cable does not have transmit and receive crossed properly or you're using an incorrect com port. See Appendix A for other error messages.

Load PSTERM.COM in the alternate system first, specifying the `/2:px` option if needed. On the host system, you'll need to install Periscope using the `/AV` and/or `/AK` installation options after PSTERM is running on the alternate system. Be sure to specify the `:px` syntax if you use other than port 1 at the fast speed. See the description of the `/AV` and `/AK` installation options in Chapter 5 for more information.

To terminate PSTERM on the alternate system, press **Alt-Ctrl**.

7.5 RECORD AND ALIAS DEFINITIONS (RS)

Use RS to verify and size a record definition file. RS reads a .DEF file containing record and alias definitions and creates a Periscope definition (.PSD) file. RUN loads these definitions to provide support for Periscope's `DR` command and commands that use the aliases.

To run RS, enter **`RS filename`** from the DOS prompt. RS assumes an extension of .DEF. When you install Periscope, specify the `/R` installation option with the size RS shows. You can load the .PSD file while in Periscope using the `LD` command.

> You must edit .DEF files created prior to Version 5 to change **F1** through **F10** to **Ctrl-F1** through **Ctrl-F10**, since beginning with Version 5, **F1** through **F9** refer to key assignments for the function keys **F1** through **F9**. (**F10** is represented as **F0**.) Also, you must regenerate all .PSD files created prior to Version 5.

Figure 7-1 shows a section of the PS.DEF file. It contains two alias definitions and a record definition. The third line of the file starts a record definition named `FCB`.

Record Definitions

The format for a record definition is:

- A backslash and the record name of up to sixteen characters on one line
- One or more field definitions that contain the following separated by commas:
  - The field name of up to ten characters;
  - the display type (any display format) plus an optional bracket or brace indicating a near or far pointer re-

spectively;
- the field length (total length in hex bytes);
- optional comments preceded by a semi-colon (no preceding comma)

```
\c1=k;dr cs:0 .psp;
\c2=dr cs:5c .fcb;
\FCB                    ; File Control Block
Drive,b,1               ; Drive 0=default, 1=A, 2=B, etc.
File,b,8                ; File name
Ext,b,3                 ; File extension
Block #,w,2             ; Current block number
Rec Size,w,2            ; Logical record size
File Size,d,4           ; File size
Date,w,2                ; Date of last update
Res.,+,a                ; Reserved for DOS
Rec #,b,1               ; Current relative record number
Rel Rec #,d,4           ; Relative record number from beginning of file
```

Figure 7-1. A Section of the PS.DEF File

Place each field definition on a line by itself. If the field display type is long real (**L**), the length must be a multiple of eight. If the field display type is long integer (**X**), double word (**D**), or short real (**S**), the length must be a multiple of four. If the field display type is word (**W**), integer (**I** or **Y**), or number (**N**), the length must be a multiple of two. The length of any one field and the total length of the record may be from **1** to **FFFF**H. A field display type of **+** skips over the indicated number of bytes without displaying anything.

The record definition can reference the contents of near and far pointers. To use a field as a pointer, add a bracket (**[**) for a near pointer or a brace (**{**) for a far pointer immediately after the field display type. A near pointer always uses a word offset from the current segment and a far pointer always uses a double word segment:offset.

For example, the field definition **farptr,b{,10** uses the double word at the current location as a far pointer and displays the target of that pointer in '**b**'yte format for **10**H bytes, with a name of **farptr**. Similarly, the field definition **nearptr,d[,4** uses the word at the current location as a near pointer and displays the target of that pointer in '**d**'ouble word format for **4** bytes, with a name of `nearptr`.

You must individually describe each pointer in the record definition. **Periscope** supports only one level of indirection. To remind yourself that the field is a pointer, you may wish to start the field name with a bracket or brace.

Aliases

The other type of definition in a .DEF file is the alias. An alias is a two-character mnemonic that represents a string of up to 64 characters. You may enter an alias as a line in a .DEF file or by using the **`EA`** command (see Chapter 9).

You may use aliases in various ways:

- To assign commands to function keys (**F1** through **F10**, **Ctrl-F1** through **Ctrl-F8**, **Alt-F1** through **Alt-F10**, and **Shift-F1** through **Shift-F10**)
- To store special strings for use by Periscope (see the **MP**, **MX**, and **X0** through **X3** aliases below)
- To assign character strings or commands to any alias that is not reserved by Periscope and then to execute the alias by typing `^` and the two-character alias name. You can chain (not nest) aliases by embedding a **`^xx`** at the end of one alias to start up the next one. Watch out for infinite loops, such as `^Y1` chaining to `^Y2`, which chains to `^Y1`.

The first two lines of the file in Figure 7-1 contain alias definitions for function keys **Ctrl-F1** and **Ctrl-F2**. While in Periscope, you can execute these aliases by pressing **Ctrl-F1** or **Ctrl-F2**.

The format for an alias definition in a .DEF file is:

- A backslash
- The two-character alias name
- An equal sign
- The one to 64-character string to be assigned to the alias

No spaces are allowed until after the equal sign. If you want multiple commands, use a semi-colon to separate the commands. If you want the command to be executed immediately, place a semi-colon at the end of the line. There may be up to 128 alias definitions in a .DEF file.

Reserved Aliases. There are various aliases reserved by Periscope. The reserved aliases are:

MP — The module path name for source-level debugging. If used, this alias should be the complete drive and path name, ending in a backslash.
MX — The module extension for source-level debugging. If used, this alias should be the file extension, starting with a period.

> Different linkers put different information in the .MAP and .EXE files, so Periscope may or may not need the **MP** and **MX** aliases to construct the full source file name. Periscope concatenates the path, name, and extension to get the module's full file name. If Periscope prompts you for a source file name, press **Alt-C** to see the file name Periscope tried to use. You can edit the file name and press return to try the file again. To fix the problem permanently, set the **MP** or **MX** aliases by using the `EA` command or by placing them in the .DEF file for the program. If the path is missing or incorrect, use the **MP** alias to automate source-level debugging of the program. Similarly, if the file extension is missing or incorrect, use the **MX** alias. Note that the path name used with the **MP** alias should end in a backslash.

X0 — The commands executed when RUN transfers control to Periscope on entry to the program.
X1 — The commands executed on entry to Periscope each time control is transferred from your program to Periscope.
X2 — The commands executed after each Periscope command.
X3 — The commands executed on exit from Periscope each time Periscope transfers control to your program.
C0 — The commands to restore the last Periscope window settings (**Ctrl-F10**).
C9 — The commands to restore the original (default) Periscope window settings (**Ctrl-F9**).
Fx — Defines the contents of the menu bar. Used with xxKEYS.PSD. See NOTES.TXT. (These are marked read-

only.)

These are the aliases you can custom-define in the .DEF file:

A1 through A9 and A0 — Assign string to **Alt-F1—Alt-F10.**
C1 through C8 — Assign string to **Ctrl-F1—Ctrl-F8.**
(**Ctrl-F9** and **Ctrl-F10** are reserved by Periscope.)
F1 through F9 and F0 — Assign string to **F1—F10.**
S1 through S9 and S0 — Assign string to **Shift-F1—Shift-F10.**

See also Sections 8.3 and 8.5.

7.6 PERISCOPE/EM'S PROGRAM LOADER (RUN)

The program RUN can load data files, .COM and .EXE files, or no file at all.

Start RUN by entering

```
RUN [/2] [/T|/X[:nnn]] [<file>] [<com-
mand line>]
```

at the DOS prompt, where `<file>` is the path, file name, and extension (.EXE, .COM or other) of the file you want to load. If you do not specify the file extension, RUN presumes .COM, then .EXE, then no extension.

When you enter `RUN` with no arguments, RUN loads no file. It sets the first instruction to INT 20H, the DOS return, to prevent accidental execution of meaningless data. Use this technique to clear and initialize Periscope. RUN performs these tasks when you execute it:

- Resets Periscope's display address to the PSP segment at offset 100H.
- Clears some of Periscope's tables, including the software trace buffer, source file buffer, screen buffers, record definition table, and symbol table.
- Disables any breakpoints that are set to avoid possible interference with the current program being debugged.
- Reverses out any code breakpoints that are set, so as not to leave an INT 3 in the code.
- Clears the breakpoints and watch variables for a program if the date/time of the program is different than the last program so that one program's breakpoints/watch vari-

ables won't be used on another program.

- Resets the Display windows to the standard values.

Loading a data file with RUN enables you to patch the file. If you load a data file, be sure to use the `QR` command to quit Periscope and return to DOS. Using the `QC` or `G` commands would have unpredictable results.

Enter the same command-line after the filename as you would enter if you were running from DOS. RUN adjusts the FCBs and command line in the PSP to look like the target program was started directly from DOS.

If you're loading an executable program (.COM or .EXE file), RUN loads the related symbol table if it finds a .PSS or .MAP file. It loads the related record definition table if it finds a .PSD or .DEF file.

To locate the symbol file, RUN first searches the specified directory for a file of the form **`filename.PSS`**. If it finds this file, it uses it to load the program's symbols. If it doesn't find a .PSS file or if the date of the .PSS file precedes the date of the executable file, it searches the directory for a file of the form **`filename.MAP`**. If it finds it, RUN executes the TS program to generate a .PSS file.

RUN executes TS with no command-line options, so TS assumes a .MAP file. If you've set the Periscope path (**`SET PS=`**), RUN uses it to find TS.COM. Otherwise, it uses the DOS path. If you have a copy of Peter Norton's TS.EXE in your DOS path, you'll need to set the Periscope path to avoid executing the wrong TS.

If no symbols are available or the symbols will not fit in the allocated space, RUN clears the symbol table. If it finds source lines in the .PSS file, RUN sets Periscope for Source-only mode. If it finds no source lines, it defaults to mixed (Both) mode.

To locate the record definition file, RUN first searches the specified directory for **`filename.PSD`**. If it does not find this file, it then searches for **`filename.DEF`**. If doesn't find this file, it then searches the Periscope directory, which you can set by entering **`SET PS=xxx`**, where **`xxx`** is the path required to find the file PS.PSD. If it doesn't find PS.PSD, it searches for PS.DEF. It uses the .PSD or .DEF file to load Periscope's record and alias definition tables. If it doesn't find a definition file, it clears the record and alias definitions. If it finds an error in the .DEF file, it will partially load the record definition table.

Once RUN loads the symbol and definition tables, it relocates itself upward and reads the target program into memory beginning at RUN's original location, and performs any segment relocation required by .EXE files. It sets the register pair BX:CX to the size in bytes of the target file. It sets other registers according to the rules for loading .COM and .EXE files (see the DOS technical reference manual).

Starting with DOS 3.00, the drive, path, and filename of the loaded program is stored at the end of the environment space. Since RUN does not use the EXEC function to load programs (unless you use the `/T` or `/X` option), this area shows RUN rather than the target program as the loaded program . The environment space is of variable length and is followed by DOS's memory allocation blocks, so it is not safe for anything but DOS to modify the environment. If your program uses this information, specify the `/T` or `/X` options described below.

RUN loads your program exactly DOS would load it under the same conditions, so you can use RUN to load memory-resident programs. Until you use RUN again, the record definition, alias, and symbol tables are preserved.

Finally, RUN passes control to the resident portion of Periscope. When you've finished debugging your program, you can exit Periscope in one of three ways: use the `G` command with no breakpoints set, use the `QC` command to quit Periscope and continue execution, or use the `QR` command to quit Periscope and return to DOS. If you use the last option, be sure that all output files are closed and that any interrupt vectors your program has modified have been reset to their original values.

If you're using two symbol tables, you can have RUN load the program's symbols into the second symbol table using `RUN /2 <file>`. If you do not specify this option, RUN loads the symbols into the first symbol table. You can only use this option if you specified two `/T:nnn` installation options when you installed Periscope.

You can temporarily increase the symbol space Periscope uses with two RUN options, `/X[:nnn]` and `/T`. When you specify either of these options, RUN uses the DOS EXEC function to load your program.

Allocate temporary symbol space by entering the `/X:nnn` option immediately after RUN. The optional `nnn` is from `1` to `1FFH` (511) KB. For example, if the symbol size was set for 4KB, and the program TEST needs 8KB, enter `RUN /X:8 TEST.EXE` to allocate the temporary symbol table and load the program with the DOS EXEC function.

Enter the `/T` option immediately after `RUN` to allocate the exact amount of symbol space required by the program, with at most 1KB to add symbols dynamically. This option also uses the DOS EXEC function to load the program.

When running Periscope from EM memory or the Model I Board's memory, the only effect of the `/X` and `/T` options is to use the DOS EXEC function to load your program.

If you experience problems using RUN, you should use the `/X[:nn]` option or the `/T` option. They both use the DOS EXEC function, so your program loads approximately 8KB higher in memory than otherwise, but the program name in the environment space is correct. To use the EXEC function, add `/T` or `/X[:nnn]` immediately after RUN. For example, enter `RUN/T FTOC`. If an EXEC error occurs, RUN displays error 182, which usually indicates a bad path name.

You must load programs linked with PLINK or .RTLink with the `/T` option or the `/X` option since these programs look for their name in the environment space. Periscope handles this automatically if you use a symbol file.

7.7 LOADING SYMBOL TABLES (SYMLOAD)

Use this program to load Periscope's symbol tables from within your program when your program manages overlays or when you do not use RUN to load your program. You can also use the `LS` command to load symbols on the fly.

SYMLOAD is a memory-resident routine that is run once per DOS session. The `/Y` option removes the current copy from memory. SYMLOAD attaches itself to an interrupt vector so your program can access it as desired. The default interrupt used by SYMLOAD is 67H, but you can change this if you need to. SYMLOAD uses DOS calls to read a symbol file, so DOS must not be busy for SYMLOAD to work.

To install SYMLOAD, enter **`SYMLOAD /I:nn`**, where `nn` is the interrupt number you will use to access SYMLOAD. Be sure that Periscope is installed, since it is required for SYMLOAD to work. You must specify the `/I:nn` command-line entry only when you want SYMLOAD to use an interrupt other than 67H. If you do specify an interrupt, be sure that the interrupt is not already used by another program.

Once you've installed SYMLOAD, you may access it from your program by performing the appropriate interrupt. The registers used on entry are:

BX—The value of your program's PSP segment. If your program is an .EXE file, add 10H to the PSP segment. The symbols' segments will be relocated relative to the value passed in this register.

DS:DX—Points to a .PSS file name in ASCIIZ format with a required extension of .PSS. For example, to load C:SAMPLE.PSS, DS:DX would point to the string C:SAMPLE.PSS followed by a binary zero.

On return, register AH contains the status of the operation. The possible values of AH are:

0 - Successful symbol table load
1 - Error reading .PSS file (DOS error returned in AL)
2 - Periscope symbol table too small for .PSS file

3 - Logical error in .PSS file
4 - Periscope is not installed
5 - Logical error in symbol table

If SYMLOAD returns a status of zero, it has successfully loaded the symbol table. Note that it relocates all addresses relative to the segment address passed in register BX, except for symbols whose segment was already in the range of F000H to FFFFH.

SYMLOAD uses register AL to return additional error information if a read error occurred.

7.8 DEBUGGING DEVICE DRIVERS (SYSLOAD)

SYSLOAD allows you to load any of the .COM programs in the Periscope package at CONFIG.SYS time as a device driver. By being able to load Periscope as a device driver, you can greatly ease the debugging of your own device drivers.

To use SYSLOAD, place a line of the form **`device=sysload.sys arguments`** in the CONFIG.SYS file. The **`arguments`** field may be one or more of the following:

`/I` is an optional argument that generates an INT 3 to activate Periscope just before jumping to the specified program.

`/Q` is an optional argument that sets quiet mode and displays serious error messages only. This option is useful when loading a transient program.

`/P=c:filename.ext arg1 arg2 ...` is required and must appear last. It specifies the drive, path, and file name of the program to be loaded, followed by the arguments the program needs. Be sure to specify the full file extension although only .COM programs can be handled at present. Note that the part of DOS which loads device drivers also converts all characters to upper case. Thus case sensitive arguments cannot be passed to the program.

To load Periscope via SYSLOAD.SYS, use a line similar to

`device=sysload.sys /p=c:\peri\ps.com/a`
in CONFIG.SYS. Follow this line with the invocation of your device driver. Embed an INT 2 or an INT 3 in the strategy and/or interrupt entry points of the driver. When the driver executes, Periscope displays its screen. You can then use the `LS` command to load the driver's symbols and begin debugging the driver.

If you're using DR-DOS, use the DR-DOS INSTALL command instead of SYSLOAD. Make sure the PS*.COM and PM*.COM files are in the root directory of the boot disk.

7.9 GENERATING SYMBOL TABLES (TS)

The TS program verifies and sizes .MAP or other symbol files, and generates a Periscope symbol file with an extension of .PSS. To run TS, enter `TS progname` from the DOS prompt. Unless you specify options that indicate otherwise (see below), TS assumes a file extension of .MAP and assumes that the .MAP file is in the same format as the sample file FTOC.MAP. For help when running TS, enter `TS ?`. TS needs approximately 180K of work space plus the size of the .PSS file that it generates.

The options available are:

`/32` **(Periscope/32 only)** — Extract 32-bit symbol information from a .MAP or .EXE file.

If you're using MASM 5.10B or MASM 6.00, use a .MAP instead of an .EXE if possible. There are problems with local code and data variables generated by these two versions of MASM.

`/A xxxx` — Add a variable-length prefix to symbols.

`/B` — Read an .EXE file produced by Borland's TLINK.

Periscope supports local symbols for Borland products when compiled and linked with the appropriate options.

Periscope does not currently support typing, structures, or register variables. You may need to use the `-r-` option with Turbo C to force register variables off if you have problems accessing all local variables. Also, watch out for libraries with extended dictionaries. You may need to rebuild the library without the `/e` switch.

If you experience problems reading a Borland .EXE file, use the TDSTRIP utility to create a .TDS file, then run TS using that file. For example, to generate a .TDS file for FOO.EXE, enter **`TDSTRIP -S FOO`**, then enter **`TS FOO.TDS /B/V`** to read the symbols from the .TDS file.

`/C` — Read a DeSmet .MAP file produced by CWare's C compiler.

`/D` — Read a LINK86 .SYM file produced by Digital Research's LINK86.

`/E` — Read a CodeView .EXE file produced by Microsoft's LINK.

Local symbols are available for Microsoft products when compiled and linked with the appropriate options. You must run TS with the **`/E`** option to generate a .PSS file that includes local as well as public and line number symbols.

If you have problems generating symbols from an .EXE file, try loading the program under CodeView and then use the **`X`** command to display symbols. If you get an error, call Microsoft. If all else fails, try using the .MAP file. You won't have any local symbols, but at least you can use an editor to fix any problems.

`/Fx` — Filter leading character **`x`** from public symbols.

For example, Microsoft C uses leading underscores on public symbols. If you use **`/F_`**, TS will remove any leading underscores from the symbols.

A different filter function enables you to limit the size of your symbol table when you are only debugging a few modules in a large program. To use this filter function, create an ASCII text file with an extension of .PSF and one statement per line in the format **`X=NAME`**, where **`X`** is **`L`** for line number or **`P`** for public symbol. **`NAME`** is the name of the module or public symbol.

The match is made on as many characters as are entered as **NAME**, and the match is case-sensitive. For example, the line **P=A** would filter out all public, local data, or local code (but not stack-based) symbols starting with **A**, while the line **L=FTOC** would filter out all lines in the module **FTOC** or **FTOC????**, where **????** may be any characters.

If you have some symbols that you don't want included in the symbol table, edit the .MAP file and delete the desired symbols or insert braces as desired to turn off symbol generation. A left brace (**{**) turns symbol generation off, and a right brace (**}**) turns it back on. Put the braces in the beginning of existing lines. Do not add any new lines to the file! Be careful when saving the .MAP file. Don't let any TABs or high-bit characters into the file.

/LA — Accept line numbers that are out of sequence. Use this option for programs produced by Borland's C compiler and other optimized code.

/LD — Discard all line numbers that are out of sequence. Use this option to discard all lines for a module after a discontinuity, such as CodeView information for an .ASM program that uses includes.

/P — Read an .RTLink .MAP file produced by PocketSoft's .RTLink. This option reads public and line-number symbols from the .RTLink .MAP file. No local symbol support or VML support is available for .RTLink as of this writing.

/Q — Read a PLINK .EXE file produced by PLINK. This option reads public and line-number symbols from the PLINK .EXE file. No local symbol support is available for PLINK.

If TS finds static symbols with embedded brackets in a PLINK .EXE file, TS converts the brackets to underscores. For example, FOO[BAR.C] becomes FOO_BAR.C_, unless you use the **/S** option, in which case TS does not output the static symbols into the .PSS file.

/RP — Remove path from line-number records.

/RX — Remove file extension from line-number records.

The `/RP` and `/RX` options are intended for use with PLINK only, since it is inconsistent in the filenames it generates for line-number symbols. When using these options, be sure to use the **MP** and **MX** aliases to provide the needed path and extension.

`/S` — Filter static variables from a PLINK .EXE file.

`/V` — Verbose mode: show statistics on the number and type of symbols.

7.10 CUSTOMIZING PERISCOPE/EM (USEREXIT)

The USEREXIT program illustrates Periscope's ability to perform user-written code. Use user-written code to perform breakpoint tests (see the `BU` command) and user exits (see the `/U` command).

Install the user-written code as a memory-resident program using an available interrupt from 60H to FFH. You must install the program before you install Periscope. Also, you must specify the Periscope installation option `/I:nn`, where `nn` is the interrupt vector used to access the user-written code, and you must include a signature of `PS` in the resident routine in the word preceding the interrupt entry point.

The registers used on entry are:

AH — Contains the breakpoint test number of one to eight or the user exit number of nine to FFH.

AL — Always zero.

DS:SI — Points to Periscope's data area (see the file USEREXIT.ASM for the layout of the table). This table contains the values of various variables used by Periscope. Any changes to the variables in this table are passed back to Periscope.

ES:BX — Points to a user service routine in Periscope. Access this routine with a far call. Use register AH to indicate the function desired. When using this routine, all registers are preserved. The functions available are (see the sample program USEREXIT.ASM for details):

- Display nul-terminated string on the screen using Periscope's display handler
- Get command line using Periscope's keyboard handler
- Search symbol table using ASCIIZ string
- Search symbol table using an address

On return from a **user breakpoint**, register AL should be set to a binary one to indicate a hit. Any other value indicates that no breakpoint is to be taken.

On return from a **user exit**, register AL indicates whether the exit code has set a command to be executed by Periscope. If AL equals 2, Periscope reads the command line passed back from the user exit. The command line must start with a semicolon and end with a carriage return. A user exit may use BIOS functions as desired. Periscope assumes any screen output is done via the user service routine described above.

Do not attempt to perform DOS functions from user-written code as DOS may be busy! You do not need to preserve the values of any registers other than SS and SP on return to Periscope. If your routine needs more than 32 words of stack space, switch to an internal stack, but be sure to switch back to the original stack before returning.

To install USEREXIT, run the program from DOS. Then install Periscope using the installation option `/I:60`. When Periscope is active, try using `BU 1` and then `GT` to get to a point where DOS is not busy. Try `/U 9` as an example of a user exit modifying the command line. If you have a numeric co-processor, try `/U 87` to display the numeric processor status. To display the BASIC-style string `A$`, enter `/U D A$`. To display DOS's memory allocation, enter `/U 88`.

The user exit `/U A` sets a range of values for the CS register to be used by the user breakpoint `BU 2`. The range must be entered as a segment:offset pair, with the colon. To enable the user breakpoint, enter `BU 2` and then `GA`, `GM`, or `GT` as needed.

USEREXIT reflects these additional reserved user exits:

- **F0** — invoked after each command is entered, so that your code can modify the command line as needed.
- **F1** — invoked before a short boot (`QS`), so that your code can perform any required cleanup.
- **F2** — invoked before a normal boot (`QB`).
- **F3** — invoked before a long boot (`QL`).

To debug user exits, do the following:

- Load Periscope normally, without the `/I:nn` installation option.
- Load your user exit code.
- Enter `RUN/T PS /X/X/I:nn` to load a test copy of Periscope, adding any installation options you like at the end.
- Set a code breakpoint on the user exit using `BC {0:nn * 4}`, where `nn` is the interrupt number.
- Use `G` to get to the test copy of Periscope. Note that the prompt is an asterisk.
- Enter `/U nn` to activate your user exit. Execution will stop in the real Periscope at the start of the routine.
- Debug your code.
- When done, use `QC` at the asterisk prompt to terminate the test copy of Periscope.

Periscope/32 users: The USEREXIT program supports remote mode.

7.11 WHEN DOS IS BUSY (WAITING)

If you're debugging TSRs that do not hook INT 21, a simple program running in the foreground can keep DOS available so you can use source-level debugging while your program is running. The program WAITING.COM continuously calls the BIOS keyboard handler while looking for a **Ctrl-C**. When a **Ctrl-C** is pressed, control is returned to DOS.

Use RUN to load your program and set whatever breakpoints you desire. When the DOS prompt is displayed, run WAITING. Then when debugging your program, DOS won't be busy.

Reference

8

- **Optional Menus**
- **Command Summary**
- **Command Parameters**
- **Command Editor**
- **Function Keys**
- **Shortcut Keys and Keyboard Assignments**

This chapter introduces the optional menus you can use to issue commands, summarizes the commands, defines the command parameters, describes Periscope's command editor, and defines the shortcut keys and keyboard assignments. You'll find a detailed description of the commands in Chapter 9.

8.1 OPTIONAL MENUS

Periscope's menu system is an optional way that you as a new or occasional user can quickly and easily find and execute the appropriate Periscope command. Once you are an expert user, you can use it to find commands you infrequently use.

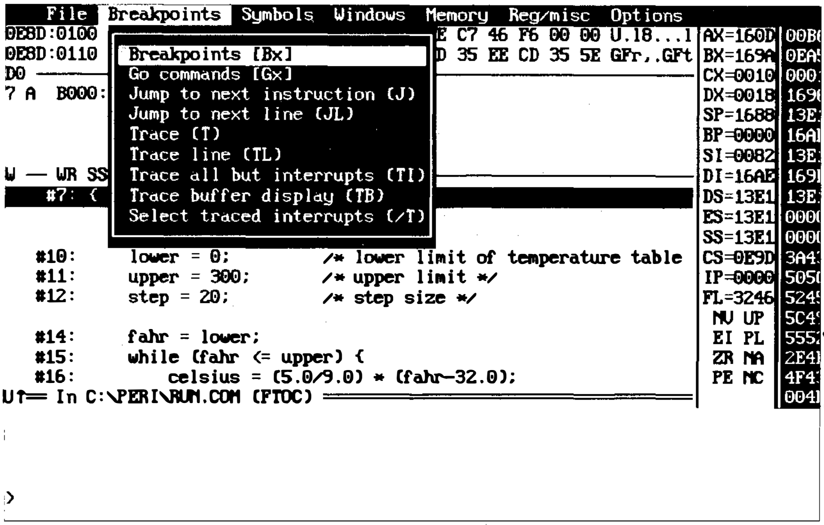
You can enable or disable the menu system when you run CONFIG. If you enable it, you'll be able to call it up any time you need it while you're debugging. It will use one line at the top of the display and load the menu system code and the on-line help and interrupt comments, all of which consume approximately 60KB of memory.

If you're running Periscope in conventional DOS memory, the additional 60K that the menu system requires may affect your ability to debug your programs. You can reduce the size required by renaming the file PSINT.DAT so that the interrupt comments won't be loaded. We do recommend that you leave the on-line help intact, but if you want to eliminate it, you can rename the file PSHELP.TXT as well.

When you enable the menu system, the top line of Periscope's display starts with the prompt `Alt-M: Menu` to remind you to press **Alt-M** to activate the menu system. The rest of this line usually displays a description of the function key assignments from the xxKEYS.PSD file, where xx is PS, CV, or TD for Periscope, CodeView, or Turbo Debugger key usage respectively. See Section 8.5 for more information.

When you activate the menu system by pressing **Alt-M**, you can then use the left and right arrow keys to select the desired top-level menu, or you can press **Alt-x**, where **x** is the first character of the menu name. It is not possible to go directly to a specific top-level menu before you press **Alt-M** since Periscope is already using some of the **Alt-key** combinations for other functions.

Within a menu, you may use the up arrow, down arrow, **Home**, and **End** keys or the character corresponding to the first letter of the option you want to select. (If more than one option starts with the same character, press that letter until the option you want is selected.)



```
    File  Breakpoints  Symbols  Windows  Memory  Reg/misc  Options
0E8D:0100  ┌──────────────────────────────────┐E C7 46 F6 00 00 U.18...1 AX=160D 00B
0E8D:0110  │ Breakpoints [Bx]                 │D 35 EE CD 35 5E GFr,.GFt BX=169A 0EA
D0 ─────── │ Go commands [Gx]                 │───────────────────────── CX=0010 000
7 A  B000: │ Jump to next instruction (J)     │                          DX=0018 169
           │ Jump to next line (JL)           │                          SP=1688 13E
           │ Trace (T)                        │                          BP=0000 16A
           │ Trace line (TL)                  │                          SI=0082 13E
W ── WR SS │ Trace all but interrupts (TI)    │───────────────────────── DI=16AE 169
   #7: {   │ Trace buffer display (TB)        │                          DS=13E1 13E
           │ Select traced interrupts (/T)    │                          ES=13E1 000
           └──────────────────────────────────┘                          SS=13E1 000
   #10:    lower = 0;          /* lower limit of temperature table   CS=0E9D 3A4
   #11:    upper = 300;        /* upper limit */                     IP=0000 505
   #12:    step = 20;          /* step size */                       FL=3246 524
                                                                      NU UP  5C4
   #14:    fahr = lower;                                              EI PL  555
   #15:    while (fahr <= upper) {                                    ZR NA  2E4
   #16:        celsius = (5.0/9.0) * (fahr-32.0);                     PE NC  4F4
U↑═ In C:\PERI\RUN.COM (FTOC) ══════════════════════════════════════         004

>
```

8-1. Periscope Breakpoints Menu

There are three basic types of options available from the top-level menus:

Options (commands) that require additional information. When you select one of these options, Periscope prompts you for the additional information required. All of the usual editing keys (except **Esc**) are available. After the command executes, Periscope redisplays the top-level menu.

Options (commands) that require no additional information. When you select one of these options, the command immediately executes, then Periscope redisplays the top-level menu. Some options are 'toggles' and have an up or down arrow in front of the option to indicate that they are turned on (up arrow) or off (down arrow).

Sub-menus. For these options, Periscope displays a sub-menu. Choose the appropriate entry and press **Enter**. Sub-menu options show brackets instead of parentheses around the Periscope command at the end of the line.

There are only two levels of menus — the top-level menus and one level of sub-menus. Throughout the menu system, you can use **Esc** to exit the current level. If you are at the top-level menu display, **Esc** returns to the Periscope prompt. If you are at a lower level in the menu system, **Esc** returns you to the top-level menu.

Periscope displays the command generated by the menu system, so that you can learn as you use Periscope. While in the menu system, you can get help on a command by pressing **F1** or by pressing **?** with the desired option selected. No help is available for sub-menus (where the command is in brackets instead of parentheses), but help is available for the items within the sub-menus.

The vast majority of Periscope commands are built into the menu system. For information on commands that are not, please see Chapter 9.

8.2 COMMAND SUMMARY

This summary is an introduction to the various types of commands available. The commands fall into fourteen groups, as follows:

1. Breakpoints. There are four categories of breakpoints: Code, Monitor, Hardware, and Debug register.

You set **Code breakpoints** on specific addresses in your program with the `BC` (Breakpoint on Code) command. When the instructions at the specified addresses execute, execution stops and the debugger displays its screen. These breakpoints are 'sticky' since Periscope "remembers" them until you explicitly clear them with the `BC` or `BA` commands. You can set temporary code breakpoints by entering an address after the `G` (Go) command. Periscope does not remember these breakpoints after the `G` command that set them executes. All of the Periscope Go commands activate Code breakpoints, which run at full speed except when you use `GA`, `GM`, or `GT`.

You set **monitor breakpoints** on a wide class of dynamically-evaluated conditions with these commands: `BB` (Byte), `BF` (Flag), `BI` (Interrupt), `BL` (Line), `BM`

(Memory), **BP** (Port), **BR** (Register), **BU** (User), **BW** (Word), and **BX** (eXit). They are all 'sticky', or remembered until you explicitly clear them. You activate them with the **GA**, **GM**, and **GT** commands only, which can cause the system to run much slower than full speed.

You set **hardware breakpoints** on the real-time CPU events that occur during the execution of your program with the **HB** (Hardware Bit), **HC** (Hardware Controls), **HD** (Hardware Data), **HM** (Hardware Memory), and **HP** (Hardware Port) commands. These commands are not covered in this manual. Please see the Model IV manual for details.

Debug register breakpoints are available on 80386 and higher machines. You set these breakpoints on reads, writes, and executes of memory with the **BD** (Breakpoint on 80386 Debug Registers) command. They are activated by all of the Go commands, and run at full speed except when you use the **GA**, **GM**, or **GT** commands.

Table 8-1 summarizes program execution speeds for the various types of breakpoints and the various Go commands.

The breakpoint command terms set, clear, enable, and disable may be confusing. Here's what they mean. When you **set** any 'sticky' breakpoint, it will be in effect each time you use the appropriate Go command. You can keep it from being in effect by either disabling it or clearing it. If you **clear** it, Periscope no longer knows about it. If you **disable** it, Periscope knows it's there, but does not use it until you **enable** it. Periscope displays disabled breakpoints with a leading minus sign.

To clear all breakpoints, enter **BA *** (for software breakpoints) or **HA *** (for hardware breakpoints). To clear an entire group of breakpoints, enter **Bx *** or **Hx ***, where **x** indicates the group you want to clear, such as Byte, Memory, Port, Word, etc. To clear an individual breakpoint, re-enter the breakpoint (except for the **HB** and **HC** breakpoints). Periscope's program loader, RUN, always disables all hardware and software breakpoints, except for the **HC** breakpoints. A useful habit to develop is to display all breakpoints with **BA** and **HA** before executing the Go command, so that you know for sure what breakpoints are set and enabled.

GO COMMAND	BREAKPOINT TYPES	SPEED
G	C, D	Full speed
G+	C, D	Full speed
G=	C, D	Full speed
GA	C, D, M	Very slow
GH	C, D, H	Full speed
GM	C, D, M, H	May be slow
GR	C, D	Full speed
GT	C, D, M	Very slow
BREAKPOINT TYPES: C=CODE; D=DEBUG REGISTER; M=MONITOR; H=HARDWARE		

Table 8-1. Execution Speeds of Various Breakpoints

2. Controls: Use the clear screen commands (**K** and **KI**) to clear Periscope's screen. Use the Color command (**/C**) to set Periscope's screen color. Use the Window command (**/W**) to set Periscope's windows. Use the Data window select command (**/D**) to select the active Data window when you're using multiple Data windows. Use the Quiet command (**/Q**) to temporarily suppress Periscope's screen output.

3. Disassembly: The Unassemble commands (**UA**, **UB**, and **US**) disassemble memory in Assembly-only, Both-source-and-assembly, and Source-only modes respectively. You can use 16-bit (**16**) or 32-bit (**32**) disassembly as needed.

4. Disk I/O: These commands include Load Absolute (**LA**) and Load File (**LF**), the corresponding Write commands (**WA** and **WF**), and the Name command (**N**).

5. Display: The Display commands (**Dx**) display memory in various formats, including ASCII, byte, double word, effective address, integer, long real, signed integer, record, short real, word, long integer, signed long integer, and ASCIIz. Use the Watch commands (**W**) to display memory in any of the above formats in a Watch window.

6. Execution: These commands include the Trace and Trace Line commands (**T** and **TL**), the Jump and Jump Line commands (**J** and **JL**), and the Go commands (**Gx**) that execute your program under various conditions. To trace all but interrupts, use the (`TI`) command. Use the Breakpoint commands (`Bx` and `Hx`) to set code, monitor and hardware breakpoints. See the discussion of Periscope's breakpoint commands earlier in this chapter.

7. Hardware: Use the Hardware commands (`Hx`) to set hardware breakpoints and controls and to display the hardware trace buffer **when you have Model IV hardware installed**. See the separate Model IV manual for details.

8. I/O: Use the Input (`I` and `IW`) and Output (`O` and `OW`) commands to read and write I/O ports, respectively.

9. Modify: Use various commands to modify memory, including the Enter commands (`E`, `EB`, `ED`, and `EW`), the Fill command (`F`), and the Move command (`M`).

10. Search: The Search commands search memory in a variety of ways, including two data searches (`S` and `SD`), an address search (`SA`), a disassembly search (`SU`), and two stack searches, one for calls (`SC`) and one for returns (`SR`).

11. Status: The status commands include the Register command (`Rx`) which you use to display, change, save, compare, and restore the machine registers and flags. View the software trace buffer with the Traceback commands (`TB`, `TR`, and `TU`). You can save, compare, and restore the interrupt vectors using the `IR`, `IC`, and `IS` commands. To save the state of a debug session, use the `WB` command. To later restore the state, use the `LB` command.

12. Symbols: Load and write the symbol table using the Load Symbols (`LS`) and Write Symbols (`WS`) commands. The Load Definitions (`LD`) and Write Definitions (`WD`) commands load and write the record definition table. Use the Enter Symbol command (`ES`) to add symbols on the fly. Use the Near command (`/N`) to search for the nearest symbols to a specified address. Use the Remove symbol command (`/R`) to delete a symbol from the symbol table. Use the Segment change command (`/S`) to modify the segment for a group of symbols.

13. Periscope/32-specific Commands. Please see Appendix

C.

14. Miscellaneous. Other commands include an in-line Assembler (`A` and `AU`), a Compare memory command (`C`), an Enter Alias command (`EA`), Hex arithmetic (`H`), Quit (`Qx`), View file (`V` and `VS`), and translate addresses and numbers (`XA`, `XD`, and `XH`).

Also available are **option** (/) commands, including the Echo command (`/E`) which echoes Periscope's screen output to a disk file. Use the User exit command (`/U`) to execute user-written code from within Periscope. Use the eXit command (`/X`) to exit to DOS.

The Key capture command (`/K`) captures keyboard input to a file. The `/1` and `/2` commands allow you to switch to the first and second symbol tables, respectively. The DOS Access command (`/A`) turns DOS access on and off. To copy Periscope's screen to the other display, use the dupe (`/"`) command. To toggle the display of line-number symbols and source-level debugging status, use the `/L` command.

8.3 COMMAND PARAMETERS

You may enter Periscope commands in upper or lower case. Use either a space or a comma to delimit parameters within a command. A delimiter is required when the command is only one character in length, after a symbol, and between two numbers.

Periscope/32 users: You cannot use a comma as a general delimiter. See Appendix C.

Each command requires at least a single-character mnemonic. All but a few commands require additional input.

Periscope's command parameters are listed below in alphabetical order. Square brackets (`[ ]`) in the command syntax indicate an optional entry. (Brackets actually entered in a command line indicate that the address is to be used as a near pointer.) An ellipsis (`...`) indicates a repetitive entry.

`*` — An asterisk in the first position of a Periscope command line causes the entire line to be treated as a comment.

? — Indicates a variable.

| — Indicates a logical OR.

Periscope/32 users: Use a stile (|) to separate the segment and offset in an address when you want Periscope/32 to use real-mode style address conversion (segment times 16 plus offset). See Appendix C.

$ — Use the dollar sign or 'here' indicator with the Display commands to replace the display address and more easily display some types of data. It assumes a value equal to one more than the last byte previously displayed. For example, if you want to page through memory displaying 200H bytes at a time, you can use `D $ L200` rather than having to specify an address each time. Similarly, you can use the `DR` command to display repeating record definitions. For example, use `DR $ RECORD` to display a repeating fixed-length record.

[ ] — Brackets around an address indicates that the offset is a near pointer to another offset within the specified segment. The trailing bracket is optional. For example, if the word at CS:250H contains 1234H, `U [CS:250]` disassembles memory starting at CS:1234.

Periscope/32 users: Periscope/32 also supports double brackets ([[ ]]) to indicate a near 32-bit pointer.

{ } — Braces around an address indicates that the segment and offset are a far pointer to another segment and offset pair. The trailing brace is optional. For example, to disassemble INT 10H, enter `U {0:10*4}`. This command uses the offset at 0:40H and the segment at 0:42H, which is interrupt vector 10H.

Periscope/32 users: Periscope/32 also supports double braces ({{ }}) to indicate a far 32-bit pointer.

O. — A prefix of `O.` before a symbol name extracts the offset portion of a symbol name. For example, to set AX to the offset portion of the symbol NEWPAGE, use `R AX O.NEWPAGE`.

S. — A prefix of **S.** before a symbol name extracts the segment portion of a symbol name. For example, assume the symbol ARRAY points to 1234:5678. To display memory at 1234:0000, you could use **D S.ARRAY:0**.

T. — A prefix of **T.** indicates that a decimal number follows. For example, to display memory at offset 2000 decimal, enter **D T.2000**.

W. — A prefix of **W.** extracts the word at the target address and uses it as the argument. For example, to display memory at offset 1000H in the segment referenced by the symbol DOSSEG, enter **D W.DOSSEG:1000**.

+, -, *, / — These arithmetic operators perform inline arithmetic. They are evaluated from left to right. For example, **dd 0:21*4**, **r ip ip+1**, **d ss:sp-4**, and **u bx+si-5** are all valid.

<address> — The address of a memory location, composed of a segment and an offset separated by a colon. Alternately, you can use registers for either or both numbers, or you can use a valid symbol for both the segment and offset. For some commands, you may omit the segment. Possible addresses include **1000:1234**, **DS:SI**, and **PRINTLINE**.

> **Periscope/32 users:** Periscope/32 assumes that addresses are in selector:offset format rather than segment:offset format. In most cases the selector and segment are interchangeable. See Appendix C for details.

<alias> — An alias is a two-character shorthand notation that can have an associated character string of up to 64 characters. There are currently 38 alias names defined and eight aliases reserved for use with Periscope. (You may define others.) You can enter aliases in a .DEF file or with the **EA** command.

See Section 7.5 and the **EA** command in Chapter 9 for details on aliases.

<arithmetic operator> — The arithmetic symbols +, -, *, and /, for addition, subtraction, multiplication, and division, respectively.

<byte> — A one- or two-digit hexadecimal number from 0 to **FF** or an 8-bit register.

<command> — A Periscope command, such as **US** (unassemble source) or **D** (display in current format).

<decimal number> — A decimal number from 0 to 65535. No punctuation is allowed.

<drive> — A single-digit number corresponding to a disk drive, where 0 equals drive A, **1** equals drive B, etc. Also you may use **A:**, **B:**, etc.

<file> — A file name, including drive, path, and extension as needed.

<flag> — A flag register. The possible values and two-character mnemonics are shown in Table 8-2.

Periscope/32 users: Please see the note at the end of the Register command in Chapter 9 for additional flag mnemonics.

FLAG	SET (=1)	CLEAR (=0)
Overflow	OV	NV
Direction	DN (STD)	UP (CLD)
Interrupt	EI (STI)	DI (CLI)
Sign	NG (negative)	PL (positive)
Zero	ZR (zero)	NZ (non-zero)
Auxiliary carry	AC	NA
Parity	PE (even)	PO (odd)

Table 8-2. Flag Register Values/Mnemonics

<format> — The formats available with the display and watch commands: **A** (ASCII), **B** (byte), D (double word), **E** (effective address), **I** (integer), **L** (long real), **N** (number), **R** (record), **S** (short real), **W** (word), **X** (long integer), **Y** (long signed integer), and **Z** (ASCIIz). See the Display (**D**) commands in Chapter 9 for details.

<length> — The number of bytes affected by a command. This may be represented by `L nnnn` where `nnnn` is a hexadecimal number from `1` to `FFFF`. It may also be represented by a number following an address. In this case the length is calculated as the number plus one minus the offset. For example, `D CS:100 L 100` and `D CS:100 1FF` (1FF plus 1 minus 100) both have a length of 100H.

You may substitute a register name for the number in either format. Periscope uses the current value of the register for the number.

You may also use a symbol for the length argument. The segment associated with the symbol must be the same as the segment referenced in the preceding address and the offset must not be less than the offset referenced in the address.

Periscope/32 users: Periscope/32 supports 32-bit lengths.

`<list>` — A list of byte(s) and/or string(s). For example `03 'COMMAND COM' 12 34` is a list composed of a byte, a string, and two trailing bytes.

`<name>` — A one- to 64-character name used as an alias.

`<number>` — A one- to four-digit hexadecimal number from `0` to `FFFF`. If you use a register name, Periscope substitutes its current value for the number.

Periscope/32 users: Periscope/32 supports 32-bit hex numbers.

`<offset>` — The one- to four-digit hexadecimal number or register representing the offset into the specified segment.

Periscope/32 users: Periscope/32 supports 32-bit offsets.

`<pointer>` — A bracket (`[`) or brace (`{`) used to indicate a near or far pointer respectively.

Periscope/32 users: Periscope/32 also supports double brackets (`[[`) and double braces (`{{`) to indicate near and far 32-bit pointers.

`<port>` — The one- to four-digit hexadecimal number associated with an I/O port.

`<range>` — An address and a length. For example `CS:100 L 100` and `0:0 FF` are ranges. You can use two symbols if they both reference the same segment and if the offset of the second symbol is greater than or equal to the offset of the first symbol.

`<register>` — A machine register. The 16-bit registers are AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, FS, GS, SS, CS, and IP. The 8-bit registers are AH, AL, BH, BL, CH, CL, DH, and DL.

Periscope/32 users: You can also use the 32-bit registers EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI, and EIP.

`<sectors>` — Two hexadecimal numbers representing the starting relative sector number and the total number of sectors (max 80H). The sector numbering scheme is the one used by DOS interrupts 25H and 26H.

`<segment>` — A one- to four-digit hexadecimal number or register representing one of the six segment registers (CS, DS, ES, FS, GS, SS).

Periscope/32 users: Substitute selector for segment in remote mode.

`<string>` — A quoted list of ASCII characters. You may use either single or double quotes to delimit the string. To enter a string containing an embedded quote, use the other form of quote to delimit the string, or enter two quotes where you want a single embedded quote.

`<symbol>` — A name corresponding to an address or a record definition. RUN loads symbols from a .PSS file when it loads the corresponding program. You may optionally precede a symbol name with a period, which forces Periscope to treat the name as a symbol. For example, to disassemble memory starting at the symbol PRINTLINE, enter `U .PRINTLINE`.

Periscope evaluates symbols first, before numbers and registers. This can cause conflicts and/or confusion if a symbol has the same name as a valid register or number (e.g. AX or A123). Be careful not to confuse symbol names with addresses. When Periscope processes the command `D A123`, it first tries to use A123 as a symbol name. If it doesn't find a symbol with that name, it tries to use it as a hex number. To inhibit its use as a hex number, precede the address with a period, `D .A123`.

Use a tilde (~) to force the name not to be treated as a symbol. For example, to display memory at address F00 when a symbol of the same name exists, enter `D ~F00`.

Use symbols also to reference record definitions read from a .DEF file. For example, to display the FCB, enter `DR CS:5C FCB`.

`<test>` — Compare two values. The possible tests are `LT` (less than), `LE` (less than or equal), `EQ` (equal), `NE` (not equal), `GE` (greater than or equal), and `GT` (greater than).

8.4 COMMAND EDITOR

Periscope has a CED-like command editor. It stores previously-entered commands in a 512-byte circular buffer. It saves all lines greater than the minimum length set in CONFIG, except when **Alt-C** or **Alt-D** is used to repeat a previous line. It supports the following editing keys:

Backspace — Delete the character to the left of the cursor and back up one character.
Ctrl-End — Erase the remainder of the command line, starting at the current cursor position.
Ctrl-Left — Move to the start of the previous word in the command line.
Ctrl-PgDn — Remove the current command line from the circular edit buffer.
Ctrl-PgUp — Clear the entire circular edit buffer.
Ctrl-Right — Move to the start of the next word in the command line.
Del — Delete a character from the command line.
Down Arrow — Display the next command line from the circular buffer.

End — Move the cursor to the end of the command line.
Esc — Erase the current command line.
Home — Move the cursor to the start of the command line.
Ins — Toggle insert mode on and off.
Left Arrow — Move the cursor one position to the left.
Right Arrow — Move the cursor one position to the right.
Tab — Space.
Up Arrow — Display the previous command line from the circular buffer.

8.5 FUNCTION KEYS

When you configure Periscope (see Chapter 3), you can assign function keys **F1** through **F10** to make Periscope's key usage like that of Borland's Turbo Debugger, or like that of Microsoft's CodeView, or assume Periscope's defaults. The default function keys and their **Alt** key equivalents are:

KEY	ALT KEY	DESCRIPTION
F1	**Alt-T**	Code timing toggle
F2	**Alt-N**	Screen swap toggle
F3	**Alt-C**	Copy previous line
F4	**Alt-D**	Copy previous line plus return
F5	**Alt-A**	Display alias definitions
F6	**Alt-G**	Set pause mode
F7	**Alt-E**	Display record definitions
F8	**Alt-I**	Display symbols
F9	**Alt-H**	Call trace
F10	**Alt-O**	Display user screen

Since the Periscope defaults duplicate **Alt** key assignments, you may want to assign **F1** through **F10** in other ways, using aliases **F1** - **F9**, and **F0**. If you define **F0** through **F9** via aliases, the alias definitions will override the default function key assignments. Please see the description of RS in Section 7.5 and the `EA` command in Chapter 9 for more information on aliases.

8.6 SHORTCUT KEYS AND KEYBOARD ASSIGNMENTS

^F1-^F9, and ^F0 — Use to execute F1 through F10.

!CA-!CZ — Use to execute Ctrl-A through Ctrl-Z.

!AA-!AZ — Use to execute Alt-A through Alt-Z.

!A1-!A9 — Use to execute Alt-1 through Alt-9.

Alt-3 — Toggle the vertical 80386 register display on and off.

You must use at least one window for this key sequence to have any effect.

Alt-A— Display the alias definitions.

Place the cursor at the beginning of the command line to display all aliases. To display alias names that start with certain characters, enter the characters, then press **Alt-A**. For example, to display all alias names starting with the letter C, type C at the start of a command line, then press **Alt-A**. Be sure not to enter any spaces before or after the characters.

Alt-B — Toggle printer double spacing on and off when Ctrl-P is used.

This has no effect when you use **Shift-PrtSc**.

Alt-C— Copy the remainder of the previous command line into the current command line.

This copies up to, but does not include the carriage return. The command line is not added to the circular edit buffer again.

Alt-D — Copy the remainder of the previous command line into the current command line and add a carriage return at the end.

For repetitive commands, you can use **Alt-D**.

Alt-E — Display the record definitions.

Place the cursor at the beginning of the command line to display all record definitions. To display record definitions that start with certain characters, enter the characters, then press **Alt-E**. For example, to display all record definitions starting with PS, enter `PS` at the start of the command line and press **Alt-E**. Be sure not to enter any spaces before or after the characters.

Alt-G — Select one of three pause modes: Pause on; Pause/clear on; and Pause off.

The `Pause on` mode displays a message when the screen is full, then waits for you to press a key before it scrolls another screen full of information into view. `Pause/clear on` differs from the `Pause on` mode in that it clears the screen after you press a key, then displays data from the top of the screen. This technique allows for much quicker updating of the second and subsequent screens, but loses the prior screen as soon as you press a key. The `Pause off` mode continually scrolls the non-windowed area of the screen. Periscope defaults to `Pause on`. It ignores the pause mode when the `/E` (echo) mode or **Ctrl-PrtSc** is active.

Alt-H — Toggle call tracing on and off.

When you enter this mode, Periscope displays the message `Call trace on`. Then when you enter a `GA` or a `GT` command, Periscope displays any CALL instructions executed. It shows nested calls by leading spaces before the address, up to a maximum of 16 levels deep. For example, part of the call trace of the FTOC program (using `GT`) is:

```
170D:0016 E8BD02          CALL   __CHKSTK
 170D:0072 E8A905          CALL   __PRINTF
  170D:0630 E8BD01          CALL   __STBUF
  170D:0640 E85103          CALL   __OUTPUT
   170D:099A E839F9          CALL   __CHKSTK
```

This mode works best on a dual-monitor system. When a procedure label or source code is shown along with the disassembled call, the label or source code is correctly indented, but the associated call is not. When you turn this mode off, Periscope displays the message `Call trace off`.

Alt-I — Display the address and name of the symbol table entries.

Place the cursor at the beginning of the command line to display all symbols. To display symbols that start with certain characters, enter the characters, then press **Alt-I**. For example, to display all symbols starting with the letter A, type **A** at the start of a command line, then press **Alt-I**. Be sure not to enter any spaces before or after the characters.

When Periscope displays local code or local data symbols, it places an asterisk immediately after the address. When it displays a symbol that is not currently loaded (using PLINK or .RTLink), it places it in parentheses. When you enter a line number, enter the full name.

To display all symbols in a segment, enter `/<number>`, where `<number>` is the segment, then press **Alt-I**. To display all symbols at a specific address, enter `/<address>`, where `<address>` may be any of the legal address formats, then press **Alt-I**. To display stack data variables, enter `//[<name>]`, where the optional `<name>` is the name of the procedure, then press **Alt-I**. Stack data variables are only displayed when you use `//`.

Alt-L — Toggle the display of commands generated by the menu system.

Alt-M — Activate Periscope's menu system (assumes you enabled it when you ran CONFIG).

Alt-N — Toggle screen swap on and off when using Periscope on a single monitor.

This has no effect when you use the `/A`, `/AD`, or `/AV` installation options. When off, this mode keeps Periscope from displaying the program's screen when you use a Jump or Trace command. If you're tracing code that doesn't modify the display, you may want to turn screen swap off to avoid the 'flash' caused by the restoration of the program's screen during each Jump or Trace command. Be sure to turn screen swap back on before executing code that will cause the program under test to output to the screen. If screen swap is off when a Go or Quit command is used, Periscope swaps the screen anyway.

Alt-O — Switch from Periscope's screen to the program's screen when using only one monitor.

If you used the `/A`, `/AD`, or `/AV` installation option or if screen swap is off, this has no effect. To return to Periscope's screen from the program's screen, press any key.

Alt-P — Generate a form feed to the parallel printer.

Alt-R — Toggle the vertical register display on and off.

You must use at least one window for this to have any effect.

Alt-S — Toggle the vertical windowed stack display on and off.

You must use at least one window for this to have any effect.

Alt-T — Toggle code timing on and off.

When you enter this mode, Periscope displays the message `Code timing on`. Periscope times your code's execution in increments of 838 nanoseconds (the standard 55 millisecond timer-tick divided by 64K), and displays the timed value as a decimal number. The maximum event length you can time is 655,359,999 times 838 nanoseconds, or approximately nine minutes. Periscope uses I/O ports 40H and 43H.

This method is very accurate, except for very short duration events of less than 20 to 30 ticks. If Periscope displays a time of less than 20 to 30 ticks or displays `N/A`, run the stand-alone timer test program (available in the UTILS SIG of our BBS) or time the code multiple times and take an average.

To turn code timing off, press **Alt-T**. Periscope displays the message `Code timing off`. Note that your code runs at full speed when code timing is on. Use of the `GA` or `GT` commands turns the code timing off. Code timing can conflict with the `/K:1` installation option, since this option turns the system timer off while you're in Periscope.

Alt-W — Toggle the current window from the Data window to the Watch window to the Disassembly window.

The current window has an up arrow in the separator line following the window and a double separator line.

Ctrl-B — Set a code breakpoint on the instruction at the top of the Disassembly window.

If a breakpoint is already set, clear it. When using overlays, be sure to set breakpoints at a symbol or source line.

Ctrl-Break — Cancel the current Periscope command.

Ctrl-E — Move the source disassembly display to the end of the source file.

If the source file is too big for its buffer, **Ctrl-E** will not go all the way to the end of the source file.

Ctrl-G — Go to the address shown at the top of the Disassembly window.

Ctrl-PrtSc or Ctrl-P — Toggle printer echo of screen output on and off.

Any control codes or special characters other than carriage return and line feed are suppressed. Only the non-windowed area of the screen is printed.

Ctrl-R — Set the current instruction (CS:IP) to the instruction at the top of the Disassembly window.

Ctrl-S — Suspend output until another key is pressed.

Ctrl-T — Move the source disassembly display to the beginning of the source file.

PadMinus (gray minus key on numeric pad) — Move backward one line in the current window.

To enter a minus sign, use the other minus key.

PadPlus (gray plus key on numeric pad) — Move forward one line in the current window.

To enter a plus sign, use the other plus key.

PgDn — Page forward through the current window.

PgUp — Page backward through the current window.

8

Semi-colon — A pseudo carriage return.

Use the semi-colon to enter multiple commands on one line. For example, if you're tracing through a program that requires repetitive Go and Enter commands, you could enter `G NEWPAGE;EW PAGENO 0` to go to the procedure NEWPAGE and modify memory starting at PAGENO. After you enter the line once, you can use **Alt-D** to repeat it.

Shift-PrtSc — Print the entire screen to the parallel printer.

Be careful if control codes are displayed on the screen with the Display or Xlate commands. Use **Ctrl-PrtSc** to avoid output of control codes to the printer.

Command Reference

■ Commands

This chapter describes in detail all Periscope (/EM and /32) commands, except those that are only available when you have the Model IV hardware installed. Please see the Model IV manual for details on Model IV-specific commands. See Chapter 8 for a summary of all commands and other related information, and Appendix C for a summary of the differences between Periscope/EM and Periscope/32.

9.1 COMMANDS

Command: help (?)

Syntax: `? [<command>]`

Description: Use this command to display help on Periscope's commands. If the help file is not available, it displays a command summary.

Examples:

`?` displays a command summary.

`? DD` displays help for the Display Double word command when the on-line help file is loaded.

Command: Assemble to memory (A)

Syntax: `A [<address>]`

Description: Use this command to assemble instructions to memory.

To use the in-line assembler, enter `A [<address>]` at the Periscope prompt. Periscope displays the specified `<address>`, or if you do not enter an address, it displays CS:IP. Enter the instructions to be assembled and press return. To terminate the assembly, press return when the cursor is at the beginning of a new line.

The assembler supports all of the real-mode opcodes up through the 80286 and 80287. It does not support the protect mode opcodes of the 80286, 80386, and later. If you use the LOCK prefix, place it on a separate line preceding the instruction it affects.

Periscope supports various forms of the opcodes, including synonyms such as JE and JZ, etc. There are two special cases: String primitives such as MOVS must explicitly reference a byte or word (MOVSB or MOVSW), and you must enter a far return as RETF.

Jump or call instructions generate the shortest form of call for the address specified. When referencing memory, be sure to use brackets around the address field to differentiate it from a direct reference.

When using symbols, you may precede the symbol name with a period. If you are referencing the contents of a symbol, be sure to put the symbol name in brackets (e.g., `MOV AX, [PAGENO]`). To get the offset of a symbol into a register, do not use the brackets (e.g., `MOV AX,PAGENO`). You may also use symbols as arguments to JMPs and CALLs.

Periscope also supports the DB pseudo-op, allowing you to enter hex or ASCII characters into memory.

Example:

To assemble an instruction at 1234:5678 to jump to the symbol NEWPAGE, enter `A 1234:5678` and press return. Then enter `JMP NEWPAGE` and press return. Press return again to exit the in-line assembler.

Command: Assemble then Unassemble (AU)

Syntax: `AU [<address>]`

Description: Use this command to assemble then disassemble an instruction.

Example:

To assemble an instruction at CS:IP to move the value of the symbol TOTMEM to register AX, enter `AU` and press return. Then enter `MOV AX, [TOTMEM]` and press **Enter**. Periscope disassembles the instruction and displays the next prompt. Press **Enter** again to exit.

Command: software Breakpoints All (BA)

Syntax: `BA [?] [*] [+] [-] [A]`

Description: Use this command to display (?), clear (*), enable (+), and/or disable (-) all currently-set breakpoints, and/or to toggle (A) the monitor breakpoint AND mode for the currently-set software breakpoints. If you enter **BA** only, it displays all currently-set breakpoints. It displays disabled breakpoints with a leading minus sign.

The AND mode ANDs the results of monitor breakpoints in different classes. Multiple breakpoints within a class (i.e. multiple register breakpoints) are still ORed together, but the result of one class is ANDed with the results of other classes.

Each class of currently-set monitor breakpoints must indicate a 'hit' for a breakpoint to be generated when the AND mode is on. For example, if the **BB**, **BI**, and **BM** breakpoints are all set and enabled, all must be true for a breakpoint to be generated when AND mode is on. If multiple breakpoints are set within a class, such as two register breakpoints, only one true condition within the class is needed to make the entire class true.

Since RUN disables all breakpoints, use this command to re-enable previously-set breakpoints after using RUN.

Examples:

Assume that a Byte breakpoint is set for the symbol LINECOUNT equal to 38H, and that a Register breakpoint is set for CX less than 5. No other breakpoints are set.

BA or **BA ?** displays both breakpoints:
BB LINECOUNT EQ 38 and **BR CX LT 0005**.

BA * clears both breakpoints.

BA + enables both breakpoints. (It enables Line and eXit breakpoints only if you have previously set them with **BL +** or **BX +** and then disabled them.)

BA - disables both breakpoints. (It disables Line and eXit breakpoints only if you have previously set them with **BL**

+ or `BX +`.)

Assume you want to watch for a RETurn where SP is greater than or equal to the current value:

`BA * A; BX +; BR SP GE SP` clears all breakpoints, sets AND mode on, sets the exit breakpoint on, and sets a breakpoint when SP is greater than or equal to its current value.

Command: Breakpoint on Byte (BB)

Syntax: `BB [<address> <test> <byte>] [?] [*] [+] [-] [...]`

Description: Use this command to set a monitor breakpoint when a byte of memory meets a test.

You may set up to eight byte breakpoints at one time. Multiple breakpoints may be set on a single input line, along with the breakpoint clear (`*`), display (`?`), enable (`+`), and disable (`-`) functions. If any of the tests pass, a breakpoint is taken.

If you do not specify a segment in the address, Periscope uses the current data segment. This breakpoint stops execution of a program on the instruction following the instruction that changed the specified byte of memory.

Periscope remembers these breakpoints until you clear them. Re-enter a previously set breakpoint to clear it and display the message `Breakpoint cleared`. Be careful to display all breakpoints before using the Go command to make sure the breakpoints you've got are the ones you want.

Examples:

`BB LINECOUNT EQ 38` sets a Byte breakpoint for the memory location corresponding to LINECOUNT.

`BB * DS:123 GT F0 ?` clears all Byte breakpoints, sets one, and then displays the Byte breakpoint.

`BB` or `BB ?` displays all Byte breakpoints.

Command: Breakpoint on Code (BC)

Syntax: `BC [<address>] [?] [*] [+] [-] [!<number>] [...]`

Description: Use this command to set a code breakpoint when an instruction executes. You can specify a software pass count by entering a number from `1` to `FFFF`H after the exclamation mark. This command performs the same function as addresses you enter with the Go command, except that these breakpoints are "sticky" whereas addresses you enter with the Go command are temporary.

If you do not specify a segment in the address, this command presumes CS. You can use any form of the Go command to activate Code breakpoints, but only the `GM` command activates the software pass counter.

You can enter multiple breakpoints on a single input line along with the breakpoint clear (`*`), display (`?`), enable (`+`), and disable (`-`) functions.

To clear a previously set breakpoint and display the message `Breakpoint cleared`, re-enter it. Be careful to display all breakpoints before using the Go command to make sure the breakpoints you've got are the ones you want.

When you set a code breakpoint, Periscope highlights the instruction in the Disassembly window unless it is the current line, in which case it places an arrow at the start of the line.

Examples:

`BC PRINTLINE` sets a Code breakpoint for the memory location corresponding to PRINTLINE.

`BC * CS:123 ?` clears all Code breakpoints, sets one, and then displays the Code breakpoint.

`BC {0:21*4` sets a Code breakpoint at the entry point for Interrupt 21H.

`BC` or `BC ?` displays all Code breakpoints.

Command: Breakpoint on 80386 Debug Registers (BD)

Syntax: `BD [<address>] [L<byte> R|W|X] [?] [*] [+] [-] [...]`

Description: Use this command to take advantage of the 80386 debug registers. You can set up to four limited real-time hardware breakpoints on instruction execution, memory writes, or memory access (read or write).

In the syntax, `<byte>` is `1`, `2`, or `4`, representing a byte, word, or double word respectively. The `R`, `W`, and `X` indicate read/write, write, and execute respectively. The segment portion of the `<address>` defaults to `DS`. Be sure to override it as needed.

Be sure to set execution breakpoints on the first byte of the instruction, including any prefixes. For read/write breakpoints, proper alignment is necessary if you want to avoid missed breakpoints. If the length is `2`, the address should be on a word boundary. Similarly, if the length is `4`, the address should be on a double word boundary.

To use this command, you must have an 80386 or later CPU and you must have indicated that you want to use the debug registers when you configured Periscope (see Chapter 3).

Do not reboot the system while a `BD` breakpoint is in effect, since it can hang the system. If the breakpoint is hit during the boot process, the system tries to access Periscope, which is no longer resident. In general, you should clear or disable `BD` breakpoints before your program terminates.

Examples:

`BD F000:FEED` sets a breakpoint on instruction execution in ROM.

`BD 0:46C L2 W` sets a breakpoint on a write to the system clock.

Command: Breakpoint on Flag (BF)

Syntax: `BF [<flag>] [?] [*] [+] [-]`

Description: Use this command to set a monitor breakpoint on a single `<flag>` register value. For example, to check for unsuccessful DOS calls where the carry flag is set, set a code breakpoint at the return point, enter `BF CY` to set a flag breakpoint, and enter `GM` to begin execution.

You can set only one flag breakpoint at any time, along with the breakpoint clear (*), display (?), enable (+), and disable (-) functions.

Periscope remembers these breakpoints until you clear them. Be careful to display all breakpoints before using the Go command to make sure the breakpoints you've got are the ones you want.

Periscope/32 users: Periscope/32 does not support breakpoints on the IOPL status, although it supports the other 32-bit flags.

Examples:

`BF OV` sets a breakpoint when the overflow flag is on.

`BF -` disables the flag breakpoint.

Command: Breakpoint on Interrupt (BI)

Syntax: `BI [<byte>] [?] [*] [+] [-] [#] [...]`

Description: Use this command to set a monitor breakpoint when a software interrupt executes. You'll get to the next occurrence of any software interrupts from 0 to FFH.

Enter the interrupt number as a `<byte>`. You can enter multiple interrupt numbers on a single line, along with the breakpoint clear (*), display (?), enable (+), and disable (-) functions. To set breakpoints on all interrupts, enter `BI #`.

If the interrupt of interest is in RAM, don't use this command. Use a Go command instead to get to the interrupt at full speed. For example, to get to Int 21H, enter `G {0:21*4}`.

Periscope remembers these breakpoints until you clear them. Re-enter a breakpoint to clear it and display the message `Breakpoint cleared`. Be careful to display all breakpoints before using the Go command to make sure the breakpoints you've got are the ones you want.

Examples:

`BI * 13` clears all Interrupt breakpoints and then sets a breakpoint on Interrupt 13H.

`BI #` sets all Interrupt breakpoints.

`BI` or `BI ?` displays all Interrupt breakpoints.

9

Command: Breakpoint on Line (BL)

Syntax: `BL [?] [*] [+] [-]`

Description: Use this command to set a monitor breakpoint at the next source code line. This breakpoint gets you to the next instruction that corresponds to a source line of a program. You may want to use the `JL` command instead.

If your program is executing and you press the Break-out Switch, chances are very good that you will stop the program in DOS, in BIOS, or in a library routine. This breakpoint is a convenient (but slow) method of getting back to the source program. It requires source line numbers to be in the symbol table.

After you set the Line breakpoint, enter `GA` or `GT` to execute to the next source line. Since this command can be very slow, use a `G` command to get a known execution address if possible.

You must turn on Line breakpoints for the first time with

`BL +`. After you do that, you can then use `BA` to enable and disable the Line breakpoint. Periscope remembers this breakpoint until you clear it.

You may also use the `SR` and `SC` commands to analyze the stack to determine the next source line.

Example:

`BL +` turns the Line breakpoint on so that a subsequent `GA` or `GT` command will stop when it reaches the next instruction that corresponds to a source code line.

Command: Breakpoint on Memory (BM)

Syntax: `BM [<address> <address> R` and/or `W` and/or `X] [?] [*] [+] [-] [...]`

Description: Use this command to set a monitor breakpoint when a range of memory will be read, written, and/or executed.

The two addresses may have different segments, but the second `<address>` must not be lower in memory than the first `<address>`. If you do not specify a segment in the address, Periscope uses the current data segment. You may use a range instead of the two addresses. You may also substitute registers and valid symbols (see the `<address>` command parameter in Section 8.3).

You may set up to eight breakpoints at one time. The read (`R`) or write (`W`) breakpoints will occur only if a read or write starts in the specified range. The execute (`X`) breakpoint will occur only if CS:IP is in the specified range. If any of the tests pass, a breakpoint is taken. This breakpoint stops execution of the program on the instruction that will read, write, or execute in the specified range of memory.

This breakpoint will not detect a change caused by code that is not traced. It will never see changes made by a hardware interrupt. If you're using the `GT` command, be sure that the you're tracing the appropriate interrupts.

You may enter multiple breakpoints on a single input line, along with the breakpoint clear (*), display (?), enable (+), and disable (-) functions. Periscope remembers these breakpoints until you clear them. Re-enter a breakpoint to clear it and display the message `Breakpoint cleared`. Be careful to display all breakpoints before you use the Go command to make sure the breakpoints you've got are the ones you want.

Periscope/32 users: Periscope/32 assumes the high and low selectors are the same.

Examples:

`BM DATASTART DATAEND W` sets a Memory breakpoint for writes from DATASTART through DATAEND. Any instruction that writes to this range of memory causes a breakpoint to be taken, before the instruction executes.

`BM * SS:SP SS:FFFF RW ?` clears all Memory breakpoints, sets a breakpoint to trap any reads or writes to the memory from SS:SP (the current stack position) to SS:FFFF (the top of the stack segment), and displays the Memory breakpoint.

`BM` or `BM ?` displays all Memory breakpoints.

Command: Breakpoint on Port (BP)

Syntax: `BP [<port> [<port>] I` and/or `O] [?] [*] [+] [-] [...]`

Description: Use this command to set a monitor breakpoint when a range of I/O ports will be read and/or written as the result of an instruction.

The second port must be greater than or equal to the first port. You can set up to eight port breakpoints at one time. A breakpoint will occur only if an IN or an OUT occurs to a port in the specified range. If any of the tests pass, a breakpoint is taken. This breakpoint stops execution of a program on the instruction that will read or write the specified range of ports.

You may set multiple breakpoints on a single input line, along with the breakpoint clear (`*`), display (`?`), enable (`+`), and disable (`-`) functions. Periscope remembers these breakpoints until you clear them. Re-enter a previously set breakpoint to clear it and display the message **`Breakpoint cleared`**. Be careful to display all breakpoints before using the Go command to make sure the breakpoints you've got are the ones you want.

Examples:

`BP 310 31F I` sets a Port breakpoint for ports from 310 to 31F. Any instruction that reads from this range of ports causes a breakpoint to be taken, before the instruction is executed.

`BP * 304 O ?` clears all Port breakpoints, sets a breakpoint to trap any writes to port 304, and displays the Port breakpoint.

`BP` or `BP ?` displays all Port breakpoints.

Command: Breakpoint on Register (BR)

Syntax: **`BR [<register> <test> <number>] [?] [*] [+] [-] [...]`**

Description: Use this command to set a monitor breakpoint when a **`<register>`** meets a **`<test>`**.

You may set up to one test per register, on any of the 8-bit or 16-bit registers except FS and GS, at one time. If any of the tests pass, a breakpoint is taken. This breakpoint stops execution of a program on the instruction following the instruction that changed the specified register.

You can set multiple breakpoints on a single input line, along with the breakpoint clear (`*`), display (`?`), enable (`+`), and disable (`-`) functions.

Periscope remembers these breakpoints until you clear them. Re-enter a previously-set breakpoint to clear it and display the message `Breakpoint cleared`. Be careful to display all breakpoints before using the Go command to make sure the breakpoints you've got are the ones you want.

Periscope/32 users: Periscope/32 does not yet support setting breakpoints on the 32-bit registers.

Examples:

`BR CX EQ 0123` sets a breakpoint when register CX is equal to 123H.

`BR * ES NE DS ?` clears all Register breakpoints, sets one, and then displays it. DS is used for its current value only.

`BR` or `BR ?` displays all Register breakpoints.

Command: Breakpoint on User test (BU)

Syntax: `BU [<number>] [?] [*] [+] [-] [...]`

Description: Use this command to enable a user-written monitor breakpoint. The user breakpoints permit breakpoint tests not provided by Periscope. The `<number>` may vary from `1` to `8`, indicating one of eight possible user breakpoints.

To use this breakpoint, install Periscope with the `/I` installation option, then install a program similar to USER-EXIT.ASM described in Section 7.10 before you run the debugger. When you enter a User breakpoint command, Periscope displays an error if it cannot find the signature of the user exit code. The user routine must set register AL to 1 if a breakpoint is to be taken before returning control to Periscope. Otherwise, no breakpoint will be taken.

You can set multiple breakpoints on a single input line, along with the breakpoint clear (`*`), display (`?`), enable (`+`), and disable (`-`) functions. Periscope remembers these breakpoints until you clear them. Re-enter a previously-set breakpoint to clear the breakpoint and display the message **`Breakpoint cleared`**. Be careful to display all breakpoints before using the Go command to make sure the breakpoints you've got are the ones you want.

Examples:

Assuming that a user-written interrupt handler has been installed using INT 60H and that Periscope was installed with the `/I:60` installation option:

`BU 1` enables User breakpoint number 1.

`BU 9` returns an error since the User breakpoint range is from one to eight.

`BU` or `BU ?` displays all User breakpoints.

Command: Breakpoint on Word (BW)

Syntax: `BW [<address> <test> <number>] [?] [*] [+] [-] [...]`

Description: Use this command to set a monitor breakpoint when a word of memory meets a test.

You can set up to eight breakpoints at one time. If you do not specify a segment in the address, Periscope uses the current data segment. If any of the tests pass, a breakpoint is taken. This breakpoint stops execution of a program on the instruction following the instruction that changed the specified word of memory.

You can set multiple breakpoints on a single input line, along with the breakpoint clear (`*`), display (`?`), enable (`+`), and disable (`-`) functions. Periscope remembers these breakpoints until you clear them. Re-enter a previously-set breakpoint to clear it and display the message **`Breakpoint cleared`**. Be careful to display all breakpoints before using the Go command to make sure the breakpoints you've got are the ones you want.

Examples:

`BW CHARCOUNT EQ 1234` sets a Word breakpoint for the memory location corresponding to CHARCOUNT.

`BW * DS:123 GT SI ?` clears all Word breakpoints, sets one, and then displays it. SI is used for its current value only.

`BW` or `BW ?` displays all Word breakpoints.

Command: Breakpoint on eXit (BX)

Syntax: `BX [?] [*] [-][+]`

Description: Use this command to set a monitor breakpoint on return from a subroutine or interrupt handler.

With this breakpoint set, execution continues until Periscope finds a RET, RETF, or IRET instruction. It is a convenient method of executing until the program is about to transfer control to another procedure.

You must turn on the eXit breakpoint for the first time with `BX +`. You can then disable and enable it with the `BA` command. Periscope remembers this breakpoint until you clear it.

Examples:

`BX +` turns the eXit breakpoint on so that a subsequent `GA` or `GT` command will stop when a RET, RETF, or IRET instruction occurs.

`BX` or `BX ?` displays the status of the eXit breakpoint.

Command: Compare (C)

Syntax: `C <range> <address>`

Description: Use this command to compare two blocks of memory a byte at a time.

Periscope displays the address and value of the byte from the first block of memory and the value and address of the corresponding byte from the second block of memory, only when the bytes are different. It displays nothing for bytes that match.

Since this command accepts two addresses as input, the two blocks of memory may be in different segments. If you do not specify a segment, Periscope uses the current data segment. The length part of the `<range>` parameter indicates how much memory you want to compare.

For example, assume that you want to compare memory location 3000:0000 with 3000:0010 for 8 bytes. Enter `C 3000:0 L 8 3000:10`. The result might be:

```
3000:0000  88  00  3000:0010
3000:0001  02  66  3000:0011
3000:0003  04  27  3000:0013
```

The above display shows three bytes that are different. Each line shows the first address, the value of the first address, the value of the second address, and the second address. Since the other five lines are not displayed, the values of these bytes are the same.

Periscope/32 users: Lengths can be greater than 64K in remote mode.

Examples:

`C DS:SI L 100 ES:DI` compares 100H bytes starting at DS:SI with 100H bytes starting at ES:DI.

`C 123 L CX 456` compares memory starting at DS:123 with memory starting at DS:456. The number of bytes compared is the current value of register CX.

`C FCB1 L 25 FCB2` compares memory starting at the symbol FCB1 with memory starting at the symbol FCB2 for 25H bytes.

Command: Display using current format (D)

Syntax: `D [<range>]`

Description: Use this command to display a block of memory in the current display format.

When you install Periscope, the Display command defaults to a Byte format. Subsequent Display commands use the most recent explicit format. See both the descriptions of the various display formats on the following pages and the information applicable to all display formats in the following three paragraphs.

The syntax for all of the Display commands except `DE` and `DR` is very flexible. If you enter `Dx`, where `x` is the format, Periscope displays memory starting where the last Display command left off. If you enter `Dx <number>`, Periscope presumes the `<number>` to be an offset, the segment to be DS, and the length to be 80H. If you enter `Dx <number> <length>`, it presumes the `<number>` to be an offset and the segment to be DS. If you're using a Data window, the size of the window overrides the `<length>` parameter.

When you're not displaying information in a window and one or more lines in the middle of the display are multiple occurrences of the same number, Periscope suppresses the line(s) and displays a message of the form `* NNNN Lines Of XX Skipped *` in place of the line(s). `NNNN` is the number (in hex) of lines skipped and `XX` is the byte value found in all bytes of the skipped lines.

If you're using a Data window and the window is active, you can use the **PgUp**, **PgDn**, **PadMinus**, and **PadPlus** keys to move forward and backward through memory.

Examples:

`D` displays memory starting where the last Display command left off.

`D ES:DI` displays memory starting at ES:DI for a length of 80H.

`D LINECOUNT L 1` displays memory starting at the symbol LINECOUNT for a length of 1 using the current format.

Command: Display using ASCII format (DA)

Syntax: `DA [<range>]`

Description: Use this command to display a block of memory in ASCII.

Each line of the display shows the starting segment and offset and up to 64 bytes of ASCII characters. This command displays all characters as is, except for the control characters nul, backspace, tab, carriage return, and line feed. It converts nuls to spaces and the other three control characters to periods. It begins a new line when it finds a CR/LF. When it finds a tab character, it moves the output position to the next tab stop.

If you're using a Data window with this format, the **PgUp** and **PadMinus** keys do not keep the display aligned since the data is variable length.

Examples:

`DA` displays memory starting where the last Display command left off.

`DA FILENAME L20` displays memory starting at the symbol FILENAME for a length of 20H bytes.

`DA ES:DI` displays memory starting at ES:DI for a length of 80H.

Command: Display using Byte format (DB)

Syntax: `DB [<range>]`

Description: Use this command to display a block of memory in hex and ASCII.

The display shows the starting segment and offset, up to 16 bytes, and their ASCII representation for each line. A dash appears between the eighth and ninth bytes for readability.

This command ignores the high-order bit for the ASCII display, i.e., it ands the original byte with a value of 7FH. Also, it displays any bytes from zero to 1FH as periods.

Examples:

`DB` displays memory starting where the last Display command left off.

`DB LINECOUNT L 1` displays the byte at the symbol LINECOUNT.

`DB ES:DI` displays memory starting at ES:DI for a length of 80H.

Command: Display using Double word format (DD)

Syntax: `DD [<range>]`

Description: Use this command to display a block of memory in double word format.

This format is useful for examining data that is stored as a word offset followed by a word segment. Each line of the display shows the starting segment and offset and up to four pairs of segments and offsets.

This command shows your program's interrupt vectors for interrupts 8, 9, 10H, 15H, 16H, 17H, 1BH, 1CH, 23H, and 24H, with an asterisk after the doubleword indicating this is a virtual value. The other display commands show the real values, which are Periscope's current values.

Periscope/32 users: The doubleword format is eight digits separated in the middle by a comma.

Examples:

`DD` displays memory starting where the last Display command left off.

`DD 0:0 L 20` displays the interrupt vectors 0 through 7.

Command: Display Effective address (DE)

Syntax: `DE`

Description: Use this command to display the effective address of any reads or writes performed by the current instruction. It has no arguments.

The display shows the address, in Byte format, of any reads or writes performed by the instruction at CS:IP. If you've set up a Data window, the window displays the current effective address automatically before each instruction executes.

If the current instruction reads memory or reads and writes memory, this command displays the effective address of the read. If the instruction writes memory, it displays the effective address of the write.

Examples:

If the current instruction is `LODSB`, the `DE` command displays memory in Byte format starting at the read address, DS:SI.

If the current instruction is `MOV [0123],AX`, the `DE` command displays memory starting at DS:123H.

If the current instruction is `MOVSW`, the `DE` command displays memory starting at DS:SI but does not display the write address of ES:DI.

Command (Periscope/32 only): Display Global/local descriptor table (DG)

Syntax: `DG [<range>]`

Description: Use this command to display the global (GDT) or local (LDT) descriptor table.

Periscope/32 treats the first address in the `<range>` as a selector number from `0` to `FFFF`. The optional second argument is the ending selector. If bit two of the selector is zero, Periscope/32 displays entries from the GDT. Otherwise, it displays entries from the LDT.

Example:

One line of a sample display below shows the selector, base, limit, access rights byte, P (present), A (accessed), G or L (GDT or LDT), default privilege level, and type (Read/Write in this case).

```
0030 Base=0000,0000 Limit=FFFF,FFFF AR=93 PAG DPL=0 Read/Write
```

9

Command: Display using Integer format (DI)

Syntax: `DI [<range>]`

Description: Use this command to display a block of memory in unsigned integer (word) format.

This format is useful for examining data that is stored as an unsigned word integer. Each line of the display shows the starting segment and offset and up to eight decimal numbers. The number displayed may be from `0` to `65535`.

Examples:

`DI` displays memory starting where the last Display command left off.

`DI DS:SI L 20` displays memory starting at DS:SI for a length of 20H bytes.

`DI ARRAY` displays memory starting at the symbol ARRAY.

Command: Display using Long real format (DL)

Syntax: `DL [<range>]`

Description: Use this command to display a block of memory in long real (quad word) format.

This format enables you to examine data that is stored as an eight-byte floating point number in 8087 (IEEE) format. Each line of the display shows the starting segment and offset and up to two numbers in scientific notation.

Examples:

`DL` displays memory starting where the last Display command left off.

`DL DS:SI L 20` displays memory starting at DS:SI for a length of 20H bytes.

`DL ARRAY` displays memory starting at the symbol ARRAY.

Command: Display using Number format (DN)

Syntax: `DN [<range>]`

Description: Use this command to display a block of memory in signed integer (word) format.

This format enables you to examine data that is stored as a signed word integer. Each line of the display shows the starting segment and offset and up to eight decimal numbers. The decimal numbers shown may vary from `0` to `32767` (0H to 7FFFH) and from `-32768` to `-1` (8000H to FFFFH).

Examples:

DN displays memory starting where the last Display command left off.

DN DS:SI L 20 displays memory starting at DS:SI for a length of 20H bytes.

DN ARRAY displays memory starting at the symbol ARRAY.

Command: Display using Record format (DR)

Syntax: **DR [!] <address> <symbol>**

Description: Use this command to display a block of memory in an easy-to-read format using a previously-created record definition.

```
Int 20      CD 20                                                  M
Topmem      9FFF
Long Call   9A                                                     .
DOS Func    F01D:FEF0
Term Addr   03B7:01DC
Brk Addr    03B7:014B
Err  Addr   03B7:0156
Caller      03B7
File Tbl    01 01 01 00 02 03 04 FF-FF FF FF FF FF FF FF FF ................
            FF FF FF FF                                     ......
Environ     0D8E
Last Stack  C800:080C
Files       0014
File Ptr    0DAF:0018
DOS Ver     05 00                                           ..
FCB1        00 20 20 20 20 20 20 20-20 20 20 20 00 00 00 00 .           ....
FCB2        00 20 20 20 20 20 20 20-20 20 20 20 00 00 00 00 .           ....
            69 7A 65 20                                     ize
```

Figure 9-1. Sample Display Using the DR Command

This format enables you to examine data that is part of a record, such as the PSP or an FCB, and to define a structure. Each line of the display shows a field name and the data for the field in any display format supported by Periscope. You can display any area of memory using any record definition. When you use the optional exclamation point, the address of each field appears on the line before the field.

To use a record format, a record definition (.DEF) file, must exist. RUN loads the record definitions from the .DEF file. You can add record definitions to the .DEF file using a text editor. See the sample file PS.DEF and the description of RS in Section 7.5.

Enter **DR ES:0 PSP** to get a display similar to Figure 9-1.

```
\PSP                    ; Program Segment Prefix
Int 20,b,2              ; DOS return
Topmem,w,2              ; Amount of memory in paragraphs
Res.,+,1                ; Reserved for DOS
Long Call,b,1           ; Long call to DOS function dispatcher
DOS Func,d,4            ; CS:IP of DOS function dispatcher
Term Addr,d,4           ; CS:IP of DOS terminate address
Brk Addr,d,4            ; CS:IP of Ctrl-Break exit address
Err Addr,d,4            ; CS:IP of critical error exit address
Caller,w,2              ; PSP of caller
File Tbl,b,14           ; File table
Environ,w,2             ; DOS 2.00 Environment segment
Last Stack,d,4          ; SS:SP after last DOS call
Files,w,2               ; File table size
File Ptr,d,4            ; File table pointer
Res.,+,8                ; Reserved for DOS
DOS Ver,b,2             ; DOS version after Setver
Res.,+,1a               ; Reserved for DOS
FCB1,b,10               ; The first FCB read from the command line
FCB 2,b,14              ; The second FCB read from the command line
```

Figure 9-2. Definition of the PSP from the File PS.DEF

The syntax for this command is less flexible than that of the other Display commands. You must enter an address and a record name. The address should include a segment, since the address used for this command is separate from the address used for the other Display commands.

Examples:

Assume that the records PSP and FCB are defined (as in the file PS.DEF).

`DR CS:0 PSP` displays the PSP, using memory starting at CS:0.

`DR CS:5C FCB` displays the first FCB in the PSP, which starts at CS:5C.

`DR FCB1 FCB` displays the FCB starting at the address referenced by the symbol FCB1. Note that the symbol table is used for the first symbol and the record definition table is used for the second symbol.

Command: Display using Short real format (DS)

Syntax: `DS [<range>]`

Description: Use this command to display a block of memory in short real (double word) format.

This format enables you to examine data that is stored as a four-byte floating point number in 8087 (IEEE) format. Each line of the display shows the starting segment and offset and up to two numbers in scientific notation.

Examples:

`DS` displays memory starting where the last Display command left off.

`DS DS:SI L 10` displays memory starting at DS:SI for a length of 10H bytes.

`DS ARRAY` displays memory starting at the symbol ARRAY.

Command (Periscope/32 only): Display TSS (DT)

Syntax: `DT`

Description: Use this command to display the task state segment (TSS). The display shows the task's backlink selector, LDT selector, I/O Map Base, stack segment for privilege levels 0, 1, and 2, general registers, segment registers, and flags.

Example:

```
dt
Link=0000  LDT Selector=0000  I/O Base=0068
SS:ESP 0=0030:8001,0D98  1=0000,0000  2=0000:0000,0000
EAX=0000,0000  EBX=0000,0000  ECX=0000,0000  EDX=0000,0000
ESP=0000,0000  EIP=0000,0000  ESI=0000,0000  EDI=0000,0000   FS-0000
 SS-0000        CS-0000        DS-0000        ES-0000        GS-0000
EBP=0000,0000  EFL=0000,0000 SM NR NN NV UP DI PL NZ NA PO NC IO V86
```

Command (Periscope/32 only): Display interrupt descriptor table (DV)

Syntax: `DV [<range>]`

Description: Use this command to display the interrupt descriptor table.

Example:

One line of a sample display below shows the vector number, the gate type, address, the access rights byte, P (present), and the default privilege level.

```
0002 Int 0028:8001,C530 AR=EE P DPL=3
```

Command: Display using Word format (DW)

Syntax: `DW [<range>]`

Description: Use this command to display a block of memory in word format.

This format enables you to examine data that is stored as words rather than as bytes. It reverses out the 'back words' style of storage used by the 8086 family. Each line of the display shows the starting segment and offset and up to eight words.

Examples:

`DW` displays memory starting where the last Display command left off.

`DW SS:SP FFFF` displays the stack from SS:SP to the top of the stack segment.

`DW POINTER` displays memory starting at the symbol POINTER.

Command: Display using long integer format (DX)

Syntax: `DX [<range>]`

Description: Use this command to display memory in long integer (four-byte) format. This format enables you to examine data that is stored as an unsigned long integer.

Each line of the display shows the starting segment and offset and up to four decimal numbers. The number displayed may be from `0` to `4,294,967,295`.

Examples:

`DX` displays memory starting where the last Display command left off.

`DX ARRAY` displays memory starting at the symbol ARRAY.

Command: Display using long signed integer format (DY)

Syntax: `DY [<range>]`

Description: Use this command to display memory in a long signed integer (four-byte) format.

Each line of the display shows the starting segment and offset and up to four decimal numbers. The number displayed may be from `-2,147,483,648` to `+2,147,483,647`.

Examples:

`DY` displays memory starting where the last Display command left off.

`DY DATAPOINTS` displays memory starting at the symbol DATAPOINTS.

Command: Display using asciiZ format (DZ)

Syntax: `DZ [<range>]`

Description: Use this command to display a block of memory in nul-terminated ASCII format.

This command is the same as the `DA` command, except that it ends the display when it finds a nul (binary zero). If you're using a Data window, the display continues to the end of the window. See the description of the `DA` command above for more information.

Examples:

`DZ` displays memory starting where the last Display command left off.

`DZ FILENAME` displays memory starting at the symbol FILENAME and continues until a nul is found or 80H bytes have been displayed.

Command: Enter (E)

Syntax: `E <address> [<list>]`

Description: Use this command to modify memory.

You must specify the segment and offset for the `<address>` to avoid accidental changes to memory. If you specify the optional `<list>`, the command modifies the specified memory and terminates. If you do not specify the list, an interactive mode begins. This mode allows you to examine and optionally modify individual bytes starting at the specified address.

For example, if you enter `E 2000:123` and press return, the interactive mode begins. Periscope displays the address and the current value of the byte as `2000:0123 xx`, where `xx` is the current value. To modify this value, enter the hex number (`0` through `FF`). Periscope echoes your input unless it's invalid, such as `G9` or too many digits.

Press the space bar to go to the next byte. Press the hyphen key to back up one byte. Use the backspace key to discard a single digit. Use the return key to terminate the interactive mode.

Examples:

`E CS:5C 0 "FILENAMEEXT"` modifies the value of CS:5C through CS:67 to contain a binary zero and the string FILENAMEEXT.

`E 404:100` starts the interactive mode and displays `0404:100 80`. To change this value to 88H, type `88`. To display the next byte, press the space bar. To change the byte at offset 104H to 0, enter `0` when the byte is displayed. To back up to offset 102H, press the hyphen key as many times as needed to get back to it. When you've finished your changes, press **Enter**.

Command: Enter Alias (EA)

Syntax: `EA <alias> [<name>]`

Description: Use this command to define or redefine an alias, which is a two-character shorthand notation for a one-to-64-character `<name>`. If you enter this command without a name parameter, the `<alias>` is deleted.

To display the current aliases, press **Alt-A.** You can assign other aliases as needed and then activate them as commands by using `^XX`, where `XX` is the two-character alias name. You cannot modify a read-only alias, such as **FX**, with this command.

For more information on aliases, see RS in Section 7.5.

Example:

`EA X2 D ES:DI` defines alias **X2** as `D ES:DI`. This command is executed after each Periscope command, a convenient way to constantly monitor the current value of ES:DI.

Command: Enter Bytes (EB)

Syntax: `EB <address> <list>`

Description: Use this command to modify memory a byte at a time. It is similar to the Enter command, but has no interactive mode. Each item in the `<list>` is a byte value. You may enter one or more bytes. You must specify the segment and offset for the `<address>`.

Example:

`EB FILENAME 'A:FILE' 0` modifies memory at the symbol FILENAME to contain the nul-terminated string A:FILE.

Command: Enter Doublewords (ED)

Syntax: `ED <address> <address> [...]`

Description: Use this command to modify memory a doubleword at a time. It is similar to the Enter command, but has no interactive mode. Each `<address>` in the command line is composed of a segment and offset or a symbol. You must specify the segment and offset for the first address, which is the destination. The second and subsequent addresses are the values to be written.

Periscope/32 users: Periscope/32 accepts either a 32-bit offset or a segment, colon, and 16-bit offset.

Example:

`ED 0:0 1234:5678` modifies INT 0 (divide overflow) to point to 1234:5678.

9

Command (Periscope/32 only): Access memory anywhere in the target system (EM)

Syntax: `EM <address>`

Description: Use this command to access memory anywhere in the target system.

Enter a 32-bit address and Periscope/32 returns a selector that you can use to access memory starting at the address you entered. You must enter the address as a 32-bit address, not in segment:offset format.

Example:

To access memory on the monochrome display adapter at B000:0000, enter (comma is optional):

```
EM B,0000
```

Command: Enter Symbol (ES)

Syntax: `ES <address> <symbol>`

Description: Use this command to define or redefine symbol table entries.

You must specify a segment and offset for the `<address>`. The `<symbol>` name must be 32 characters or less and may be preceded by a period. This command adds symbols to the end of the symbol table, regardless of any duplicate names in prior public, line, or local symbols. Use the `/R` command to delete any conflicting symbol names.

Examples:

`ES CS:100 START` defines a symbol named START to have a segment equal to the current value of CS and an offset of 100H.

`ES ES:DI OUTDATA` defines a symbol named OUTDATA to have a segment and offset equal to the current values of ES and DI, respectively.

Command: Enter Words (EW)

Syntax: `EW <address> <number> [...]`

Description: Use this command to modify memory a word at a time. It is similar to the Enter command, but has no interactive mode. Each `<number>` is a word. You may enter one or more words. You must specify the segment and offset for the `<address>`.

Example:

`EW ARRAY 1 2 3` writes three words starting at ARRAY.

Command: Fill (F)

Syntax: `F <range> <list>`

Description: Use this command to fill a block of memory with a byte/string pattern.

You must specify a segment and offset for the address portion of the `<range>`. The length specifies the number of bytes affected. The `<list>` is the pattern copied into the specified range of memory. If the length of the list is less than the length of the range you specify, the command copies the list as many times as needed to fill the range. Conversely, if the length of the list is greater than the length of the range, it does not copy the extra bytes.

Periscope/32 users: Lengths may be greater than 64K in remote mode.

Examples:

`F ES:0 L 1000 0` writes binary zeroes to memory starting at ES:0 for a length of 1000H bytes.

`F DS:SI L CX "test"` writes the string `test` to memory starting at DS:SI. If CX is 3, only `tes` is copied. If CX is 8, test is copied exactly two times, etc.

`F ARRAY ENDARRAY 0` zeroes memory from the symbol ARRAY up to and including the symbol ENDARRAY.

Command: Go (G)

Syntax: `G [<address>] [...]`

Description: Use the Go command to set temporary code breakpoints, activate sticky code breakpoints and debug register breakpoints, and execute the program you're debugging. (See Section 8.2 for a general discussion of breakpoints.) This command activates code breakpoints and debug register breakpoints only. To activate monitor breakpoints, use the `GA`, `GM` or `GT` commands.

You can use `Ctrl-G` to go to the line displayed at the top of the Disassembly window.

If you specify any `<address>` parameters on the command line, this command replaces the byte at each of the addresses with a CCH, the single-byte breakpoint. When Periscope regains control via any method, it restores the original byte. The addresses you enter on the command line are temporary code breakpoints. You may specify up to four of these breakpoints. If you do not specify a segment, the command uses the current code segment.

To set up to sixteen "sticky" code breakpoints, use the `BC` command. To set debug register breakpoints, use the `BD` command, which allows you to set up to four limited real-time hardware breakpoints on instruction execution, memory writes, or memory access (read or write).

If you enter `G` with no addresses, any sticky code and debug register breakpoints are activated. If no sticky code or debug register breakpoints are set, program execution continues until completion or until you press the Break-out Switch.

Periscope remembers code breakpoints until you clear them or until you re-run Periscope. If you have set code or debug register breakpoints and want to continue program execution without using any of the breakpoints, you can disable all breakpoints using `BA -`, or you can use the `QC` command.

You cannot set code breakpoints in ROM. (Use the debug register breakpoints for this!) Code breakpoints require that Periscope be able to exchange the original byte with CCH before starting the Go command. Since the setting of a code breakpoint in the middle of an instruction can have unpredictable results, set code breakpoints using symbol names where possible.

Examples:

`G PRINTLINE` sets a temporary code breakpoint at the address equal to the symbol PRINTLINE and starts execution of the program.

`G FF00:0000` returns an error since the address is in ROM.

`G` begins execution of the program with no temporary code breakpoints.

`G 123` sets a temporary code breakpoint at CS:123 and starts execution of the program.

Command: Go plus (G+)

Syntax: `G+`

Description: Use this command to set a temporary code breakpoint on the next instruction, activate any code breakpoints set with the `BC` command, and execute the program you're debugging. `G+` activates code and debug register breakpoints only. Other than setting a temporary breakpoint on the following instruction, it is identical in function to the `G` command described above.

Example:

If IP is 200H and the current instruction is INT 21H:

`G+` sets a temporary breakpoint at 202H, which is after the INT instruction.

Command: Go equal (G=)

Syntax: `G=[<address>] [...]`

Description: Use this command to set CS:IP to the first **`<address>`**, set any indicated temporary code breakpoints, activate any sticky code breakpoints set with the `BC` command, and execute the program you're debugging. This command activates code and debug register breakpoints only.

Other than setting CS:IP to the first address entered, this command is identical in function to the `G` command described above. The equal sign must appear immediately after the letter `G`.

Examples:

`G=PRINTF` sets CS:IP to the value indicated by the symbol PRINTF and executes the program.

`G=123` sets CS:IP to CS:123 and executes the program.

Command: Go using All (GA)

Syntax: `GA [<address>] [...]`

Description: Use this command when you want to trace ALL instructions. Otherwise, it is identical to the `GT` command.

This command activates code breakpoints, debug register breakpoints, and monitor breakpoints. It always traces ALL the way through ALL software interrupts. The `GT` command single steps through instructions, except when it encounters a software interrupt. Then it checks the interrupt trace table (see the `/T` command). If the interrupt is not in the table, `GT` does not trace through the interrupt. This can cause `GT` to miss breakpoints. The `GA` command is slower than `GT`, but more dependable.

It is not possible to trace into hardware interrupts with this command. Use Periscope Model IV for this purpose.

Example:

See the examples under the `GT` command.

Command: Go using Monitor (GM)

Syntax: `GM [<address>] [...]`

Description: Use this command to go at full speed to a certain point and then evaluate the monitor breakpoints. If any of the monitor breakpoints indicate a hit, Periscope displays its screen. Otherwise full speed execution resumes.

This command activates code breakpoints, debug register breakpoints, and hardware breakpoints. It evaluates the monitor breakpoints only when a code, debug register, or hardware breakpoint occurs.

Unless you are using Periscope Model IV, this command acts as a combination of the `G` and `GT` commands. Consider the situation where you need to watch a buffer for an end of file marker. Using `GT`, this would usually be very time-consuming. If you enter a monitor test (such as `BR AL EQ 1A`) and then use `GM` to go to the appropriate place in your code, most of the code will be executed at full speed. Each time a code or debug register breakpoint is reached, the monitor breakpoint(s) are evaluated, rather than after every instruction.

Example:

`GM PRINTLINE` sets a temporary code breakpoint at the address equal to the symbol PRINTLINE. When a code, debug register, or hardware breakpoint occurs, the monitor breakpoints are evaluated. If any of the monitor breakpoints indicate a hit, Periscope displays its screen. Otherwise, full-speed execution resumes.

Command: Go to Return address on stack (GR)

Syntax: `GR`

Description: Use this command to analyze the stack for the first return address and set a temporary code breakpoint at that address.

This command uses the same logic as the `SR` command. If it does not find a return address, it displays an error. This command has no arguments. It ignores any addresses you enter after `GR`.

Example:

`GR` examines the stack for a return address and then goes to that address.

Command: Go using Trace (GT)

Syntax: `GT [<address>] [...]`

Description: Use this command to trace in a single-step mode that evaluates the monitor breakpoints after each instruction. This command activates code breakpoints, debug register breakpoints, and monitor breakpoints.

This command puts the system into a mode where it analyzes every instruction executed by your program to see if a breakpoint has been reached. This analysis can slow down the execution of your program by a factor of 100 to 1000 or more, but in many cases is the only way to find an elusive bug (unless you're using Periscope Model IV). Since this command is slow, try to use the normal Go command to get as close to the problem as possible, then use `GT`.

Periscope remembers the monitor breakpoints until you clear them or re-run Periscope. If you have code, debug register, and/or monitor breakpoints set and want to continue program execution without using any of the breakpoints, you can disable all breakpoints (using `BA -`) or use the `QC` command.

To make sure you've got the correct breakpoints, get in the habit of checking the breakpoint settings before using this command. Enter `BA` to display the current breakpoints before entering `GT`.

For temporary and sticky code breakpoints, this command performs in the same fashion as the Go command. After a breakpoint, use the `TB` command to see the instructions preceding the instruction that caused the breakpoint.

When this command encounters a software interrupt, it executes it step-by-step if and only if the interrupt number is set in the trace table (see the description of the `/T` command). If the breakpoint you're trying to find is in an interrupt or is caused by an interrupt, you should use the `GA` command, since it traces all software interrupts.

It is not possible to trace into hardware interrupts with this command. Use Periscope Model IV for this purpose.

If you're using the `GT` command and the program you're debugging starts running at full speed, an ill-behaved interrupt has probably modified the trap (single-step) flag. To find the problem interrupt, press the Break-out Switch and then use `TB` to see the last code that Periscope traced. Note the last entry in the software trace buffer. The interrupt shown is the one that caused Periscope to lose control.

You've got two possible solutions: either use the `/T` command to force tracing of the offending interrupt or use the `GA` command to force tracing of all interrupts.

Examples:

`GT PRINTLINE NEWPAGE` sets temporary code breakpoints at the addresses equal to the symbols PRINTLINE and NEWPAGE.

`GT` begins execution of the program with no temporary code breakpoints, only sticky code, debug register, and monitor breakpoints that are enabled.

`GT ES:456` sets a temporary code breakpoint at ES:456.

Command: Hex arithmetic (H)

Syntax: **`H <number> <arithmetic operator> <number>`**

Description: Use this command to perform hexadecimal arithmetic.

Addition, subtraction, multiplication, and division are available, using the standard **`<arithmetic operator>`**s.

Each `<number>` must be in hex and may be from one to four hex digits. If you enter a register name in place of one of the numbers, this command uses its current value for the number.

Multiplication returns two words separated by spaces. The first word is the high-order part. Division returns two words separated by the letter `R`. The first word is the quotient and the second is the remainder.

See also the `X` (translate) command.

⇨ **Periscope/32 users:** Numbers can be up to eight digits each.

Examples:

`H 1234/123` gives an answer of `0010 R 0004`.

`H 1234*123` gives an answer of `0014 B11C`.

Model IV Hardware (H series) Commands—please see the Model IV manual.

Command: Input (I)

Syntax: `I <port>` or `IW <port>`

Description: Use this command to read an I/O port. The first form reads a byte while the second form reads a word.

The port number may be from `0` to `FFFF`H, although the IBM PC only supports ports from zero to 3FFH. Any larger number is effectively ANDed with 3FFH. The value retrieved by reading the port displays on the line following the command.

Examples:

`I 100` performs a byte read of port 100H and displays the byte input.

`IW DX` performs a word read of the port indicated by register DX and displays the word input.

Command: Interrupt Compare (IC)

Syntax: `IC`

Description: Use this command to compare the current value of the interrupt vectors with their previously-saved values.

`IC` is available only after you have used an `IS` command to save the interrupt vectors. After you've used an `IR` command, this command is disabled until you've issued another `IS` command.

Assuming the vector for Interrupt 0 had been changed, the display of the `IC` command might be `00 0294:588F 2345:6789`, where the first field is the interrupt number, which is followed by the old interrupt vector and the new (current) interrupt vector.

Periscope/32 users: Periscope/32 does not support this command in remote mode.

Example:

Assuming the interrupt vectors were previously saved using the `IS` command, enter `IC` to compare all vectors with their saved values.

Command: Interrupt Restore (IR)

Syntax: `IR`

Description: Use this command to restore the interrupt vectors to a previously-saved state.

This command is usable only after you've used an `IS` command to save the interrupt vectors. After you've performed an Interrupt Restore, the Interrupt Restore command is disabled until you've performed another Interrupt Save.

Periscope/32 users: Periscope/32 does not support this command in remote mode.

Example:

Assume the interrupt vectors were previously saved using the `IS` command. Enter `IR` to restore all vectors to their values saved by the `IS` command.

Command: Interrupt Save (IS)

Syntax: `IS`

Description: Use this command to save the interrupts for later comparison or restoration.

The Interrupt Save command saves the current state of the machine's interrupt vectors in case you need to restore the vectors to that state at some later point. For example, assume you're debugging a program that modifies some of the interrupt vectors. If you need to terminate execution of the program, you can restore the interrupt vectors and then use the `QR` command to return to DOS.

To use the Interrupt Save command, enter `IS` at the Periscope prompt. Later, you can restore the vectors to their saved state by using the `IR` command.

To prevent accidental restoration of the vectors, the `IS` command sets a flag that is cleared by the `IR` command. When this flag is cleared, the `IR` command generates an error.

Periscope/32 users: Periscope/32 does not support this command in remote mode.

Example:

Enter `IS` to save the interrupt vectors. At any point later, the `IR` command may be used to restore the vectors to their saved state.

Command: Jump (J)

Syntax: `J`

Description: Use this command as a shorthand form of Go, to execute (step) to the next instruction.

This command executes the current instruction at full speed, avoiding tracing through the execution of CALL, INT, LOOP or other repeated instructions. It performs the same function as a temporary code breakpoint set on the next instruction. The difference is that you don't have to stop and compute the address and then enter a Go command. Jump does it for you. If the current instruction is any form of a RET, IRET, or JMP (including conditional jumps) Periscope traces one instruction (to follow the code) instead of using a temporary code breakpoint.

There is one condition under which this command does not work. When you're tracing ROM you cannot use code breakpoints, since you can't write to ROM.

Generally speaking, it is safe to use this command in place of the Trace command. There are some cases that present a problem, however. One possibility is a LOOP instruction that passes control downwards rather than upwards. Others include CALLs or INTs that do not return control to the next instruction.

Examples:

Assume that the current instruction is INT 21. Enter `J` to place a temporary code breakpoint at the instruction after the INT 21 and execute the INT at full speed.

Assume that the current instruction is RET. Enter `J` to trace to the next logical (not physical) instruction.

Command: Jump Line (JL)

Syntax: `JL`

Description: Use this command to step to the next source-code line.

The `JL` command performs Jump (step) commands until it finds the next source line. This is a quick method of moving through a high-level language program at the source-code level. If it finds no source line symbols, this command acts like the `J` command. See also the `TL` command.

Example:

Assume the current instruction is line 10 of the first source-code module. Enter `JL` to step to the next logical source-code line.

➪ If your compiler does not generate line numbers for every line, some lines may be skipped.

Command: clear (K)

Syntax: `K`

Description: Use this command to clear the Periscope screen and regenerate any windows. It has no arguments.

Example:

`K` clears the screen.

Command: clear and Initialize (KI)

Syntax: `KI`

Description: Use this command to initialize the monitor, clear the Periscope screen, and regenerate any windows. It has no arguments.

Use this variant of the clear command if Periscope's screen is not in text mode on entry to Periscope. Do not use this command if you've set 43-line or 50-line mode. It will revert the screen to 25-line mode.

Example:

`KI` performs a mode set and clears the screen.

Command: Load Absolute disk sectors (LA)

Syntax: `LA <address> <drive> <sectors>`

Description: Use this command to load absolute disk sectors into memory.

The segment defaults to CS if you do not specify a segment in the `<address>`. The `<drive>` is either a single-digit number indicating the disk drive (0=A, 1=B, etc.) or `A:`, `B:`, etc. The `<sectors>` parameter is the starting sector number and the number of sectors to be read. The maximum number of sectors that can be read in one operation is 80H, which equals 64K bytes.

To use this command, DOS must not be busy. This command uses DOS interrupt 25H. See the DOS manual for information on the numbering of the absolute disk sectors.

Periscope/32 users: Periscope/32 does not support this command in remote mode.

Examples:

`LA DS:100 A: 10 20` loads data into memory starting at DS:100 from drive A, starting at sector number 10H for 20H sectors.

`LA 100 B: 0 4` loads data into memory starting at CS:100 from drive B, starting at sector 0 for 4 sectors.

Command: Load Batch file (LB)

Syntax: `LB <file>`

Description: Use this command to load a batch or script file that contains Periscope commands.

To use this command, DOS must not be busy. The input file defaults to an extension of .PSB. You may create the file with the `WB` command, the `/K` command, or a text editor. You can place any legal Periscope commands in the file.

Example:

After using `WB SAVE` to save the breakpoint and window settings to the file SAVE.PSB, use `LB SAVE` to later reload the breakpoint and window settings.

Command: Load alias and record Definitions (LD)

Syntax: `LD *` or `LD <file>`

Description: Use this command to clear and/or load alias and record definitions from a .PSD `<file>` created by RS.

The asterisk clears the alias and record definitions. You can only load .PSD files, not .DEF files, with the `LD` command. See the description of RS in Section 7.5. To load a file, DOS must not be busy.

Examples:

`LD *` clears all alias and record definitions.

`LD FTOC` loads the alias and record definitions from the file FTOC.PSD.

Command: Load File from disk (LF)

Syntax: `LF [<address>]`

Description: Use this command to load a file from disk into memory. Before you can use this command, you must specify a file name with the Name command.

Specify where the file is to be loaded with the optional `<address>`. If you do not specify the address, `LF` assumes CS:100. To use this command, DOS must not be busy.

You can load any type of file into memory with the `LF` command. After you've loaded the file, BX and CX indicate the size of the file in bytes and no other processing occurs. `LF` does not relocate or strip the headers of .EXE files.

You should generally use RUN to load and execute a program, since it loads the symbol table and performs relocation for .EXE files. Use the `LF` command to load a file into memory to examine or modify it.

9

This command cannot load into memory beyond the end of DOS memory. For a 640K system, the maximum address is 9000:FFFF.

Periscope/32 users: Periscope/32 does not support this command in remote mode.

Examples:

`LF DS:1000` loads the file defined by a Name command into memory starting at DS:1000.

`LF` loads the file defined by a Name command into memory starting at CS:100.

Command: Load Symbols from disk (LS)

Syntax: `LS *` or `LS <segment> <file>`

Description: Use this command to load a Periscope symbol file (.PSS) into the symbol table.

Use the asterisk to clear the symbol table.

Enter the relocation factor to be added to the segment values found in the .PSS file in the `<segment>`. For .COM files, this is the value of the PSP segment or CS. For .EXE files, this is the value of the PSP segment plus 10H. To have Periscope calculate the correct value for an .EXE file that is the currently active program, use `$` as a segment value. The `<file>` is the path and file name of the .PSS file.

Use this command to load .PSS files, not .MAP files. Periscope checks the version to ensure that the symbol file used is compatible with the current version of Periscope. To use this command, DOS must not be busy.

Periscope/32 users: You cannot use `LS $` in remote mode since it refers to the PSP on the host system.

Examples:

`LS *` clears the symbol table.

`LS CS SAMPLE` loads the file SAMPLE.PSS into the symbol table, relocating the segments by the current value of CS.

`LS $ FTOC` loads the file FTOC.PSS into the symbol table at PSP +10H for the program FTOC.EXE.

Command: Move (M)

Syntax: `M <range> <address>`

Description: Use this command to copy a block of memory to another location in memory.

You must specify the segment and offset for both addresses. If the source block and target block overlap, the move into the target block is performed without loss of data. The source segment and target segment may be different.

Periscope/32 users: Lengths in remote mode can be greater than 64K. Periscope/32 uses only the offset to calculate the direction of the copy.

Examples:

`M 1000:0 L 100 1000:80` copies 100H bytes from the source block (1000:0 to 1000:FF) to the target block (1000:80 to 1000:17F).

`M 1000:80 L 100 1000:0` copies 100H bytes from the source block (1000:80 to 1000:17F) to the target block (1000:0 to 1000:FF).

`M DS:SI L CX ES:DI` copies CX bytes from the source block (DS:SI) to the target block (ES:DI), where all values are the current contents of the respective registers.

Command: Name (N)

Syntax: `N <file>`

Description: Use this command to enter data into the PSP for disk I/O and for naming files to be read or written by Periscope using the `LF` and `WF` commands.

This command copies the `<file>` parameter to the unformatted parameter area in the PSP, starting at CS:80H. After it copies the name into CS:80H, it uses the DOS parsing function to parse the first two file names in the command line into the FCBs at CS:5CH and CS:6CH. If it finds an invalid drive id for a file, it generates a message and sets register AL or AH to FF, indicating the first or second file, respectively.

It requires that the PSP's address has been set by RUN and that the first four bytes of the PSP contain the bytes `CD 20` followed by the top of memory size in paragraphs.

It copies all data entered after the `N` until it finds a semi-colon or a carriage return. If it cannot find the PSP, you can still use the `LF` and `WF` commands, presuming that DOS is not busy.

Periscope/32 users: Periscope/32 does not support this command in remote mode.

Examples:

`N C:COMMAND.COM` copies the file name to Periscope's internal file buffer and to the unformatted parameter area at CS:80H and then parses the file name into the FCB at CS:5CH.

Command: Output (O)

Syntax: `O <port> <byte>` or `OW <port> <word>`

Description: Use this command to write to an I/O port. The first form writes a byte and the second form writes a word.

The `<port>` may be from `0` to `FFFF`H, although the IBM PC only supports ports from zero to 3FFH. Any larger number is effectively ANDed with 3FFH. The `<byte>` value output to the port may be from `0` to `FF`H. The `<word>` value output may be from `0` to `FFFF`H.

Examples:

`O 100 FF` outputs FFH to port 100H.

`OW DX 1234` outputs 1234H to the port indicated by register DX.

`O DX AX` returns an error since register AX represents a word. Use `OW DX AX` to output a word.

Command: Quit (Q)

Syntax: `Q`, `QB`, `QC`, `QL`, `QR` or `QS`

Description: Use this command to display the reason Periscope was activated and to exit Periscope.

The possible entry reasons are:

`01` - A trap 1 (unexpected single-step) occurred.
`06` - A trap 6 (illegal instruction) occurred.
`0D` - A trap D (segment wraparound) occurred.
`BR` - A monitor breakpoint was taken.
`DR` - An 80386 debug register breakpoint occurred.
`GO` - A code breakpoint was taken.
`HW` - A hardware breakpoint occurred (Periscope Model IV).
`P1` - Parity error 1 (motherboard) occurred.
`P2` - Parity error 2 (expansion memory) occurred. This is normal when you press the Break-out Switch.
`SW` - The Break-out Switch or hot keys (set with PSKEY) were pressed.
`TR` - An instruction was traced.
`WD` - A watchdog interrupt occurred on a PS/2 machine.

The `QB` (boot) command boots the system. It is the same as **Alt-Ctrl-Del**, clearing all of user memory and resetting the standard interrupt vectors. Use this option when the system becomes hopelessly confused or when you suspect that a runaway program may have incorrectly modified a critical area of memory. If an exception interrupt or trap occurs, you should boot the system as soon as possible.

The `QC` (continue) command returns control to the executing program after restoring the program's screen. If nothing is executing, i.e. the DOS prompt is displayed, control is returned to DOS. This method does not set any breakpoints. Use the Go command if you want to set any breakpoints.

The `QL` (long boot) command puts the system through full diagnostics, as when the system is first powered on. Use it if you need to reinstall any BIOS drivers such as the VGA.

The `QR` (return to DOS) command abandons execution of the current program and returns to DOS. This command does not close open files, which can cause problems if the files have been updated, or back out changes the program has made to interrupt vectors. It does clear the keyboard buffer. When possible, use `QC` or `G` instead of this command.

Periscope allows the `QR` command only if DOS is not busy and you used RUN.COM to set the PSP for the executing program.

If you get an error but want to return to DOS even though it is not 100% safe, you can enter `QR!` to force it. Do so at your own risk!

Periscope/32 users: This command acts like a `/X` command in remote mode since Periscope/32 stays in the foreground instead of going resident. Use `QU` or `QR!` to terminate Periscope/32 when you're running in remote mode.

The `QS` (short boot) command re-boots the system via Interrupt 19H. This method preserves most of RAM, including the interrupt vectors. Some sections of memory in the first 64K are overwritten by the boot record and DOS. This can cause problems for some memory-resident programs, such as a low-memory RAM disk. Since all interrupts are not restored, this boot option is more fragile than the other two.

Due to the unchanged vectors, some resident programs and drivers may think they're still installed. If you have problems, either quit using this option or use the user exit F1 to clean up the necessary vectors on the way out.

The short boot is not compatible with all systems. If it doesn't work in your system, try removing all device drivers and memory-resident programs. If it still doesn't work, there's probably not much hope on your system.

There are dedicated user exits for the `QB`, `QL`, and `QS` commands. They enable you to customize the cleanup process for your system before rebooting. See the sample program USEREXIT.ASM in Section 7.10 for more information.

Examples:

`Q` displays the entry reason.

`QC` exits Periscope and continues execution of the interrupted program without setting any breakpoints.

`QB` exits Periscope and performs a normal boot.

Command (Periscope/32 only): Quit, reboot host, Unhook remote driver (QU)

Syntax: `QU`

Description: Use this command to reboot the host system and unhook the remote driver on the target system when you want to use the target system as if the remote Periscope driver had never been installed.

Command: Register (R)

Syntax: **`R [<register>]`** or **`R+`** or **`R=<address>`** or **`R<byte> [<number>]`** or **`R?`**

Description: Use this command to display and modify the current values of the registers and flags.

If you enter **`R`** and press return, this command displays the current values of the registers and flags. If the current instruction performs a memory read and/or write, it displays the effective address of the read/write, along with the current value of memory at the effective address(es). Finally, it disassembles the current instruction. The information is shown in the appropriate windows if you're using windows.

To modify the registers or flags, enter **`R xx`**, where **`xx`** is the register name. Use **`FL`** for the flags. To easily move to the next instruction, enter **`R+`**. Periscope disassembles the current instruction and sets IP to the start of the next instruction.

To change CS:IP to a new address, enter **`R=<address>`**.

There are ten user registers that you may use to hold any word value you desire. In the syntax shown above, the **`<byte>`** is a number from **`0`** to **`9`**. For example, to hold the current value of SP in user register number one, enter **`R1=SP`**. Then at a later time, you could enter **`DW SS:R1`** to use the saved user register. To display the contents of a single user register, enter just the register name. To display the contents of all user registers, enter **`R?`**. These registers can be used wherever a 16-bit register is usable.

In all of the examples shown in Figure 9-3, the first two lines display the current values of the registers and the flags. See

```
EXAMPLE 1:
AX=007F  BX=0034  CX=0000  DX=0000  SP=1724  SI=0F1E  DI=1568
DS=0040  ES=00BF  SS=00BF  CS=F000  IP=E850  FL=0046  NV  UP  DI  PL  ZR  NA  PE  NC
F000:E850 74F3          JZ      E845                          ; jump

EXAMPLE 2:
AX=0000  BX=0000  CX=0100  DX=0001  SP=FFF0  SI=0000  DI=0000
DS=063A  ES=063A  SS=063A  CS=063A  IP=010E  FL=0246  NV  UP  DI  PL  ZR  NA  PE  NC
WR DS:131 = 0000
063A:010E 891E3101      MOV     [FILEOFFSET],BX

EXAMPLE 3:
AX=0000  BX=0000  CX=0100  DX=0001  SP=FFFB  SI=0000  DI=0000
DS=063A  ES=063A  SS=063A  CS=063A  IP=01AD  FL=0246  NV  UP  DI  PL  ZR  NA  PE  NC
                 P565:
063A:01AD BF2F01        MOV     DI,012F                       ; FILESEGMENT
```

Figure 9-3. Sample Displays of Registers and Flags

the **`<flag>`** command parameter in Section 8-3 for an explanation of the flag mnemonics. The last line in each of the examples in Figure 9-3 shows the disassembled instruction. The address of the instruction (CS:IP) is shown at the left, followed by the bytes that make up the instruction, and the instruction itself.

In **Example 2** shown in Figure 9-3, the third line shows that the current instruction performs a write to the word at DS:0131 and that the current value of the word is zero. If the instruction were to read memory, line three would also show that information.

The evaluation of the effective address of memory reads and writes shows the effect of any and all memory accesses before the execution of the instruction. The effective address calculations and displays for real-mode instructions are supported (up to the 80286 and the 80287), with two exceptions. The stack shown as affected by the ENTER instruction is limited to a single PUSH BP and does not include the PUSH that is done for each nesting level. The FRSTOR and

FSAVE instructions do not show the 94 bytes they read and write.

In **Example 3** shown in Figure 9-3, the third line shows P565, the name of the current address from the symbol table. This line is present only when CS:IP exactly matches an entry in the symbol table.

If the current instruction is a conditional jump (see **Example 1** in Figure 9-3), the jump is evaluated based on the current flag settings as 'jump' or 'no jump', meaning that the jump will or will not be taken, respectively. If an instruction references a byte value and the data byte is from 20H to 7FH, the ASCII equivalent of the byte is shown at the end of the line as a comment, in quotes. Illegal instructions are shown as `???`.

If an address referenced by an instruction is found in the symbol table, the symbol name is substituted for the offset (see **Example 2** in Figure 9-3). If an ambiguous reference to an address is made, the symbol name is shown at the end of the disassembled instruction as a comment (see **Example 3** in Figure 9-3). This indicates that the symbol may or may not have been used in the original instruction. Ambiguous references are generated by a move of an offset to a register, such as MOV DI,OFFSET FILESEGMENT.

An address must match exactly for the symbol to be found. The current value of the segment used by the instruction (explicit or implicit) must match the segment in the symbol table. The offset used by the instruction must also match the offset in the symbol table.

To modify a register, enter **`R <register>`**. Periscope displays the current value of the register, followed by a colon. If you enter a one- to four-digit hex number or another register name and press return, the register is changed. If you press return without entering a number, the register is not changed. The valid 16-bit register names are **`AX`**, **`BX`**, **`CX`**, **`DX`**, **`SP`**, **`BP`**, **`SI`**, **`DI`**, **`DS`**, **`ES`**, **`FS`**, **`GS`**, **`SS`**, **`CS`**, **`IP`**, and **`FL`** (flags). The valid 8-bit register names are **`AH`**, **`AL`**, **`BH`**, **`BL`**, **`CH`**, **`CL`**, **`DH`**, and **`DL`**.

To modify a flag, enter **`R FL`**. Periscope displays the current values of the flags (see the `<flag>` parameter in Section 8.3) followed by a hyphen. To change the flags, enter the desired mnemonics and press return. If you press return without entering any flag mnemonics, no flags are changed. You may enter the flags in any order, in upper or lower case, and with or without spaces between the entries. The hex value of the flag register is displayed following the mnemonic `FL`.

Changes to registers and flags are highlighted, even for the software traceback display. This makes it easy to see what changes were made since the last time Periscope's screen was displayed.

When you use windows, the register display can be put into a vertical window. This format requires 18 lines to display all the registers and flags. If fewer lines are available, the register display is truncated. When you use the vertical window, the effective address is displayed in the separator line following the Watch window. You may toggle this mode on and off by pressing **Alt-R**. You can also set vertical registers by using `R:1` in the windows specification. See the `/W` command for more information.

If you're using an 80386 or later system, a vertical display of the 80386 registers is also available. Press **Alt-3** to toggle 386 registers on and off. You can also use `R:3` in the windows specification. Note that the high part of the 32-bit registers and the FS and GS registers are never highlighted when changed.

Periscope/32 users: The register display supports 32-bit registers in two formats: a horizontal four-line display and a vertical display, both of which display the full 32-bit registers and flags.

Periscope/32 restricts changes to the segment registers while the processor is in protect mode. The new value cannot go beyond the limit of the GDT or LDT and must indicate a valid selector.

Periscope/32 indicates that the size of segment register is 16 bits with a minus sign (`DS-1234`) and that its size is 32 bits with an equals sign (**DS=0030**).

You can assign values of up to 32 bits to the **R0** through **R9** register variables.

The flags display (**RF**, **RFL**) shows more mnemonics and the state of the processor (protect, real or V86):
VM/SM—Virtual mode/not virtual mode
RF/NR—Resume flag/not resume flag
NT/NN—Nested task/not nested task
Ix—IOPL level, where **x** is from 0 to 3

Periscope/32 displays the CPU state at the end of the flags display as **PMx**, **V86**, or **REAL**, where **x** is the ring (0 to 3).

Examples:

R displays all registers and flags, the effective address for reads and/or writes, and disassembles the current instruction. It also forces re-display of the Disassembly window if one is used.

R AX displays the current value of register AX and prompts you for the new value. Press return to leave the register unchanged, or enter a one- to four-digit hex number and press return to change the register.

R AX CX is a one-line method of changing the value of register AX to the current value of register CX.

R+ moves the instruction pointer to the beginning of the next instruction.

R=_MAIN sets CS:IP to the symbol _MAIN.

R FL displays the current flags, followed by a hyphen. If you want to change the zero flag from NZ to ZR, enter **ZR** and press return. You can also enter **R FL ZR**.

R0=AX saves the current value of register AX to user register R0.

R0 displays the current value of user register R0.

R? displays the current value of all non-zero user registers.

Command (Periscope/32 only): Register Extended FLags (R EFL)

Syntax: **R EFL**

Description: Use this command to enter the numeric value for the extended flags dword. It is an alternative to entering the flag mnemonics with the **RF** or **RFL** command.

Command: Register Compare (RC)

Syntax: **RC**

Description: Use this command to compare the current value of the registers with their previously-saved values.

RC is available only after you've used an **RS** command to save the registers. After you've used an **RR** command, this command is disabled until you've used another **RS** command.

Assuming that register CX has been changed, the display of the **RC** command might look like that shown in Figure 9-4, where the first line shows the register names, the second line shows the values of the registers at **RS** time, and the third line shows the changed registers at **RC** time. Double quote marks indicate unchanged values.

```
      AX   BX   CX   DX   SP   BP   SI   DI   DS   ES   SS   CS   IP   FL
@RS: 0000 0000 0000 0000 FFFE 0000 0000 0000 1382 1382 1382 1382 0100 0246
@RC:  ''   ''   ''   ''  FFF8  ''   ''   ''   ''   ''   ''  026A 143F  ''
```

Figure 9-4. Display of RC Command

Periscope/32 users: This command shows only the changed registers in a list format.

Example:

Assuming the registers were previously saved using the `RS` command, enter `RC` to compare all registers with their saved values.

Command: Register Restore (RR)

Syntax: `RR`

Description: Use this command to restore the registers to a previously-saved state.

You can use this command only after you've used an `RS` command to save the registers. After you've performed a Register Restore, `RR` is disabled until you've used `RS` again to save the registers.

Example:

Assume the registers were previously saved using the `RS` command. Enter `RR` to restore all registers to their values saved by the `RS` command.

Command: Register Save (RS)

Syntax: `RS`

Description: Use this command to save the registers for later comparison or restoration.

The Register Save command saves the current state of the machine's registers and flags in case you need to restore the registers to that state at some later point. For example, assume you're debugging a subroutine. In many situations, it is very convenient to save the machine's registers and then start debugging the subroutine. If you discover a problem, you can then restart the subroutine by restoring the registers from their saved values.

To use the Register Save command, enter `RS` at Periscope's prompt. Later, you can restore the registers to their saved state with the `RR` command. This command does not restore any data areas. To prevent accidental restoration of the registers, the `RS` command sets a flag that is cleared by the `RR` command. If this flag is not set, `RR` generates an error.

Example:

Enter `RS` to save the machine registers. The `RR` command may then be used to restore the saved register values, and the `RC` command may be used to compare register values with the saved values.

Command (Periscope/32 only): display eXtended Registers (RX)

Syntax: `RX`

Description: Use this command to display the extended registers: GDTR, IDTR, LDT, TR, CRx, DRx, and exception error code.

Command: Search (S)

Syntax: `S [!x] <range> <list>`

Description: Use this command to search memory for a byte/string pattern.

It searches the block of memory specified by the `<range>` for the pattern specified by the `<list>`. If it finds a match, it displays the starting address and symbol name, if any, and continues the search for matches at the next byte. If it does not find any matches, it displays nothing. If you do not specify a segment in the address, it uses the current data segment.

A wildcard search capability is available using the optional

!x parameter. You can use the wildcard character **x** in the list parameter to indicate a wildcard field. For example, to search for all occurrences of the byte string **EB**H, a wildcard, and 90H (short jumps followed by a NOP), use `S !? CS:100 LCX EB ? 90`. Avoid using hex characters for the wildcard. Use **?** or a period when possible to minimize confusion. Note that the wildcard character need not be in quotes, but any other non-hex field does.

Examples:

`S CS:IP L 200 CD 21` searches memory from the current instruction (CS:IP) for 200H bytes for the pattern `CD 21`. Any matches are displayed in segment:offset format.

`S PRINTLINE L 50 C "Page"` searches 50H bytes starting at the address of the symbol PRINTLINE for the byte `0CH` followed by the string `Page`.

Command: Search for Address reference (SA)

Syntax: `SA <range> < address>`

Description: Use this command to search memory for references to a specified address. It does not show any source code, just disassembled instructions.

This command can be thought of as a disassembly that only shows instructions that reference an address of interest. To use it, specify a **<range>** to be searched and the **<address>** to be searched for. If you do not use a symbol name for the address, be sure to specify the segment register. For example, if you're searching for a procedure reference, specify CS.

You can use this command to find JMPs and CALLs to a procedure or to find locations in your program where a data variable is accessed. Any instruction that references the specified address is displayed.

References to stack data variables are not shown by this command.

Examples:

`SA CS:100 L 200 CONVERT` searches from CS:100 for 200H bytes for any references to the address represented by the symbol CONVERT.

`SA PSTART PEND DS:0` searches from the address represented by PSTART through the address represented by PEND for references to DS:0.

Command: Search for Calls (SC)

Syntax: `SC [<byte>]`

Description: Use this command to search the stack for references to CALLed subroutines and software INTerrupts. If it finds a match, it displays the disassembly of the CALL or INT. This technique can help you determine the calling sequence used by a program and to unravel nested code. `SC` is similar to `SR`. `SR` analyzes the stack looking outward, while `SC` analyzes the stack looking inward. Use the optional `<byte>` field to override the default length of 10H stack entries.

The results are usually accurate, but cannot be guaranteed. For example, if a PUSH instruction saves a value on the stack that is the same as the address of the instruction after a CALL instruction, a false hit will occur.

This command does not interpret hardware interrupts since there is no interrupt in the instruction stream to indicate what happened.

Periscope/32 users: The optional argument is the maximum number of lines to display. The maximum number of stack entries to search is 16 times that number.

Example:

`SC` searches the stack for CALLS and INTS. If it finds any, it displays the disassembled instruction. Note that the most recent item is displayed first.

Command: Search then Display (SD)

Syntax: `SD <range> <list>`

Description: Use this command to search memory for a byte/string pattern and then display the matches. This command is the same as the Search command, except that it displays any matches in byte format. If you're using a Data window, it displays the matching address in the active window. After you press a key, it continues the search. See the Search command for more information.

Example:

`SD CS:0 FFFF "Hello"` searches memory from CS:0 to CS:FFFF for the string `Hello`. If any matches are found, they are displayed in byte format.

Command: Search for Return address (SR)

Syntax: `SR [<byte>]`

Description: Use this command to search the stack for return addresses. It examines each stack item to see if it contains an address after a CALL or INT instruction.

This command is similar to the `SC` command. `SR` analyzes the stack looking outward, while `SC` analyzes the stack looking inward. Use the optional `<byte>` field to override the default length of 10H stack entries.

When `SR` finds an address after a CALL or INT instruction, it displays the address of the instruction and the offset to the nearest symbol after the CALL or INT. This technique can help you determine the calling sequence a program uses and to unravel nested code.

The results are usually accurate, but cannot be guaranteed.

For example, if a PUSH instruction saves a value on the stack that is the same as the address of the instruction after a CALL instruction, a false hit will occur.

Some programs may manipulate the stack in ways that cause this command to fail.

This command does not interpret hardware interrupts since there is no interrupt in the instruction stream to indicate what happened.

Periscope/32 users: The optional argument is the maximum number of lines to display. The maximum number of stack entries to search is 16 times that number.

Example:

`SR` searches the stack for return addresses. If any are found, the return address and the offset from the next lower symbol are displayed. The address shown is the instruction following a CALL (near or far) or an INT. Note that the most recent item is displayed first.

Command: Search for Unassembly match (SU)

Syntax: `SU <range> <list>`

Description: Use this command to search memory for instructions that match a pattern.

Think of this command as a disassembly that only shows instructions that match a specified pattern. To use it, specify a `<range>` that is to be searched and the pattern that is to be searched for. For example, to find all MOVSB instructions, enter `"MOVSB"` (in quotes) as the `<list>` argument.

To find all occurrences of MOV SP, enter `"MOV SP"`. The command converts lower case input to upper case before it starts the search. It starts in the mnemonic field and goes through the end of the argument field. It ignores any blanks in the `<list>`, so `"MOVSP"` is the same as `"MOV SP"`.

11 occurrences of MOV SP, enter `"MOV SP"`. The command converts lower case input to upper case before it starts the search. It starts in the mnemonic field and goes through the end of the argument field. It ignores any blanks in the `<list>`, so `"MOVSP"` is the same as `"MOV SP"`.

This command does not find source-code lines or procedure labels, just disassembled instructions.

Examples:

`SU CS:100 L 200 "MOV SS"` searches from CS:100 for 200H bytes for any instructions that contain `MOV SS`.

`SU PSTART PEND "POP"` searches from the address represented by PSTART through the address represented by PEND for POP instructions.

9

Command: Trace (T)

Syntax: `T [<number>]`

Description: Use this command to trace through the current program one instruction at a time (i.e., to single step).

If you do not enter the optional `<number>`, this command executes one instruction of the program you're debugging then returns control to Periscope. If you enter a `<number>`, it executes that number of instructions before returning to Periscope. After each trace, it displays the registers, effective address, and current instruction.

If you're using a single-monitor system to trace code that does not do any screen writes, you may want to use **Alt-N** to turn the screen swap off. This eliminates the annoying flash caused by the program's screen being restored in case the instruction updates the screen.

Unlike the Go command, you can use the Trace command to trace through ROM, since it works by changing the trap flag and not by modifying the code being traced.

Examples:

`T` traces the execution of a single instruction.

`T 3` traces the execution of the next three instructions.

`T CX` traces the execution as many times as indicated by the current value of the CX register. If CX is currently 100H and the next instruction changes it to zero, the trace will still be performed 100H times.

Command: Trace Back / Trace Registers / Trace Unasm (TB/TR/TU)

Syntax: `TB | TR | TU [*] [![#<number>]]`

Description: Use these commands to view the software trace buffer. This circular buffer contains from zero to 2016 entries. You can change the default size (1KB, or 32 entries) with the `/B:nn` installation option. (Do not confuse this buffer with the real-time hardware trace buffer available with Periscope Model IV.)

Whenever you exit Periscope via any execution command, Periscope makes an entry in the software trace buffer. Each entry contains the machine registers and an ascending sequence number. When you display the buffer with the `TB` command, it shows, for each entry, the registers, the sequence number, and a symbolic disassembly of the instruction indicated by the saved CS:IP.

Periscope makes an entry in the software trace buffer each time it relinquishes control to your program, so watch out for possible discontinuities. If you're using the `T`, `GA`, or `GT` commands, there's no problem since these commands single-step. However, if you're using the `G` or `J` commands, Periscope doesn't "see" all instructions, so there will be discontinuities in the trace buffer. Note that the disassembly uses the current contents of memory at the saved CS:IP so the disassembly may be incorrect if you have changed the instructions. If the trace buffer is empty, Periscope ignores a **TB**, **TR** or **TU** command.

To display the entire trace buffer continuously (i.e., to dump it to a file or to the printer), use the exclamation point. Enter the optional **#<number>** to begin the dump at a specified sequence number.

Clear the trace buffer by entering **TB ***.

You enter a full-screen display mode whenever you enter this command without an ***** or **!**. The only way to exit this mode is to press **Esc**. The keys available are: **Home, End, Up, Dn, PgUp, PgDn,** and **Esc.** If you press any other key, Periscope displays the message **Press Esc to end full-screen mode**.

You can position the buffer by entering a **#<number>**, where **<number>** is the sequence number. If you enter a number that's too low, the first entry in the buffer is shown. Similarly, if you enter a number that's too high, the last entry in the buffer is shown.

```
AX=0000  BX=0000  CX=008B  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000    #0001
DS=15E6  ES=15E6  SS=15E6  CS=15E6  IP=0137  FL=0346 NV UP EI PL ZR NA PE NC
                START:
15E6:0137 E81700          CALL    GETMEM

AX=0000  BX=0000  CX=008B  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000    #0002
DS=15E6  ES=15E6  SS=15E6  CS=15E6  IP=0151  FL=0346 NV UP EI PL ZR NA PE NC
                GETMEM:
15E6:0151 B106            MOV     CL,06

AX=0000  BX=0000  CX=0006  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000    #0003
DS=15E6  ES=15E6  SS=15E6  CS=15E6  IP=0153  FL=0346 NV UP EI PL ZR NA PE NC
15E6:0153 BE0200          MOV     SI,0002
```

Figure 9-5. Software Trace Buffer Using the TB Command

Figure 9-5 shows three consecutive entries in the format used by the **TB** command.

The **TR** and **TU** commands are subsets of the **TB** command. **TR** displays just the registers and sequence number, and **TU** displays just the disassembled instruction. The same records shown in Figure 9-5 are shown in Figure 9-6 in the format displayed by the **TR** command. You can switch among any of the three formats by keying **B**, **R**, or **U**.

The disassembly mode in effect when you view the trace buffer determines the number of lines you see for each instruction. If Source mode (**US** command) is in effect, you'll see one line per instruction. If Assembly mode (**UA** command) is in effect, you'll see two lines per instruction. If Both mode (**UB** command) is in effect, you'll see three lines per instruction. You can switch disassembly modes (see the **U** command) to control the length of the screen display of the software trace buffer.

```
AX=0000  BX=0000  CX=008B  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000    #0001
DS=15E6  ES=15E6  SS=15E6  CS=15E6  IP=0137  FL=0346 NV UP EI PL ZR NA PE NC

AX=0000  BX=0000  CX=008B  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000    #0002
DS=15E6  ES=15E6  SS=15E6  CS=15E6  IP=0151  FL=0346 NV UP EI PL ZR NA PE NC

AX=0000  BX=0000  CX=0006  DX=0000  SP=FFFC  BP=0000  SI=0000  DI=0000    #0003
DS=15E6  ES=15E6  SS=15E6  CS=15E6  IP=0153  FL=0346 NV UP EI PL ZR NA PE NC
```

Figure 9-6. Software Trace Buffer Using the TR Command

Periscope/32 users: These commands use the horizontal 386 register display.

Due to the number of 32-bit registers Periscope/32 saves in each trace buffer entry, you'll see only 16 entries per 1KB of software trace buffer memory. The maximum number of trace buffer records is 16 times 63, or 1008, assuming you installed Periscope/32 with the **/B:3f** installation option.

Examples:

TB shows both the register and disassembly display of the software trace buffer.

TU shows just the disassembly display.

TB * clears the software trace buffer and resets the sequence number to zero.

Command: Trace all but Interrupts (TI)

Syntax: `TI`

Description: Use this command to trace (single step) all instructions except interrupts.

If a software interrupt is the current instruction, this command sets a temporary code breakpoint on the next instruction and executes the interrupt at full speed.

Example:

If the current instruction at offset 120H is INT 21H, the `TI` command sets a temporary code breakpoint at offset 122H and executes to that point at full speed.

9

Command: Trace Line (TL)

Syntax: `TL`

Description: Use this command to single step until the next source-code line is executed.

The `TL` command performs Trace commands until it finds the next source line. You can use this command to get back to your source code if you see no source code when Periscope comes up. Since this command single steps the code while checking for a source line, it can be slow. In many situations, the `JL` command will be faster. If finds no source line symbols, this command acts like the `T` command.

Example:

Assume the current instruction is in a library routine. Enter `TL` to single step your code until an instruction corresponding to a source line is executed.

Command: Unassemble memory (UA/UB/US)

Syntax: `U[A|B|S] [<range>]`

Description: Use this command to disassemble memory in the Assembly mode, source-and-assembly (Both) mode, or Source-only mode as set by the `UA`, `UB`, and `US` commands respectively.

Source is the default mode. If you set the source file buffer to zero (`/E:0` installation option) or if Periscope is unable to find line number symbols, Source-and-assembly (Both) is the default mode. If you enter `U`, the default is the last mode set.

The syntax for this command is very flexible. If you enter `U`, the disassembly starts where the last `U` command left off. The `G`, `J`, `R`, and `T` series commands reset the starting point to CS:IP. If you enter `U <number>`, Periscope presumes the `<number>` to be an offset and CS to be the segment. If you enter `U <number> <length>`, it presumes the `<number>` to be an offset, and CS to be the segment. If you're using a Disassembly window, the size of the window overrides the length parameter.

If you're using a Disassembly window and it is active, you can use the **PgUp**, **PgDn**, **PadPlus**, and **PadMinus** keys to move forward and backward through memory. When you use the up arrow or **PgUp** key to go backwards in the Disassembly window, the value of IP will not go below zero. When you disassemble memory without a Disassembly window, the default length for disassembly is 18H bytes or lines, depending on the mode used.

When you use `UA` mode (see Figure 9-7), Periscope displays all available symbols, including public and line symbols. (You can turn line symbol display off using the `/L` command.) The maximum backwards movement of the `UA` Disassembly window is 21H lines. Any larger Disassembly window will not move back a full page.

```
                  _MAIN:
1392:0000 55            PUSH    BP
1392:0001 8BEC          MOV     BP,SP
1392:0003 B81000        MOV     AX,0010
1392:0006 9A6C029A13    CALL    __CHKSTK
                  FTOC#10:
1392:000B C746F60000    MOV     WORD PTR [BP-0A],0000
                  FTOC#11:
1392:0010 C746F22C01    MOV     WORD PTR [BP-0E],012C
                  FTOC#12:
1392:0015 C746F41400    MOV     WORD PTR [BP-0C],0014
```

Figure 9-7. Disassembly of FTOC Program in UA Mode

Periscope supports disassembly of real and protect mode opcodes for CPUs from the 8088 to the 80486. To set 16-bit or 32-bit disassembly, see the **16** and **32** commands. If Periscope encounters an ambiguous reference to an address, it shows the symbol name at the end of the disassembled instruction as a comment. This indicates that the symbol may or may not have been used in the original instruction. Ambiguous references are generated by a move of an offset to a register, such as **MOV DI,OFFSET TMEMORY**.

```
 #7: {
        int lower, upper, step;
        float fahr, celsius;
#10:    lower = 0;        /* lower limit of temperature table */
#11:    upper = 300;      /* upper limit */
#12:    step = 20;        /* step size */
```

Figure 9-8. Disassembly of FTOC Program in US Mode

Periscope evaluates register memory references of the current instruction, taking into account the current values of machine registers. For example, for the instruction **MOV AX, [BP-10]**, it shows the symbol name STEP as a comment when the corresponding local symbol is used and is in the scope of the current procedure. Also, it evaluates references such as **MOV [BX+SI-20],AX**, showing any symbols it finds.

When you use **US** mode, you'll see source and line numbers, but no other symbols unless the source is not available.

If the source file date/time stamp is more recent than the program's date/time stamp, however, Periscope displays the warning message, `Source file more recent than program`.

Use the `US` command to turn on source-level debugging (see Figure 9-8). This mode shows a minimum of assembly code, i.e., it shows assembly code until the first source line is found and then shows just the source code. You can use the **PadPlus, PadMinus, PageUp**, and **PageDown** keys to move through your source file.

The source-line naming convention is composed of a module name of up to eight characters, an asterisk, and the line number from 1 to 65535. For example, line 10 of FTOC is referred to as `FTOC#10`. While 'in' the module, the shorthand form `#10` may be used to reference line 10.

Periscope always displays the line number of the first source line for reference in a Disassembly window. If the line number is a valid symbol, it appears in the format `#nnnnn:`, where `nnnnn` is the line number. Otherwise, it appears as `(nnnnn)`. When the current instruction is a conditional jump, Periscope displays the `jump` or `no jump` comment in the separator line of the Disassembly window.

Periscope shows the module name in the separator line of a Disassembly window. It shows the module name as the first line of the disassembly if you're not using a Disassembly window. When possible, use a Disassembly window so you can scroll through the source file.

Periscope should never prompt you for a file name for source-level debugging. If you are ever prompted for a source file name, press **Alt-C** to see the name Periscope tried to use. Edit the file name as needed and press return. To fix the name permanently, change your make file to specify the full path name for your source files. If you press return without entering a file name, source disassembly is disabled. To display the file name prompt again, enter `UA` and then enter `US`.

To display source code, these conditions must be met:

- DOS must not be busy.
- A source file buffer must be available (if you installed

Periscope with **/E:0**, the **US** command is not available).

- Line symbols must be available for Periscope to be able to associate an instruction with a source-code line.
- You must be disassembling memory at a source line symbol.

To test for these conditions, see the **/L** command.

```
                _MAIN:
   #7: {
1392:0000 55             PUSH    BP
1392:0001 8BEC           MOV     BP,SP
1392:0003 B81000         MOV     AX,0010
1392:0006 9A6C029A13     CALL    __CHKSTK
   #10:     lower = 0;          /* lower limit of temperature table */
1392:000B C746F60000     MOV     WORD PTR [BP-0A],0000
   #11:     upper = 300;        /* upper limit */
1392:0010 C746F22C01     MOV     WORD PTR [BP-0E],012C
   #12:     step = 20;          /* step size */
1392:0015 C746F41400     MOV     WORD PTR [BP-0C],0014
```

Figure 9-9. Disassembly of FTOC Program in UB Mode

When you use **UB** mode (see Figure 9-9), Periscope displays any available source code and suppresses line symbols. This mode shows the source code, followed by the assembly code generated by the source code.

Periscope/32 users: The disassembly shows an address of the form xxxx:yyyy,yyyy. The maximum backward movement is 18 lines.

Whenever it encounters a new selector, the disassembly defaults to 16 or 32 bits depending on the mode of the selector. You can override this with the **16** or **32** command. Periscope/32 remembers the settings for up to eight selectors.

Normally, Periscope/32 uses the DOS memory allocation chain to display the owner of the current disassembly address at the bottom of the Disassembly window. It does not do this in remote mode.

Examples:

U FTOC# disassembles memory in the current or default mode, starting at the first source line in FTOC. Use this technique to find the first source line in a module if you're not sure what the line number is.

`U NEWPAGE` disassembles memory in the current or default mode, starting at the symbol NEWPAGE. The default length of 18H bytes is used.

`UA FTOC#10 L 1` disassembles memory in symbolic Assembly mode for one instruction starting at the symbol FTOC#10.

`UB FTOC#10` disassembles memory in source-and-assembly (Both) mode starting at line 10 in FTOC.

`UB` disassembles memory in source-and-assembly (Both) mode starting where the last `U` command left off. If the `G`, `J`, `R` or `T` commands were used, the disassembly starts at CS:IP.

`US FTOC#10` disassembles memory in Source-only mode starting at line 10 in FTOC.

Command: View file (V)

Syntax: `V <file>`

Description: Use this command to view a text file from within Periscope. Don't try to use it to display a file that's not in ASCII!

The `<file>` is any legal file name, including drive, path, file, and extension. To use this command, DOS must not be busy. A source file buffer must be available. If you installed Periscope with `/E:0`, this command is not available.

Periscope displays the file in the non-windowed area of the screen, unless you have set up a View window. It displays the line numbers in the lower left-hand corner of the screen. Use **PgUp** and **PgDn** to page up and down through the file. Use the up and down arrow keys to move up or down one line at a time. Use **Home** and **End** to move to the start and end of the file. Use the right arrow key to display beyond column 80 and the left arrow key or **Enter** to get back. When you're finished viewing the file, press **Esc** to return to Periscope's prompt. If View finds an EOF character (1AH) in the file, it treats it as the end of the file.

A simple string search is available. Enter a forward slash and the text to be located. The search begins on the second line from the top of the screen. The search is not case sensitive. When found, the string is displayed at the top of the screen. To repeat a search, enter a forward slash and press **Alt-D**. If no match is found, View displays the last line in the file.

To position to a specific line, enter a pound sign (#), the decimal line number (`1` to `65535`), and press **Enter**. View displays the line.

If you're using a View window, the window is static after you press **Esc** to return to the Periscope prompt. If you change the windows or clear the screen, you'll lose the View window. Use the View command again to re-display the file.

Example:

`V C:PS.DEF` displays the file PS.DEF. Use the **PgUp**, **PgDn**, **Up**, **Down**, **Left**, **Right**, **Home**, and **End** keys to move through the file. When done, press **Esc** to return to the Periscope prompt.

Command: View Source file (VS)

Syntax: `VS`

Description: Use this command to view the current source file. It has no arguments and is available only when source-level debugging has been turned on using the `UB` or `US` commands. This command functions exactly like the View command described above.

Example:

`VS` displays the current source file. When finished viewing the file, press **Esc** to return to the Periscope prompt.

Command: Watch (W)

Syntax: `W*` or `W<byte> *` or `W[<byte>] [<format>] [<pointer>] <range>`

Description: Use this command to watch memory locations and I/O ports.

Use the Watch command in conjunction with a Watch window, which may be up to eight lines long. Each line shows the watch variable number, the format type in parentheses, the symbol name or address, and the value. The display of each watch variable is limited to one line on the screen. The formats available are the same as with the Display command, plus `P` to watch an I/O port.

For example, to display the variable STEP in integer format, enter `W I STEP` or `W0 I STEP L2`. When you do not specify a `<byte>`, Watch uses the next available value. To watch an I/O port, enter `W P <port>`. To clear the entire Watch window, enter `W*`. To clear an individual watch variable, use `Wn *`, where `n` is from `0` to `7`. To display all watch settings, use `W?`

The Watch command supports the use of pointers. To use this feature, enter a bracket (`[`) or a brace (`{`) after the display format and before the address. If the bracket/brace immediately follows the `<format>` (no space!), Watch dynamically evaluates the pointer each time it displays the watch item.

If the bracket/brace does not immediately follow the `<format>` (one or more spaces), it evaluates the pointer when you enter the command but remains static after that time. For example, for `W B{FOO`, Watch dynamically evaluates the far pointer FOO, whereas for `W B {FOO`, it only evaluates FOO when you enter the command.

Examples:

`W0 I AMOUNT` displays the variable AMOUNT in integer format.

`W3 *` clears the watch variable number 3.

`W P 310` displays the value read from port 310H.

`W*` clears all watch variables.

`W?` displays all watch settings.

Command: Write Absolute disk sectors (WA)

Syntax: `WA <address> <drive> <sectors>`

Description: Use this command to write memory to absolute disk sectors.

The segment defaults to CS if you do not specify a segment in the `<address>`. The `<drive>` is either a single-digit number indicating the disk drive (`0`=A, `1`=B, etc.) or `A:`, `B:`, etc. The `<sectors>` parameter is the starting sector number and the number of sectors to be written. The maximum number of sectors that can be written in one operation is 80H, which is 64K bytes.

To use this command, DOS must not be busy. This command uses DOS interrupt 26H. See the DOS manual for information on the numbering of the absolute disk sectors.

When using this command, be very careful. An absolute disk write can very easily destroy the file allocation table (FAT) or the disk directory!

Usually, you will want to perform a Load Absolute, change a few bytes of memory, and then perform a Write Absolute of the data back to disk. If this is the case, be sure that the parameters you use with the Load and Write commands are the same.

Periscope/32 users: This command is not supported in remote mode.

Examples:

`WA DS:100 A: 10 20` writes data from memory starting at DS:100 to drive A:, starting at sector number 10H for

20H sectors.

`WA 100 B: 0 4` writes data from memory starting at CS:100 to drive B:, starting at sector 0 for 4 sectors.

Command: Write Batch file (WB)

Syntax: `WB <file>`

Description: Use this command to write a 'batch' file that contains Periscope commands to save the current settings for the windows and the hardware and software breakpoints. To restore the state in a later debugging session, use the `LB` command to load the file. Unless you specify an extension, `WB` uses the default .PSB.

Example:

Enter `WB SAVE` to write the breakpoint and window settings to the file SAVE.PSB. Later, use `LB SAVE` to reload the breakpoint and window settings.

Command: Write alias and record Definitions (WD)

Syntax: `WD <file>`

Description: Use this command to write the current alias definitions and record definitions to a .PSD file. The `<file>` field is the name of the file to be written, with an extension of .PSD. To use this command, DOS must not be busy.

Example:

`WD FTOC` writes the current alias and record definitions to the file FTOC.PSD.

Command: Write File to disk (WF)

Syntax: `WF [<address>]`

Description: Use this command to write a file from memory to disk. Before you do, however, you must specify a file name with the Name command.

The optional `<address>` specifies where the memory image of the file begins. If you do not specify an address, `WF` uses CS:100. To use this command, DOS must not be busy.

You can use this command to write any type of file to disk. Before you write the file, be sure that BX and CX indicate the size of the file in bytes. Do not attempt to write an .EXE file unless it was loaded with the `LF` command. An .EXE file loaded by RUN is missing its header and is executable only at its current address.

Periscope/32 users: This command is not supported in remote mode.

Examples:

`WF DS:1000` writes the file defined by a Name command from memory to disk starting at DS:1000.

`WF` writes the file defined by a Name command from memory to disk starting at CS:100.

Command: Write Symbols to disk (WS)

Syntax: `WS <segment> <file>`

Description: Use this command to write a Periscope symbol (.PSS) file using the current symbol table.

The `<segment>` contains the relocation factor that is subtracted from the current symbol segment before the file is written. For .COM files, this is the value of the PSP segment or CS. For .EXE files, this is the value of the PSP segment plus 10H. The `<file>` is the path and file name of a .PSS file. To use this command, DOS must not be busy.

You cannot use this command to write a .MAP file. It only writes .PSS files. It does not change the symbol tables.

If an error occurs when writing the symbol file, the symbols may be left with the relocation factor subtracted. If this happens, you can recover the symbols using `WS 0 <file>` to write the symbols without further relocation. Then enter `LS *` to clear the symbol table, followed by `LS <segment> <file>` to restore the symbol table.

Examples:

`WS CS SAMPLE` subtracts the current value of CS from the symbol's segments and writes the file SAMPLE.

`WS 0 C:TEST` subtracts zero from the symbol's segments and writes the file C:TEST.PSS.

Command: translate (X)

Syntax: `X|XH <number>` or `XA <address>` or `XD <decimal number>`

Description: Use `X` or `XH` to translate a one- to four-digit hexadecimal `<number>` to its decimal, octal, binary, and ASCII equivalents. You may also use it to perform in-line arithmetic.

Use `XA <address>` to translate an address (segment and offset) into its equivalent five-byte absolute address. The absolute address is calculated by multiplying the segment by 10H and adding the offset to the result.

Use `XD <decimal number>` to translate a one- to five-digit decimal number to its hexadecimal, octal, binary, and ASCII equivalents. The number must be from `0` to `65535`. The number may not have any punctuation, such as commas or periods. You can translate numbers larger than 65535, but the high order part is lost.

Periscope/32 users: This command supports hex numbers of up to eight digits. It supports decimal numbers up to 4,294,967,295. It does not support octal numbers.

In remote mode, **XA** adds the offset argument to the base of the selector argument, then returns the result. Otherwise it uses the normal calculation method, (segment*16)+offset.

Examples:

X 5051 displays:

```
5051h    20561d    050121o    0101 0000 0101 0001b    PQ
```

XA 1234:5678 displays:

```
179B8
```

XD 20561 displays:

```
5051h    20561d    050121o    0101 0000 0101 0001b    PQ
```

Command: Use 16 or 32-bit disassembly (16/32)

Syntax: `16` or `32`

Description: `16` sets 16-bit disassembly. `32` sets 32-bit disassembly for use on 80386 or later CPUs.

There is currently no effective address support for 32-bit instructions.

Examples:

`16` switches to 16-bit disassembly.

`32` switches to 32-bit disassembly.

Command: Option ditto (copy Periscope's screen)

Syntax: `/"`

Description: Use this command to copy Periscope's screen onto the other display. It is usable only when you're using a dual-monitor system with one color display and one monochrome display using an MDA.

Example:

`/"` copies Periscope's screen to the other display.

Command: Options 1 and 2 (switch symbol tables)

Syntax: `/1` or `/2`

Description: Use this command to switch to an alternate symbol table.

To use it, you must specify two symbol tables when you install Periscope, using two `/T:xxx` installation options. The `/1` command makes the first symbol table active and the `/2` command makes the second symbol table active.

You can use the `LS` command to manually load the current symbol table or you can use RUN to load the desired symbol table. RUN defaults to loading symbol table 1, but loads table 2 when you use `RUN /2 <file>`.

When debugging a program that uses overlays, you must load the symbol table that contains overlays last.

Examples:

`/1` makes the first symbol table active.

`/2` makes the second symbol table active, presuming that you specified two `/T` installation options when you installed Periscope.

Command (Periscope/32 only): Option 3 (enable/disable ring 3 debugging)

Syntax: `/3`

Description: Use this command to enable or disable ring 3 (1 through 3 for Windows) debugging by an application-level debugger, such as CodeView for Windows.

Command: Option 4 (toggle internal 486 cache)

Syntax: `/4`

Description: Use this command to enable or disable the cache that is an integral part of the 80486 chip.

Example:

`/4` changes the state of the internal 486 cache, i.e., if it was enabled, it will be disabled and vice versa.

Command: Option A (toggle DOS Access)

Syntax: `/A`

Description: Use this command to enable or disable DOS access by Periscope.

The `/A` command toggles Periscope's use of DOS. When used once, it turns Periscope's use of DOS off. When used again, it turns Periscope's use of DOS back on.

Example:

`/A` enables/disables DOS access by Periscope.

Command: Option C (display and set Colors)

Syntax: `/C [<byte>]`

Description: Use this command to display and set the screen colors.

The optional `<byte>` is the color attribute. If you do not enter a number, this command displays the colors for 00 to 7FH. If you enter a number, it sets the screen color as in the `/C` installation option.

Examples:

`/C` displays the available screen colors.

`/C 17` sets the Periscope screen color to white on blue.

Command: Option D (Data window select)

Syntax: `/D [<byte>]`

Description: Use this command to select the active Data window when you're using more than one Data window.

The active window is the one modified by display commands. The inactive window(s) display memory in the same format as when they were last active. If you're using only one Data window, this command has no effect. If you do not enter a number in the `<byte>` parameter, it makes the next Data window active (as indicated by the up arrow after the window number). If you enter a number that corresponds to a Data window (`0-3`), it makes that window active.

Examples:

`/D` makes the next Data window active. If there are three Data windows, the first use of this command makes the second window active. The second use of this command makes the third window active. The third use of this command makes the first window active, etc.

`/D 3` makes the last Data window active, assuming four

Data windows are in use.

Command: Option E (Echo screen to a file)

Syntax: `/E [<file>]`

Description: Use this command to echo Periscope's screen output to a `<file>`.

This command causes all non-windowed output to be written to a disk file at the same time it is being written to the screen. To begin this mode, enter `/E` followed by a file name and a carriage return. While active, the command prompt shows `/E` to remind you that echo mode is on. To end echo mode, enter `/E` with no file name. The usual rules about DOS availability from within Periscope apply.

Examples:

`/E D:OUTPUT` starts echo mode, using the file D:OUTPUT. Until you use another `/E` command, Periscope's non-windowed screen output is written to this file.

`/E` ends echo mode, closing the file D:OUTPUT, and returns to the standard Periscope prompt.

Command: Option K (capture Keystrokes to a file)

Syntax: `/K [<file>]`

Description: Use this command to capture keystrokes to a `<file>`.

To start keystroke capture, enter `/K <file>`. Periscope then writes all keystrokes to the file until you enter the `/K` command again. To replay the captured keystrokes, enter the `LB` command. Note that the default file extension is .PSB. While this command is in use, Periscope's prompt becomes `/K>`. It does not capture keystrokes while the menu system is active.

Examples:

`/K TEST` starts capturing keystrokes to the file TEST.PSB.

`/K` turns off keystroke capture and closes the file TEST.PSB.

Command: Option L (display Line symbols and source debug status)

Syntax: `/L`

Description: Use this command to enable/disable the display of line-number symbols when you're using the `UA` mode, and to display the status of various items required for source-level debugging.

If it finds no line symbols, this command displays the message `No lines found`. If you see this message, correct your compile and link options to get line numbers in the symbol file. They are required for source-level debugging.

If you've set the source buffer size to zero with the `/E:0` installation option, this command displays the message `No buffer found`. If you see this message, reinstall Periscope with a source file buffer, since it is required for source-level debugging.

If DOS is busy, it displays the message `DOS busy`. Periscope uses DOS to access your source file, so DOS must not be busy for source-level debugging.

If it finds no problems, it displays the message `lines found`.

Example:

`/L` toggles the display of line-number symbols when you use the `UA` mode. It also displays the messages described above.

Command: Option N (Nearest symbols)

Syntax: `/N [<address>]`

Description: Use this command to search for the symbols nearest to the specified `<address>`.

This command displays up to three symbols on up to three lines: the next lower symbol (preceded by `>`), the equal symbol (preceded by `=`), and the next higher symbol (preceded by `<`).

It can help you get your bearings when you've interrupted an executing program by showing you the nearest symbols. If you do not enter an `<address>`, it assumes CS:IP. It displays the nearest symbol that is located lower in memory on the first line, the symbol for the specified address on the next line, and the nearest symbol that is located higher in memory on the next line. If it finds no lower, equal, or higher symbol, it displays nothing.

Periscope/32 users: This command searches for symbols only within the same selector.

Examples:

Assume three symbols X, Y, and Z located at 1000:100, 1000:200, and 1000:300 respectively.

`/N 1000:200` displays:

```
> X
= Y
< Z
```

`/N 1000:0` displays:

```
<X
```

Command: Option Q (Quiet)

Syntax: `/Q`

Description: Use this command to turn off Periscope's display output until the end of the current command line.

This is useful when you want to suppress Periscope's output to the screen. For example, when you're using the `/E` command to echo Periscope's output to a file, you can suppress the display using this command.

Example:

`/Q;U IP L1000` turns off the display output until the disassembly is complete.

Command: Option R (Remove symbol)

Syntax: `/R <symbol>`

Description: Use this command to remove public and line symbols (but not local symbols) from the symbol table. For instance, since Periscope evaluates symbol names before register names, a symbol named AX would disable references to register AX unless you used this command to remove the symbol.

Examples:

`/R AX` removes the symbol AX, leaving no conflict with the register named AX.

`/R FTOC#10` removes the symbol FTOC#10. Be careful when removing line-number symbols, since you cannot reenter them with the `ES` command.

Command: Option S (Segment change)

Syntax: `/S <segment> <segment>`

Description: Use this command to make global changes to the values of segments in the symbol table. It searches the entire symbol table for symbols, including local data symbols, with a segment that matches the first `<segment>`. When it finds a match, it changes the symbol's segment to the second `<segment>`. Use this command to adjust the segments of symbols when a program relocates its code or data areas.

Periscope/32 users: This command adjusts both the segment and offset. The syntax is `/S segment:address segment:address`, which provides maximum flexibility in relocating symbols for Windows.

Example:

`/S 1234 DS` changes the segment of all symbol table entries that are currently 1234 to the current value of DS.

Command: Option T (Trace interrupt table)

Syntax: `/T [?] [*] [#] [<byte>] [...]`

Description: Use this command to force tracing of interrupts when you use the `GT` command.

Some interrupt service routines turn the trap flag off when returning status information in the flag registers. If Periscope does not trace all the way through such routines when you're using the `GT` command, the program can get out of Periscope's control and begin executing at full speed.

The known troublesome interrupts are 13H, 15H, 16H, 1AH, 20H, 25H, 26H, 2FH, 40H, and 41H. When you initially install Periscope, it flags these interrupts for forced tracing. Using this command, you can change the interrupts that will be traced when you use `GT`.

The possible command arguments are `*`, `#`, `?`, and numbers from `0` to `FF` (always presumed hex). The `*` clears all traps and the `#` sets all traps. A `?` displays the current trace list. The `<byte>` hex number toggles the state for that interrupt from off to on or vice-versa.

Warnings:

- Interrupt 21H should be in the trace list whenever you use function 4BH (Exec).
- When an interrupt is not traced, Periscope becomes dormant until the second instruction after the INT XX instruction.
- If you use a `GT` command when the current instruction (CS:IP) is an interrupt, the interrupt is always traced.
- If you have problems with the `GT` command losing control, try using the `GA` command.

Examples:

`/T #` forces tracing of all interrupts.

`/T * 21` clears all interrupts and then forces tracing of Int 21H.

Command: Option U (User exit)

Syntax: `/U <byte> [<address>]`

Description: Use this command to perform user-written code from Periscope.

To use this command, you must install a program similar to USEREXIT.ASM and you must install Periscope with the `/I` option. The `<byte>` you enter after the `/U` command must be from `9` to `FF`H. It cannot be a literal. Periscope passes the number to the user-written program in register AH. It passes other information, including the optional `<address>` you enter on the command line. You may follow this parameter with additional free-form information, which means you cannot stack commands on the line following a `/U` command. When you enter a user exit command, Periscope displays an error if it cannot find the signature of the user exit code.

USEREXIT displays the status of the numeric processor. To use USEREXIT, load it before you install Periscope, then install Periscope with the `/I:60` installation option. From within Periscope, enter `/U 87` to display the status of the numeric processor.

The user exits from F0 to FFH are reserved for Periscope. See Section 7.10 for more information.

Example:

Assuming that a user-written interrupt handler has been installed using INT 60H and that Periscope was installed with the `/I:60` installation option, `/U 9` executes user exit number 9.

Command: Option W (Window setup)

Syntax: `/W [<token>] [[:<byte>] [.<color>]] [...]`, where `<token>` is `D`, `R`, `S`, `U`, `V`, or `W`; `<byte>` is the length in lines; and `<color>` is the color attribute; or `/W -<token>`; or `/W +<token>`; or `/W ?`

Description: Use this command to change Periscope's windows. Its use and syntax are identical to the `/W` installation option.

Periscope can window Data, Register, Stack, Unassembly, View and/or Watch information. Once you establish windows, Periscope displays the windowed data at a constant location on the screen and updates the windows after each command (it updates the View window only when you use the View command).

Indicate the type of data to be windowed with the `<token>`, which can be `D`, `R`, `S`, `U`, `V`, and/or `W`. The tokens are optional and may be in any order. If you omit a token, Periscope will not window the corresponding type of information. It displays the windows in the same order as you place the tokens on the input line, except for the Stack and vertical Register windows. These windows are always on the right-hand side of the screen.

The `<byte>` parameter defines the length (number of lines) of the horizontal window in hex (except for the `R` window). If you do not specify a length, Periscope displays a default number of lines. The maximum length for any one window and the total area that you can window is four lines less than the screen length, including a separator line following each window. When you specify a length, at least one space must follow the number. If you're using 43-line mode or 50-line mode, the windows can get quite large. Since Periscope regenerates the windows after each command, large windows can slow Periscope's response time. The default and minimum number of lines for each of the horizontal window types (vertical window lengths are determined by the total number of horizontally-windowed lines) are shown in Table 9-1.

	DEFAULT	MINIMUM
DATA	4	1
REGISTER	2	2
UNASM	4	1
VIEW	4	1
WATCH	4	1

Table 9-1. Periscope Window Lengths

You can individually set the colors of the windows, using a hex number from 1 to 7F in the `<color>` parameter. The numbering scheme is the same as that used by the `/C` installation option, and by the `/C` command. To set a Data window of five lines using color 1FH, use `/W D:5.1F`.

Data Window. The `D` token sets up a Data window to show data in any of the display formats except `DR` (Display Record). The window continues to show the same address until you use another Display command. When you use RUN to enter Periscope, it sets the display address to DS:100.

You may use up to four Data windows, with each window showing a different range and using a different display format. For example, if you want to set up three Data windows with lengths of 4 lines, 2 lines, and 6 lines, respectively, enter `/W D:4 D:2 D:6`. To change the active Data window, use the `/D` command. When the start address of a

Data window matches a symbol, Periscope displays the symbol name in the separator line at the end of the window. When the Data window is active, you may use the **PgDn**, **PgUp**, **PadPlus**, and **PadMinus** keys to move forward and backward through memory.

Register Window. Use the `R` or `R:2` tokens to set up a horizontal Register window to show register and flag information. The length is fixed at two lines. Periscope shows the effective address of any memory reads or writes in the separator line following this window.

Use the `R:1` and `R:3` tokens to establish a vertical window on the right-hand side of the screen to show standard and 80386 register displays, respectively. The displays require 18 lines of window space for the full register set and the flags. Periscope truncates the register displays if fewer lines are available. Other windows must exist for a vertical Register window to exist. You may toggle these vertical windows off and on using **Alt-R** (standard) and **Alt-3** (80386). When you use either of these windows, Periscope displays the effective address in the separator line following the Watch window.

Stack Window. Use the `S` token to set up a vertical window on the right-hand side of the screen to show the stack. The length of the Stack window is equal to the total number of lines contained in the other types of windows. A Stack window cannot exist without other windows. A chevron in the left margin of the window indicates the current value of the BP register. You can toggle the Stack window off and on using **Alt-S**. Read the stack from the upper right-hand corner of the screen downwards. If you choose not to use a Stack window, you can always view the stack using `DW SS:SP`.

Disassembly Window. Use the `U` token to set up a Disassembly window to show disassembled instructions. Periscope initially uses CS:IP as the address for the disassembly, and resets it to CS:IP each time you use a `G`, `J`, `R`, or `T` series command. You can disassemble any area of memory by using the `U` command with the desired address. Periscope highlights sticky code breakpoints when it shows them in the Disassembly window. It shows the current instruction in reverse colors. (Some window colors may cause the reverse color to be invisible. For example, color 7E on a monochrome EGA causes the reverse bar to be invisible.)

The maximum backwards movement of a Disassembly window is 21H lines. Any larger Disassembly windows will not move back a full page. If you 'lose' the reverse video bounce bar, use the `R` command to re-display it. When the Disassembly window is active, you may use the **PgDn**, **PgUp**, **PadPlus**, and **PadMinus** keys to move forward and backward through memory.

Periscope shows the current location in the separator line following the Disassembly window. It uses the DOS memory allocation blocks to get the actual program name when possible. Two caveats: the lookup is based on CS only and it will display `In RUN.COM` when in a program loaded by RUN unless you used **`RUN /T`** or **`RUN /X`**.

View Window. Use the `V` token to establish a View window to display a text file. The View command uses the space reserved by the window. See the description of the View command for more information.

Watch Window. Use the `W` parameter to set up a Watch window to display the contents of up to eight memory locations or I/O ports. Once you establish a Watch window, use the Watch command to specify the locations to be "watched". When the Watch window is active, you may use the **PadPlus** and **PadMinus** keys to scroll through the window. See the Watch command for more information.

Use **Ctrl-F9** to restore the original window settings in effect when you installed Periscope. Use **Ctrl-F10** to restore the most recent window settings.

Use the **`/W -<token>`** and **`/W +<token>`** commands to decrease or increase the length of a window by one line. They will not create or remove a window. To decrease the length of the Data window, use `/W -D`. To increase the length of the Watch window, use `/W +W`. When decreasing a window's length, the minimum resulting length is one line. When increasing a window's length, the maximum resulting length is determined by the maximum length for the screen.

The `/W ?` command displays the current window settings.

Periscope/32 users: The Stack window can be either 16 or 32 bits wide, depending on the width bit in the stack selector. Due to the wider stack, you should avoid the vertical register display. Also, any invalid memory addresses display as question marks.

The default window settings for Periscope/32 show a two-line Data window, followed by the horizontal registers, followed by a two-line Watch window and and an eight-line Disassembly window. The vertical Stack window is on the right side of the screen.

Periscope/32 interprets the Register window settings as follows:
`R:1` is interpreted and saved as `R:3` (vertical display)
`R:2` is interpreted and saved as `R:4` (horizontal display)

Examples:

`/W D:8.1F R` windows data in the first eight lines of the screen using color 1F (white on blue) followed by two lines of register information. A total of 12 lines are used for windows, including the two separator lines.

`/W SR U:7` windows register information in the first two lines of the screen, followed by seven lines of disassembly. A total of 11 lines are used for windows, including separator lines, so the specified Stack window is 11 lines long.

Command: Option X (eXit to DOS)

Syntax: `/X`

Description: Use this command to exit to DOS.

DOS must not be busy and memory must have been freed using DOS function call 4AH. To return to Periscope, enter `EXIT` at the DOS prompt. While at the DOS prompt, don't execute RUN, PS, or change the state of the system (e.g. change monitors).

Example:

`/X` exits Periscope and displays the DOS prompt, assuming memory has been freed and that DOS is not busy. When you're ready to return to Periscope, enter `EXIT` at the DOS prompt.❖

APPENDIX A

Messages

- **Informational Messages and Prompts**
- **Error Messages**

You'll find details on informational messages, prompts, warnings, and error messages displayed by Periscope in this chapter.

A.1 INFORMATIONAL MESSAGES AND PROMPTS

Programs in the Periscope package generate the following messages and prompts, listed in alphabetical order:

Break

Periscope displays this message when you press the Breakout Switch or otherwise generate an NMI while Periscope is already active.

Breakpoint cleared

Periscope displays this message when you re-enter a breakpoint.

DOS 3.10 or later required

Periscope needs MS-DOS or IBM-DOS 3.10 or later.

EOI issued for IRQ x

Periscope displays this message when it must issue an End of Interrupt for a hardware interrupt. The IRQ level designated by **x** indicates the interrupted activity: **0** = timer (INT 8); **1** = keyboard (INT 9); etc.

Trap xx

On an 80286 or later CPU, Periscope can intercept three types of exceptions, unexpected single-step (INT 1), illegal opcode (INT 6), and segment wraparound (INT 0DH). The **xx** is `01`, `06` or `0D`.

Grrr!

Periscope displays this message when the watchdog timer on an IBM PS/2 machine activates Periscope after detecting a hung system.

Parity error 1

Periscope displays this message when a motherboard parity error occurs. Make sure you've entered the correct response to the Periscope configuration option that asks if you have a PC/XT motherboard with an 80286 or later turbo card. If your configuration is correct, run your system diagnostics.

Parity error 2

Periscope generates this message when a parity error occurs in the expansion bus or when the you press the Break-out Switch.

> This message can also occur when you've installed two Periscope boards in your system. Try plugging the Break-out Switch into the other board.

Press Esc to end full-screen mode

Periscope displays this message when it's in full-screen mode and you enter an unexpected keystroke.

Source buffer too small

Periscope displays this message when the source file buffer is less than 1/32nd the size of the source file you're using. Increase the size of the buffer using the `/E` installation option for better performance.

Source file?

Periscope displays this prompt when it cannot find the source file. Press **Alt-C** to see the name Periscope tried to use. Correct the name and press return. Use the **MP** and **MX** aliases as needed to automate the source file access.

Source file more recent than program

Periscope displays this message when the date/time stamp on the source file is later than the date/time of the program file. An incorrect source line may be displayed if the two files do not match.

Warning: Timer interrupts (IRQ 0) are turned off via port 21H

Periscope has found that the system clock has been turned off. This is not a normal condition, so we are alerting you to it.

Warning: Keyboard interrupts (IRQ 1) are turned off via port 21H

Periscope has found that the keyboard has been turned off. This is not a normal condition, so we are alerting you to it.

A.2 ERROR MESSAGES

Periscope's error messages are numbered. Each program has been assigned a range of numbers for easy cross-reference. The error numbers and corresponding programs are:

- 01 through 39—resident portion of PS.COM
- 40 through 89—transient/installation portion of PS.COM
- 90 through 99 — RS.COM
- 100 through 129 — TS.COM
- 130 through 139 — SYMLOAD.COM
- 140 through 149 — INT.COM
- 150 through 159 — PSKEY.COM
- 160 through 169 — PSTEST.COM (see Model I Board installation addendum)
- 170 through 189 — RUN.COM
- 190 through 199 — SYSLOAD.SYS
- 200 through 229 — POPPS.COM (see PopUp Periscope manual)
- 230 through 239 — CONFIG.COM
- 240 through 249 — SETUP.COM
- 260 through 269 — RX.COM (see Periscope/Remote for DOS addendum)
- 270 through 279 — PSTERM.COM
- 400 through 414 — PS4TEST (see Model IV manual)

A list of the possible error messages and an explanation of each follows:

01—Invalid command

An unknown Periscope command was entered. Enter **?** to display the commands available.

02—Invalid/missing address

An address was expected, but was not found or was found to be invalid. You may enter the address as a symbol or a one- to four-digit segment, a colon, and a one- to four-digit offset. You may substitute a register name for the segment or offset.

03—Invalid/Missing segment

Some commands that modify memory require an explicit segment to reduce the chance of accidental memory modifications. Enter the segment as a number or register, or use a symbol for the address.

04—Invalid/missing length

The length argument was not found or was found to be invalid. If you enter `L nnnn`, the number `nnnn` must be greater than zero. If you enter it as an offset, the number must be greater than or equal to the first offset. If you enter a symbol, the symbol's segment must equal the first segment and the symbol's offset must be greater than or equal to the first offset.

05—Unexpected input

After completion of a command, an unexpected entry was found. If you want to enter multiple commands, place a semicolon between the commands.

06—Missing list

No list was found for the Fill or Search commands. These commands require a byte/string list.

07—Missing quote

The trailing single or double quote was not found for a list.

08—Invalid/missing operator

An expected argument was not found. Check the command syntax and try again.

09—Number is not decimal

A decimal number must be composed of the digits `0` through `9` with no punctuation.

10—Invalid/missing number

A required number was not found or was found to be invalid. The number must be from one to four hex digits or a valid register name. For some commands, the number is limited to two hex digits or the 8-bit registers. If part of a list, the number must be one or two digits; you cannot use a register name.

11—Invalid/missing register

The register name must be `AX`, `BX`, `CX`, `DX`, `SP`, `BP`, `SI`, `DI`, `DS`, `ES`, `FS`, `GS`, `SS`, `CS`, `IP` or `FL`; or one of the 8-bit registers `AH`, `AL`, `BH`, `BL`, `CH`, `CL`, `DH`, or `DL`. The register name may be in upper or lower case.

If this error occurs from the in-line assembler, it may mean that the register specified does not fit the instruction or is illegal (e.g., PUSH AL or POP CS).

12—Invalid flag

The valid flag names are `OV`, `NV`, `DN`, `UP`, `EI`, `DI`, `NG`, `PL`, `ZR`, `NZ`, `AC`, `NA`, `PE`, `PO`, `CY`, and `NC`, in upper or lower case.

13—Too many breakpoints

Too many breakpoints are set for the command. See the command you're using in Chapter 9 for the limits.

14—Wrong version in PSS file

The version number found in the .PSS file is not current. Regenerate the .PSS file using TS.

15—Address range is too big (Periscope/32 only)

The `HM` command and the hardware trace buffer `/A` command convert a contiguous virtual address range into one or more contiguous physical address ranges. The number of contiguous physical address ranges exceeds the capacity of the Model IV Board or the `/A` internal buffers. Try limiting the virtual address range.

16—Can't modify memory

An attempt was made to set a Code breakpoint in memory that could not be modified. The memory is not present, is read-only (ROM), or was not correctly updated with the CCH code needed for a Code breakpoint.

17—Second address/port less than first

The second address or port number is less than the first number. Enter an address or port number greater than or equal to the first. For commands requiring a range, the second offset was found to be less than the first offset. For example, the command `D 0: 100 80` is invalid, since 80 is less than 100.

18—Unknown symbol

An unknown symbol was referenced. The symbol may be preceded by a period and must be followed by a delimiter such as a space, carriage return, or semi-colon. The maximum symbol length is 32 characters. To display the symbol names and addresses from the symbol table, press **Alt-I**. To display the record definition table, press **Alt-E**.

19—Table full or invalid

The record or symbol table was found to have a logical error or is completely full. Try using an undefined record or symbol in a display statement. If this error occurs, the table has a logical error, otherwise the table is full.

If the symbol table is full, reload Periscope with a larger `/T` installation option. If the record definition table is full, reload Periscope with a larger `/R` installation option.

If the table is invalid, chances are good that it has been garbled. For the symbol table, use the `LS` command to clear and reload symbols. For the record definition table, use the `LD` command to clear and reload definitions.

20—DOS busy

The Load, Name, View, Unassemble Source, Write, Echo and DOS eXit commands use DOS function calls. Since DOS is not re-entrant, Periscope tests to be sure that DOS is available (not busy). It does this by checking a flag set by DOS. This flag must be zero and interrupts must be enabled for Periscope to allow DOS functions. If you receive this message, use the Go command to get back to your program's code or go to the next invocation of INT 28H, using `G`

{0:28*4}. Then try the command again.

21—Not enough memory

Insufficient memory is available to perform the Load or eXit to DOS command. When using the `/X` command, you must first release memory back to DOS.

22—Invalid drive

One of the drive names specified in the Name command is invalid. Register AL or AH is set to FFH if the first or second file name, respectively, had an invalid drive identifier.

23—Can't open file

Periscope was unable to open a file for input or output. If you're loading a file into memory, check the name you specified to the Name command. If you're writing a file, check that the filename is legal, the file is not a read-only file, and room exists in the directory for the file. This error can also occur if too many files are open.

24—Invalid window specification

The parameters specified with the `/W` option were found to be in error. The window specification may contain the tokens `D`, `R`, `S`, `U`, `V` and `W` in any order, in upper or lower case. If you enter a number, it must be of the form `X:nn`, where `X` is the token, and `nn` is the number of lines. For the `R` token, the number may be `1`, `2`, or `3`. A number must be followed by a space, a slash (indicating the start of another installation option), a carriage return, or a color setting in the format `.cc`, where `cc` is a number from `01` to `FF`. The total number of windowed lines, including a separator line for each window, must be 21 or less. If you're using 43-line or 50-line mode, the window sizes may be larger by the corresponding number of lines.

25—Read/write error

A fatal error occurred when reading or writing a file or absolute sectors. Check the disk and filename and retry the command. Also check to see if DOS has become busy while you're using the `/E` or `/K` command.

26—Function not available

This error indicates that the desired command is not available. It can occur under various conditions:

- when you use an **IC** or **IR** command and you haven't previously used an **IS** command to save the interrupt vectors
- when you use an **RC** or **RR** command and you haven't previously used an **RS** command to save the registers
- when you use a **TB**, **TR**, or **TU** command but installed Periscope with the **/B:0** option (no trace buffer)
- when you use a **UB**, **US** or **V** command but installed Periscope with the **/E:0** option (no source buffer)
- when you use an **LS** command but installed Periscope with a **/T:0** option (no symbols)
- when you use a **BU** or **/U** command but either did not install Periscope with a **/I:nn** option or the user exit program has been corrupted
- when you use an **LD** command but installed Periscope with the **/R:0** option (no definitions)
- when you use a **QR** command and DOS is busy or you did not use RUN to enter Periscope
- when you use a **/2** command but no second symbol table is present
- when you use the **/R** command to remove a local symbol
- when you use the **BD** command on a CPU prior to the 80386
- when you use the **/4** command on a CPU prior to the 80486

27—Unknown mnemonic

An unknown mnemonic was specified to the in-line assembler. The assembler knows the mnemonics for the 8086, 8087, 8088, 80186, 80286, and 80287 processors. For the 80286, only the real-mode opcodes are supported. Check the mnemonic and try again. Prefixes other than segment overrides must be on a separate line preceding the instruction they affect.

28 Byte, Word, Dword, Qword or Tenbyte pointer needed

An ambiguous instruction was specified to the in-line assembler. Some instructions, such as **MOV [SI],1**, require a width indicator of byte or word. You would enter the instruction as `MOV byte ptr [SI],1` or `MOV word ptr [SI],1`, respectively. 8087/80287 instructions may require a width indicator of `D`, `Q`, or `T` for double word, quad word, or ten byte respectively.

29—Invalid memory reference

An instruction that incorrectly references memory was specified to the in-line assembler. Check the register(s) and offset specified in the instruction to be sure that the memory reference is legal. For example, `MOV AX,[DX]` is not legal, but `MOV AX,[BX]` is legal.

30—Invalid argument(s)

There are too many or too few arguments for the mnemonic specified. Check the number of arguments and try again. Note that the 80286 multiply immediate instruction must always be entered in the three-argument format.

31—File too large

The size of the file being loaded is too large to fit in the first 640K of memory. Use a lower load address if possible.

32—PSP not found

The Name command was not able to locate the PSP. You can ignore this error if you need to read or write a file with the `LF` or `WF` commands. If you are trying to format the PSP, use RUN to re-enter Periscope.

35—COMSPEC not found

The `/X` command was unable to find the COMSPEC parameter in the environment block. DOS may be garbled. Reboot and try again.

36—Linear to physical address error (Periscope/32 only)

Either no physical address currently exists for the virtual address specified or a selector was specified that is not currently in use.

37—Range conflicts with Periscope

The range specified crosses into Periscope's memory or ports. Specify another range that does not interfere with Periscope.

38—Internal error
An error has occurred in Periscope's paged memory allocation or in Model IV's hardware breakpoint settings. Please call Technical Support if you get this error.

39—Invalid HM/HP settings
See the Model IV manual.

40—Number must be 1 to 4 hex digits (0-9, A-F)
All numbers associated with Periscope installation options are in hex format for consistency. For the `/B`, `/C`, `/E`, `/I`, `/K`, `/R`, `/S`, and `/V` options, the number must be one or two hex digits.

41—Not enough memory above/below 640K
Insufficient memory is available to install Periscope. If 'below', check the amount of available memory using CHKDSK. If 'above', increase the memory available via your memory manager. Boot the system or reduce the space Periscope requires by adjusting the installation options.

42—Invalid installation option
An unexpected entry was found in the installation options. To display the valid installation options, enter `PS ?` from the DOS prompt, or use `PS *` to install Periscope.

43—Interrupt must be 08H, 09H, 10H, 15H, 16H, 17H, or 1CH
The `/V` option specified an interrupt number other than the ones listed.

44—Error using protected memory -- run PSTEST
Periscope was not able to properly install itself in the protected memory on the Model I Board. Check the port setting on the board and the memory and ports specified with the `/M` and `/P` installation options, if any. If the problem per-

sists, run PSTEST (see the Model I Board's installation instructions).

45—Error reading xxx-- run CONFIG

The transient portion of Periscope (PS) was not able to read the indicated file. Be sure that both PS and PS1 are in the same directory. If you can't find PS1, you probably haven't run CONFIG. When you're using Model IV, PS4 is also required and must be in the same directory.

46—Copy of program in protected memory is invalid

The copy of Periscope in the protected memory does not agree with the temporary copy in RAM. Check that the Model I Board is properly seated in the expansion slot and that the chips on the board are properly seated in their sockets. If the problem persists, run PSTEST (see the Model I Board's installation instructions).

47—Screen size must be from 0 to 40H (64) KB

The size of the program's screen specified with the `/S` option must be from `0` to `40`H KB. Note that the number is in hex!

48—Symbol table size must be from 0 to 1FFH (511) KB

The size of the symbol table specified with the `/T` option must be from `0` to `1FF`H KB. Note that the number is in hex!

49—INT 2 does not point to Periscope -- check for conflicts!

The NMI vector is not pointing to Periscope. Check to make sure that your display adapter is not running in an emulation mode where it is using NMI. See Section B.2 for more information.

50—Record table size must be from 0 to 20H (32) KB

The size of the record definition table specified with the `/R` option must be from `0` to `20`H KB. Note that the number is in hex!

51—Unable to remove Periscope from memory

Periscope was unable to release memory. Use MEM or a similar program to check the contents of memory.

52—Unable to read Help or Comment file

An error occurred reading PSHELP.TXT or PSINT.DAT. Rerun CONFIG to regenerate these files.

53—Port number must be from 100H to 3FCH

The port number specified with the `/P` option must be from `100`H to `3FC`H. Note that the number is in hex!

54—Memory specification conflicts with memory used by DOS

The memory address specified with the `/M` option conflicts with DOS memory. Use a higher address, outside the range of DOS memory.

55—Color attribute must be from 01H to FFH and foreground color must not equal background color

The number specified with the `/C` or `/W` installation option indicates a color combination that will display nothing, i.e., the foreground and background colors are the same. Choose another color and remember that the number is in hex!

56—Incorrect window specification

See the explanation of Error 24, above.

57—Unable to read/write response file

An error occurred when Periscope tried to read or write the response file. Check the file name or disk and try again. Note that Periscope ignores any installation options you enter after the response file name. For example, `PS /c:17 @c:std` sets the color attribute to 17H and then reads the rest of the options from the file c:std. If you enter `PS @c:std /c:17`, the color attribute is not set to 17H.

58—Trace buffer size must be from 0 to 3FH (63) KB

The size of the software trace buffer specified with the `/B` option must be from `0` to `3F`H KB. Note that the number is in hex!

59—Invalid user interrupt vector

The user interrupt vector specified with the `/I` option must be from `60H` to `FFH`. The interrupt handler must be already installed using the specified interrupt. Periscope checks for the presence of the interrupt handler by reading memory at the interrupt's segment and offset. The word prior to the interrupt entry point must equal `PS`. See the sample program USEREXIT.ASM in Section 7.10 for more information.

60—Load segment for Periscope tables must be from xxxx to yyyy

The load segment specified with the `/L` option must be greater than the current value of the PSP plus 10H paragraphs. The load segment must also be less than the top of memory minus 3000H paragraphs. If the PSP is C00H and the top of memory is A000H, then the allowable range for the load segment is C10H through 7000H.

61—Source buffer size must be from 0 to 8KB

The size of the source buffer specified with the `/E` option must be from `0` to `8`KB.

62—IRQ masks must be from 0 to FFH

The values specified for the IRQ mask must be entered as one byte from `0` to `FF`H or two bytes from `0` to `FF`H, separated by a space.

63—Incompatible version of Periscope/Remote used

The version of Periscope/Remote software running on the target system is incompatible with the version of Periscope running on the host system.

64—Unable to use 43- or 50-line mode

An EGA is required for 43-line mode. A VGA is required for 50-line mode.

65—Defective CPU (stack change was interrupted).

Replace ASAP!

The 8088 CPU in your system is an early version of the chip that does not protect the instruction after the stack segment is modified. This defect can cause problems when tracing through DOS, using a numeric processor, and when using Periscope. The CPU should be replaced as soon as possible.

66—Incorrect software for Periscope hardware or CPU type

An attempt was made to run the Periscope software when it is configured for a board that is not in the system or for a board that is not supported on this CPU.

67—Unable to read trace buffer--run PS4TEST

See the Model IV manual.

68—Unable to initialize COM port

An error occurred when Periscope attempted to initialize the COM port for alternate PC or remote support. Check the port number and speed settings on both sides and try again. Also confirm that your null-modem cable completely matches the specifications given in the file NOTES.TXT.

69—Error reading PMx.COM--run SETUP

The menu program file PMx.COM was not found. Re-run SETUP (Chapter 3).

70—Interrupt 1CH does not point to an IRET instruction

Periscope's dynamic configuration was not able to find an IRET instruction to correlate INT 1CH to. If you encounter this error, please call Technical Support.

71—Invalid com port number (1-8) or speed (S, M, or F)

When you use an alternate PC or run in active remote mode, the com port number may be specified as `/AV:px`, `/AK:px`, or `/2:px`, where `p` is the port number and `x` is the speed (`S`low, `M`edium, or `F`ast). Both systems must use the same speed.

72—Installation option invalid when remote debugging is

used

When you debug in active remote mode, you cannot use the `/AK` or `/AV` installation options.

73—/M or /P installation option invalid for PS/2 system

The Model I/MC Board is no longer produced. See the manual that came with your board.

74—/Q installation option invalid when /AK or /AV used

When you use an alternate PC, the `/Q` installation option is not allowed. This is because the alternate PC support requires a com port that will not be initialized at ROM-scan time.

75—Error reading xxKEYS.PSD

This error is displayed when Periscope is unable to read the file xxKEYS.PSD, where xx is PS, CV, or TD for Periscope, CodeView, or Turbo Debugger function key emulation, respectively. If the file exists, but is not in the current directory, use the DOS Set command (`SET PS=C:\XXXX`) to set the Periscope path. If the file does not exist, rerun SETUP from the original Periscope disk or a backup copy.

The xxKEYS.PSD loaded in the read-only area must be 511 bytes or less. If it is too large, modify the .DEF file used to generate it.

76—Periscope Model IV board not found

See the Model IV manual.

77—Unable to load Periscope; remove PopUp and try again

PopUp Periscope and Periscope cannot co-exist.

78—Error using XMS memory (code x)

An XMS allocation error occurred. Please check XMS memory using MEM or MANIFEST. If problems persist, contact Technical Support.

90—DEF file not found

RS was not able to find a file of the specified name with an extension of .DEF.

91—Unable to read DEF file

An error occurred reading the .DEF file. Check the file name and try again.

92—Line xxxxx of DEF file is not in correct format

The .DEF file is not in the format expected. The line number indicates the line in the .DEF file where the error occurred. Check the format of the .DEF file as defined in the description of RS in Section 7.5.

93—Not enough memory

Insufficient memory is available for RS to load the .DEF file. Check the amount of available memory using CHKDSK and re-boot as needed.

94—Unable to write PSD file

An error occurred when RS attempted to write the .PSD file. Check the file name and try again.

100—xxx file not found

TS was not able to find a file of the specified name with an extension of .MAP, .SYM, or .EXE.

101—Unable to read xxx file

An error occurred reading the .MAP, .SYM, or .EXE file. Regenerate the file and try again.

102—Line xxxxx of xxx file is not in correct format

The .MAP, .SYM, or .EXE file is not in the format expected. The line number indicates the line in the .MAP file where the error occurred.

If you've used a text editor to modify the .MAP file, be sure to save it in its original format with no embedded tab characters or high bits set. If this error occurs on an unmodified .MAP file, please call Technical Support.

103—Not enough memory

Insufficient memory is available for TS to load the .MAP file. Check the amount of available memory using CHKDSK and re-boot as needed.

104—Unable to write PSS file
An error occurred when TS attempted to write the .PSS file. Check the file name and try again.

105—Unknown EXE symbol type
TS encountered an unknown symbol type in the .EXE file. Please call Technical Support if you get this error.

106—Unknown PLINK symtable type - x
TS encountered an unknown record type in the symbol table at the end of the PLINK file. Please call Technical Support if you get this error.

107—Symbol table overflow at line xxxxx
The compressed symbol table is too large. Use a text editor to modify the .MAP file. You can either delete lines from the .MAP file or comment them using braces, where `{` begins commenting and `}` ends it.

108—Invalid option
An invalid command-line option was used. Enter `TS ?` to display the valid options.

109—EXE file contains inconsistent symbol information
This error occurs when line numbers from two or more modules are competing for the same address space. Try changing any tricky segment usage or turning off symbol generation for any assembly routines in included files.

110—Line numbers out of sequence at line xxxxx
Use the `/LA` or `/LD` options to accept or discard out-of-sequence line numbers.

111—Borland Pascal overlays are not supported

130—Periscope Version x.xx not installed

SYMLOAD cannot run without the corresponding version of Periscope installed. Install the correct version of Periscope and restart SYMLOAD.

140—File not found

INT was unable to read the specified file. Check the file name and try again.

141—Unable to read or write file

An error occurred reading or writing the indicated file. Check the disk and try again.

142—Invalid option

An invalid command-line option was used. Enter `INT ?` to display the valid options.

150—INT x does not point to Periscope!

PSKEY cannot invoke Periscope using the indicated interrupt, which must point to Periscope. Use INT to display the current vectors. If an interrupt has been corrupted, use RUN to refresh them and try again.

160 through 169—See the Model I Board installation addendum.

170—INT 2 does not point to Periscope -- check for conflicts!

See the explanation of Error 49 above.

171—EXE Header not found

A file with an extension of .EXE was specified, but the header record identifying the file as a valid .EXE file was not found. Regenerate the .EXE file and restart RUN.

172—Unable to read file

An error occurred reading the file. Check to be sure the disk is ready and that the file size shown by DIR indicates the true file size.

173—Not enough memory

Insufficient memory is available for RUN to load the desired program. Check the amount of available memory using CHKDSK and re-boot as needed.

174—Periscope Version x.xx not installed

RUN cannot run without the corresponding version of Periscope installed. Install the correct version of Periscope and restart RUN.

175—Periscope not installed correctly

RUN was unable to modify the protected memory. Reload Periscope and try again.

176—Internal error

See the explanation of Error 38 above.

177—Unable to load DEF file

A .DEF file was found for a .COM or .EXE file, but RUN was unable to load it correctly. Check the format of the .DEF file and the size required for the .DEF file using RS.

If the space required by the .DEF file is greater than the record table size, as much of the .DEF file is loaded into the record table as possible. Use **Alt-E** to see the records that were loaded. Note that the last record will usually be only partially defined.

178—PSS file larger than symbol table - no symbols loaded

An error occurred when RUN attempted to load the .PSS file into the symbol table. Usually the .PSS file is larger than the space reserved for the symbol table when Periscope was installed. If this is the case, restart Periscope with a larger symbol table, or use the `/T` or `/X` option with RUN. Otherwise check the file and try again.

179—Symbol table full or invalid

A logical error was found in the symbol table during final processing. Regenerate the .PSS file and try again.

180—PSS file date/time is prior to program's date/time

This warning message indicates that the date/time stamp on the program file is more than one minute later than that of the .PSS file. This is to be expected if the program file has been patched. Otherwise, it indicates that you're using an obsolete .PSS file. Proceed with caution!

181—Unable to execute TS

RUN was unable to EXEC TS.COM. Manually run TS and then re-execute RUN.

182—Exec failure

An error occurred when using the DOS EXEC function to load your program. Check the path name and try again. If the problem persists, call Technical Support.

183—Invalid Exec symbol size

The size indicated with the `/x:` option must be from `0` to `1FF`H KB.

184—PSD file larger than record table - no records/aliases loaded

An error occurred when RUN attempted to load the .DEF or .PSD file into the record definition table. Usually the .DEF/.PSD file is larger than the space reserved for the record definition table when Periscope was installed. If this is the case, restart Periscope with a larger record definition table. Otherwise check the file and try again.

185—File not found

The program to be loaded by RUN was not found. Check the file name and try again.

186—Wrong version number in PSS file

See the description of Error 14 above.

187—Remote timeout

An error occurred when RUN tried to establish communications with the remote PC. Check to make sure that the null-modem cable is securely connected on both ends and that the target system is still functioning.

188—Only one symbol table found

The `/2` option requires two symbol tables. When PS.COM is loaded, be sure that two `/T` installation options are specified. For example, `PS /T:20/T:40` would set up two tables.

190—Program terminated without resident request

The program specified by the `/P` option did not terminate and stay resident. This is a warning message that can be suppressed with the `/Q` option.

191—Unable to read file

The file specified with the `/P` option cannot be read.

192—Invalid option

An invalid command-line option was found in SYSLOAD.SYS. Check the options used with the description of SYSLOAD.SYS in Section 7.8.

230—Unable to read PSx.COM

CONFIG was unable to read the indicated file. Check the disk and try again.

200 through 229—See the PopUp Periscope manual.

231—Interrupt 1CH does not point to an IRET instruction

Interrupt 1CH does not point to an IRET instruction. Check to make sure that no device drivers or memory-resident programs are installed and reboot the system as needed. If this does not clear up the problem, please call Technical Support for assistance.

232—Unable to read file

An error occurred when CONFIG attempted to read a file. Rerun SETUP and try again.

233—Wrong model or version of Periscope

The version or model number found in PS2.COM does not equal the expected version or model. Rerun SETUP and try again.

234—Unable to write file
An error occurred when CONFIG attempted to write a file on the target disk. Check the disk and try again.

235—File too large
An internal error has occurred. Reboot and try again. If the problem persists, please call Technical Support.

236—Format error in PSINT.TXT at line xxxxx
The interrupt comment file is not in the correct format. Rerun SETUP and try again.

240—File PERI.PGM not found
Rerun SETUP and try again.

241—Invalid drive or directory specified. Please try again.
The target drive or directory is not available. Please check the name and try again.

242—Not enough memory
Insufficient memory is available for SETUP to start the self-extract process. Check the amount of available memory using CHKDSK and re-boot as needed.

243—Unexpected error during self-extract program
This usually indicates that the target disk is full. Check the available disk space and try again.

244—Disk error
A disk error has occurred during the self-extract process. Check the target drive and try again.

260 through 269—See the Periscope/Remote for DOS addendum.

270—xxxx timeout (Function=yy, byte=zz)
This indicates a read or write timeout when PSTERM tried

to read or write a byte across the serial link to Periscope. Retry several times, then if errors persist, select the Fail option. If things do not clear up, select the Abort option and restart PSTERM.COM and PS.COM.

271—Port x does not exist

The serial port (default `1`, changeable with the `/2:px` option) is not installed in the computer.

272—UART appears defective

The serial port failed internal diagnostic testing. Replace your serial port and try again.

273—Cannot transmit, cable is wired improperly

The serial cable does not have transmit and receive crossed in both directions. As a result, Periscope cannot send and receive data properly.

274—CTS stuck high, check cable wiring and other serial port

One of two things can cause this problem. Either the serial cable does not have RTS/CTS lines crossed in both directions, or the serial port on the computer that did not display this message is configured to always report `Clear To Send` (CTS).

275—CTS unavailable, cable does not have RTS/CTS lines crossed

The serial cable does not have RTS and CTS lines crossed in both directions. As a result, Periscope cannot handshake properly.

276—No interrupt, check hardware configuration and TSR programs

The receive character interrupt did not generate. This could happen when there are two or more serial ports configured for the same address, such as two ports configured as COM1, or a TSR program is using the same port or the same interrupt request line with a different port (or different hardware, such as a mouse.)

400 through 414—See the Model IV manual.✦

Technical Support and Trouble-shooting

- Technical Support
- Troubleshooting

Tech support procedures and the most common problems are discussed in this appendix. Please check to see if your problem is discussed before calling Technical Support. Also, please either call us or use our BBS rather than writing or FAXing if at all possible. It's much faster and easier to resolve problems on the telephone or BBS than by mail or FAX.

B.1 TECHNICAL SUPPORT

If you are a registered Periscope user, you'll receive free Technical Support when you call the technical support number. We also run a BBS, so if calling is not convenient, you can use the BBS as a forum for your questions and/or problems. We will return Technical Support calls to registered users in the U.S. and Canada when we are not available during Technical Support hours. **Please see the back of the cover page for phone numbers and hours of operation.**

When you call with a problem, please have your registration number handy. If you purchased your Periscope recently from The Periscope Company (TPC) but have not sent your registration card in, please have both your invoice number and your registration number available. If you did not purchase your Periscope from TPC, we will assist you in the installation and configuration of your Periscope, but your registration card must be on file before we will provide any additional Technical Support. You may wish to send your card to us by overnight courier if you purchased your Periscope from a dealer and need Technical Support right away.

Please be prepared to answer these questions when you call:

- Which Periscope product (Model I, Periscope/EM, or Model IV) and version (e.g., Version 5.40) are you using?
- What brand and model of computer are you using?
- What version of DOS, Windows, and/or OS/2 are you using?
- What boards are installed in your system?
- What device drivers and/or memory-resident software are you using?
- What Periscope installation options are you using?
- What is the problem you're experiencing?

B.2 TROUBLESHOOTING

386MAX Users

If you're using 386MAX by Qualitas, Inc., note the following:

- When you load Periscope in high memory, make sure

you have at least 124K of contiguous memory available.
- Don't try to use the `AC` options with PSKEY.
- Use version 2.20 or later of 386MAX to avoid conflicts with Periscope.

DOS Usage

If you try to use DOS from Periscope and get the message `DOS busy`, enter `G {0:28*4}` to set a breakpoint the next time INT 28H executes. When the CS:IP is at the start of INT 28H, Periscope allows you to use DOS.

Periscope Version and Model

The version number (Arabic number) is the release number of the software, such as 5.40. The model number (Roman numeral) identifies the hardware used with the software. To determine which version and model you are using, check the default value of item 7 in the Watch window. The version number is followed by an alphabetic suffix, which indicates the model. Some of the possible values are **E** (Periscope/EM), **M** (Model I, Rev 3), and **K** or **L** (Model IV).

Traps (Exception Interrupts)

If your program suddenly pops into Periscope with a trap, one of three things has happened. Either an unexpected single-step has occurred (indicated by trap 1), an illegal instruction has been executed (indicated by trap 6) or a segment wraparound (a read or write of a word at offset **FFFFH** indicated by trap 0DH) has occurred. The illegal instruction is usually caused by an unbalanced stack, such as when a routine is called as a near procedure, but returns as a far procedure. Similarly, mismatched PUSHes and POPs frequently lead to a trap.

Break-out Switch

If your Break-out Switch does not work, check the following items:
- Make sure that your machine supports NMI. If you're using a system that has only 8 chips per 64/256 KB, such as the Tandy 1000, you're probably out of luck.
- Use INT to display the values of the interrupt vectors. INTs 1, 2, and 3 should have the same segment, with

B

offsets ascending by 5.

- Try using CLEARNMI to periodically clear out any blockage of the Break-out Switch.
- Microsoft's EMM386.SYS does not correctly handle NMI under some conditions. If you must use this driver, do not use the Break-out Switch.

Memory Usage

If you don't have a monochrome monitor, watch out for memory corruption if your memory manager attempts to use the mono region (B000-B7FF). If this region becomes corrupted, it can cause the UMB chain to be destroyed.

Port Usage

If you configure Periscope as Model I but don't have the board in the system, be sure to use the `/N` installation option to avoid access to ports 300 and 301.

Symbols and Source Code

If you have problems displaying source code, enter the `/L` command while in Periscope. If you get the message **`No lines found`**, check your compile and link options. If you get the message **`DOS busy`**, no source code will be displayed until you get to a point that DOS is not busy. If this doesn't solve the problem, check the items listed under the disassemble source (`US`) command. Do not expect to be able to access source code when debugging a TSR program (unless you're using the Periscope utility WAITING.COM), since DOS is usually busy.

If you have problems accessing symbols, press **Alt-I** to display the symbols available. If you do not find the symbols you expect, check the compile and link options you used to generate the program. Also, if you configured Periscope to preserve the case of symbols, be sure that the symbols are entered in the same case as shown by the **Alt-I** display.

Unexpected Entry into Periscope

If Periscope comes up unexpectedly, enter `Q` to display the entry reason. See the description of the Quit command in Chapter 9 for more information.

Interrupt Vectors

Since Periscope temporarily replaces the values of some interrupt vectors while its screen is displayed, most display commands will show the values of Periscope's vectors. To see your program's vectors, use the doubleword format, `DD`. The values followed by an asterisk indicate your program's vectors.

Disassembly Window

If the reverse video bounce bar disappears, enter `R` to force it to reappear.

Program Screen

If your program's screen is not being correctly restored after Periscope's screen is displayed, check to be sure you've allocated enough memory for saving and restoring screens. In a single-monitor system, the default is 4KB, which is enough for text mode only. If your program uses graphics mode, we strongly recommend that you use a dual-monitor system. It will save much wear and tear on you and your display adapter.

An EGA or a VGA driving a color display can co-exist with a monochrome display. Be sure to use a plain Jane display adapter, i.e., one without any graphics ability. You can use one of the original IBM mono cards or the Dell mono card or equivalent.

Another option is to have Periscope's screen displayed on another PC. See the `/AV` installation option.

Other Problems

If you encounter problems other than the ones mentioned above, check to see if the situation is repeatable. First, try it on the same machine under the same circumstances. If the problem persists, try removing resident programs and device drivers. If the problem goes away, try to isolate the conflicting program. Finally, try to repeat the problem on another machine. The more information you have when you call for Technical Support, the better, since that will help us help you more quickly and effectively. ✦

Periscope/32

- **Introduction to Periscope/32**
- **Differences between Periscope/EM and Periscope/32**

This appendix describes Periscope/32, a 32-bit-aware derivative of Periscope/EM, and summarizes how it differs from Periscope/EM. We've noted specific differences throughout the manual as well.

If you're using Periscope/32 as the host debugger in a remote debugging environment with a Periscope remote driver running on the target, you'll find detailed installation instructions and environment-specific usage information in the addenda accompanying the products that include it, such as *Periscope/32 for Windows* or *Periscope/32 for OS/2*. If you're running another (non-Periscope) remote driver on the target, see the documentation included with the driver for implementation-specific details. If you're using the *Periscope 32-bit Toolkit*, see the source files on the disk for additional information.

C.1 INTRODUCTION TO PERISCOPE/32

Periscope/32 is a 32-bit-aware version of the Periscope/EM software. It runs

- as a standalone debugger in a single-machine DOS environment (with or without the Periscope Model IV hardware), and
- as the host debugger in a two-system, active remote environment, with a supporting remote driver running on the 32-bit target (again, with or without the Periscope Model IV hardware).

In the remote environment, the driver software running on the target computer acts as Periscope/32's "agent", enabling you to use Periscope/32 to debug system-level software running in that environment.

C.2 DIFFERENCES BETWEEN PERISCOPE/EM AND PERISCOPE/32

Since Periscope/32 is derived from Periscope/EM, most of the differences between between the two are a result of the differences in the 16:16 memory model of Periscope/EM versus the 16:32 memory model of Periscope/32. Other differences are due to the fact that Periscope/32 is normally used as a host debugger in a remote debugging environment whereas Periscope/EM is normally used as a standalone debugger in a single-system environment.

The differences between Periscope/EM and Periscope/32 are summarized below. We have detailed specific differences throughout the manual.

Differences between Periscope/32 (32-bit model) and Periscope/EM (16-bit model)

- Periscope/32 shows invalid or out of range memory with double question marks (`??`). This can occur when an offset is beyond the limit for the selector or when an offset is beyond 0FFFFH in real mode. Periscope/32 usually displays inaccessible memory with one or more question marks, but some commands, such as the disassemble command, show invalid memory as **`FF`**.

- You cannot use the comma as a general delimiter in Periscope/32, since it is used as optional punctuation for numbers longer than four digits. For example, the command `O 21,FF` is legal when you're using the Periscope/EM, but not when you're using Periscope/32. Use `O 21 FF` instead.
- Periscope/32 generally displays eight-digit offsets with four digits, a comma, and four more digits for better readability. The only exception to this is in disassembly, where the comma can be confusing. For eight-digit numbers that you enter, the comma is optional and does not indicate position. So, for example, the command `R EAX 1234,5` is the same as `R EAX 12345`, but is not the same as `R EAX 1234,0005`. Periscope/32 ignores a comma you input unless it is in the middle of a string or you're using the in-line assembler.
- Periscope/32 supports four types of pointers:
  `[` — near 16-bit pointer (16-bit offset)
  `{` — far 16-bit pointer (16-bit offset, 16-bit segment)
  `[[` — near 32-bit pointer (32-bit offset)
  `{{` — far 32-bit pointer (32-bit offset, 16-bit selector)
- Periscope/32 assumes that addresses are in selector:offset format rather than in segment:offset format. It supports 32-bit offsets, while Periscope/EM supports 16-bit offsets. In most cases, the selector and the segment are interchangeable. There is one special case use of a selector in Periscope/32. Using a selector value of 0, you can reference Virtual 86 (V86) memory while in any mode. The allowable offsets are from 0 to 10FFF0, which is one megabyte plus 64K minus 16 bytes (the XMA limit). So, to reference memory at the V86 address of 2000:1234, you could use `0:2000*10+1234` or `0:21234`. You can also indicate that real-mode style address conversion (segment times 16 plus offset) is to be applied to an address by separating the segment and offset by a stile (`|`) instead of a colon (`:`). This means that you could also reference the above address as `2000|1234`. When you use the stile, Periscope/32 converts the address to the `0:xxxxxx` format.
- If you're debugging code that stays in V86 mode, use the normal addressing. If you're debugging code that moves from V86 to protect mode and back, use the `0:xxxxxx` addressing so that the address will be valid regardless of the mode you're currently in. Note that this is especially important if you use any sticky breakpoints such as `BC`

C

or BD. When you're running in V86 mode, some instructions cause a Trap D, including pushf, popf int, iret, cli, and sti. When these instructions execute, control transfers to the ring 0 exception handler so that it can emulate the instruction. For some of these instructions, the ring 0 emulation may cause single stepping to regain control at the second instruction after the emulated instruction. For example, assume the following code sequence:

```
sti
pop ax
pop bx
```

If you single step the `sti` instruction, control returns to Periscope/32 at the `pop bx` instruction. The `pop ax` instruction executed, but because of the way some ring 0 exception handlers work, Periscope/32 does not see it. If you want, you can force the display of the `pop ax` instruction by setting a code breakpoint at that address.

Differences between Periscope/32 running in local mode vs. Periscope/32 running in remote mode
Following are the differences between Periscope/32 running standalone in a single-system environment (local mode) and Periscope/32 running in the host system in a two-system environment, with a Periscope remote driver running in the target system (remote mode):

- When you're running in remote mode, Periscope/32 does not go resident. It stays in the foreground on the host system so that DOS use is never a problem. To get to the DOS prompt, use the `/X` or `QR` commands. While in DOS, the target system is suspended. To return to Periscope, use the `EXIT` command at the DOS prompt. To completely terminate Periscope/32, use the `QR!` command. **Warning:** use of `QR!` bails out of Periscope/32 without doing a full cleanup. You should use this command with caution. You can also use the `QU` command to reboot the host system and unhook the Periscope remote driver on the target system so that the target runs as if the driver had not been installed.
- In remote mode, Periscope/32 does not emulate the tracing of a software interrupt. Instead, it attempts to set a

breakpoint at the first instruction of the interrupt. If this is not possible (as in a BIOS interrupt), it uses the `J` command to step over the interrupt. If you want to stop inside a real-mode BIOS interrupt, use `BD {0:nn*4`, where `nn` is the desired interrupt.

- If a Trap D occurs inside the Periscope remote driver, Periscope/32 displays the message **`Fatal error in Periscope/Remote`**. Should you ever see this message, please mail or fax a screen dump to Technical Support.
- In remote mode, Periscope/32 does not load into the Model I Board's memory or EM memory, even if a board or supporting memory manager is present.
- The default symbol table size for Periscope/32 in remote mode is 80H (128) KB.
- Periscope/32's screen indicates that it is in remote mode by displaying the prompt as `/2>` instead of `>`.
- Periscope/32 in remote mode shows `PSRx Version 5.xx` as item 7 in the Watch window instead of the normal `PERISCOPE/32 5.xx`.
- In remote mode, the `QS` command has the same effect as the `QB` command.
- The USEREXIT program supports remote mode. It can read and write memory on the target system. It can also download and execute code on the target system and upload a string from the target. See the file USEREXIT.ASM for details.
- When you use the `BC`, `Gx`, or `Jx` commands that can set a code breakpoint on memory that is not valid (out of limits or unloaded), you'll get an Error 16 from Periscope/32. For example, if you're at the last instruction of a program and the current instruction is a CALL, you cannot use the `J` and `JL` commands, since the current instruction is the end of the current segment. In this case, use one of the Trace or Go commands instead. Note that if you set a code breakpoint on memory using an LDT selector that is valid when you issue the `BC` command, but invalid when the actual breakpoint occurs, Periscope/32 ignores the breakpoint and displays no message!
- Periscope/32 in remote mode does not allow you to use the `/A`, `/AK`, and `/AV` installation options. It assumes that you're using a single-monitor system and that you're using the COM port to connect the host to the target rather than to an alternate PC.

Command differences and Periscope/32-specific commands

The Periscope remote driver does not do any file I/O on the target system. All file-oriented commands are implemented on the host system. These include the following commands: **LB**, **WB**, **LD**, **WD**, **LS**, **WS**, **V**, **VS**, **UB**, **US**, **/E**, **/K**, **HR**, **HT**, **HU**, **HW**, and **/X**. These commands are not available in remote mode: **LA**, **WA**, **LF**, **WF**, **N**, **IC**, **IR**, and **IS**.

The following commands differ between Periscope/EM and Periscope/32 or are specific to Periscope/32: **BF**, **BM**, **BR**, **C**, **DD**, **DG**, **DT**, **DV**, **ED**, **EM**, **F**, **H**, **IC**, **IR**, **IS**, **LA**, **LF**, **LS**, **M**, **N**, **QR**, **QU**, **R**, **R EFL**, **RF**, **RFL**, **RC**, **RX**, **SC**, **SR**, **TB**, **TR**, **TU**, **U**, **UA**, **UB**, **US**, **WA**, **WF**, **X**, **XD**, **XH**, **XA**, **/3**, **/N**, **/S**, and **/W**.

See Chapter 9 for details on using these commands.⌖

Index

INDEX