Periscope

MODEL III

The Undercover Debugger
For The IBM Personal Computer

by Brett Salter

Periscope

The Undercover Debugger
For The IBM Personal Computer

by Brett Salter

Periscope Manual (Version M31)

SOFTWARE BY: Brett Salter
MANUAL BY: Sharon Bailey & Brett Salter

PUBLISHED BY:
The Periscope Company, Inc.
14 Bonnie Lane
Atlanta, GA 30328 USA
Phone: 404/256-3860

SALES: 800/722-7006
TECH SUPPORT: 404/256-3372 (9am-5pm Eastern time)

We will attempt to fix errors you find in the software if you will provide us a way to recreate the problem on our computer systems. We welcome your suggestions and comments regarding improvements and enhancements. Periscope's evolution is guided by its users.

The entire risk as to the performance of this package is with the purchaser. TPC has carefully reviewed the materials provided with this package, but does not warrant that the operation of the items included in this package will be uninterrupted or error-free. TPC assumes no responsibility or liability of any kind for errors in the package or for the consequences of any such errors.

Table of Contents

Please Read—Important Information

- **Money-Back Guarantee**
- **Warranties**
- **FCC Compliance**

Money-Back Guarantee (All Models)

If you purchase Periscope from The Periscope Company, Inc. (TPC) and the package does not perform to your satisfaction, you may return it within 30 days of our shipping date, and we will refund your purchase price, excluding shipping charges. Other suppliers may or may not offer a similar guarantee.

Return authorization must be obtained from TPC at 404/256-3860 prior to return. Returned packages must be received within the 30-day guarantee period, must be marked with the return authorization number, must contain all materials, including the registration card, and must arrive in as-new condition. If the return authorization is not clearly marked on the package, it will be refused. If the package is damaged or incomplete, or if it is received after the 30-day guarantee period, a minimum 15% restocking fee may be deducted from the amount refunded. **Note:** A separate fee of 15% *may* be deducted on returns of Model III unless there is an unresolvable conflict with the user's computer system.

Warranties

TPC warrants the enclosed printed circuit board and/or break-out switch (all models except II-X) to be in good working order for a period of one year from the date of purchase as a new product.

Should this hardware fail to perform properly any time within the one-year warranty period, TPC will, at its option, repair or replace it at no cost except as set forth in this warranty. Replacement parts or products will be furnished on an exchange basis only. Replaced parts and/or products become the property of TPC. No warranty is expressed or implied for damage caused by accident, abuse, misuse, natural or personal disaster, or unauthorized modification.

Warranty service may be obtained by sending the board and/or switch to TPC during the warranty period. *Return authorization must be obtained at 404/256-3860 prior to return, and proof of purchase date must be included. If shipped by mail or any common carrier, owner agrees to insure and accept all liability for loss or damage, to prepay all shipping charges, and to use a shipping container equivalent to the original packaging.*

The Periscope diskette, manual, and binder (all models) are warranted to be free from defects for ninety days from the date of purchase. If any materials are found defective during the 90-day warranty period, call TPC at 404/256-3860 for warranty service.

FCC Compliance

The equipment contained in this package complies with the requirements in Part 15 of FCC Rules for a Class A computing device. Operation of this equipment in a residential area may cause unacceptable interference to radio and TV reception, requiring the operator to take whatever steps are necessary to correct the interference. Shielded cable should be used with this unit to insure compliance with the Class A limits.

Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Preface

- **Registration, Updates, Upgrades, And Tech Support**
- **The Four Models of Periscope**
- **Learning Periscope**
- **Using The Manual**

Registration, Updates, Upgrades, And Tech Support

Be sure to complete and return the enclosed registration card. It entitles you to an automatic first software update, free of charge, and notice of future updates, for which there is a nominal charge.

(**Note:** Keep your registration card in a safe place if you do not send it in immediately. Without it, you must provide us with proof-of-purchase and pay the current update fee to become registered and receive your first software update.)

Registered users may upgrade to a higher-priced model of Periscope for the difference in price plus a nominal upgrade fee (currently $10). For instance, if you purchase Model II, then later decide you need the protected memory offered by Model I, it would cost you $180 (the Model I price of $345 less the Model II price of $175 plus $10) to trade in your Model II for a Model I. The current list prices, which may or may not be what you actually paid, are always used to calculate the price difference. Call us at 404/256-3860 for details and/or a return authorization number.

Registered users also receive free technical support by calling 404/256-3372, Monday through Friday, from 9 AM

to 5 PM Eastern time. *Please call rather than writing if possible—it's much faster and easier to resolve problems on the telephone than by mail.* When you call with a problem, please be prepared to answer these questions:

- What Model (I, II, II-X, or III) and version (e.g., 3.10) of Periscope are you using?
- What brand and model of computer are you using (e.g., IBM XT)?
- What version of DOS are you using (e.g., PC DOS 2.1)?
- 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?
- Are you able to repeat the problem on the same computer system or on another computer system?

The Four Models Of Periscope

This manual describes four different models of Periscope. The four models include very similar software, but there are differences in the hardware. We use an arrow (➧) to point out key model differences in this manual.

➧ **Periscope I** includes a memory board with 56K of write-protected RAM and a remote break-out switch. The protected RAM protects Periscope's software from programs that overwrite low memory, where debugger software normally resides. And since Periscope resides in the board's memory, it doesn't compete with your program for space in normal RAM.

The switch located on the Model I board's mounting bracket enables you to stop an executing program and enter Periscope at any time. The optional remote switch that plugs into the phono jack on the mounting bracket performs exactly the same function. This break-out capability means that you can interrupt the system any time, to debug the executing program or just to see what's going on. It is especially valuable when your system is hung.

➧ **Periscope II** includes a remote break-out switch, but no protected memory board. The switch taps into an expansion slot that is already in use, so no additional slot is required. (The break-out switch with Model I requires a slot because it is implemented via the board.) The Model II break-out switch performs the same function described above for the Model I break-out switch.

➧ **Periscope II-X** is software only, with no break-out switch or memory board. Because it does not include a break-out switch, it does not support Interrupt 2 (NMI), generated by the switch. (However, you can install your own break-out switch to generate Interrupt 2, then configure Periscope as a Model II.) Model II-X works on the IBM convertible, which uses Interrupt 2 for its own purposes, and may work in other environments where there's a conflict with the other models of Periscope.

➧ **Periscope III** includes a remote break-out switch, and a memory board with 64K of write-protected RAM, a real-time trace buffer, and hardware breakpoints. The memory protection and break-out capability are the same as for Model I (except there's 8K more protected memory). The difference is that the Model III board also contains the logic to monitor the system bus, and with your program executing at full speed, captures vital execution information and/or stops at specified hardware breakpoints. There are additional Periscope commands, used only with Model III, that enable you to deal with the Model III board functions.

Learning Periscope

Please read or at least browse through the manual and take the tutorials in Chapters IV and V before you begin using Periscope.

Information on using Periscope III is included throughout the manual. However, there is a special chapter (Chapter XI) on hardware-assisted debugging that Periscope III users should read before reading the other chapters.

When you're ready to get started, here are the things you must do:

Step 1—Back up the Periscope disk and configure Periscope (See Chapter II.)
Step 2—Install the hardware (See Chapter III.)
Step 3—Take the tutorials (See Chapters IV and V.)
Step 4—Install the software (See Chapter VI.)
Step 5—Load your program (See Chapter VIII.)

Using The Manual

Here's how the manual is organized:

Chapter I—Introduction—gives you an overview of Periscope, its features, benefits, and requirements.

Chapter II—Configuring Periscope—gives you a step-by-step procedure for backing up, then configuring the Periscope software; also describes each file on the disk.

Chapter III—Installing the Hardware—describes how to check your configuration for conflicts with the default settings of either of the Periscope boards, how to change your Periscope board's settings if you have conflicts, and how to install the board and break-out switch.

➧ For Periscope II, a separate section covers how to install the break-out switch.

Chapter IV—Tutorial: Using Periscope—walks you through a session using Periscope to 'debug' a simple assembler program.

Chapter V—Tutorial: Periscope and 'C'—walks you through a session using Periscope to 'debug' a simple 'C' program.

Chapter VI—Installing the Periscope Software—describes how to install the Periscope software; explains the installation options and the alternate methods of installing Periscope.

Chapter VII—Debugging with Periscope—defines the quit options, keyboard usage, the command parameters, and Periscope's commands.

Chapter VIII—RUNning Your Program—describes how the program loader, RUN.COM, works.

Chapter IX—Using The Periscope Utilities—describes how to use the various utility programs included in the package.

Chapter X—Technical Notes—discusses debugging theory, NMI use, CPU differences, DOS, debugging techniques, debugging device drivers and non-DOS programs, Periscope internals, the Periscope (I and III) boards, and the IBM Enhanced Graphics Adapter (EGA).

Chapter XI—Using Periscope III—describes hardware-assisted debugging with Periscope III.

Appendix A—Error Messages—explains Periscope's error messages.

I—Introduction

- **Welcome!**
- **Items Included In The Periscope Package**
- **What Makes Periscope Different**
- **Features And Benefits**
- **Symbols—Your Road Map Through Memory**
- **System Requirements**

Welcome!

Thank you for choosing Periscope as your debugging system. You've made your life as a software developer easier, because Periscope gives you the tools you need to find and fix your programs' bugs quickly! The sooner you begin using Periscope, the sooner you'll start saving yourself many hours of debugging time. While you're getting acquainted with Periscope, remember that we support what we sell. If you run into a problem or have a question, please call Tech Support at 404/256-3372.

Items Included In The Periscope Package

The first four items below are included with all models of Periscope. The switch, board, and umbilical (items 5, 6 and 7) are included only with the indicated models.

1) A registration card

The registration card is important...don't lose it! It entitles you to a free first software update, notices of

subsequent updates, and free technical support by calling 404/256-3372. For more information, see the Preface.

2) A diskette, with the Periscope software and related files

The diskette contains the debugger software, plus sample programs and files used in the tutorials (Chapter IV and V); Periscope utility programs (Chapter IX); and a Periscope demo program. See Chapter II for a description of each file on the diskette.

3) A quick-reference card

The quick-reference card provides you with a concise list of Periscope's commands, command parameters, installation options, and keyboard usage, all of which are described in detail in Chapters VI and VII.

4) This manual

This manual's purpose is to provide you with the training you need to learn Periscope (Chapters IV and V) and with the technical information you need to become an expert Periscope user (Chapters VI through X). We strongly urge you to read the manual!

5) A remote break-out switch (All Models except II-X)

The switches located on the Model I and III boards' mounting brackets, the optional remote switches that plug into the phono jacks on the Model I and III boards' mounting brackets, and the optional remote switch that comes with Model II perform exactly the same function. They enable you to interrupt the system at any time, often saving you from rebooting or powering down and up again. See Chapter III for installation instructions.

6) A board (Models I & III)

The Model I board includes protected RAM plus support for the break-out switch. Because Periscope resides in the board's protected memory, the debugger software can't be overwritten by a runaway program, plus normal RAM is available for your program. The Model III board includes protected RAM and support for the break-out switch, and it supports hardware breakpoints and provides a real-time trace buffer. See Chapter III for installation instructions.

7) An Umbilical Socket (Model III only)

The Model III package includes an umbilical that plugs into the 8087/80287 socket on the motherboard to pick up the one signal (indicating whether a memory read is an instruction fetch) not available on the bus. This umbilical does not preempt use of the numeric processor. See Chapter III for installation instructions.

What Makes Periscope Different

Reviewers agree that Periscope is hard to beat for solid, dependable, tough debugging. Periscope is the only debugger ever chosen as Product of the Month by award-winning PC Tech Journal. Explaining why Periscope was chosen, Technical Editor Jeff Duntemann says that Periscope has *"an extraordinarily clean and innovative design"* and an *"overall aura of quality too strong to ignore."* In Ward Christensen's review, Periscope was dubbed as *"unrivaled for it's flexibility,"* and Mr. Christensen concluded that *"...Periscope's diverse features, affordable price, and portability place it in a class by itself."* Jim Kronman, who reviewed Periscope II-X for Computer Language wrote afterwards, *"I am very impressed with Periscope...it has become my 'heavy duty' debugger of choice..."*

Periscope was originally conceived as a crash-recovery system, so one of its key differences is its thorough crash recovery capability. The protected memory included with Models I and III keeps runaway programs from interfering with the operation of Periscope. The break-out switch included with all models except II-X lets you check out problems when they occur, without having to stop and load your program under debugger control.

➧ Periscope II and II-X do not offer the same level of crash recovery as Periscope I and III since Models II and II-X do not include a write-protected memory board.

You can use Periscope to debug device drivers, non-DOS programs, interrupt handlers, and memory-resident programs. You can even debug DOS because Periscope uses BIOS rather than DOS function calls for most of its operations.

Periscope is highly dependable because it saves the state of the machine when it's activated, reverts to a fresh, power-on machine state for its own use (unless you specify otherwise), then restores the saved state when done. Even when hardware interrupts are disabled, you can recover the machine by pressing the break-out switch included with all models except II-X.

You can now choose from a full line of Periscope debugging systems, beginning with software-only Model II-X through the powerful hardware-assisted Model III. Periscope III, in addition to providing the known dependability and comprehensiveness of the other models, offers tremendously-powerful capabilities for debugging real-time systems, like a real-time trace buffer several times the size normally provided. It uses a superset of Periscope commands, so once you're familiar with one of the other models of Periscope, you just have to learn an additional dozen commands to take full advantage of Model III's special hardware functions!

Features And Benefits

Since Periscope's job is to make software development quicker and easier for programmers, it's designed to be easy to learn and use, reliable, comprehensive, and fast.

These features make Periscope easy to learn and use:

- Optional on-line help.
- DOSEDIT-like command-editing.
- Commands that are a logical extension of DEBUG's.
- Optional windows that you can define and redefine to display disassembly, stack, register, file, and data information. You can define up to four data windows, and you can control both foreground and background colors for each window.
- Multiple memory display formats, including ASCII, byte, integer, signed integer, word, double word, short real, long real, and ASCIIZ string; plus, define and display memory using your own memory display templates.
- Symbol support, including source code and line numbers. (See the discussion on symbols later in this chapter.)
- The ability to assign command sequences to function keys.
- The ability to switch from your program's screen to Periscope's screen and back with a single keystroke. If you use a two-monitor system Periscope automatically uses the monitor not used by your program.

These features make Periscope reliable:

- Dependable crash recovery—Periscope saves the interrupt vectors it uses each time it is activated, then restores them when done.
- You can safely interrupt the system any time with the break-out switch, then continue running as if Periscope had never been used; Periscope doesn't use DOS except

for file access, so you can debug your program even if DOS is not available.
• Periscope protects itself; runaway programs can't touch the protected memory.

➧ The code in Models II and II-X resides in normal RAM and is not protected.

These features make Periscope comprehensive:

• A symbolic in-line assembler.
• Over 75 software breakpoints, including sticky and temporary code breakpoints; breakpoints on register values, byte and word values, and reads and writes to ranges of memory and I/O ports using various tests; the ability to write your own breakpoint tests; and more.
• Trace and traceback capability.
• Disassemble any location in memory, displaying the symbols, line numbers, and code from your source program.
• Search memory for string/byte patterns, procedure and address references, and instructions.
• Time your code in increments of 838 nanoseconds.
• View text files.
• Read and write disk files and absolute disk sectors.
• Read and write I/O ports.
• Compare two locations in memory, to move a block of memory from one location to another, and to make changes to memory.
• Execute calls and interrupts at full speed.
• Display, change, save, and restore the registers and flags.
• Translate from hex to decimal and vice-versa, and to perform hex arithmetic.
• Support for the EGA, including 43-line mode when using a single monitor.
• User exits, so you can execute your own code from within Periscope.

Periscope is FAST because it's written entirely in assembler!

Symbols—Your Road Map Through Memory

Periscope is symbolic, meaning it allows you to use data and procedure names from your source program when you're debugging. This symbolic capability speeds up debugging tremendously, since you do not have to look for a particular sequence of instructions to find a certain section of code, nor do you have to remember the location of code or data—you can access it by name. You can even use source line numbers and your high-level

source code if your compiler provides the information Periscope needs.

For example, assume that a program you're debugging calls a subroutine named PRINT_LINE, and you want to go to the first call of this subroutine in your program. You would enter 'G PRINT_LINE'. ('G' is the Periscope Go command; see Chapter VII.) If symbols were not available, you'd have to know the address of the subroutine in order to get to it. If you were to disassemble this same program, the disassembly would display 'PRINT_LINE' wherever it is referenced in the program.

Symbols are read from the MAP file generated by your linker. Periscope supports the IBM, Microsoft, Phoenix, and DRI linkers as well as providing limited support of the Cware DeSmet and Manx Aztec C compilers. Periscope also provides symbol support, including source lines and code, for compiled Turbo Pascal programs if you use TdebugPLUS by TurboPower Software. If your compiler generates line number references in the object files it produces, Periscope will give you line numbers and source code. For more specific information on symbols, see the tutorials in Chapters IV and V, the sections in Chapter IX on the TS.COM and PUBLIC.COM utility programs, and the description of RUN.COM in Chapter VIII.

System Requirements

The system requirements for Periscope are:

- IBM PC, XT, or AT; Compaq; or other close compatible such as Zenith Z-150, Columbia, Leading Edge, Sperry and many others.
- PC/MS-DOS 2.00 or later
- 64K available RAM
- 1 disk drive
- 80-column monitor

➧ Model III works on the IBM PC, XT, and AT, the Compaq 8088 and 80286 systems, and other very close compatibles. It does NOT work on: IBM XT/286; Compaq 8086 and 80386 Deskpros; AT&T 6300 and 6300+; IBM Personal System/2; any zero wait-state machine; any machine containing a 'Turbo' card. Call Tech Support if you have compatibility questions or problems.

The Model I board may be installed in an expansion chassis. The model III board must be installed in the system unit.

II—Configuring Periscope

- **Backing Up And Configuring Periscope**
- **Files Contained On The Periscope Disk**

Backing Up And Configuring Periscope

When you receive this package, back up and configure Periscope using the following procedure:

- Place your DOS diskette in drive A and enter 'DISKCOPY A: B:'.

- When prompted, insert the Periscope diskette in drive A, and a blank diskette in drive B. Press any key to perform the copy.

- When DISKCOPY is complete, remove the original diskette and store it in a safe place.

- Reboot the system without any device drivers or memory-resident programs installed. (You may skip this step only if you have an IBM or Compaq system.)

- Place the Periscope back-up diskette in drive A and enter 'A:CONFIG'.

- Answer the prompts on the full-screen display (shown below) and press F10 to configure Periscope. The configuration program generates the selected model of the Periscope software and copies the files from the distribution disk to the drive specified, using a sub-directory named 'PERI'. The last prompt controls the minimum command length that is added to Periscope's command buffer. If any errors occur, check the error messages in Appendix A and retry as needed. For more information on the configuration program, CONFIG.COM, see Chapter IX. **Note:** If you select Model II-X and you want

to define hot keys to activate Periscope, please read the information in Chapter IX on PSKEY.

```
Important! This program should be run without any device drivers or memory-
resident programs installed unless you're using an IBM or Compaq system!

Periscope Model (choose one):
  Ø - Model I (protected memory board)
  1 - Model II (break-out switch)
  2 - Model II-X (software only -- no NMI support)
  3 - Model III (hardware breakpoint board)
Enter choice (Ø-3) [Ø]

Target drive (A-Z)? [C] (uses/creates '\PERI\')

Copy Periscope ancillary files to target drive (Y/N)? [Y]

Minimum command length to be added to edit buffer (Ø-9) [Ø]
```

```
Key usage
Enter - next field
Home - first field
End - last field
Up - prior field
Down - next field
Esc - exit to DOS
F1Ø - configure PS
```

Files Contained On The Periscope Disk

The files on the distribution diskette are:

CLEARNMI.COM—Despite the name, the non-maskable interrupt (NMI) used by the break-out switch can indeed be masked out. This memory-resident utility program attaches to the timer interrupt and clears the non-maskable interrupt ports once a second.

CONFIG.COM—This utility program is used to configure the Periscope software, moving it from the distribution disk onto your working disk.

FTOC.C—This is the source for the Fahrenheit-to-Celsius program from the Kernighan and Ritchie text. This program is used in the 'C' tutorial in Chapter V.

FTOC.DEF—This is a Periscope record definition file used when debugging the FTOC program.

FTOC.EXE—This is the executable code for the FTOC program used in Chapter V.

FTOC.MAP—This is the MAP file produced by the linker for the FTOC program. It contains symbol and line number references.

INT.COM—This utility program is used to save, display, or compare the values of the interrupt vectors. It is typically used to determine the interrupt vectors used by a resident program.

PS.DEF—This ASCII text file contains some sample Periscope record definitions that are read when RUN.COM is used to load a program.

PS.PGM—This is the Periscope program for Models I, II, and II-X. It is converted to PS.COM by the configuration program, CONFIG.COM.

PS3TEST.COM—This program is used to run trace buffer and breakpoint tests on the Periscope III board.

PSDEMO.COM—This program is a billboard-style demonstration of Periscope. It is not copyrighted and may be freely distributed. All other files on this disk are copyrighted and may not be distributed.

PSH.PGM—This is the Periscope program for Model III. It is converted to PS.COM by the configuration program, CONFIG.COM.

PSHELP.TXT—This is the optional help file that is loaded into RAM by PS.COM when the /H installation option is specified. This file is a normal ASCII text file which can be modified as needed using a text editor.

PSHELP2.TXT—This is the short form of the help file. It gives the syntax and a short description for each command. To use this file instead of the standard help file, rename it to PSHELP.TXT.

PSINT.TXT—This is the optional interrupt comments file that is loaded into RAM by PS.COM when the /H installation option is specified. This file is a normal ASCII text file which can be modified as needed using a text editor.

PSKEY.COM—This memory-resident utility program is used to select hot keys to activate Periscope via the keyboard, including SysReq, Shift-PrtSc, and shift key combinations.

PSTEST.COM—This program is used to run memory diagnostics on the Periscope I and III boards.

PUBLIC.COM—This utility program is used to generate public statements for most data items and procedures in an assembly program—giving you the best possible symbol support for debugging.

READ.ME—This file contains any changes made to this manual since it was last published.

RS.COM—This utility program is used to verify a record definition (DEF) file and determine the record table size required by the DEF file. It enables you to efficiently allocate space for the record definition table.

RUN.COM—This is the program loader. It is used to load a program or data file into memory, read the program's symbol table and record definition table if available and pass control to Periscope.

SAMPLE.ASM—This is the source code for the sample assembly program used in the tutorial.

SAMPLE.COM—This is the executable code for the sample assembly program used in the tutorial.

SAMPLE.MAP—This is the MAP file produced when the sample program is linked. It is used to provide symbolic references.

SYMLOAD.COM—This utility program lets your program change Periscope's symbol table while your program is running. It is useful for programs that manage their own overlays or are running under another environment, such as TopView.

SYSLOAD.SYS—This utility program is used to load any COM file as a device driver. It can be used to load Periscope at CONFIG.SYS time, allowing you to debug device drivers.

TS.COM—This utility program is used to verify a MAP file and determine the symbol table size required by the MAP file. It enables you to efficiently allocate space for the symbol table and to generate a Periscope symbol file (PSS). This symbol file is optional for the standard linker (IBM/Microsoft), but is required for other linkers.

USEREXIT.ASM—This sample program illustrates user breakpoints and user exits from Periscope, including a DOS availability test and a display of the 8087/80287 status.

USEREXIT.COM—This is the executable code generated from USEREXIT.ASM.

III—Installing The Hardware

- **Checking For Conflicts**
- **Setting The DIP Switches**
- **Installing The Periscope I Board**
- **Installing The Periscope II Break-Out Switch**
- **Installing The Periscope III Board**

➧ If you're using Periscope II-X, skip this chapter.

➧ If you're using Periscope II, skip to the fourth section in this chapter (Installing the Periscope II Break-Out Switch).

➧ If you're using Periscope I or III, read the first two sections, which apply to both boards. Then read the third section (Installing the Periscope I Board) if you have Model I or the last section (Installing the Periscope III Board) if you have Model III.

Checking For Conflicts

The Model I board uses 56K of memory and two consecutive I/O ports. The Model III board uses 64K of memory and four consecutive I/O ports. For proper operation, the memory and ports used by either board must not be used by any other device. When you receive your board, the DIP (Dual In-line Package) switches are set to use memory starting at D000:0000 and I/O ports starting at 300H.

This area of memory is above DOS memory and the area reserved for screen buffers. The board's default setting may conflict with other add-on memory cards that use

this area for a RAM disk or ROM device drivers. Expanded memory boards typically use this area. Check the documentation for any non-standard expansion cards to see if this is the case.

The I/O ports used by both boards are in the block (300H to 31FH) reserved by IBM for a prototype card. If you have a prototype card in your system, you'll need to check to see which ports, if any, it uses. The use of these ports conflicts with the 3Com Ethernet card, which uses ports 300H through 30FH. If you have this card, try using port 310H. Other expansion options may also use Periscope's default I/O ports. Check the documentation for any non-standard expansion cards to see if this is the case. Note that the true range of I/O ports available is from zero to 3FFH, since the IBM PC only supports the ten low-order bits of a port address.

If you find no conflicts with the memory or I/O ports used by your Periscope board, skip the next section on setting the DIP switches.

Setting The DIP Switches

There are two DIP switches on both the Periscope I board and the Periscope III board. The eight-position switch labeled SW1 controls the I/O ports used by the board. The four-position switch labeled SW2 controls the starting address of memory used by the board.

SW1 is preset to use I/O ports starting at 300H. The switches may be set to indicate any I/O ports on a four-byte boundary. You must not set the switches so that they conflict with other ports in the system. Certain ports are off-limits. These include zero to 100H and others used by expansion cards in your system. If port 300H is not available, try 310H. Consult the IBM Technical Reference Manual and the documentation for your non-standard expansion cards to avoid conflicts.

The switches can be read by laying the board on a table, component (chip) side up, with the top of the board facing you and the mounting bracket to your left. From this vantage point, the address can be read as a binary number. Switches eight and seven make up the first hex number, switches six, five, four and three make up the second hex number, and switches two and one make up the two high bits of the third hex number. When the switch is OFF (up or away from you), it corresponds to a one. When the switch is ON (down or towards you), it corresponds to a zero.

The three hexadecimal numbers correspond to the port address, 00xx yyyy zz00, where 00xx is the first number, yyyy is the second number, and zz00 is the third number. For example, when you receive the board, switches seven and eight are OFF (equal to one) and all others are ON (equal to zero). This is the same as the bit pattern 0011 0000 0000, which is 300H. To change the port setting to 304H, move switch one to the OFF position. Notice that the two missing bits of the first and third numbers are always zero.

The following table illustrates the switch settings.

DIP SWITCH SW1 (I/O PORTS)

	'Hundreds'		'Tens'				'Units'	
Starting Port	S8	S7	S6	S5	S4	S3	S2	S1
000	ON	ON	ON	ON	ON	ON	ON	ON
100	ON	OFF	ON	ON	ON	ON	ON	ON
200	OFF	ON	ON	ON	ON	ON	ON	ON
300	OFF	OFF	ON	ON	ON	ON	ON	ON
300	OFF	OFF	ON	ON	ON	OFF	ON	ON
310	OFF	OFF	ON	ON	ON	OFF	ON	ON
320	OFF	OFF	ON	ON	OFF	ON	ON	ON
330	OFF	OFF	ON	ON	OFF	OFF	ON	ON
340	OFF	OFF	ON	OFF	ON	ON	ON	ON
350	OFF	OFF	ON	OFF	ON	OFF	ON	ON
360	OFF	OFF	ON	OFF	OFF	ON	ON	ON
370	OFF	OFF	ON	OFF	OFF	OFF	ON	ON
380	OFF	OFF	OFF	ON	ON	ON	ON	ON
390	OFF	OFF	OFF	ON	ON	OFF	ON	ON
3A0	OFF	OFF	OFF	ON	OFF	ON	ON	ON
3B0	OFF	OFF	OFF	ON	OFF	OFF	ON	ON
3C0	OFF	OFF	OFF	OFF	ON	ON	ON	ON
3D0	OFF	OFF	OFF	OFF	ON	OFF	ON	ON
3E0	OFF	OFF	OFF	OFF	OFF	ON	ON	ON
3F0	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON
300	OFF	OFF	ON	ON	ON	ON	ON	ON
304	OFF	OFF	ON	ON	ON	ON	ON	OFF
308	OFF	OFF	ON	ON	ON	ON	OFF	ON
30C	OFF	OFF	ON	ON	ON	ON	OFF	OFF

The first section of the table illustrates the use of switches seven and eight to set the 'hundreds' part of the address, the second section illustrates the use of switches three through six to set the 'tens' part of the address, and the third section illustrates the use of switches one and two to set the 'units' part of the address.

DIP switch SW2 is preset to use memory starting at D000:0000. The switches may be set to indicate any area in memory on a 64K boundary. You must not set the switches so that they conflict with other memory installed in the system.

Certain ranges of memory are off-limits. These include—0000:0000 to, but not including x000:0000, where x is the number of 64K banks of RAM installed; A000:0000 to A000:FFFF if an EGA or other enhanced display adapter is installed; B000:0000 to B000:FFFF since this range is used by the monochrome and color/graphics adapters; C000:0000 to C000:FFFF if an EGA is installed or the system is an XT; E000:0000 to E000:FFFF if the system is an AT; and F000:0000 to F000:FFFF on all systems. For example, if your system has 512K, memory below 8000:0000 is already used by RAM, and is therefore off-limits.

The starting memory address can be read by laying the board on a table, component (chip) side up, with the top of the board facing you and the mounting bracket to your left. From this vantage point, the address can be read as a binary number. Switches four, three, two, and one make up the first hex number. When the switch is OFF (up or away from you), it corresponds to a one. When the switch is ON (down or towards you), it corresponds to a zero.

The hexadecimal number corresponds to the highest part of the absolute memory address, x0000 or x000:0000, where x is the first number. For example, when you receive the board, switches four, three, and one are OFF (equal to one) and switch two is ON (equal to zero). This is the same as the bit pattern 1101, which corresponds to D000:0000.

If memory starting at segment D000H is already in use, try using E000H if the system is a PC or an XT. To change the memory setting to E000:0000, move switch two to the OFF position and switch one to the ON position. If the system is an AT, try using C000H. To change the memory setting to C000:0000, move switch one to the ON position.

The following table illustrates the switch settings.

	DIP SWITCH SW2 (MEMORY)			
Starting Address	S4	S3	S2	S1
0000:0000 (0 K)	ON	ON	ON	ON
1000:0000 (64 K)	ON	ON	ON	OFF
2000:0000 (128 K)	ON	ON	OFF	ON
3000:0000 (192 K)	ON	ON	OFF	OFF
4000:0000 (256 K)	ON	OFF	ON	ON
5000:0000 (320 K)	ON	OFF	ON	OFF
6000:0000 (384 K)	ON	OFF	OFF	ON
7000:0000 (448 K)	ON	OFF	OFF	OFF
8000:0000 (512 K)	OFF	ON	ON	ON
9000:0000 (576 K)	OFF	ON	ON	OFF
A000:0000 (640 K)	OFF	ON	OFF	ON
B000:0000 (704 K)	OFF	ON	OFF	OFF
C000:0000 (768 K)	OFF	OFF	ON	ON
D000:0000 (832 K)	OFF	OFF	ON	OFF
E000:0000 (896 K)	OFF	OFF	OFF	ON
F000:0000 (960 K)	OFF	OFF	OFF	OFF

Installing The Periscope I Board

Before installing the board, be sure that the power is off and that the power cord is removed from the PC! To complete the installation, you'll need a small screwdriver.

Step 1—Open the PC by removing 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.

Step 2—The board can be installed in any one of the available expansion slots on the system board, except the slot nearest the power supply in an XT. If you do not plan to use the remote break-out switch, the left-most expansion slot will be the most convenient for reaching the switch located on the board's mounting bracket. Select an available expansion slot and remove the metal bracket from the back panel for that slot, using a small screwdriver. The metal bracket may be discarded, but be sure to save the retaining screw.

Step 3—Align the board with the expansion slot and lower it until the edge connector is resting on the expansion slot receptacle on the system board. Press the board straight down until it seats in the expansion slot. Install the retaining screw through the board's bracket into the PC's back panel and tighten it.

Step 4—Slide the cover of the system unit back over the machine and install the cover mounting screws.

Step 5—If you plan to use the remote break-out switch, install it now, while the power is still off. Remove the dummy plug from the phono jack mounted on the bracket and insert the phono plug that is connected to the remote switch.

If you do not plan to use the remote break-out switch, leave the dummy plug in place. This will help prevent the accidental use of this jack for some other potentially dangerous use—such as the connection of a composite monitor.

Step 6—Re-connect all peripherals and replace the power cord.

Step 7—Boot the system. If you haven't configured the Periscope software, do so now—see Chapter II for more information.

Step 8—Install Periscope by executing PS.COM. If you changed the DIP switches, you'll need to specify the memory and/or port settings as installation options. For example, if the memory was changed to E000:0000 and the port was changed to 304H, enter 'PS /M:E000 /P:304'. See Chapter VI for more information on the installation options. If an error occurs, see Appendix A for an explanation of the error.

Step 9—After installing Periscope, press the remote break-out switch or the switch located on the board's mounting bracket. The Periscope screen should be displayed, showing the current values of the registers and a disassembly of the current instruction. To continue, enter 'G' and press return.

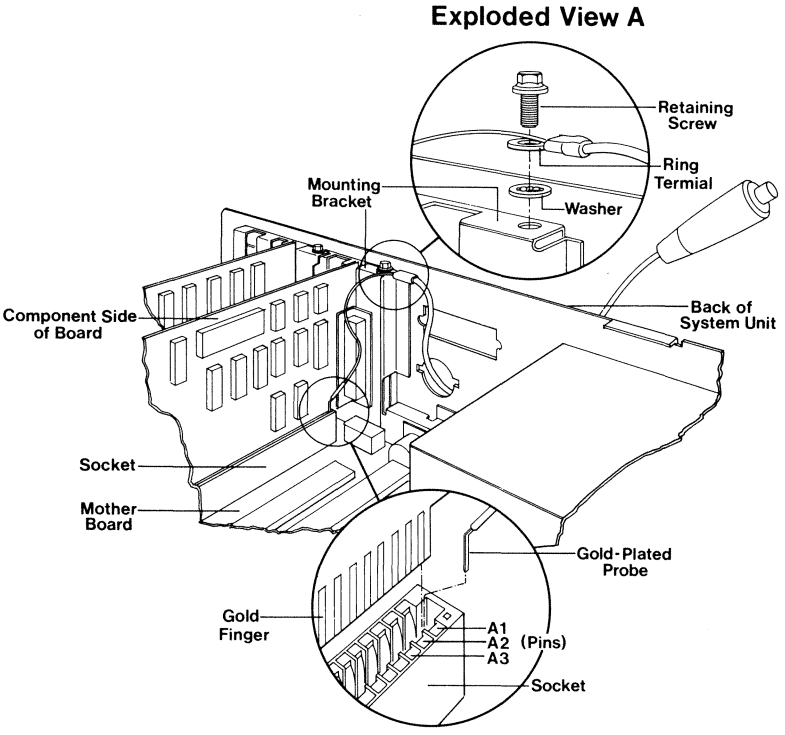
Do not press the break-out switch before PS.COM is installed—you'll get a parity error and have to turn the power off and back on.

If the break-out switch doesn't work when Periscope is first installed on a PC or XT, check the DIP switch setting for the 8087. If you have an 8087 installed, the switch should be OFF. If you don't have an 8087, the switch should be ON.

Installing The Periscope II Break-Out Switch

Before installing the break-out switch, be sure that the power is off and that the power cord is removed from the PC! To complete the installation, you'll need a small screwdriver and a bright light source, such as a

flashlight. Please refer to the illustration below. It shows the switch installed in an IBM PC. Keep in mind that there may be slight physical differences if you're using another machine.



Step 1—Open the PC by removing 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.

Step 2—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) and the other is a gold-plated probe (see Exploded View B).

Route the connector end of the cable assembly into the PC from the back of the system unit. There are several possible ways of routing the cable, 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 crimped and possibly damaged.

Step 3—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. Note—If this slot 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 the illustration. 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—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 the probe to be attached securely.

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 it so that the probe 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 the illustration. **Note:** 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 uninsulated part of the probe from contacting anything other than the desired pin.

Step 5—Double check the placement of the probe. It should be in the pin 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 an extra dummy hole at the end of the socket. Be sure not to 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 the machine and install the cover mounting screws.

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

Step 9—Boot the system. If you haven't configured the Periscope software, do so now—see Chapter II for more information.

Step 10—Install Periscope by executing PS.COM. If an error occurs, see Appendix A for an explanation of the error.

Step 11—After installing Periscope, press the break-out switch. Periscope's screen should be displayed, showing the current values of the registers and a disassembly of the current instruction. To continue, enter 'G' and press return.

Do not press the break-out switch before PS.COM is installed—you'll get a parity error and have to turn the power off and back on.

If the break-out switch doesn't work when Periscope is first installed on a PC or XT, check the DIP switch setting for the 8087. If you have an 8087 installed, the switch should be OFF. If you don't have an 8087, the switch should be ON.

Installing The Periscope III Board

Before installing the board, be sure that the power is off and that the power cord is removed from the PC! To complete the installation, you'll need a small screwdriver.

Step 1—Open the PC by removing the cover mounting screws on the rear of the system unit. Slide the cover of the system unit forward and remove the cover from the system unit.

Step 2—The board can be installed in any one of the available full-length slots on the system board. In an AT,

be sure to remove the plastic cover from the high bus connectors and to install the board in a 16-bit expansion slot. In a PC or XT, leave the plastic cover on the high bus connectors so that the exposed fingers won't cause a short-circuit.

Select an available expansion slot and remove the metal bracket from the back panel for that slot, using a small screwdriver. The metal bracket may be discarded, but be sure to save the retaining screw.

Step 3—Locate the 8087 or 80287 socket on the motherboard. It's a 40-pin socket usually located near the CPU. If you have an 8087 or 80287 installed, note the direction of the 'nose' or notch in the chip and then remove the chip using a small screwdriver or an IC extractor. Install the 8087 or 80287 into the 40-pin umbilical socket supplied with Periscope III. Be sure that the notch on the chip is aligned with the notch on the umbilical socket. **Note:** The motherboard socket may be backwards in some machines. Check your reference manuals to be sure of the direction.

Insert the umbilical socket into the socket on the motherboard, being careful to align the notches on the two sockets. Now press the socket firmly into place. It is very important that the umbilical socket is securely fitted into the motherboard socket! If the umbilical is not installed correctly, the Periscope software will not load.

Step 4—Plug the cable attached to the umbilical socket into the four-pin header near the bracket on the Periscope III board. The connector is keyed so that it can be attached in one position only—with the cable pointing toward the front of the system unit.

Step 5—Align the Periscope III board with the expansion slot and lower it until the edge connector is resting on the expansion slot receptacle on the system board. Press the board straight down until it seats in the expansion slot. Install the retaining screw through the board's bracket into the PC's back panel and tighten it. Route the umbilical cable near the back panel of the system so that it won't be pinched when the cover is installed.

Step 6—If you plan to use the break-out switch located on the board's mounting bracket, screw the enclosed black Bakelite switch cap onto the end of the switch that goes through the board's mounting bracket.

Step 7—Replace the cover of the system unit and install the cover mounting screws.

Step 8—If you plan to use the remote break-out switch, install it now, while the power is still off. Remove the dummy plug from the phono jack mounted on the bracket and insert the phono plug that is connected to the remote switch.

If you do not plan to use the remote break-out switch, leave the dummy plug in place. This will help prevent the accidental use of this jack for some other potentially dangerous use—such as the connection of a composite monitor.

Step 9—Re-connect all peripherals and replace the power cord.

Step 10—Boot the system. If you haven't configured the Periscope software, do so now—see Chapter II for more information.

Step 11—Install Periscope by executing PS.COM. If you changed the DIP switches, you'll need to specify the memory and/or port settings as installation options. For example, if the memory was changed to E000:0000 and the port was changed to 304H, enter 'PS /M:E000 /P:304'. See Chapter VI for more information on the installation options. If an error occurs, see Appendix A for an explanation of the error.

Step 12—After installing Periscope, press the remote break-out switch or the switch located on the board's mounting bracket. Periscope's screen should be displayed, showing the current values of the registers and a disassembly of the current instruction. To continue, enter 'G' and press return.

If the break-out switch doesn't work when Periscope is first installed on a PC or XT, check the DIP switch setting for the 8087. If you have an 8087 installed, the switch should be OFF. If you don't have an 8087, the switch should be ON.

Step 13—To confirm the correct operation of your Periscope III board, execute the program PS3TEST.COM. If you've changed the DIP switches, you'll need to specify the memory and/or port settings as in Step 11 above. If an error occurs while running PS3TEST, please see the instructions on running PS3TEST in Chapter IX.

IV—Tutorial: Using Periscope

- **Tutorial Using Assembler Program**

This chapter takes you through a guided tour of Periscope, using a simple assembly language program. This tutorial demonstrates some of Periscope's debugging commands, but for more detailed information, you'll need to see Chapter VII.

Before you can take the tutorial, you'll need to configure the software (see Chapter II) and install the hardware (see Chapter III). Make the Periscope directory the default directory and then enter 'PS /H /T:2' from the DOS prompt. This will load Periscope into memory, load the on-line help file and interrupt comment file, and allocate 2KB for symbol tables. If you've changed the DIP switches on the board, be sure to enter the memory and/or port options on this line as well.

The program used in this tutorial is named SAMPLE.COM. This simple program displays the total and the available memory in the system. There are four files associated with this program. They are:

PS.DEF—Contains record definitions for the PSP (Program Segment Prefix) and the FCB (File Control Block) and other records

SAMPLE.ASM—The source code for SAMPLE.COM

SAMPLE.COM—The executable program

SAMPLE.MAP—The MAP file produced by the linker when the /M option is used

Periscope uses the DEF file to read keyboard, alias, and record definitions into memory. This file is created with an editor as a standard ASCII text file. In the file

PS.DEF, there are two sample keyboard definitions and three record definitions. Record definitions can be used to display any area of memory in an easy-to-read format. There's a utility program to verify and determine the amount of memory required by a DEF file. See the description of RS.COM in Chapter IX for more information.

Periscope uses the MAP file in order to replace the RAM addresses normally supplied while debugging with the more meaningful labels used in the program being debugged. The MAP file can be created when you link your program by specifying a MAP file and the '/M' option. This file contains the public code and data addresses and their labels. Periscope then uses these symbols to display the labels rather than the numeric addresses. The more symbols you have, the easier it is to debug your program—so we've provided a program to generate as many PUBLIC statements as possible. Also, there's a program to verify and size MAP files and convert non-standard MAP files to Periscope's format. See the descriptions of PUBLIC.COM and TS.COM in Chapter IX for more information.

Before you start debugging SAMPLE.COM, use the DOS TYPE or PRINT command to print a listing of SAMPLE.ASM for reference. The program's mainline code starts at START and ends at DOSRET. The mainline calls three procedures. The first procedure, GETMEM, is called once to retrieve the total memory and the available memory. CONVERT is called twice to convert the memory size from hex to ASCII. Finally, DISPLAY is called once to display the results.

Now that Periscope is installed and you have a listing of SAMPLE.ASM for reference, start the program loader RUN by entering 'RUN SAMPLE.COM' and pressing the return key.

RUN displays the address of the PSP and informational messages. Then the display switches to Periscope's screen and the first instruction of the program is displayed—

```
AX=0000  BX=0000  CX=008B  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=0C73  ES=0C73  SS=0C73  CS=0C73  IP=0100  FL=0246 NV UP EI PL ZR NA PE NC
                SAMPLE:
0C73:0100 EB35          JMP     START
```

Registers BX and CX show the size of the program. Registers DS, ES, SS, and CS all show the PSP segment since this is a COM program. The actual number will vary, depending on the version of DOS and any memory-resident programs you're using.

Also, notice the symbols—the current instruction is labeled SAMPLE, since the name was defined as PUBLIC. The address of the jump is START, not an offset. If you need to know the offset, enter 'U START' to disassemble memory starting at the symbol START.

For help, enter '?' and press return. A command summary is displayed, from which you can see the possible commands. Now enter '? D' to get help on the Display command.

To display the PSP, enter 'DB CS:0'. This shows the first 128 bytes of the PSP in Byte format. For a more useful display, enter 'DR CS:0 PSP'. The record definition 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.COM in Chapter IX). To see the record definitions available, press F7 before entering anything after the Periscope prompt.

To move to the next instruction, enter 'T'. Now you're at the instruction labeled START. Enter 'U' to disassemble the next few lines—

```
              START:
0C73:0137 E81700          CALL    GETMEM
0C73:013A A13301          MOV     AX,[TOTMEM]
0C73:013D BF1001          MOV     DI,0110                         ; TMEMORY
0C73:0140 E82400          CALL    CONVERT
0C73:0143 A13501          MOV     AX,[FREMEM]
0C73:0146 BF2C01          MOV     DI,012C                         ; AMEMORY
0C73:0149 E81B00          CALL    CONVERT
0C73:014C E83400          CALL    DISPLAY
            DOSRET:
0C73:014F CD20            INT     20              ; Program terminate
              GETMEM:
0C73:0151 B106            MOV     CL,06
0C73:0153 BE0200          MOV     SI,0002
0C73:0156 8B04            MOV     AX,[SI]
```

Notice the two lines with the commented references to TMEMORY and AMEMORY. These instructions are ambiguous references to items in the symbol table. This situation occurs when a number is moved to a register. Periscope cannot be sure that the references exist in the source program, so it displays the symbol as a comment. In this case, the instructions actually reference the symbols shown, but a situation where 100H is moved to a register would give a false reference to the program entry point, SAMPLE.

The SA (Search Address) command is used to search for address references. Enter 'SA START DISPLAY CONVERT' to search from START to DISPLAY for references to the procedure CONVERT. Also, try 'SA START DISPLAY TOTMEM' to search the same range of memory for references to the data variable TOTMEM.

Before continuing, set a sticky code breakpoint at the end of the program, DOSRET. To do this enter 'BC DOSRET' and press return.

To see the symbols available, press F8. Now, set a word breakpoint on the value of the variable TOTMEM changing from zero to any other value. To do this enter 'BW TOTMEM NE 0' and press return. To see the current breakpoints, enter 'BA ?'. The result is—

```
BC  DOSRET
BW  TOTMEM NE ØØØØ
```

Now enter 'GT' to execute the program with both of the breakpoints activated. Execution will stop at the instruction after the one that moves register AX into TOTMEM—

```
AX=Ø1ØØ  BX=ØØØØ  CX=ØØØ6  DX=ØØØØ  SP=FFFC  BP=ØØØØ  SI=ØØØ2  DI=ØØØØ
DS=ØC73  ES=ØC73  SS=ØC73  CS=ØC73  IP=Ø15D  FL=Ø212 NV UP EI PL NZ AC PO NC
ØC73:Ø15D 8CCB          MOV     BX,CS
```

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. Press the Esc key to display Periscope's prompt. Now, clear the word breakpoint by entering 'BW *'. Check the breakpoints by entering 'BA ?'. Only the one code breakpoint should be present.

To see the current value of TOTMEM, enter 'DI TOTMEM L2'. This is the total memory in K—note that the result is in decimal, not hex. To show the value in hex, use 'DW TOTMEM L2'. To convert back to decimal, use the translate command. Enter 'X nnnn', where 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.

Use the Jump command to trace through the next few instructions. Enter 'J' and press return. Then press F4 to repeat the previous command. When you get to the 'RET' instruction, use the 'J' command one more time to get back to the mainline code.

Clear the screen by entering 'K' and pressing return. Then disassemble the number conversion routine by entering 'U CONVERT DISPLAY'. To get back to the current instruction, enter 'R' and press return.

To check the number conversion routine, CONVERT, use the Jump command three times to get to the instruction after

the first call to CONVERT. Display the converted value by entering 'DA TOTAL' or 'DA TMEMORY'. This value should agree with the converted value of TOTMEM displayed previously.

Since the sticky code breakpoint for DOSRET is still in effect, enter 'G' to go to DOSRET. Alternately, if the sticky breakpoint did not exist, you could enter 'G DOSRET'.

To view the source file for this program, enter 'V SAMPLE.ASM' and press return. Use the PgUp, PgDn, Home, End, Up, and Down arrow keys to move through the file. When done, press the Esc key to return to the Periscope prompt.

To set Periscope's windows, enter '/W DRSU'. This sets up four windows showing data, register, stack, and disassembly information. The presence, order, and length of these windows can be changed on the fly. Try using Ctrl-F9 and Ctrl-F10 to call up Periscope's standard monochrome and color window settings. Now turn the windows off again by using '/W'.

The remaining commands are not germane to this sample program, but we'll explore some of them here.

To display the interrupt vectors starting with INT 20H, enter 'DD 0:20*4'.

To change the value of register CX, enter 'R CX' and press return. When the colon prompt is displayed, enter '10' and press return. Alternately, you can do the same thing in one line by entering the number after the register—try entering 'R CX 10'. Afterwards, check the result—enter 'R' and press return to display the registers.

Display memory at offset 200H by entering 'DB DS:200'. Now clear this unused memory by using the Fill command. Enter 'F DS:200 L 80 " "' to store 80H bytes of spaces. Now, enter 'D 200' again—

```
0C73:0200  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
           * 0006 LINES OF 20 SKIPPED *
0C73:0270  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
```

See how the six lines of spaces in the middle of the display were omitted?

Use the Enter command to modify memory—'E DS:240 "test"' enters the string 'test' starting at DS:240.

The Search command is used to find a byte/string pattern. Enter 'S 100 400 "test"' to search from offset 100H to offset 400H for the string 'test'. It will be found at offset 240.

The Compare command is used to compare two blocks of memory. Enter 'C 200 L CX 240' to compare the 10H bytes (the value of register CX) starting at offset 200H to the 10H bytes starting at offset 240H. The result might be:

```
0C73:0200  20  74  0C73:0240
0C73:0201  20  65  0C73:0241
0C73:0202  20  73  0C73:0242
0C73:0203  20  74  0C73:0243
```

The first four bytes are different, since the first block contains spaces and the second block contains the string 'test'. The remaining twelve bytes are the same, so nothing is displayed for them.

To copy one block of memory to another, the Move command is used. Enter 'M DS:240 24F DS:200' to copy the contents of 240H through 24FH to 200H through 20FH. Use 'D 200' to verify the move.

To perform hex arithmetic, enter 'H' followed by a number, an operator (+, -, *, or /) and a second number and press return. For example, to subtract 20H from 10H, enter 'H 10-20'.

To quit Periscope, enter 'Q'. The quit options are then displayed. Enter 'R' to return to DOS or 'C' to continue and then press return. Either response ends the debugging session.

V—Tutorial: Periscope And 'C'

- **Tutorial Using 'C' Program**

This chapter takes you through a guided tour of Periscope from the high-level language perspective. This tutorial demonstrates some of Periscope's debugging commands, but for more detailed information, you'll need to take the tutorial in Chapter IV and also refer to Chapter VII.

Before you can take the tutorial, you'll need to configure the software (see Chapter II) and install the hardware (see Chapter III). Make the Periscope directory the default directory and then enter 'PS /H /T:2' from the DOS prompt. This will load Periscope into memory, load the on-line help file and interrupt comment file, and allocate 2KB for symbol tables. If you've changed the DIP switches on the board, be sure to enter the memory and/or port options on this line as well.

The program used in this tutorial is named FTOC.EXE. The four files associated with this program are:

FTOC.C--The source code for FTOC.EXE.

FTOC.DEF—Contains keyboard, alias, and record definitions.

FTOC.EXE—The executable program.

FTOC.MAP—The MAP file produced by the linker when the /LI and /M options are used.

Periscope uses the FTOC.C file to display source code.

Periscope uses the FTOC.DEF file to read keyboard, alias, and record definitions into memory. This file is created with an editor as a standard ASCII text file. In the file FTOC.DEF, there are four sample keyboard definitions, two

alias definitions, and one record definition. See the description of RS.COM in Chapter IX for more information about DEF files.

Periscope uses the MAP file to replace RAM addresses normally supplied while debugging with more meaningful labels from the program being debugged. The MAP file can be created when you link your program by specifying a MAP file and the '/LI' and '/M' options. This file contains the public code and data addresses and their labels. Periscope displays these labels (symbols) rather than the numeric addresses. The more symbols you have, the easier it is to debug your program!

The compiler options to generate line number information in the object file vary from one vendor to the next. The command-line options we're aware of are '-d' for Lattice, '/Zd' for Microsoft, and '-x0', '-x1', or '-x2' for CI.

Some symbol names will vary from one compiler to another. We've used Lattice C, Version 3.00 to compile this sample program. If you're using another compiler, different symbol names will be seen for compiler-generated symbols.

Before you start debugging FTOC.EXE, use the DOS TYPE or PRINT command to print a listing of FTOC.C for reference. Note that the declarations of the variables LOWER, UPPER, STEP, FAHR, and CELSIUS are outside the main—which causes these variables to be made global and available as symbols. Local, stack-based variables are not available as symbols. (Local symbol support is a planned enhancement, however.)

Now that Periscope is installed and you have a listing of FTOC.C for reference, enter 'RUN FTOC.EXE' and press return. RUN displays the address of the PSP and informational messages. Then the display switches to the debugger screen and the first instruction of the program is displayed. **Note:** You won't see any C source code yet, since this instruction is in the prologue, which is written in assembler.

Press Ctrl-F9 (monochrome) or Ctrl-F10 (color) to set Periscope's standard windows. The data window is shown at the top of the screen, then the register and disassembly windows. The vertical stack window is shown at the right, with the current stack pointer at the top. The value of the BP register is indicated by a chevron at the left of the stack window.

To monitor the variables FAHR and CELSIUS in the data window, enter 'DS FAHR'. Since CELSIUS immediately follows FAHR in memory, both variables can be seen on one line, labeled FAHR. Neither variable is initialized yet, so their present values can be ignored.

Press function key F5 to see the two aliases defined in the FTOC.DEF file. Aliases are a method of identifying items to Periscope. The 'MP' alias identifies a module path and the 'MX' alias identifies a module extension. Both are used for source-level debugging. The 'MP' alias defines the path name to be used when displaying a source file and the 'MX' alias defines the file extension. Recent versions of the linker put the module name in the MAP file, so Periscope concatenates the path, name, and extension to get the module's full file name.

The EA (enter alias) command is used to modify aliases. The syntax of this command is: 'EA <alias> [<name>]', where the alias must be exactly two characters and the name must be from zero to 16 characters. If the current drive is not 'C:', change the 'MP' alias by entering 'EA MP X:' where X is the module path. If you enter 'EA MP' with no argument, the alias is deleted. When a path name is used, end the MP alias with a back-slash! The 'MX' alias correctly shows '.c', so no change is required.

For compiler products that put the module name and extension into the MAP file, the 'MX' alias should not be used. Periscope can read a module name of up to eight characters and an extension of up to four characters (including period) from the MAP file.

The current instruction is in the prologue or initialization code. To get to the actual program, enter 'G MAIN' (use 'G _MAIN' for Microsoft C). Periscope now shows the C source code with the intervening assembler instructions in the disassembly window. Ordinarily, if the MP and MX aliases are correctly set and a recent version of LINK is used, you won't be prompted for the source file name. If you are ever prompted for the name, press the F3 key to see what file Periscope attempted to use. You can edit this file name and press return to try again.

Now, go to line 10 of the program by entering 'G A10'. Now that you're on a source line, enter 'JL' to get to the next source line in the program. The Jump Line command uses the Jump command until the 'current' instruction is the beginning of a source line. To use it, you must be on a source line!

Periscope has three disassembly modes: UA for assembly only; UB for both source and assembly; and US for source-only. The source-only display shows assembly code only until the first source line reference is found and then switches into source-only mode. Enter 'UA', 'US', and then 'UB' to see the difference in the display. Unless the '/E:0' installation option is used, Periscope assumes 'UB' mode. See Chapter VII for more information.

The 'J' command executes to the next sequential assembly level instruction. The 'T' command is the same as the 'J' command, except that it will trace into called routines and interrupts, instead of keeping to the next sequential instruction. Try both commands a few times and note how the disassembly window changes.

Now, enter 'G PRINTF' to get to the next call of the library procedure PRINTF. At this point, you're away from your program's source code, so no source lines are shown. To get back to the next source code line, enter 'BL+;GT' to turn line breakpoints on and execute in 'monitor' mode until the next instruction corresponding to a source line is executed. This method of finding your source is very slow, but can be a handy way of getting back to your source code. You'll now end up at line 18 in the program. Check the values of FAHR and CELSIUS as displayed in the window versus the values just displayed by the program.

Enter 'G A18' a few times to go to the next execution of line 18. Notice how the values of FAHR and CELSIUS in the data window change each time.

We've set up a sample record definition called 'VARS' that describes the data variables in this program. To display the variables starting at the symbol LOWER, enter 'DR LOWER VARS'. Since a keyboard definition was set in the DEF file, you can also use Ctrl-F4 to call out this command.

To view the source file for this program, enter 'VS' and press return. Use the PgUp, PgDn, Home, End, Up, and Down arrow keys to move through the file. When done, press the Esc key to return to the Periscope prompt.

To quit Periscope, enter 'Q'. The quit options are then displayed. Enter 'C' to continue and then press return to end the debugging session.

VI—Installing The Periscope Software

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

Installation Options

To load Periscope, enter 'PS' from the DOS prompt, followed by the desired installation options. (See the next section in this chapter for alternate start-up methods). Periscope has the following default values (in alphabetical order by installation option):

43—The standard 25-line mode is assumed.

A—One monitor is assumed.

B—The software trace buffer is presumed to be 1K (enough for 32 entries).

C—The screen color is low-intensity white on black.

D—The original INT 13H (disk) vector is not restored before a short boot.

E—The source file buffer is presumed to be 1K.

F—Slow color output is presumed, unless an EGA is used.

H—On-line help and interrupt comments are not available.

I—No user interrupt exit is available.

L—Periscope's external tables are loaded as low in memory as possible.

M—The write-protected memory starts at D000:0000.

N—For Periscope I and III, the software is loaded into the protected memory.

P—The ports used by the board start at 300H.

Q—The system is not a PC or XT with a 80286 'turbo' card.

R—The record table size is presumed to be 1K.

S—The program's screen size is presumed to be 4K, which is sufficient for text mode only.

T—The symbol table size is presumed to be 1K.

V—The BIOS interrupt vectors used by Periscope (8H, 9H, 10H, 15H, 16H, 17H, and 1CH) are set to their power-on values each time Periscope is used, and restored to their original values when Periscope is exited.

W—Windows are turned off.

Z—External tables are not suppressed.

➧ For Periscope II and II-X, the M, N, P, and Z installation options do not apply.

Use the installation options described below to change the above defaults. All options contain a slash (/) and a single letter mnemonic, except for the '/43' option. Some of the options include a number. If a number is used, it is always preceded by a colon (:) and the number is always in hexadecimal notation. Periscope can be installed multiple times per DOS session, but each install allocates more memory.

The installation options are:

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

/43—Use the Enhanced Graphics Adapter (EGA) 43-line mode. When this option is used, Periscope supports a single monitor in 43-line mode, presuming that the display is already in 43-line text mode when the resident portion of Periscope is activated. This option reserves 7KB of memory for Periscope's screen and 7KB for the application's screen. Dual-monitor support is not currently available for 43-line mode. Periscope will use 25 lines

unless the screen was already in 43-line text mode. If Periscope is set to 43-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 debug screen. This option indicates 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 the break-out switch is pressed or when a program is loaded with RUN.COM. If this option is used, the /S option is ignored and no memory is reserved for the program's or Periscope's screen buffer.

/B:nn—Set the size of the software trace buffer to something other than 1K. This option is used 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 number of K desired. The number may be from zero to 3FH K, allowing a maximum of 2016 entries. Remember that the input is in hex!

➧ Periscope III's hardware trace buffer is always 8192 entries deep. See Chapter XI.

/C:nn—This option sets the color attribute for Periscope's display. The two digit number is from 1 to FFH. The border of the screen is set to the same color as the background. (If an Enhanced Color Display (ECD) is used on an EGA, no border color can be set.) To calculate the number you want (I use 17H—gray on blue) see your machine's technical reference manual or the table below.

The layout of the color attribute bits is:

Bit number	7	6	5	4	3	2	1	Ø
Use	X	R	G	B	H	R	G	B
		background color				foreground color		

X - blink if 1, else no blink
R - red gun on if 1, else off
G - green gun on if 1, else off
B - blue gun on if 1, else off
H - high-intensity if 1, else normal

The RGB combinations are:

```
Green plus Blue is Cyan
Red plus Blue is Magenta
Red plus Green is Brown (Yellow if high intensity)
Red plus Green plus Blue is Gray (White if high intensity)
```

Assuming that you want cyan on black, use 0000 0011 binary, or '/C:3'. There are some illegal color combinations that Periscope won't allow. These include 0, 8, 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. This option is used with certain RAM disk software to re-point the diskette interrupt vector to BIOS before using the short boot option. It is needed only if the RAM disk software you use modifies the original interrupt 13H to point to memory that is corrupted by a short boot.

/E:nn—Set the size of the source file buffer to something other than 1K. This option is used to improve performance when the Unassemble Source or View file commands are used. The one or two-digit hexadecimal number nn is the number of K desired. The number may be from zero to 10H K. Remember that the input is in hex!

➧For Models I and III, the source file buffer is placed in the protected memory if space is available and if the buffer size is 1KB. If the size is larger than 1KB, it is always placed in low (DOS) memory.

/F—Set fast output for a color monitor. This option should be used if you have a 'snow-free' color card. The original IBM color card suffers from 'snow' when you attempt to write directly to the screen buffer. A snow-free card allows you to write directly to the screen without any interference. If you have one of these snow-free cards, use this option to speed up the screen display. This option is assumed if an EGA is used.

/H—Install the on-line help and interrupt comments. If this option is specified, the files PSHELP.TXT and PSINT.TXT are loaded into RAM and will be available from Periscope. The files must be in the current directory or must be able to be found via the Periscope path. To set the Periscope path, enter 'SET PS=XXX' at the DOS prompt, where XXX is the desired 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. It can be edited as needed 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. Two versions of Periscope's help file are provided. PSHELP.TXT is the larger version, containing syntax, use, and an example of each of Periscope's commands. PSHELP2.TXT is the smaller version, containing syntax and usage descriptions, but no examples. To use the smaller help file, rename it to PSHELP.TXT and use the /H installation option.

The file PSINT.TXT is a normal ASCII text file. It can be edited as needed to add or remove interrupt information. Each line contains an interrupt number, a space, a function number (value of register AH), a space, and a description of up to 30 characters followed by a carriage return and line feed. The two numbers may be from 00 to FF. If the function number is '..', the associated description is displayed for any value of AH or when the interrupt is not the current instruction.

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

/I:nn—This option is used to allow access to user-written code from Periscope. The program must be a memory-resident routine that is already installed using an interrupt from 60H to FFH. It must meet the specifications as defined for the program USEREXIT.ASM in Chapter IX. The two-digit hexadecimal number nn must be from 60H to FFH.

/J and /K—These options are discontinued in Version 3. See the description of PSKEY.COM in Chapter IX for the current hotkey technique.

/L:nnnn—Load Periscope's tables starting at the specified segment. Normally, Periscope's external tables are loaded as low as possible (just after DOS). If you're debugging non-DOS programs, this option can be used to place Periscope's tables 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 1000H paragraphs. For example, if the PSP is C00H and the top of memory is 5000H, the limits for this option would be C10H through 4000H.

/M:nnnn—Set the protected memory segment to something other than D000H. Periscope I has 56KB of protected memory. Periscope III has 64KB of protected memory. The four-digit hexadecimal number nnnn represents the segment to be used. Be sure that the segment used does not conflict with other memory installed in the system

and that the desired segment is indicated by the DIP switches on the board. If you must change the default setting, be sure to carefully follow the memory switch setting procedures in Chapter III.

➧ The '/M' installation option is not available for Periscope II or II-X.

/N—Run the Periscope I or III software without using the protected memory. If you need to run the Model I software without the Periscope I board, use this option. When this option has been used, the software does not clear the Model I switch. If you want to use the switch, you can clear it by outputting any value to the second port on the board (e.g. '0 301 0' for the standard configuration). For Model III, this option loads the software into normal system memory, but the Model III board must still be present for Periscope III to work.

➧ The '/N' installation option is not available for Periscope II or II-X.

/P:nnn—Set the starting protected memory port to something other than 300H. Periscope I uses two consecutive ports and Periscope III uses four consecutive ports. Be sure that the ports used do not conflict with other ports installed in the system and that the desired port is indicated by the DIP switches on the board. If you must change the default setting, be sure to carefully follow the port switch setting procedures in Chapter III.

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.

➧ The '/P' installation option is not available for Periscope II or II-X.

/Q—Indicates a hybrid system that has a PC or XT style motherboard and an 80286 CPU board. This option must be specified if your system is an AT&T 6300 Plus or if you have a 286 'turbo' card in a PC-class machine.

/R:nn—Set the size of the record definition table to something other than 1K. This option is used when debugging programs with large DEF files and large resultant record tables. The one or two-digit hexadecimal number nn is the number of K desired. The number may be from zero to 20H K. Remember that the input is in

hex! See the description of RS.COM, which determines the record table size required for a DEF file, in Chapter IX.

/S:nn—Set the size of the program's screen buffer to something other than 4K. This option is ignored if the /A option is used. It is used when debugging programs that use the color-graphics adapter (CGA) or EGA. It is only required for single-monitor systems where a 4K screen buffer is too small. The one or two-digit hexadecimal number nn is the number of K desired, from zero to 20H K. If you need 16K, enter '/S:10'. Remember that the input is in hex! The maximum size allowable is 32K, using '/S:20'. 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 on the EGA that need more than 32KB, a dual-monitor system is required to preserve your graphics screens.

/T:nn—Set the size of the symbol table to something other than 1K. This option is used when debugging programs with large MAP files and a large resultant symbol table. The one or two-digit hexadecimal number nn is the number of K desired. The number may be from zero to 3FH (63) K. Remember that the input is in hex! See Chapter IX for information on TS.COM, which determines the symbol table size required for a MAP file and generates compact memory-image symbol files.

/V:nn—Indicate a BIOS interrupt vector that is to be left alone. Normally, each time you enter Periscope, it saves your program's interrupt vectors then temporarily resets the interrupt vectors it uses to their power-on default values. This is done to make Periscope as dependable as possible. When control is returned to the interrupted program, the interrupt vectors are restored to their values on entry to Periscope.

If you have a situation where you want Periscope to use your modified interrupt vectors while it is running, use the /V:nn option, where nn is the one or two digit hexadecimal interrupt number. The possible numbers are 8 (timer), 9 (keyboard), 10H (video), 15H (cassette/ scheduler), 16H (keyboard I/O), 17H (printer), and 1CH (timer control). Note that each vector must be entered 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 these changes cannot be overridden.

Note: The '/V:10' option is no longer required for an EGA.

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.COM and use '/V:9' to be able to switch screen modes from within Periscope.

- If you're using a serial printer, run MODE.COM and use '/V:17' to be able to access your printer from within Periscope.

- If you're using AST Fastdisk Version 1.40, use '/V:15' to avoid problems writing to the RAM disk from within Periscope.

/W—This option is used to set Periscope's windows. Please see 'Option W' in Chapter VII for the syntax and a complete description. (The window specification can be entered as an installation option when PS.COM is run or as a command from within Periscope.) Periscope's standard window settings for monochrome and color systems are available via Ctrl-F9 and Ctrl-F10 respectively.

/Z—This option is used to keep Periscope from using any memory external to the Periscope board. It overrides other options that allocate memory outside of the memory on the board. A warning message is displayed when it cancels options already selected.

➧ The '/Z' installation option is not available for Periscope II or II-X.

The installation options can be entered 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 /W D:8 R U: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 16K to save the program's screen on a single-monitor system.

PS /T:20 /V:10 /H—Reserve 32K 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 zero K to over 128K. These buffers are located in normal RAM using the terminate-and-stay-resident function of DOS. The protected memory on the Periscope board is used for the code and data areas of Periscope and as much of the external tables as will fit. In many cases, no memory is required other than the Periscope board's protected memory.

If any errors are encountered during the initialization process, Periscope 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—try running RUN.COM instead. If Periscope is not installed, RUN will display an error message.

Alternate Start-up Methods

Periscope can be installed via a full-screen display that lets you choose among all possible options. To use this method, enter 'PS *' at the DOS prompt. The screen shown below is displayed.

```
Periscope Version 3.10  Copyright 1986, The Periscope Company, Inc.
                                                    Key usage
43 - Use 43-line mode on an EGA (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 (0-10H kb) [01]         Esc - exit to DOS
F - Use fast color output (y/n)? [N]                F9 - write response
H - Install help and interrupt comments (y/n)? [N]     file & install PS
I - User interrupt vector number (0-FFH) [00]       F10 - install PS
L - Load Periscope tables at segment [0000]
M - Protected memory segment (xxxx-E000) [D000]
N - Run without using protected memory (y/n)? [N]
P - Protected memory port (100-3FC) [300]
Q - Hybrid system: PC or XT with 80286 CPU (y/n)? [N]
R - Record definition table size (0-20H kb) [01]
S - Screen buffer size (0-20H kb) [04]
T - Symbol table size (0-3FH kb) [01]
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]
W - Window types & lengths: Data  (0-14H) [00]  Register (0 or 2) [00]
          Stack (Y/N) [N]    Unasm (4-14H) [00]    View (0-14H) [00]
Z - Use zero RAM other than protected memory (y/n)? [N]
```

All 'information' messages are suppressed when this screen is used—only error messages are displayed. When function key F9 is used, this full-screen display generates a response file named PS that can later be used to start Periscope with 'PS @PS'.

Normally, we recommend that you invoke PS.COM from an AUTOEXEC.BAT file to ensure its presence each time the system is booted. If, however, you sometimes need different Periscope options for debugging different types of programs, the response file is for you. This file is an ASCII text file that contains any PS.COM installation options. For example, if you create a file named C:STD that contains '/B:4 /V:10 /A', you can enter 'PS @C:STD' to load Periscope using the options found in the file named C:STD. You can use 'PS /C:17 @C:STD' to set the color attribute and then retrieve the options from the file named C:STD. Any options entered after a response file name are ignored. For example, 'PS @C:STD /C:17' would not set the color attribute. The file name used for a response file may be any legal file name.

VII—Debugging With Periscope

- **The Quit Options**
- **Keyboard Usage**
- **Command Parameters**
- **The Periscope Commands**

The Quit Options

When you use 'Q' to quit Periscope, the quit options are displayed. The prompt is:

```
(XX) Boot (B), Continue (C), Debug (D), Long boot (L), Return to DOS (R), or
Short boot (S)?_
```

The first value in parentheses indicates the method that was used to activate Periscope. The possible values are:

BR—A monitor breakpoint was taken.
EX—An exception interrupt (6 or DH) occurred on the 80286 CPU.
GO—A code breakpoint was taken.
HW—A hardware breakpoint occurred (Periscope III only).
P1—Parity error 1 (motherboard) occurred.
P2—Parity error 2 (expansion memory) occurred.
SW—The break-out switch was pressed. **Note:** this also appears for entry via hotkeys set with PSKEY.COM.
TR—An instruction was traced.

➧ When the Periscope II break-out switch is used, 'P2' is shown instead of 'SW' and the message 'Parity error 2 (break-out switch)' is displayed on entry to Periscope. This is normal for Model II and should be ignored.

The **Boot (B)** option performs the same function as Alt-Ctrl-Del, clearing all of user memory and resetting the standard interrupt vectors. This option can be used when the system is hopelessly confused or when you suspect

that a runaway program may have incorrectly modified a critical area of memory. If an exception interrupt occurs, you should boot the system as soon as possible.

The **Continue (C)** option 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 to set breakpoints.

The **Debug (D)** option is used to return to the Periscope prompt.

The **Long boot (L)** option puts the system through full diagnostics, as when the system is first powered on. It should be used if you need to reinstall any BIOS drivers such as the EGA.

The **Return to DOS (R)** option is used to abandon execution of the current program and return to DOS. Note that any open files will not be closed—this can cause problems if the files have been updated. Any changes the program has made to interrupt vectors will not be backed out. See the description of the Interrupt Save and Restore (IS and IR) commands. Also, the keyboard buffer is cleared. When possible, use 'QC' or 'G' instead of this command.

The **Short boot (S)** option is used to re-boot 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.

Periscope can be activated by unexpected events. When one of these events occurs, the appropriate message is displayed: 'Parity error 1'; 'Parity error 2 (break-out switch)'; or 'Exception interrupt'. Parity error 1 indicates a parity error on the motherboard. Parity error 2 indicates a parity error in the expansion bus (this message is also triggered by Model II's break-out switch). An exception interrupt occurs on an 80286 machine when an illegal instruction is executed or when a segment wraparound occurs.

Keyboard Usage

Periscope has a DOSEDIT-like command editor. Previous commands are kept in a circular buffer that is 512 bytes in size. All lines above the minimum length set by CONFIG.COM are saved, except when F3 or F4 is used to repeat a previous line. The following editing keys are supported:

Home—Move the cursor to the start of the command line.

End—Move the cursor to the end of the command line.

Up Arrow—Display the previous command line from the circular buffer.

Down Arrow—Display the next command line from the circular buffer.

Left Arrow—Move the cursor one position to the left.

Right Arrow—Move the cursor one position to the right.

Ins—Toggle insert mode on and off.

Del—Delete a character from the command line.

Backspace—Delete the current keystroke and back up one character.

Ctrl-End—Erase from the current cursor position to the end of the command line.

Ctrl-Left—Move to the start of the previous word in the command line.

Ctrl-Right—Move to the start of the next word in the command line.

Esc—Erase the command line.

Ctrl-PgDn—Remove the current command line from the circular edit buffer.

Ctrl-PgUp—Clear the entire circular edit buffer.

The function keys used by Periscope are:

F1—Toggle code timing on and off. When this mode is turned on, the message 'Code timing on' is displayed.

Any subsequent Go or Jump (but not Jump Line) commands time your code in increments of 838 nanoseconds (the standard 55 millisecond timer-tick divided by 64K). On return to Periscope, the timed value is displayed as a decimal number. This method is very accurate, except for very short duration events of less than 20 to 30 ticks. Due to the overhead imposed by Periscope's entry and exit code, extremely short duration events are not timed correctly. If the time displayed is less than 20 to 30 ticks or is displayed as 'N/A', run the stand-alone timer test program (available on request) or exercise the code multiple times and take an average. The maximum event length that can be timed is 655,359,999 times 838 ns, or approximately one hour. The I/O ports used by this technique are 40H and 43H. To turn code timing off, press F1 again. The message 'Code timing off' is then displayed. Note that your code is run at full speed when the code timing is on.

F2—Toggle screen swap on and off when Periscope is used on a single monitor. This key has no effect when the /A installation option is used. When off, this mode keeps Periscope from displaying the program's screen when a Go, Jump, or Trace command is used. 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 Go, 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.

F3—Copy the remainder of the previous command line into the current command line. This key copies up to, but does not include the carriage return. The command line is not added to the circular edit buffer again.

F4—Same as F3, except that a carriage return is added at the end of the command line. For repetitive commands, you can use just one keystroke—F4.

F5—Display the current aliases. If the cursor is at the beginning of a command line, all aliases are displayed. You can display alias names that start with a character sequence by entering the desired characters and then pressing F5. For example, to display all alias names starting with 'C', enter 'C' at the start of a command line and press F5. Note that the alias name search is case sensitive and that the search is for the alias name, not the alias itself! Be sure not to enter any spaces before or after the search name.

F6—This key selects 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 and waits for a key press before scrolling another screen full of information into view. 'Pause/clear on' differs from the 'Pause on' mode in that it clears the screen after a key is pressed and 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 a key is pressed. The 'Pause off' mode continually scrolls the non-windowed area of the screen.

F7—Displays the current record definitions. If the cursor is at the beginning of a command line, all record definitions are displayed. You can display record definitions that start with a character sequence by entering the desired characters and then pressing F7. For example, to display all record definitions starting with 'PS', enter 'PS' at the start of a command line and press F7. Be sure not to enter any spaces before or after the search name.

F8—Displays the address and name of the symbol table entries. If the cursor is at the beginning of a command line, all symbols are displayed. You can display symbols that start with a character sequence by entering the desired characters and then pressing F8. For example, to display all symbols starting with the letter 'A', enter 'A' at the start of a command line and press F8. Be sure not to enter any spaces before or after the search name.

F9—Toggle call tracing on and off. When this mode is turned on, the message 'Call trace on' is displayed. When a GA or GT command is used, any CALL instructions executed are displayed on Periscope's screen. Nested calls are indicated by leading spaces before the address, up to a maximum of 16 levels deep. For example, the call trace of the FTOC.EXE program from the beginning to MAIN (using GT) is:

```
2A9C:01C2 E80715          CALL   RBRK
 2A9C:170A E89409          CALL    RBRK
2A9C:02B6 E84C19          CALL   MAIN
 2A9C:1C3D E8E401          CALL  _GETFC
 2A9C:1C89 E86DE6          CALL   MAIN
```

This mode is best used on a dual-monitor system. If you have one monitor, turn screen swap off using F2 if possible. A procedure label or source code may be shown along with the disassembled call, in which case the label or source code is correctly indented, but the associated call is not. When this mode is turned off, the message 'Call trace off' is displayed.

F10—Switches from Periscope's screen to the program's screen if only one monitor is being used. If the /A installation option was used or if screen swap is off, this key has no effect. To return to Periscope's screen from the program's screen, press any key.

Alt-F1 through Alt-F10—Saves the current command line of up to 64 characters. Enter a command and press Alt and a function key to save the command for later recall using Ctrl-Fn. To get a carriage return after the saved command, enter a semi-colon as the last character of the command before saving it. Key assignments may be read from the DEF file—see the description of RS.COM in Chapter IX.

Ctrl-F1 through Ctrl-F10—recalls the command line saved by Alt-Fn. The recall function can be used anywhere within a command. To easily remember the key usage, think of Alt-Fn as 'A'ssign and Ctrl-Fn as 'C'all.

Other keys are defined as follows:

Alt-S—Toggles the vertical windowed stack display on and off. At least one window must be used for this key to have any effect.

Ctrl-Break—Cancels the current command and returns to Periscope's prompt.

Ctrl-PrtSc—Toggles 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-S—Suspends output until another key is pressed.

PadMinus (gray minus key on numeric pad)—Moves backward one line in the current window. This command affects the current data window if the last command is a display command or if no disassembly window is in use. To enter a minus sign, use the other minus key.

PadPlus (gray plus key on numeric pad)—Moves forward one line in the current window. This command affects the current data window if the last command is a display command or if no disassembly window is in use. To enter a plus sign, use the other plus key.

PgDn—Pages forward through the current window. This command affects the current data window if the last command is a display command or if no disassembly

window is in use. Otherwise this key affects the disassembly window.

PgUp—Pages backward through the current window. This command affects the current data window if the last command is a display command or if no disassembly window is in use. Otherwise this key affects the disassembly window.

Semi-colon—This character is used as a pseudo carriage-return. Use it to enter multiple commands on one line. For example, if you're tracing through a program that requires repetitive Go and Fill commands, you could enter 'G NEWPAGE;F PAGENO L2 0' to go to the line labeled NEWPAGE and fill memory starting at PAGENO. After the line has been entered once, you can use F4 to repeat it.

Shift-PrtSc—Prints the entire screen to the parallel printer. Be careful if control codes have been displayed on the screen with the Display or Xlate commands—use Ctrl-PrtSc to avoid output of control codes to the printer.

Command Parameters

Periscope is command driven. Commands may be entered in upper or lower case. Either a space or a comma may be used to delimit parameters within a command. A delimiter is required when a sub-function is omitted, after a symbol, and between two numbers.

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

The various parameters used by Periscope are defined below, in alphabetical order. Brackets ([ ]) in the command syntax are used to indicate an optional entry. (Note that brackets actually entered in a command line are used to indicate that the address is to be used as a near pointer.) An ellipsis (...) is used to indicate a repetitive entry.

$—The dollar sign or 'here' indicator can be used 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, the DR command can be used to display

repeating record definitions. For example, 'DR $ .RECORD' can be used to display a repeating fixed-length record.

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

<alias>—A two-character short-hand notation for a Periscope variable. The valid aliases are:

MP—The module path name for source-level debugging
MX—The module extension for source-level debugging
X1—The command executed on entry to Periscope
X2—The command executed after each Periscope command
X3—The command executed on exit from Periscope

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

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

<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.

<flag>—A flag register. The possible values and two-character mnemonics are:

FLAG	SET (=1)	CLEAR (=Ø)
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)
Carry	CY (STC)	NC (CLC)

<function>—The Periscope command, such as D (Display memory), or U (Unassemble).

<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.

A register name may be substituted for the number in either format. The current value of the register is used for the number.

A symbol may also be used 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.

<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 file name, including drive, path, and extension as needed. May also be any name for use as an alias.

<number>—A one- to four-digit hexadecimal number from 0 to FFFF. If a register name is used, its current value is substituted for the number.

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

<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. Two symbols may be used 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, SS, CS, and IP. The 8-bit registers are AH, AL, BH, BL, CH, CL, DH, and DL.

<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.

****—A one- to four-digit hexadecimal number or register representing one of the four segment registers (Code, Data, Extra, or Stack).

<string>—A quoted list of ASCII characters. Either single or double quotes may be used 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 the single embedded quote is desired.

<sub-function>—The mnemonic used with most commands. For example, to display memory in word format, enter 'DW', where 'W' is the sub-function. The sub-function must follow the function immediately—no intervening spaces are allowed. This is necessary to differentiate between a sub-function and an address. For example, consider 'DD' and 'D D'. The first command displays data in double word format starting at the current segment and offset. The second command displays data in the current format starting at offset D in the current segment.

<symbol>—A name corresponding to an address or a record definition. Symbols are loaded from a PSS or MAP file when the corresponding program is loaded by RUN. A symbol name may be optionally preceded by a period. (If a period precedes a name, the name is forced to be a symbol.) For example, to disassemble memory starting at the symbol 'PRINT_LINE', enter 'U .PRINT_LINE'.

Symbols are evaluated 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—the command 'D A123' first tries to use A123 as a symbol name. If a symbol is not found, A123 is used as a hex number. To inhibit its use as a hex number, use 'D .A123'.

Symbols are also used to reference record definitions read from a DEF file. For example, to display the FCB, enter 'DR CS:5C FCB'.

<test>—Used to 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).

[ ]—Brackets around an address are used to indicate that the offset is to be used as 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.

{ }—Braces around an address are used to indicate that the segment and offset are to be used as 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.

O.—A prefix of 'O.' before a symbol name extracts the offset portion of a symbol name. For example, to modify the instruction pointer to the offset portion of the symbol NEW_PAGE, use 'R IP O.NEW_PAGE'.

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'.

+, -, *, /—These arithmetic operators may be used to perform inline arithmetic. The operators + (add), - (subtract), * (multiply), and / (divide) are evaluated from left to right. For example, 'dd 0:21*4', 'r ip ip+1', 'd ss:sp-4', 'u bx+si-5' are all acceptable.

The Periscope Commands

Periscope's commands are described on the following pages. Because of the number, complexity, and importance of the breakpoint commands, a few words of introductory information may help you understand the different types.

There are three categories of breakpoints: Code breakpoints, Monitor breakpoints, and Hardware breakpoints. **Code breakpoints** are set on specific addresses in your program. When the instructions at the specified addresses are about to be executed, a breakpoint is taken. **Monitor breakpoints** are a wide class of dynamically-evaluated conditions. **Hardware breakpoints** are set with the Periscope III board only. They are set on the real-time bus events that occur during the execution of your program.

Code breakpoints are set via the 'BC' (Breakpoint on Code) command. These breakpoints are 'sticky' code breakpoints since they are remembered until explicitly cleared with the 'BC' or 'BA' commands. Temporary code breakpoints are set when an address is entered with the 'G' (Go) command. They are not remembered after the execution of the 'G' command that set them. All of the Go

commands (G, G=, GA, GH, GM, GT) activate Code breakpoints.

Monitor breakpoints are set via these commands: 'BB' (Byte), 'BI' (Int), 'BL' (Line), 'BM' (Memory), 'BP' (Port), 'BR' (Register), 'BU' (User), 'BW' (Word), and 'BX' (eXit). They are all 'sticky', or remembered until explicitly cleared. They are activated by the GA, GM, and GT commands only.

Hardware breakpoints are set via the 'HM' (Hardware Memory) and 'HP' (Hardware Port) commands. The 'HB' (Hardware Bit), 'HC' (Hardware Control), and 'HD' (Hardware Data) breakpoint commands are used to qualify the Hardware Memory and Hardware Port breakpoints. All Hardware breakpoints are 'sticky', or remembered until explicitly cleared.

Breakpoint command terms that may be confusing at first are: display, set, clear, enable, and disable. 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 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. Disabled breakpoints are displayed with a leading '-'.

To clear all breakpoints, use the 'BA *' command (for the software breakpoints) or the 'HA *' command (for the hardware breakpoints). To clear an entire group of breakpoints, use the 'Bx *' command or the 'Hx *' command, 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.COM, 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.

See the Help command later in this chapter for information on using Periscope's on-line help function.

Command: Assemble to memory

Syntax: A [<address>]

Description: This command assembles instructions to memory.

To use the in-line assembler, enter 'A [<address>]' when Periscope's prompt is displayed and press return. The assemble address is then displayed. If no address is specified, CS:IP is used. 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 8086, 8087, 8088, 80186, 80287, and real-mode 80286 opcodes. The protected-mode opcodes of the 80286 are not supported. If a prefix instruction other than a segment override (CS:, DS:, ES:, or SS:) is used, it must be on a separate line preceding the instruction it affects. Various forms of the opcodes are supported, 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 a far return must be entered 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, the symbol name may be preceded by 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. Symbols may also be used as arguments to JMPs and CALLs.

For instructions that require the phrase 'BYTE PTR' or 'WORD PTR' to specify the width of the operation, use B or W, in upper or lower case, instead. Some 8087 and 80287 instructions require a width indicator of D, Q, or T for Double word, Quad word, or Ten byte respectively.

Periscope 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 NEW_PAGE, enter 'A 1234:5678' and

press return. Then enter 'JMP NEW_PAGE' and press return. Press return again to exit the in-line assembler.

Command: Assemble then Unassemble

Syntax: AU [<address>]

Description: This command is the same as the Assemble command described previously, except that it disassembles an instruction immediately after assembling it.

This immediate feedback was originally used to debug the in-line assembler. We left it in Periscope since users reacted positively to it.

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 return. The instruction is disassembled and then the next prompt is displayed. Press return again to exit the in-line assembler.

Command: display, clear, enable, and/or disable All software Breakpoints

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

Description: This command displays (?), clears (*), enables (+), and/or disables (-) the currently-set software breakpoints. If you enter 'BA' only, display is assumed, and all currently-set breakpoints are displayed. Disabled breakpoints are displayed with a leading '-'.

Examples:

Assume that a Byte breakpoint has been set for the symbol LINE_COUNT equal to 38H and that a Register breakpoint has been set for CX less than 5. No other breakpoints have been set.

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

'BA *' clears both breakpoints.

'BA +' enables both breakpoints. (Line and eXit breakpoints are enabled only if they have been previously turned on with 'BL +' or 'BX +' and then disabled.) **Note:** Since RUN.COM disables all breakpoints, use this command to re-enable previously-set breakpoints.

'BA -' disables both breakpoints. (Line and eXit breakpoints are disabled only if they have been previously turned on.)

Command: Breakpoint on Byte

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

Description: This command is used to set a breakpoint when a byte of memory meets a test.

Up to eight breakpoints may be set at one time. If a segment is not specified in the address, the current data segment is used. If any of the tests pass, a breakpoint is taken. To trace execution with this breakpoint enabled, the GA or GT command must be used. This breakpoint stops execution of a program on the instruction following the instruction that changed the specified byte of

memory. Multiple breakpoints may be set on a single input line. The breakpoint clear (*), display (?), enable (+), and disable (-) functions may also be present on the line.

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays 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 .LINE_COUNT EQ 38' sets a Byte breakpoint for the memory location corresponding to LINE_COUNT.

'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

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

Description: This command is used to set a breakpoint when an instruction is executed.

It performs the same function as addresses entered after the Go command, except that these breakpoints are remembered. If a segment is not specified in the address, CS is presumed. Any form of the Go command may be used to enable Code breakpoints. This breakpoint stops execution of a program before the instruction at the specified address is executed. Multiple breakpoints may be set on a single input line. The breakpoint clear (*), display (?), enable (+), and disable (-) functions may also be present on the line. See the Go command for more information.

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays 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:

'BC PRINT_LINE' sets a Code breakpoint for the memory location corresponding to PRINT_LINE.

'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 Interrupt

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

Description: This command is used to set a breakpoint when a software interrupt is executed.

This breakpoint is used to get to the next occurrence of any software interrupts from 0 to FFH. Multiple interrupt numbers may be entered on a single line. To set breakpoints on all interrupts, use BI #. This breakpoint is activated only with the GA or GT command. If the interrupt of interest is in RAM, use a Go command to get to the interrupt in real time. For example, to get to Int 21H, enter 'G {0:21*4}'. The breakpoint clear (*), display (?), enable (+), and disable (-) functions may also be present on the line.

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays 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 * 21' clears all Interrupt breakpoints and then sets a breakpoint on Interrupt 21H.

'BI #' sets all Interrupt breakpoints.

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

Command: Breakpoint on Line

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

Description: This command is used to set a breakpoint when a source code line is executed.

This breakpoint is used to get to the next instruction that corresponds to a source line of a high-level language program. If your program is executing and you press the break-out switch, chances are very good that the program will be stopped in DOS, BIOS, or in a library routine. This breakpoint is a convenient method of getting back to the source program. It requires source line numbers to be in the symbol table—symbols added with the ES command will not suffice. After setting the Line breakpoint, use GA or GT to execute to the next source line. Since this command can be very slow, use a Go command to a known execution address if possible. The breakpoint clear (*), display (?), enable (+), and disable (-) functions may also be present on the line.

BL + must be used to turn on Line breakpoints for the first time—BA + (enable all breakpoints) will enable the Line breakpoint only if it has been previously turned on and then disabled. After being set, this breakpoint is remembered until it is cleared.

Note: 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 the next instruction that corresponds to a source code line is reached.

Command: Breakpoint on Memory

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

Description: This command is used to set a breakpoint when a range of memory will be read, writtten, and/or executed.

The two addresses may have different segments, but the second address must not be lower in memory than the

first address. A range may be used instead of the two addresses. If a segment is not specified in the address, the current data segment is used. Up to eight breakpoints may be set 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. To trace execution with this breakpoint enabled, the GA or GT command must be used. This breakpoint stops execution of a program on the instruction that will read, write, or execute the specified range of memory. Multiple breakpoints may be set on a single input line. The breakpoint clear (*), display (?), enable (+), and disable (-) functions may also be present on the line.

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 appropriate interrupts are being traced.

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays 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:

'BM DATASTART DATAEND W' sets a Memory breakpoint for writes from DATASTART thru DATAEND. Any instruction that writes to this range of memory causes a breakpoint to be taken, before the instruction is executed.

'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

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

Description: This command is used to set a 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. Up to eight breakpoints may be set 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. To trace execution with this breakpoint enabled, the GA or GT command must be used. This breakpoint stops execution of a program on the instruction that will read or write the specified range of ports. Multiple breakpoints may be set on a single input line. The breakpoint clear (*), display (?), enable (+), and disable (-) functions may also be present on the line.

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays 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 * 304 304 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

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

Description: This command is used to set a breakpoint when a register meets a test.

Up to one test per register may be set at one time. If any of the tests pass, a breakpoint is taken. To trace execution with this breakpoint enabled, the GA or GT command must be used. This breakpoint stops execution

of a program on the instruction following the instruction that changed the specified register. Multiple breakpoints may be set on a single input line. Any of the 16-bit or 8-bit registers may be used. The breakpoint clear (*), display (?), enable (+), and disable (-) functions may also be present on the line.

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays 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:

'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. Note that DS is used for its current value only.

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

Command: Breakpoint on User test

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

Description: This command is used to enable a user-written 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, a program similar to USEREXIT.ASM as described in Chapter IX must be installed before PS.COM is run. Also, the /I installation option must be used when PS.COM is run. On return from the user routine, register AL must be set to 1 if a breakpoint is to be taken. Any other value causes no breakpoint to be taken. Note that any other breakpoints currently set may cause a breakpoint to be taken.

The first User breakpoint in the sample program USEREXIT.ASM is used to set a breakpoint when DOS is available for file I/O. If you need to perform DOS functions after pressing the break-out switch, this User breakpoint will come in handy.

Multiple breakpoints may be set on a single input line. The breakpoint clear (*), display (?), enable (+), and disable (-) functions may also be present on the line.

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays 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 PS.COM had 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

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

Description: This command is used to set a breakpoint when a word of memory meets a test.

Up to eight breakpoints may be set at one time. If a segment is not specified in the address, the current data segment is used. If any of the tests pass, a breakpoint is taken. To trace execution with this breakpoint enabled, the GA or GT command must be used. This breakpoint stops execution of a program on the instruction following the instruction that changed the specified word of memory. Multiple breakpoints may be set on a single input line. The breakpoint clear (*), display (?), enable (+), and disable (-) functions may also be present on the line.

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays 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 CHAR_COUNT EQ 1234' sets a Word breakpoint for the memory location corresponding to CHAR_COUNT.

'BW * DS:123 GT SI ?' clears all Word breakpoints, sets one, and then displays it. Note that SI is used for its current value only.

'BW' or 'BW ?' displays all Word breakpoints.

Command: Breakpoint on eXit

Syntax: BX [?] [*] [-] [+] [...]

Description: This command is used to set a breakpoint on return from a subroutine or interrupt handler.

With this breakpoint set, execution continues until a RET, RETF, or IRET instruction is found. It is a convenient method of executing until the program is about to transfer control to another procedure.

Note that BX + must be used to turn on eXit breakpoint for the first time—BA + (enable all breakpoints) will enable the eXit breakpoint only if it has been previously turned on and then disabled. After being set, this breakpoint is remembered until it is cleared.

Examples:

'BX +' turns the eXit breakpoint on so that a subsequent GA or GT command will stop when a RET, RETF, or IRET instruction is reached.

'BX' or 'BX ?' displays the status of the eXit breakpoint.

Command: Compare

Syntax: C <range> <address>

Description: This command is used to compare two blocks of memory a byte at a time.

If any differences are found, the address and value of the first byte and the value and address of the second byte is displayed. Nothing is displayed for bytes that match. Since this command accepts two addresses as input, the two blocks of memory may be in different segments. If no segment is input, the current data segment is used. The length parameter indicates how much memory is to be compared.

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 were 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 were not displayed, the values of these bytes were the same.

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

Syntax: D [<range>]

Description: This command is used to display a block of memory in the current display format.

When Periscope is installed, Display 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 as well as the information applicable to all display formats in the following two paragraphs.

The syntax for all of the Display commands except DE and DR is very flexible. If you enter 'Dx', where x is the sub-function, memory is displayed starting where the last Display command left off. If you enter 'Dx <number>' the number is presumed to be an offset, the segment is presumed to be DS, and the length is presumed to be 80H. If you enter 'Dx <number> <length>' the number is presumed to be an offset, and the segment is presumed to be DS.

When display information is not shown in a window and one or more lines in the middle of the display are found to be multiple occurrences of the same number, the line(s) are suppressed and a message of the form '* NNNN Lines Of XX Skipped *' is displayed 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 a data window is used, the PgUp, PgDn, PadMinus, and PadPlus keys can be used to move forward and backward through memory. These keys affect the data window if the last command was a display command.

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 LINE_COUNT L 1' displays memory starting at the symbol LINE_COUNT for a length of 1 using the current format.

Command: Display using ASCII format

Syntax: DA [<range>]

Description: This command is used 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. All characters are displayed as is, except for the control characters nul, backspace, tab, carriage return, and line feed. Nuls are converted to spaces and the other three control characters are converted to periods. A new line is started when a CR/LF is found. If a tab character is found, the output position is moved to the next tab stop.

For example, if you enter 'DA TEXT' the display might look like this:

```
1350:0200  Periscope is a full-featured symbolic debugger, system monitor a
1350:0240  nd "break-out"..
1350:0250  switch for the IBM PC, XT, AT, and compatibles.
```

If a data window is used 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

Syntax: DB [<range>]

Description: This command is used to display a block of memory in hex and ASCII.

Each line of the display shows the starting segment and offset, up to 16 bytes, and their ASCII representation.

A dash is displayed between the eighth and ninth bytes for readability. If a display is not started on a paragraph boundary (i.e., the memory address is not evenly divisible by 16), a short right-aligned line is displayed for the first line. Similarly, if the display does not end on a paragraph boundary, the last line will be a short left-aligned line.

For the ASCII display, the high-order bit is ignored, i.e., a byte whose value is greater than 80H has 80H (128) subtracted from it before being displayed. Also, any bytes from zero to 1FH are displayed as periods.

For example, if you enter 'DB 0:0 L 20' or 'DB0:0 1F' the display might look like this:

```
0000:0000  5E 03 3F 08 5D 0B F0 BF-62 0B F0 BF 67 0B F0 BF   ..?.].p?b.p?g.p?
0000:0010  ED 01 70 00 54 FF 00 F0-62 0B F0 BF 05 18 00 F0   m.p.T..pb.p?...p
```

Examples:

'DB' displays memory starting where the last Display command left off.

'DB LINE_COUNT L 1' displays the byte at the symbol LINE_COUNT.

'DB ES:DI' displays memory starting at ES:DI for a length of 80H.

Command: Display using Double word format

Syntax: DD [<range>]

Description: This command is used 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 4 pairs of segments and offsets. If the number of bytes displayed is not evenly divisible by 16, the last line will be a short line.

For example, if you enter 'DD 0:0 L 20' or 'DD 0:0 1F' the display might look like this:

```
0000:0000  083F:035E    BFF0:0B5D    BFF0:0B62    BFF0:0B67
0000:0010  0070:01ED    F000:FF54    BFF0:0B62    F000:1805
```

Examples:

'DD' displays memory starting where the last Display command left off.

'DD 0:0 L 20' displays the interrupt vectors 0 through 7.

'DD {0:20*4}' displays memory starting at interrupt 20H.

Command: Display Effective address

Syntax: DE

Description: This command is used to display the effective address of any reads or writes performed by the current instruction. It has no arguments.

The display shows the address of any reads or writes performed by the instruction at CS:IP. The display is always in byte format. This display mode is best used with a Data window—the window will display the current effective address automatically before each instruction is executed.

If the current instruction reads memory, the effective address of the read is shown. If the instruction writes memory, the effective address of the write is shown. If the instruction reads and writes memory, only the read address is shown.

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: Display using Integer format

Syntax: DI [<range>]

Description: This command is used 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 8 decimal numbers. The number displayed may be from zero to 65535. If the number of bytes displayed is not evenly divisible by 16, the last line will be a short line.

For example, if you enter 'DI DS:SI L 20' the display might look like this:

```
15E6:1000      1      2      3      4  32764  32765  32766  32767
15E6:1010  32768  32769  32770  32771  65532  65533  65534  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

Syntax: DL [<range>]

Description: This command is used to display a block of memory in long real (quad word) format.

This format is used to examine data that is stored as an 8-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.

For example, if you enter 'DL DS:SI L 20' the display might look like this:

```
4981:0182  .801674102922700 E-290   .11125369876685 E-307
4981:0192  0                        .11820704873320 E-307
```

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

Syntax: DN [<range>]

Description: This command is used to display a block of memory in signed integer (word) format.

This format is useful for examining data that is stored as a signed word integer as used by BASIC and other languages. Each line of the display shows the starting segment and offset and up to 8 decimal numbers. The decimal numbers shown may vary from zero to 32767 (0H to 7FFFH) and from -32768 to -1 (8000H to FFFFH). If the number of bytes displayed is not evenly divisible by 16, the last line will be a short line.

For example, if you enter 'DN DS:SI L 20' the display might look like this:

```
15E6:1000     +1     +2     +3     +4 +32764 +32765 +32766 +32767
15E6:1010 -32768 -32767 -32766 -32765     -4     -3     -2     -1
```

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

Syntax: DR <address> <symbol>

Description: This command is used to display a block of memory in an easy-to-read format using a previously-created record definition.

This format is useful for examining data that is part of a record, such as the PSP or an FCB. Each line of the display shows a field name and the data for the field in any format supported by Periscope. Any area of memory can be displayed using any record definition.

To use a record format, a record definition, or DEF file must exist. RUN.COM 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.COM in Chapter IX. The following definition of the PSP is from the file PS.DEF.

```
\PSP            ; Program Segment Prefix
Int 20,b,2      ; DOS return
Topmem,w,2      ; Amount of memory in paragraphs
Res.,b,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
Res.,b,16       ; Reserved for DOS
Environ,w,2     ; DOS 2.00 Environment segment
Res.,b,2e       ; Reserved for DOS
FCB1,b,10       ; The first FCB read from the command line
FCB2,b,14       ; The second FCB read from the command line
```

Assuming that the definition for the PSP record has been loaded, enter 'DR CS:0 PSP' to get a display similar to the following:

```
Int 20      CD 20                                                 M
Topmem      5000
Res.        00                                                    .
Long Call   9A                                                    .
DOS Func    F01D:FEF0
Term Addr   0B42:012C
Brk Addr    0B42:0139
Err Addr    0B42:0481
Res.        42 0B 01 01 01 00 02 FF FF FF FF FF FF FF FF FF   B...............
            FF FF FF FF FF FF                                 ......
Environ     125B
Res.        E2 FF 61 12 14 00 18 00 61 12 00 00 00 00 00 00   ................
            * 0001 LINES OF 00 SKIPPED *
            00 00 CD 21 CB 00 00 00 00 00 00 00 00 00 00 00   ..M!K.........
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   .           ....
            00 00 00 00                                       ....
```

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 kept 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

Syntax: DS [<range>]

Description: This command is used to display a block of memory in short real (double word) format.

This format is used to examine data that is stored as a 4-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.

For example, if you enter 'DS DS:SI L 10' the display might look like this:

```
4981:0182  .3944305 E-30          .1057946 E-35
4981:018A  .1875863 E-29          0
```

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: Display using Word format

Syntax: DW [<range>]

Description: This command is used to display a block of memory in word format.

This format is useful for examining 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 8 words. If the number of bytes displayed is not evenly divisible by 16, the last line will be a short line.

For example, if you enter 'DW 0:0 L 20' or 'DW0:0 1F' the display might look like this:

```
0000:0000  035E  083F  0B5D  BFF0  0B62  BFF0  0B67  BFF0
0000:0010  01ED  0070  FF54  F000  0B62  BFF0  1805  F000
```

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 asciiZ format

Syntax: DZ [<range>]

Description: This command is used to display a block of memory in nul-terminated ASCII format.

This command is the same as the DA command, except that the display ends when a nul (binary zero) is found. If a data window is used, the display continues to the end of the data 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

Syntax: E <address> [<list>]

Description: This command is used to modify memory.

The segment and offset must be specified for the address, to avoid accidental changes to memory.

If the optional list is present, the specified memory is modified and the command terminates. If the list is not present, an interactive mode is started. 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 is started. The program 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). Any invalid input, such as 'G9' or too many digits, is not echoed.

Press the space bar to skip to the next byte. Press the hyphen key to back up one byte. The backspace key is used to discard a single digit. Use the return key to terminate the interactive mode. Note that the interactive mode is not compatible with the multiple command capability of Periscope—i.e., you cannot use semi-colons to 'stack' multiple commands on one line.

When moving forward with the space bar, a new line is started when the address is evenly divisible by eight. When moving backward with the hyphen key, each address is on a new line.

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 the return key.

Command: Enter Alias

Syntax: EA <alias> [<name>]

Description: This command is used to define or redefine an alias.

The alias is a two-character short-hand notation for a one- to 16-character name. See the definition of the alias parameter earlier in this chapter for more information on the available aliases. To display the current aliases, press the F5 key. If this command is used without a name parameter, the alias is deleted.

The X1, X2, and X3 aliases cause a Periscope command to be executed on entry to Periscope, after each Periscope command, and on exit from Periscope respectively. If an error occurs during this alias execution, further alias execution is suppressed until the EA command is used again.

Examples:

'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.

'EA X2' deletes alias X2.

Command: Enter Symbol

Syntax: ES <address> <symbol>

Description: This command is used to define or redefine symbol table entries.

A segment and offset must be specified for the address. The symbol name must be 16 characters or less and may be preceded by a period. The symbol table is searched for a symbol of the same name. If an existing symbol is found, the segment and offset associated with it are updated. If no match is found, a new symbol is added at the end of the symbol table. To display the current symbols, press the F8 key.

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: Fill

Syntax: F <range> <list>

Description: This command is used to fill a block of memory with a byte/string pattern.

A segment and offset must be specified for the address to avoid accidental changes to memory. The length specifies the number of bytes to be affected. The list is the pattern that is copied into the specified range of memory. If the length of the list is less than the length of the range specified, the list is copied as many times as needed to fill the range. Conversely, if the length of the list is greater than the length of the range, the extra bytes are not copied.

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

Syntax: G [<address>] [...]

Description: The Go command is used to set temporary code breakpoints, activate sticky code breakpoints, and execute the program being debugged. (See the introduction to the Commands section of this chapter for a general discussion of breakpoints and breakpoint terminology.) This command activates code breakpoints only. To activate monitor breakpoints, use the GA or GT commands. To activate hardware breakpoints, use the GH command.

If any addresses are specified on the command line, the byte at each of the addresses is replaced with a CCH, the single-byte breakpoint. When control is returned to Periscope via any method, the original byte is restored. The addresses entered on the command line are referred to as temporary code breakpoints. Up to four of these breakpoints may be used. If the address does not contain a segment, the current code segment is used.

To set sticky code breakpoints, use the BC command described earlier. This method allows you to set up to 16 sticky code breakpoints.

If 'G' with no addresses is entered, the sticky breakpoints, if any, are used. If there are no sticky breakpoints, program execution continues until the break-out switch is pressed. The sticky breakpoints are remembered until cleared or PS.COM is rerun. If you have code 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.

You cannot set code breakpoints in ROM—code breakpoints require that Periscope be able to exchange the original byte with CCH before starting the Go. 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.

When a Go command is used, Periscope invisibly traces one instruction and then performs the Go. This allows you to set a breakpoint on the current instruction to repetitively go to the same address.

Examples:

'G PRINT_LINE' sets a temporary code breakpoint at the address equal to the symbol PRINT_LINE 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 Equal

Syntax: G= [<address>] [...]

Description: This command sets CS:IP to the first address, sets any indicated code breakpoints, activates any sticky code breakpoints set with the BC command, and executes the program being debugged. This command activates code breakpoints only.

Other than setting CS:IP to the first address entered, this command is identical in function to the Go command described above. The equal sign must appear directly 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

Syntax: GA [<address>] [...]

Description: This command is the same as the GT command described below, except that ALL instructions are traced by this command. This command activates code and monitor breakpoints only.

The GT command single steps through instructions, except when a software interrupt is performed. Then the interrupt trace table (see the /T command) is checked. 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 always traces ALL the way through software interrupts. It is slower than GT, but more dependable.

Example: See the examples under the GT command.

Command: Go using Hardware (requires Periscope III)

Syntax: GH [<address>] [...]

Description: This command is used to activate code and hardware breakpoints and execute the program being debugged. This command activates code and hardware breakpoints only (not monitor breakpoints!).

This command is the same as the Go command, except that it also sets any hardware breakpoints that are enabled. Whenever the hardware breakpoints are changed, they must be completely reset. This full reset involves over one million OUT instructions, so please be patient.

The hardware breakpoints are disabled when RUN.COM is used—be sure to check the status of the hardware breakpoints using the 'HA' command before starting execution. See the description of the Go command for more information.

Examples:

'GH PRINT_LINE' sets a temporary code breakpoint at the address equal to the symbol PRINT_LINE, invokes all code and hardware breakpoints that are set and starts execution of the program.

'GH' invokes all hardware breakpoints that are set and begins execution of the program.

Command: Go using Monitor

Syntax: GM [<address>] [...]

Description: This command is used 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's screen is displayed. Otherwise full speed execution resumes. This command activates code and hardware breakpoints only—the monitor breakpoints are evaluated only when a code or hardware breakpoint occurs.

For all models of Periscope other than model III, 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 breakpoint is reached, the monitor breakpoint(s) are evaluated, rather than after every instruction.

For Periscope III, this command acts like a combination of the GH and GT commands. Consider a situation where you need to find where your program is writing to low memory. Since DOS and other programs can be legitimately writing to low memory, you need to watch for writes to low memory where the offending Code Segment points to your program. To do this with Periscope III, set a hardware breakpoint watching for writes to the desired range, then enter a monitor breakpoint (such as BR CS EQ CS), and then enter 'GM'. Each time a hardware breakpoint occurs, the monitor breakpoint(s) are evaluated. If any of the monitor breakpoints indicates a hit, Periscope's screen is displayed. Otherwise full speed execution resumes. The monitor breakpoints of interest when Periscope III is used are BB, BR, BW, and BU. The Register breakpoint is particularly powerful, since register information is not available on the bus. For more complex events, the user breakpoint tests can be used. This command can run slowly if the hardware breakpoint occurs frequently—try to qualify the hardware breakpoint as tightly as possible for best performance.

Note: Each time a hardware breakpoint occurs, a discontinuity appears in the real-time trace buffer. This happens because nothing is added to the buffer from the time the hardware breakpoint occurs until just before Periscope returns control to the interrupted program.

Since the monitor breakpoint is evaluated one or more instructions after the instruction that caused the hardware breakpoint, it is possible for the hardware breakpoint and the software breakpoint to be out of sync. This can cause an occasional false or missed breakpoint. For

example, if you're watching for writes to memory where the CS points to your program, code that changes CS during the breakpoint overrun interval can cause problems.

Examples:

'GM PRINT_LINE' sets a temporary code breakpoint at the address equal to the symbol PRINT_LINE, invokes all hardware breakpoints that are set (if Periscope III is used) and starts execution of the program. When a code or hardware breakpoint occurs, the monitor breakpoints are evaluated. If any of the monitor breakpoints indicate a hit, Periscope's screen is displayed. Otherwise, full-speed execution resumes.

Command: Go using Trace

Syntax: GT [<address>] [...]

Description: This command is the similar to the normal Go command, except that it enters a single step mode and evaluates the monitor breakpoints after each instruction. This command activates code and monitor breakpoints only.

This command puts the system into a mode where every instruction executed by your program is analyzed to see if a breakpoint has been reached. This analysis can slow down the execution by a factor of 100 to 1000, but in many cases is the only way to find an elusive bug (unless you're using Periscope III). Since this command is slow, try to use the normal Go command to get as close to the problem as possible.

The monitor breakpoints are remembered until cleared or until PS.COM is rerun. If you have code 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.

Since breakpoints are cleared when re-entered, get in the habit of checking the breakpoint settings before using this command. Enter 'BA' to display the current breakpoints before entering 'GT'. This way you can make sure that the right breakpoints have been set before starting execution.

For temporary and sticky code breakpoints, this command performs in the same fashion as the Go command described above. After a breakpoint, use the TB command to see the instructions preceding the instruction that caused the breakpoint.

When a software interrupt is encountered, it is executed 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 interrupts.

If you're using the GT command and the program being debugged 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 traced by Periscope. Note the last entry in the software trace buffer—the interrupt shown 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.

Borland's SIDEKICK can interfere with the tracing of Interrupt 21H when GT is used—force tracing of INT 21 using the /T command; use the GA command instead of GT; or better yet—remove SIDEKICK while debugging.

Examples:

'GT PRINT_LINE NEW_PAGE' sets temporary code breakpoints at the addresses equal to the symbols PRINT_LINE and NEW_PAGE.

'GT' begins execution of the program with no temporary code breakpoints—only sticky and monitor breakpoints that are enabled.

'GT ES:456' sets a temporary code breakpoint at ES:456.

Command: Help

Syntax: ? [<function><sub-function>]

Description: This command displays Periscope's commands by function and sub-function if the on-line help file has been loaded. If the help file is not available, a command summary is displayed.

Examples:

'?' displays a command summary.

'? DD' displays help for the Display Double word command if the on-line help file has been loaded.

Command: Hex arithmetic

Syntax: H <number> <arithmetic operator> <number>

Description: This command is used to perform hexadecimal arithmetic.

Addition, subtraction, multiplication, and division are available. The standard operators are used for each function. The numbers must be in hex and may be from one to four hex digits. If a register name is entered in place of one of the numbers, its current value is used 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.

Examples:

'H 1234+123' gives an answer of 1357.

'H 1234-123' gives an answer of 1111.

'H 1234/123' gives an answer of 0010 R 0004.

'H 1234*123' gives an answer of 0014 B11C.

'H DI-SI' displays the result of subtracting the current value of SI from the current value of DI.

Command: display, clear, enable, and/or disable All Hardware breakpoints (requires Periscope III)

Syntax: HA [?] [*] [+] [-] [...]

Description: This command is used to display, clear, enable, or disable the hardware breakpoints.

HA ? displays all hardware breakpoints, HA * clears all hardware breakpoints, HA + enables all hardware breakpoints, and HA - disables all hardware breakpoints. If the hardware controls (HC) are set to the default values, nothing is displayed for them when 'HA ?' is used. See the 'BA' command for software breakpoints.

Examples:

'HA ?' or 'HA' displays all hardware breakpoints.

'HA +' enables all hardware breakpoints. **Note:** since RUN.COM disables the breakpoints, use this command to re-enable the previously-set breakpoints.

Command: Hardware Bit breakpoint (requires Periscope III)

Syntax: HB [xxxx xxxx L and/or H] [?] [*] [+] [-]

Description: This command is used to set, display, clear, enable, or disable the hardware data bit breakpoint.

The Bit breakpoint is entered as a binary number of exactly eight characters. The allowable bit values are 0, 1, and X which correspond to a zero, one, and 'don't care' respectively. Only one value may be specified and the breakpoint must be used in conjunction with a HM or HP breakpoint, i.e. the specified data bits are used to qualify a Memory or Port breakpoint.

The source of the data is indicated by an 'L' for the low (8-bit) bus or 'H' for the high (16-bit) bus. The high bus is never used on a PC-class machine nor with the 8-bit memory in an AT-class machine. For 16-bit memory, the low and/or high busses may be used—see the table in the HD command below to determine bus usage. If you

are using an AT-class machine and are not sure which bus to specify, try using 'LH' to specify either bus. You may get some extraneous breakpoints, but at least you won't miss any.

HB ? displays the bit breakpoint, HB * clears the bit breakpoint, HB + enables the bit breakpoint, and HB - disables the bit breakpoint.

Examples:

'HB 1XXX XXXX L' sets the bit breakpoint for the low bus on any value containing a binary one in bit 7 and any value in bits 6 through zero.

'HB 1010 01X0 H' sets the bit breakpoint for the value A4H or A6H on the high bus.

Command: Hardware Controls (requires Periscope III)

Syntax: HC [?] [*] [#<number>] [M<byte>] [O- or O+] [S- or S+] [TB, TC, or TT]

Description: This command is used to display, clear, or set the hardware controls. The controls include the pass count, megabyte range, trace overflow stop, selective trace, and trigger location.

The pass count is entered as '#' followed by a number from one to FFFFH. It is used as a real-time breakpoint counter—when the specified number of breakpoints have occurred, Periscope interrupts the executing program. If one is specified as the pass count, the first breakpoint will interrupt the program. If the pass count is set to four, the fourth breakpoint will interrupt the program, etc. Note that the pass count must be used in conjunction with an HM or HP breakpoint.

The megabyte range is entered as 'M' followed by a number from zero to 0FH. It affects the address range used by the HM command. It is valid only on an AT-class machine that has a 16-bit bus. This number is normally zero for addresses in the first megabyte of memory, but it may be any value from zero to 0FH. If it is not zero, Hardware Memory breakpoints are generated only when memory beyond the first megabyte is referenced.

The trace overflow stop is enabled by entering 'O+' and disabled by entering 'O-'. When enabled, the overflow stop generates a breakpoint each time the hardware trace buffer fills up. This breakpoint allows you to see the bus events in 8K groups, so you can examine program flow starting at a particular point. You can use this capability to count the bus cycles required by an application in increments of 8K. This control does not require an HM or HP breakpoint.

The selective trace is enabled by entering 'S+' and disabled by entering 'S-'. When enabled, only events that would cause a hardware breakpoint are saved in the hardware trace buffer. This mode should used with a pass count that indicates the number of events desired before the Periscope screen is displayed. For example, if you want to capture the next 16 Outs to port 3B4H, set the Port breakpoint and enter 'HC * #10 S+' to clear the controls and then set them to capture only the next 10H trigger events. Finally, enter 'GH' to arm the board and begin execution.

Note: When selective trace is used, not all bus events are saved in the trace buffer, so it is not usually meaningful to disassemble the trace buffer. Also, when 'S+' is used, the trigger location is set to the bottom of the buffer ('TB', as discussed below.)

The trigger location may be set to the Top ('TT'), Center ('TC'), or Bottom ('TB') of the trace buffer. When set to the top, the trace buffer shows 8K events AFTER the trigger event, numbered from 0 to +1FFE. When set to the center, the trace buffer shows up to 4K events before the trigger event and 4K events after the trigger event, numbered from -FFF to 0 to +FFF. When set to the bottom, the trace buffer shows up to 8K events BEFORE the trigger event, numbered from -1FFE to 0. **Note:** The trigger location that is in effect when a hardware breakpoint occurs controls the contents of the trace buffer—although the trigger location may be changed after a breakpoint, the contents of the trace buffer remain the same. TC and TT are meaningful only for a hardware breakpoint, not when the break-out switch is pressed. See the description of the HT command below for more information.

Examples:

'HC * #5 TT' clears the hardware controls, sets the pass count to 5, and sets the trigger location to the top of the buffer.

'HC 0+;GH' sets the trace overflow stop on and begins execution with hardware breakpoints enabled. After 8,192 (8K) bus events have been added to the hardware trace buffer, control is returned to Periscope.

'HC S+ #20 TC' sets selective trace on with a pass count of 20H. Since selective trace is on, the 'TC' command is ignored and 'TB' is assumed.

Command: Hardware Data breakpoint (requires Periscope III)

Syntax: HD [<byte> <byte> L and/or H] [?] [*] [+] [-] [...]

Description: This command is used to set, display, clear, enable, or disable hardware Data breakpoints.

The Data breakpoints are entered as ranges of values, where the low and high values must be from zero to 0FFH. Up to 16 ranges may be specified. This breakpoint must be used in conjunction with an HM or HP breakpoint, i.e. the specified data ranges are used to qualify a Memory or Port breakpoint.

Due to the design of the hardware breakpoint tables, it is not possible to set a breakpoint where the range stored in the table is all zeroes. The breakpoint setting 'HD 0 0' is therefore not allowed. If you need to set a value of zero, use the 'HB' command.

The source of the data is indicated by an 'L' for the low (8-bit) bus or 'H' for the high (16-bit) bus. These indicators are additive, i.e. once 'L' is specified, it applies to all Data breakpoints. The high bus is never used on a PC-class machine nor with 8-bit memory in an AT-class machine. For 16-bit memory, the low and/or high busses may be used. See the table below to determine bus usage. If you are using an AT-class machine and are not sure which bus to specify, try using 'LH' to specify either bus. You may get some extraneous breakpoints, but at least you won't miss any.

Environment	Low bus	High bus
PC or XT	X	
AT/8-bit memory	X	
AT/16-bit memory		
even byte	X	
even word	LO	HI
odd byte		X
odd word*	HI(2)	LO(1)

* Access to a word at an odd address is broken into two cycles

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays the message 'Breakpoint cleared'. Be careful to display all breakpoints before using the GH or GM command to make sure the breakpoints you've got are the ones you want.

Examples:

'HD 10 1F L' allows breakpoints when the data on the lower bus is from 10H to 1FH.

'HD * 20 2F L 40 4F H' clears data breakpoints and sets breakpoints for data values from 20H to 2FH and 40H to 4FH on either the low or high bus. Note that the low or high bus indicators apply to both ranges!

Command: Hardware Memory breakpoint (requires Periscope III)

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

Description: This command is used to set, display, clear, enable, or disable Memory breakpoints.

The breakpoints are entered as two addresses or a range. Up to 16 ranges may be specified. The source of the breakpoint is indicated by an 'M', 'R', 'W' and/or 'X', indicating DMA, memory read, memory write, and code execution (instruction prefetch) respectively. These indicators are additive, i.e. once 'R' is specified, it applies to all Memory breakpoints.

Note: Although DMA is used to refresh dynamic RAM, it is not possible to set breakpoints on RAM refresh cycles.

Due to the design of the hardware breakpoint tables, it is not possible to set a breakpoint where the range stored in the table is all zeroes. The breakpoint setting 'HM 0:0

0:0' is therefore not allowed. Also, do not attempt to set memory breakpoints inside Periscope's code/data area—Error 37 will result.

A breakpoint starting at an odd address in 16-bit memory can be missed if the next lower even address is accessed as a word. In this case, the message "Warning - Odd low address may cause missed breakpoint" is displayed when the breakpoint is entered. The message can be ignored if you're sure that the memory in question will be accessed starting at the odd address.

The instruction prefetch breakpoint is generated when a byte is read into the prefetch queue, not when it is actually executed. Since the prefetch queue is four bytes long on the 8088 and six bytes long on the 8086 and 80286, instructions are read before they're actually executed. Due to the 'breakpoint overrun' phenomenon, a breakpoint does not occur immediately. So the instruction in question may or may not have been executed by the time Periscope is activated. If the desired instruction has not been executed, try setting the execution breakpoint after an instruction that flushes the prefetch queue (JMP, CALL, RET, IRET, INT, etc.).

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays the message 'Breakpoint cleared'. Be careful to display all breakpoints before using the GH or GM command to make sure the breakpoints you've got are the ones you want.

Examples:

'HM B000:0 L1 W' sets a breakpoint on writes to memory at B000:0 (the upper left-hand corner of the monochrome screen) for a length of one byte.

'HM 0:0 0:3FF R' sets a breakpoint on reads of the interrupt vector table.

'HM 0:0 CS:0 W' sets a breakpoint on writes to memory from the beginning of memory to the current code segment.

Command: Hardware Port breakpoint (requires Periscope III)

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

Description: This command is used to set, display, clear, enable, or disable Port breakpoints.

The breakpoints are entered as a range of two ports, where each value must be from zero to FFFFH, although the IBM PC fully supports ports from zero to 3FFH. Up to 16 ranges may be specified. Due to the design of the hardware breakpoint tables, it is not possible to set a breakpoint where the range stored in the table is all zeroes. The breakpoint setting 'HP 0 0' is therefore not allowed.

The source of the breakpoint is indicated by an 'I' and/or 'O' indicating an In (port read) or Out (port write), respectively. These indicators are additive, i.e. once 'I' is specified, it applies to all Port breakpoints.

After being set, these breakpoints are remembered until they are cleared. Re-entering a previously set breakpoint clears the breakpoint and displays the message 'Breakpoint cleared'. Be careful to display all breakpoints before using the GH or GM command to make sure the breakpoints you've got are the ones you want.

Examples:

'HP 308 308 I' sets a breakpoint on reads of port 308H.

'HP 310 31F O' sets a breakpoint on writes of ports from 310H to 31FH.

Command: display Hardware buffer in Raw mode (requires Periscope III)

Syntax: HR [*] or HR [!] [] [<name> or <address>]

Description: This command is used to display the real-time trace buffer in a 'raw dump' format. Each line of the display corresponds to one bus event. Each line contains an address, data, an operation, a symbol corresponding to the address if available, a sequence number, and possible other information.

The trace buffer may be cleared by entering 'HR *'. The entire buffer may be displayed using 'HR !'. If an asterisk or exclamation point is not used, a full-screen display mode is entered unless the buffer is empty. The

only way to exit this mode is to press the Esc key. The optional segment is used to decode addresses. The optional filename or address indicates a trace buffer file previously created with the HW command that is used instead of the current contents of the trace buffer.

While in full-screen mode, the keys available are: Home, End, Up, Dn, PgUp, PgDn, Left Arrow, and Esc. You can switch between buffer display modes by entering 'R', 'T', or 'U'.

You can move to any location in the buffer by entering '#' followed by the sequence number. Note that the sequence number must be from -1FFE to +1FFE. Any value outside this range is ANDed with 1FFFH.

You can search for items in the trace buffer. To search for an address, enter '/' followed by the address or symbol name. To search for low or high data values, enter '/L nn' or '/H nn' respectively, where nn is the low or high data byte. To search for an operation type, enter '/T x', where x is I (In), M (DMA), O (Out), R (Read), W (Write), or X (Fetch). The buffer search starts at the second line from the top of the screen. **Note:** When searching for an address using a symbol name that starts with 'H', 'L', or 'T', start the symbol name with a period to avoid conflict with the other search commands.

The address is shown at the start of the line. The address may appear in one of four formats—a segmented address including the segment, a colon, and an offset; a 4-digit port number; a 5-digit non-segmented address; or a 6-digit non-segmented address. The segmented address is displayed when the address can be decoded into BIOS (segment F000H), Periscope's code segment, DOS, or the current code segment. The optional segment that can be entered on the command line overrides the current code segment. To change the "decode" segment, enter 'S' followed by the segment value. The 4-digit port number is used for I/O port access. The 5-digit non-segmented address is used when Periscope is unable to decode the segment into one of the above values. If addresses above one megabyte occur on an AT-class machine, the 6-digit non-segmented address is displayed.

The second field is the data present on the bus. If a PC-class machine is used, this field is always a byte, since all bus access on a PC occurs 8 bits at a time. If an AT-class machine is used, the data field may be a high byte, low byte, or a word. The high byte is present on the high bus and is left-aligned in the data field. The low byte is present on the low (8-bit) bus and is right-

aligned. The word is shown in word format, with the high (later) byte displayed first.

The third field is the operation. The possible values are: DMA (Direct Memory Access); Fetch (code prefetch); In (I/O port read); Out (I/O port write); Probe (the external probe line has been pulled low); Read (memory read, not including code prefetch); and Write (memory write). All but the DMA and Probe operations are mutually exclusive. If an invalid combination is detected, a hex number and a question mark are displayed in the operation field. Please call Tech Support if you see this displayed. A symbol name may follow the operation if the address indicates a symbol.

The fourth field is used only on an AT-class machine when a byte access to memory has been made. It shows the other data byte in parentheses. Due to the vagaries of the 16-bit bus, we've included this orphan data byte—if you ever get any unexpected breakpoints, check this byte! If a byte access was made to the low bus, this field will show the data present on the high bus and vice-versa.

The fifth field is a sequence number. The hex number may be from -1FFE to +1FFE, depending on the trigger location. The first entry in the circular buffer is indicated by 'Top' after the sequence number. Use the Home key to get to the Top of the buffer. The last entry in the buffer is indicated by 'Bottom' after the sequence number. Use the End key to get to the Bottom of the buffer. Use the Left Arrow key to get to the center of the buffer (actually, 4K from the end of the buffer).

The other possible field follows the sequence number. If the address is within Periscope's code/data area, a 'PS' is displayed to identify the code/data as being Periscope. You may see these 'PS' entries at the beginning and end of the trace buffer—they show where Periscope was turning control over to the application and regaining control from the application respectively.

A sample hardware trace buffer display is shown below.

```
ADDRESS   DATA  OPERATION                    ORPHAN SEQUENCE

1B9A:Ø137 E8    Fetch START                  (9A)  -ØØCF
1B9A:Ø138 ØØ17  Fetch                              -ØØCE
1B9A:Ø13A 33A1  Fetch                              -ØØCD
1B9A:Ø13C BFØ1  Fetch                              -ØØCC
1B9A:Ø151 B1    Fetch GETMEM                 (Ø1)  -ØØCB
1B9A:FFFC Ø13A  Write                              -ØØCA
1B9A:Ø152 BEØ6  Fetch                              -ØØC9
1B9A:Ø154 ØØØ2  Fetch                              -ØØC8
1B9A:Ø156 Ø48B  Fetch                              -ØØC7
1B9A:Ø158 E8D3  Fetch                              -ØØC6
1B9A:Ø15A 33A3  Fetch                              -ØØC5
1B9A:ØØØ2 AØØØ  Read                               -ØØC4
1B9A:Ø15C 8CØ1  Fetch                              -ØØC3
1B9A:Ø15E D3CB  Fetch                              -ØØC2
1B9A:Ø16Ø 2BEB  Fetch                              -ØØC1
1B9A:Ø162 A3C3  Fetch                              -ØØCØ
1B9A:Ø164 Ø135  Fetch                              -ØØBF
1B9A:Ø133 8Ø    Write TOTMEM                 (35)  -ØØBE
```

Examples:

'HR' displays the last page of the trace buffer.

'HR 1234' decodes as many addresses as possible using 1234 as the segment value.

'HR *' clears the hardware trace buffer.

Command: display Hardware buffer Single entry (requires Periscope III)

Syntax: HS [*] or []

Description: This command displays the last entry in the real-time hardware trace buffer in a 'raw dump' format.

'HS *' clears the real-time hardware trace buffer, just as the other buffer display commands do. The display output is identical to that shown by the HR command. This command does not enter a full-screen mode—it displays one line showing the last entry in the hardware trace buffer and then displays the Periscope prompt. It is used to display the bus cycle that caused a hardware breakpoint. See the HR command for more information.

Example:

'GH;HS' enables hardware breakpoints and displays the last entry in the real-time hardware trace buffer after a hardware breakpoint occurs.

Command: display Hardware buffer in Trace mode (requires Periscope III)

Syntax: HT [*] or HT [!] [] [<name> or <address>]

Description: This command is used to display the real-time hardware trace buffer in a disassembly-and-data format. Code prefetch bus cycles are disassembled in symbolic form. Other bus cycles (read, write, in, out) are displayed in the 'raw' format described in the 'HR' command, except for DMA cycles, which are not shown.

The trace buffer may be cleared by entering 'HT *'. The entire buffer may be displayed using 'HT !'. If an asterisk or exclamation point is not used, a full-screen display mode is entered unless the buffer is empty. The only way to exit this mode is to press the Esc key. The optional segment is used to decode addresses. The optional filename or address indicates a trace buffer file previously created with the HW command that is used instead of the current contents of the trace buffer.

While in full-screen mode, the keys available are: Home, End, Up, Dn, PgUp, PgDn, Left Arrow, and Esc. You can switch between buffer display modes by entering 'R', 'T', or 'U'. Depending on what is found in the trace buffer, HT may show nothing. If this happens, enter 'R' to switch to the HR mode.

You can move to any location in the buffer by entering '#' followed by the sequence number. Note that the sequence number must be from -1FFE to +1FFE. Any value outside this range is ANDed with 1FFFH.

You can search for items in the trace buffer. To search for an address, enter '/' followed by the address or symbol name. To search for low or high data values, enter '/L nn' or '/H nn' respectively, where nn is the low or high data byte. To search for an operation type, enter '/T x', where x is I (In), M (DMA), O (Out), R (Read), W (Write), or X (Fetch). The buffer search starts at the second line from the top of the screen. **Note:** When searching for an address using a symbol name that starts with 'H', 'L', or 'T', start the symbol name with a period to avoid conflict with the other search commands.

The Up and Dn keys move up or down by one record—this will not necessarily result in a change of the screen.

Similarly, the PgUp key moves backward by as many records as there are lines displayed. The best results are obtained by using the PgDn key, since this key moves forward and starts a new page where the current one ends.

The disassembled lines are in the standard disassembly format, showing an address, the opcodes making up the instruction, and the instruction itself. The address is shown in segmented notation or in 5- or 6-digit form as described in the 'HR' command. To change the "decode" segment, enter 'S' followed by the segment value. If the code is executing in Periscope's code segment, a 'PS' notation appears near the end of the line. A quick way to discern disassembly lines from other lines is that the disassembly lines don't show a sequence number. Note that segment override prefixes may be separated from the instructions they affect.

When disassembling instructions, one or more bus cycle records are required per instruction. All code prefetch cycles are added to a buffer which is then disassembled. Even though code has been fetched by the processor, there is no absolute method of telling whether the code was actually executed, so Periscope attempts to infer the execution of code in the buffer. If a discontinuity in the address occurs or after an unconditional 'JMP', 'CALL', 'RET' (far or near), 'INT', or 'IRET' instruction is found, the buffer is flagged as being flushed, since the system's prefetch buffer acts in a similar fashion. These flushed instructions are indicated by an asterisk in front of the mnemonic.

Any memory or port accesses performed by an instruction (read, write, in, or out) are shown one or more lines after the instruction itself. These memory or port accesses may be the only way to determine that an instruction was actually executed. (The display does not currently match memory/port access with the appropriate instruction ... maybe one day.)

Register information is not available on the bus directly, but it is possible to deduce register values, especially by looking at the results of MOV instructions and implicit or explicit PUSHes and POPs, such as when an INT or IRET instruction is executed. You can often deduce the DS and ES registers from instructions that manipulate memory, etc.

When using the Up or Dn keys, the display can vary drastically. This is due to the cumulative effect of the buffered instructions and Periscope's flush logic. Instructions can disappear and then reappear in odd

positions. You might think of this erratic movement as being like house of mirrors, where a slight movement can cause a large distortion in what you see.

A sample hardware trace buffer display is shown below. Note the asterisk before the second instruction, indicating it was flushed from the prefetch buffer.

```
                START:
1B9A:0137 E81700         CALL   GETMEM
1B9A:013A A13301        *MOV    AX,[TOTMEM]
1B9A:FFFC 013A  Write                                        -00CA
                GETMEM:
1B9A:0151 B106           MOV    CL,06
1B9A:0153 BE0200         MOV    SI,0002
1B9A:0156 8B04           MOV    AX,[SI]
1B9A:0158 D3E8           SHR    AX,CL
1B9A:0002 A000  Read                                         -00C4
1B9A:015A A33301         MOV    [TOTMEM],AX
1B9A:015D 8CCB           MOV    BX,CS
1B9A:015F D3EB           SHR    BX,CL
1B9A:0161 2BC3           SUB    AX,BX
1B9A:0163 A33501         MOV    [FREMEM],AX
1B9A:0133 80    Write TOTMEM                    (35)  -00BE
```

Examples:

'HT B000' displays the last page of the hardware trace buffer, showing all addresses in segment B000 in segmented format.

'HT *' clears the hardware trace buffer.

Command: display Hardware buffer in Unasm mode (requires Periscope III)

Syntax: HU [*] or HU [!] [] [<name> or <address>]

Description: This command is used to display the real-time hardware trace buffer in a disassembly-only format. Code prefetch bus cycles are disassembled in symbolic form. Other bus cycles (read, write, in, out, DMA) are not shown.

The trace buffer may be cleared by entering 'HU *'. The entire buffer may be displayed using 'HU !'. If an asterisk or exclamation point is not used, a full-screen display mode is entered unless the buffer is empty. The only way to exit this mode is to press the Esc key. The optional segment is used to decode addresses. The

optional filename or address indicates a trace buffer file previously created with the HW command that is used instead of the current contents of the trace buffer.

While in full-screen mode, the keys available are: Home, End, Up, Dn, PgUp, PgDn, Left Arrow, and Esc. You can switch between buffer display modes by entering 'R', 'T', or 'U'. Depending on what is found in the trace buffer, HU may show a blank display. If this happens, enter 'R' to switch to the HR mode.

You can move to any location in the buffer by entering '#' followed by the sequence number. Note that the sequence number must be from -1FFE to +1FFE. Any value outside this range is ANDed with 1FFFH.

You can search for items in the trace buffer. To search for an address, enter '/' followed by the address or symbol name. To search for low or high data values, enter '/L nn' or '/H nn' respectively, where nn is the low or high data byte. To search for an operation type, enter '/T x', where x is I (In), M (DMA), O (Out), R (Read), W (Write), or X (Fetch). The buffer search starts at the second line from the top of the screen. **Note:** When searching for an address using a symbol name that starts with 'H', 'L', or 'T', start the symbol name with a period to avoid conflict with the other search commands.

The Up and Dn keys move up or down by one record—this will not necessarily result in a change of the screen. Similarly, the PgUp key moves backward by as many records as there are lines displayed. The best results are obtained by using the PgDn key, since this key moves forward and starts a new page where the current one ends.

The disassembled lines are in the standard disassembly format, showing an address, the opcodes making up the instruction, and the instruction itself. The address is shown in segmented notation or in 5- or 6-digit form as described in the 'HR' command. To change the "decode" segment, enter 'S' followed by the segment value. If the code is executing in Periscope's code segment, a 'PS' notation appears near the end of the line.

When disassembling instructions, one or more bus cycle records are required per instruction. All code prefetch cycles are added to a buffer which is then disassembled. Even though code has been fetched by the processor, there is no absolute method of telling whether the code was actually executed, so Periscope attempts to infer the execution of code in the buffer. If a discontinuity in the

address occurs or after an unconditional 'JMP', 'CALL', 'RET' (far or near), 'INT' or 'IRET' instruction is found, the buffer is flushed, since the system's prefetch buffer acts in a similar fashion. Flushed instructions are flagged with an asterisk when 'HT' is used, but are not shown when this command is used.

When using the Up or Dn keys, the display can vary drastically. This is due to the cumulative effect of the buffered instructions and Periscope's flush logic. Instructions can disappear and then reappear in odd positions. You might think of this erratic movement as being like house of mirrors, where a slight movement can cause a large distortion in what you see.

A sample hardware trace buffer display is shown below.

```
                START:
1B9A:0137 E81700        CALL    GETMEM
                GETMEM:
1B9A:0151 B106          MOV     CL,06
1B9A:0153 BE0200        MOV     SI,0002
1B9A:0156 8B04          MOV     AX,[SI]
1B9A:0158 D3E8          SHR     AX,CL
1B9A:015A A33301        MOV     [TOTMEM],AX
1B9A:015D 8CCB          MOV     BX,CS
1B9A:015F D3EB          SHR     BX,CL
1B9A:0161 2BC3          SUB     AX,BX
1B9A:0163 A33501        MOV     [FREMEM],AX
```

Examples:

'HU S.START' displays the last page of the hardware trace buffer, decoding all possible addresses to the segment of the symbol START.

'HU *' clears the hardware trace buffer.

Command: Hardware Write (requires Periscope III)

Syntax: HW <name> or HW <address>

Description: This command is used to save the contents of the hardware trace buffer to disk or memory. The saved trace buffer can be viewed with the HR, HT, and HU commands.

If DOS is not busy, use the first form of the command to save the trace buffer to a disk file. The file is always

49,158 bytes in length, which is the 48K trace buffer plus a 6-byte header.

If DOS is busy, use the second form of the command to save the trace buffer to memory. The address used must have an offset of zero and be in the first 640K of memory. Be careful not to overwrite DOS or your program. If COMMAND is active (i.e. the DOS prompt is shown), don't use the last 48K of memory, since this will overwrite the transient portion of COMMAND. After saving the buffer to memory, use the Go command to get to some point in your code so you can write the memory image to a disk file. When writing the file, set register BX to zero and register CX to C006H.

If it is not possible to write the trace buffer to disk or memory, enter 'HC S+;QC' to turn selective trace on and continue execution. This magic combination minimizes changes to the trace buffer so you can later save the trace buffer. **Note:** Do not use RUN.COM to re-enter Periscope since it always clears the trace buffer!

To save the trace buffer to disk from DOS, use 'PS3TEST /W'.

Examples:

'HW PSBUF.DAT' saves the current contents of the hardware trace buffer to the file PSBUF.DAT. To view the saved trace buffer, enter 'HT CS PSBUF.DAT', where CS is the "decode" segment.

'HW 8000:0' saves the current contents of the hardware trace buffer to memory starting at 8000:0. To view the saved trace buffer, enter 'HR CS 8000:0', where CS is the "decode" segment.

Command: Input

Syntax: I <port>

Description: This command is used to read an I/O port.

The port number may be from zero to FFFFH, although the IBM PC only supports ports from zero to 3FFH—any larger number is effectively ANDed with 3FFH. The byte value retrieved by reading the port is displayed on the line following the command.

Examples:

'I 100' performs a read of port 100H and displays the byte input.

'I DX' performs a read of the port indicated by register DX and displays the byte input.

Command: Interrupt Restore

Syntax: IR

Description: This command is used to restore the interrupt vectors to a previously-saved state.

This command is usable only after an IS command has been used to save the interrupt vectors. After an Interrupt Restore has been performed, the Interrupt Restore command is disabled until another Interrupt Save has been performed.

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

Syntax: IS

Description: This command is used to save the interrupts for later 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 debuggging 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' when the Periscope prompt is displayed. 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.

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

Syntax: J

Description: This command is used as a shorthand form of Go—to execute at full speed to the next instruction.

It executes the current instruction at full speed, avoiding single-stepping through the execution of CALL, INT, LOOP or other repeated instructions. This command 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 no code breakpoints can be used, 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

Syntax: JL

Description: This command is used to jump from one source-code line to the next source-code line.

This command is usable only when the current instruction corresponds to a high-level language source-code line.

The JL command performs Jump commands until the next source line is found. This is a quick method of moving through a high-level language program, keeping to the source-code lines.

Example:

Assume the current instruction is line 10 of the first source-code module. Enter 'JL' to go to the next logical source-code line. **Note:** If your compiler does not generate line numbers for every line, some lines may be skipped.

Command: Klear

Syntax: K

Description: This command clears the Periscope screen and regenerates the windows if used. It has no arguments.

Example:

'K' clears the screen.

Command: Klear and Initialize

Syntax: KI

Description: This command initializes the monitor, clears the Periscope screen, and regenerates the windows if used. It has no arguments.

This variant of the clear command should be used if Periscope's screen is not in text mode on entry to Periscope. Do not use this command if 43-line mode is set—it will revert the screen to 25-line mode.

Example:

'KI' performs a mode set and clears the screen.

Command: Load Absolute disk sectors

Syntax: LA <address> <drive> <sectors>

Description: This command is used to load absolute disk sectors into memory.

The segment defaults to CS if no segment is specified in the address. The drive is a single-digit number indicating the disk drive (0=A, 1=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. See the description of the Name command for more information. This command uses DOS interrupt 25H. See the DOS manual for information on the numbering of the absolute disk sectors.

Examples:

'LA DS:100 0 10 20' loads data into memory starting at DS:100 from drive A, starting at sector number 10H for 20H sectors.

'LA 100 1 0 4' loads data into memory starting at CS:100 from drive B, starting at sector 0 for 4 sectors.

Command: Load File from disk

Syntax: LF [<address>]

Description: This command is used to load a file from disk into memory.

The optional address specifies where the file is to be loaded. If the address is not specified, CS:100 is used. To use this command, DOS must not be busy. See the description of the Name command for more information. Before this command can be used, the Name command must be used to specify a file name.

The LF command can be used to load any type of file into memory. After the file has been loaded, BX and CX indicate the size of the file in bytes. After the file is loaded into memory no other processing occurs—EXE files

are not relocated or stripped of their headers. RUN.COM should generally be used to load and execute a program, since it loads the symbol table and performs relocation for EXE files. The LF command is useful for loading a file into memory for examination or modification.

Note: This command cannot load into memory beyond the end of DOS memory. For a 640K system, the maximum address is 9000:FFFF.

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

Syntax: LS * or <name>

Description: This command is used to load a Periscope symbol file (PSS) into the symbol table.

The asterisk clears the symbol table. If an asterisk is used, the command terminates. If it is not used, the symbols are added to the end of the symbol table. The segment contains the relocation factor that is added to the segment values found in the PSS file. 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 name is the path and file name of a PSS file. Do NOT enter the .PSS extension. The use of this command destroys the name set with the Name command.

This command cannot be used to load a MAP file—only PSS files are supported.

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.

Command: Move

Syntax: M <range> <address>

Description: This command is used to copy a block of memory to another location in memory.

The segment and offset must be specified for both addresses, to avoid accidental changes to memory. 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.

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). Since the source and target blocks overlap by 80H bytes, the move copies memory starting at 1000:FF and works down, so that the target block is an exact copy of the original source block.

'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). Since the source and target blocks overlap and the source is higher than the target, the move copies memory starting at 1000:80 and works up.

'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

Syntax: N <name>

Description: This command is used to enter data into the PSP for disk I/O and for naming files to be read or written by Periscope.

The name parameter is copied to a Periscope buffer for use with the Load and Write commands. It is then copied to the unformatted parameter area in the PSP, starting at CS:80H. After the name is copied into CS:80H, the DOS parsing function is used to parse the first two file names in the command line into the FCBs at CS:5CH and CS:6CH. If an invalid drive id is found for a file, a message is generated and register AL or AH is set to FF, indicating the first or second file, respectively.

This command copies all data entered after the N until a carriage return is found—it ignores the use of a semi-colon for entering multiple commands on one line. If the PSP cannot be found, Periscope can still be used to read or write the file, presuming that DOS is not busy.

Since the Name, Load, View, Unassemble Source, Write, Echo, and Exit to DOS commands use DOS calls, Periscope must check to be sure that DOS is not busy.

The Name command also requires that the PSP's address has been set by RUN.COM and that the first four bytes of the PSP contain the bytes 'CD 20' followed by the top of memory size in paragraphs.

The LS and WS commands destroy the file name set by this command. After using either of these commands, re-enter the desired Name command, if any.

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.

'N FILE1,FILE2/N' copies the file names to Periscope's internal file buffer and to the unformatted parameter area at CS:80H and then parses the file names into the FCBs at CS:5CH and CS:6CH.

Command: Output

Syntax: O <port> <byte>

Description: This command is used to output a byte to an I/O port.

The port number may be from zero to FFFFH, 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 zero to FFH.

Examples:

'O 100 FF' outputs FFH to port 100H.

'O DX 12' outputs 12H to the port indicated by register DX.

'O DX AX' returns an error since register AX represents a word—only bytes can be output via Periscope.

Command: Quit

Syntax: Q [<sub-function>]

Description: This command is used to exit the debugger and display Periscope's quit options.

The optional sub-function is used to pre-answer the quit option prompt. The possible combinations are QB, QC, QD, QL, QR, and QS to Quit and Boot, Continue, Debug, perform a Long boot, Return to DOS, or perform a Short boot, respectively. See the section in this chapter entitled 'Quit Options' for more information.

Examples:

'Q' exits the debugger and displays the quit options.

'QC' exits the debugger and continues execution of the interrupted program without setting any breakpoints.

'QS' exits the debugger and performs a short boot.

Command: Register

Syntax: R [<register>] or [F]

Description: This command is used to display and modify the current values of the registers and flags.

If you enter 'R' and press return, the current values of the registers and flags are displayed. If the current instruction performs a memory read and/or write, the effective address of the read/write is displayed, along with the current value of memory at the effective address(es). Finally, the current instruction is disassembled. The information is shown in the appropriate windows if windows are being used.

Some examples are shown below:

Example 1:

```
AX=007F  BX=0034  CX=0000  DX=0000  SP=1724  BP=00A0  SI=0F1E  DI=1560
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=FFFD  BP=0000  SI=0000  DI=0000
DS=063A  ES=063A  SS=063A  CS=063A  IP=010E  FL=0246 NV UP EI PL ZR NA PE NC
WR DS:0131 = 0000
063A:010E  891E3101     MOV     [FILE_OFFSET],BX
```

Example 3:

```
AX=0000  BX=0000  CX=0100  DX=0001  SP=FFFB  BP=0000  SI=0000  DI=0000
DS=063A  ES=063A  SS=063A  CS=063A  IP=01AD  FL=0246 NV UP EI PL ZR NA PE NC
              P565:
063A:01AD  BF2F01       MOV     DI,012F                       ; FILE_SEGMENT
```

In all of the above examples, the first two lines display the current values of the registers and the flags. See the table below for an explanation of the flag mnemonics. The last line in each of the examples shows the disassembled instruction. The address of the instruction (CS:IP) is shown at the left, followed by the one to six bytes that make up the instruction, and the instruction itself.

In Example 2, 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 access before the execution of the instruction. The

effective address calculations and displays for the 8086, 8087, 8088, 80186, 80287, and 80286 real mode instructions are supported, with the exceptions that 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 and that the FRSTOR and FSAVE instructions do not show the 94 bytes they read and write.

In Example 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), 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). 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). 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 FILE_SEGMENT.

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, SS, CS, and IP. The valid 8-bit register names are AH, AL, BH, BL, CH, CL, DH, and DL.

To modify a flag, enter 'R F'. Periscope displays the current values of the flags (see the table below) 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. The flags may be entered 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.

FLAG	SET (=1)	CLEAR (=Ø)
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)
Carry	CY (STC)	NC (CLC)

Examples:

'R' displays all registers and flags, the effective address for reads and/or writes, and disassembles the current instruction.

'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 IP IP+1' modifies the instruction pointer to its current value plus one.

'R F' 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 F ZR'.

Command: Register Restore

Syntax: RR

Description: This command is used to restore the registers to a previously-saved state.

This command is usable only after a RS command has been used to save the registers. After a Register Restore has been performed, Restore is disabled until another Register Save has been performed.

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

Syntax: RS

Description: This command is used to save the registers for later 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' when the Periscope prompt is displayed. Later, you can restore the registers to their saved state by using 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.

Command: Search

Syntax: S <range> <list>

Description: This command is used to search memory for a byte/string pattern.

The block of memory specified by the range is searched for the pattern specified by the list. If a match is found, the starting address of the match is displayed and the search for matches continues at the next byte. If no matches are found, nothing is displayed. If no segment is specified in the address, the current data segment is used.

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 PRINT_LINE L 50 C "Page"' searches 50H bytes starting at the address of the symbol PRINT_LINE for the byte 0CH followed by the string 'Page'.

Command: Search for Address reference

Syntax: SA <range> <address>

Description: This command is used to search memory for references to a specified address.

This command can be thought of as a disassembly that only shows instructions that reference an address of interest. To use it, specify an address range that is to be searched and the address reference that is to be searched for. If you're not using a symbol name for the address reference, be sure to specify the segment register that is to be used. 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.

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

Syntax: SC [<byte>]

Description: This command searches the stack for references to CALLed subroutines and software INTerrupts. If a match is found, the disassembly of the CALL or INT is displayed. 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.

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 instruction analyzes 32 stack entries unless the optional argument is used or unless the value of SP wraps around from FFFF to zero.

This command does not interpret hardware interrupts since there is no interrupt in the instruction stream to indicate what happened.

Example: 'SC' searches the stack for CALLS and INTS. If any are found, the disassembled instruction is displayed. Note that the most recent item is displayed first.

Command: Search then Display

Syntax: SD <range> <list>

Description: This command is used to search memory for a byte/string pattern and then display the 'find(s)'.

This command is the same as the Search command, except that any matches are displayed in byte format. If a data window is used, the matching address is displayed in the active window. After a key press, the search continues. 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

Syntax: SR [<byte>]

Description: This command searches the stack for return addresses. Each stack item is examined 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 SR analyzes the stack looking inward.

If a match is found, the address of the instruction and the offset to the nearest symbol after the CALL or INT is displayed. This technique can help you determine the calling sequence used by a program 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. This instruction analyzes 32 stack entries unless the optional argument is used or unless the value of SP wraps around from FFFF to zero.

Some programs may manipulate the stack in ways that cause this command to fail. For example, Lattice C can add an odd value to the SP register, which causes the stack to be viewed incorrectly.

This command does not interpret hardware interrupts since there is no interrupt in the instruction stream to indicate what happened.

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

Syntax: SU <range> <list>

Description: This command is used to search memory for instructions that match a pattern.

This command can be thought of as a disassembly that only shows instructions that match a specified pattern. To use it, specify an address 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.

Be sure to enter the search list as a quoted string. Note that there are seven spaces from the start of the mnemonic field to the start of the operand field—to find all occurrences of MOV SP, enter 'MOV', four spaces, and 'SP'. **Note:** this command will not find source-code lines or procedure labels—just disassembled instructions. Any lower case input is converted to upper case before the search is performed. The search starts in the mnemonic field (column 25) and goes through the end of the argument field (column 61).

Examples:

'SU CS:100 L 200 "MOVbbbbSS"' searches from CS:100 for 200H bytes for any instructions that contain 'MOVbbbbSS', where 'b' is a blank. There must be exactly four spaces after the 'MOV' for the command to work.

'SU PSTART PEND "POP"' searches from the address represented by PSTART through the address represented by PEND for POP instructions.

Command: Trace

Syntax: T [<number>]

Description: This command is used to execute the current program one instruction at a time.

If the optional number is not entered, one instruction is executed and control is returned to Periscope. If the number is entered, that number of instructions is traced. After each trace, the registers, effective address, and current instruction are displayed.

If you're using a single-monitor system to trace code that does not do any screen writes, you may want to use the F2 key 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, the Trace command can be used to trace through ROM, since it works by changing the trap flag and not by modifying the code to be 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

Syntax: TB [*] ; TR [*] ; TU [*]

Description: These commands are used to view the software trace buffer. Do not confuse this buffer with the real-time hardware trace buffer available with Periscope III.

The software trace buffer is used to save the machine registers each time Periscope is exited. This circular buffer can contain zero to 2016 entries, depending on the

installation option /B:nn. Each entry contains the machine registers and an ascending sequence number. When displayed, the buffer shows the registers, sequence number, and a symbolic disassembly of the instruction indicated by the saved CS:IP. The following example shows three consecutive entries in the format used by the TB command.

```
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
```

The trace buffer may be cleared by entering 'TB *'. If an asterisk is not used, a full-screen display mode is entered. The only way to exit this mode is to press the Esc key. While in full-screen mode, the cursor is positioned at the lower left-hand corner. The keys available are: Home, End, Up, Dn, PgUp, PgDn, and Esc. If you press any other key, the message 'Press Esc to end full-screen mode' is displayed.

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. For example, for the trace records shown above, TR shows:

```
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
```

Similarly, the TU command shows:

```
              START:
15E6:0137  E81700       CALL    GETMEM
            GETMEM:
15E6:0151  B106         MOV     CL,06
15E6:0153  BE0200       MOV     SI,0002
```

Since Periscope adds to the buffer each time it is exited, watch out for possible discontinuities in the software trace buffer. If you're using the T, GA, or GT commands, there's no problem. If you're using the G or J commands, not all instructions will be 'seen' by Periscope—the unseen instructions leave 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 the instructions have been changed. If the trace buffer is empty, a TB, TR or TU command has no effect.

When using the TB, TR, or TU commands to display the software trace buffer, you can switch among any of the three formats by keying B, R, or U, respectively.

Examples:

'TB' shows both the register and disassembly display of the end of the software trace buffer.

'TR' shows just the register display.

'TU' shows just the disassembly display.

'TB *' clears the software trace buffer and resets the sequence number to zero.

Command: Unassemble memory

Syntax: U [<range>]

Description: This command is used to disassemble memory into the 8086, 8087, 8088, 80186, 80287, and 80286 real-mode instructions.

Memory is disassembled in the ASM, Source/ASM (Both), or Source mode as set by the UA, UB, and US commands respectively. Source/ASM is the default mode, unless the /E:0 installation option was used, in which case ASM is the default mode.

The syntax for this command is very flexible. If you enter 'U', the disassembly starts where the last U command left off. The commands G, J, R, and T reset the starting point to CS:IP. If you enter 'U <number>' the number is presumed to be an offset, the segment is presumed to be CS, and the length is presumed to be 20H. If you enter 'U <number> <length>' the number is presumed to be an offset, and the segment is presumed to be CS.

Two sample disassemblies are shown below. Both are of the same range of memory, but the second listing was made using a symbol table. Note the difference in readability.

Without symbols:

```
1261:0137 E81700        CALL    0151
1261:013A A13301        MOV     AX,[0133]
1261:013D BF1001        MOV     DI,0110
1261:0140 E82400        CALL    0167
1261:0143 A13501        MOV     AX,[0135]
1261:0146 BF2C01        MOV     DI,012C
1261:0149 E81B00        CALL    0167
1261:014C E83400        CALL    0183
1261:014F CD20          INT     20
```

With symbols:

```
                START:
1261:0137 E81700        CALL    GETMEM
1261:013A A13301        MOV     AX,[TOTMEM]
1261:013D BF1001        MOV     DI,0110                 ; TMEMORY
1261:0140 E82400        CALL    CONVERT
1261:0143 A13501        MOV     AX,[FREMEM]
1261:0146 BF2C01        MOV     DI,012C                 ; AMEMORY
1261:0149 E81B00        CALL    CONVERT
1261:014C E83400        CALL    DISPLAY
              DOSRET:
1261:014F CD20          INT     20
```

Each line of the disassembly shows the address of the instruction (CS:IP) followed by the one to seven bytes that make up the instruction. Next the instruction is displayed.

If an address in the instruction is an exact match with an entry in the symbol table, the symbol name is substituted for the address. For example, in the second line, address DS:0133 is referenced. When the symbol table is searched, the name TOTMEM is found and displayed instead of 0133.

If you're debugging programs where DS and/or ES do not initially point to the data area(s), variable references such as this one are not shown until DS and/or ES are changed to point to the data area(s) within your program. Code references, such as the label START are found, since CS must be correct for the program to execute.

If an ambiguous reference to an address is made, the symbol name is shown 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.

Segment override instructions (CS:, DS:, ES:, and SS:) are shown folded into the instruction they affect except when a segment override precedes an instruction that does not have a memory operand. In this case, the segment override appears to the left of the mnemonic field. Other prefixes (REP, LOCK, etc.) are shown on a separate line preceding the instruction they affect.

If a disassembly window is used, the PgUp, PgDn, PadPlus, and PadMinus keys can be used to move forward and backward through memory. These keys affect the disassembly window unless a data window is in use and the last command was a display command.

Examples:

'U NEW_PAGE' disassembles memory starting at the symbol NEW_PAGE. The default length of 20H bytes is used.

'U CS:IP L 1' disassembles memory for one instruction at the current instruction. The same result can be achieved by using the Register command.

'U' disassembles memory starting where the last U command left off. If the G, J, R or T commands were used, the disassembly starts at CS:IP.

Command: Unassemble ASM instructions

Syntax: UA [<range>]

Description: This command is used to enable ASM-only disassembly and to disassemble memory.

Use this command to turn off source-level debugging. After the subfunction has been entered once, it is not necessary to enter it again. See the U command above for more information.

Examples:

'UA NEW_PAGE' disassembles memory starting at the symbol NEW_PAGE. The default length of 20H bytes is used.

'UA A10 L 1' disassembles memory for one instruction starting at the symbol A10.

'UA' disassembles memory starting where the last U command left off. If the G, J, R or T commands were used, the disassembly starts at CS:IP.

Command: Unassemble Both asm and source

Syntax: UB [<range>]

Description: This command is used to enable mixed ASM and source disassembly and to disassemble memory.

Use this command to enable ASM and source-level debugging. This mode shows high-level source code,

followed by the ASM code generated by the high-level code. After the subfunction has been entered once, it is not necessary to enter it again. This mode is the default unless the /E:0 installation option is used. See the descriptions of the U and US commands for more information.

Examples:

'UB .A10' disassembles memory in Source/ASM format starting at the symbol A10.

'UB' disassembles memory starting where the last U command left off. If the G, J, R or T commands were used, the disassembly starts at CS:IP.

Command: Unassemble Source

Syntax: US [<range>]

Description: This command is used to enable source disassembly and to disassemble memory.

Use this command to turn on source-level debugging of a high-level language. This mode shows a minimum of ASM code—it shows assembly code until the first source line is found and then shows just high-level source code.

If you're using a recent version of the IBM/Microsoft linker and you've set up the MP and MX aliases described in Chapter V, Periscope should never prompt you for a file name for source-level debugging. (If you haven't already done so, PLEASE take the C tutorial in Chapter V.) Otherwise, when needed, Periscope prompts for the file name corresponding to the module being disassembled. Enter the file name and press return to display the source code. If the file is not found, the prompt is displayed again. 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'. For Phoenix's PLINK, the source file name must be entered manually—no automatic module name extraction is provided by Periscope.

To display source code, the following conditions must be met:

- DOS must not be busy (see the description of the Name command for more information)

- A file buffer must be available (if PS.COM was installed with /E:0, this command is not available)

- Line symbols must be available for Periscope to be able to associate an instruction with a source-code line

- Any disk I/O errors will cause an incomplete source display or none at all

To see the correspondence between the module names and module prefixes (A, B, etc.), enter '/S FFFF FFFF'. This will display all line name records, showing the module prefix and the module name.

The MP alias should usually end with a backslash. The MP and MX aliases can be overridden with a colon and a period respectively. If one of the aliases is incorrect, you can enter the correct path, file and extension or better yet, press F3 and then edit that file name.

When a disassembly window is used, the PgDn key displays more source code. The new page will always start at a source line referenced in the symbol table. The other positioning keys (PgUp, PadPlus, and PadMinus) may show ASM code.

Examples:

'US A10' disassembles memory starting at the line symbol A10. The default length of 20H bytes is used.

'US' disassembles memory starting where the last U command left off. If the G, J, R or T commands were used, the disassembly starts at CS:IP.

Command: View file

Syntax: V <name>

Description: This command is used to view a text file from within Periscope. Don't try to use it to display a file that's not in ASCII!

The name is any legal file name, including drive, path, file, and extension. To use this command, DOS must not be busy (see the description of the Name command for more information). A file buffer must be available—if PS.COM was installed with /E:0, this command is not available.

The file is displayed in the non-windowed area of the screen, unless a View window has been set up. The line numbers displayed are shown in the lower left-hand corner of the screen. Use the PgUp and PgDn keys 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 the Home and End keys 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 the enter key to get back. When you're finished viewing the file, press the Esc key to return to Periscope's prompt. Note that Ctrl-Break cannot be used to terminate this command—Esc is the only way out.

A simple string search is available. Enter a 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 slash and press F4. If no match is found, the last record in the file is displayed.

To position to a specific line, enter a pound sign ('#'), the decimal line number (1 to 65535), and press return. The desired line is then displayed.

If a View window is used, the window is static after Esc has been pressed to return to the Periscope prompt. If the windows are changed or the screen is cleared, the View window is lost. Use the View command 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

Syntax: VS

Description: This command is used 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: Write Absolute disk sectors

Syntax: WA <address> <drive> <sectors>

Description: This command is used to write memory to absolute disk sectors.

The segment defaults to CS if no segment is specified in the address. The drive is a single-digit number indicating the disk drive (0=A, 1=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. See the description of the Name command for more information. 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 a file allocation table (FAT) or a 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 used with the Load and Write commands are the same.

Examples:

'WA DS:100 0 10 20' writes data from memory starting at DS:100 to drive A, starting at sector number 10H for 20H sectors.

'WA 100 1 0 4' writes data from memory starting at CS:100 to drive B, starting at sector 0 for 4 sectors.

Command: Write File to disk

Syntax: WF [<address>]

Description: This command is used to write a file from memory to disk.

The optional address specifies where the memory image of the file begins. If an address is not specified, CS:100 is used. To use this command, DOS must not be busy.

See the description of the Name command for more information. Before this command can be used, the Name command must be used to specify a file name.

This command can be used to write any type of file to disk. Before the file is written, be sure that BX and CX indicate the size of the file in bytes. Do not attempt to write an EXE file that was not loaded with the LF command—an EXE file loaded by RUN.COM is missing its header and cannot be written to disk.

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

Syntax: WS <name>

Description: This command is used 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 name is the path and file name of a PSS file. Do NOT enter the 'PSS' extension. Note that the use of this command destroys the name set with the Name command.

This command cannot be used to write a MAP file—only PSS files are supported. The symbol tables are left unchanged by this command.

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 <name> to write the symbols without further relocation. Then use LS * to clear the symbol table, followed by LS <name> 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.PSS.

'WS 0 C:TEST' subtracts zero from the symbol's segments and writes the file C:TEST.PSS.

Command: Xlate (translate) Hex number

Syntax: X <number> or XH <number>

Description: This command is used to translate a one- to four-digit hexadecimal number or a register to its decimal, octal, binary, and ASCII equivalents.

Example:

'X 5051' displays '5051h 20561d 050121o 0101 0000 0101 0001b PQ'.

Command: Xlate (translate) Address

Syntax: XA <address>

Description: This command is used 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.

Example:

'XA 1234:5678' displays '179B8'.

Command: Xlate (translate) Decimal number

Syntax: XD <decimal number>

Description: This command is used to translate a one- to five-digit decimal number to its hexadecimal, octal, binary, and ASCII equivalents. The number must be from zero to 65535. The number may not have any punctuation, such as commas or periods. Numbers larger than 65535 can be translated, but the high order part is lost.

Example:

'XD 20561' displays '5051h 20561d 050121o 0101 0000 0101 0001b PQ'.

Command: Option D (Data window select)

Syntax: /D [<byte>]

Description: This command is used to select the active data window when more than one data window is in use.

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 one data window is in use, this command has no effect. If no number is entered, the next data window is made active (as indicated by the up arrow after the window number). If a number that corresponds to a data window (0-3) is entered, that window is made 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 [<name>]

Description: This command is used to echo Periscope's screen output to a disk file.

All non-windowed output is 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. Do not use semi-colons to 'stack' commands when this command is used. 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 another '/E' command is used, 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 N (Nearest symbols)

Syntax: /N [<address>]

Description: This command is used to search for the symbols nearest to the specified address.

Up to three symbols are displayed—the next lower symbol (left); the equal symbol (middle); and the next higher symbol (right).

This command can help you get your bearings when you've interrupted an executing program by showing you the nearest symbols. If no address is entered, CS:IP is assumed. The nearest symbol that is located lower in memory is displayed at the beginning of the line, followed by the symbol for the specified address, which is followed by the nearest symbol that is located higher in memory. If no lower, equal, or higher symbol is found, nothing is displayed.

Examples:

Assume three symbols X, Y, and Z located at 1000:100, 1000:200, and 1000:300 respectively.

'/N 1000:200' displays:

```
'1000:0100 X              1000:0200 Y              1000:0300 Z'
```

'/N 1000:0' displays:

```
'                                                  1000:0100 X'
```

Command: Option R (Remove symbol)

Syntax: /R <symbol>

Description: This command is used to remove a symbol from the symbol table.

This command is used to eliminate undesired symbols. Since symbol names are evaluated before register names, a symbol named AX would disable references to register AX unless this command was used.

Examples:

'/R AX' removes the symbol AX, leaving no conflict with the register named AX.

'/R B123' removes the symbol B123. Be careful when removing line number symbols, since these cannot be re-entered using the ES command.

Command: Option S (Segment change)

Syntax: /S

Description: This command is used to make global changes to the values of segments in the symbol table.

The entire symbol table is searched for symbols having a segment that matches the first segment entered. If a match is found, the symbol's segment is changed to the second segment entered and the new address and the symbol name are displayed. This command is used to adjust the segments of symbols when a program relocates its data areas, such as in Microsoft BASIC, FORTRAN, and Pascal.

Examples:

'/S F00D DS' changes the segment of all symbol table entries that are currently F00D to the current value of DS.

'/S CS CS' displays the segment of all symbol table entries that match the value of CS. This is a good method for querying the symbol table without changing anything.

Command: Option T (Trace interrupt table)

Syntax: /T [?] [*] [#] [<byte>] [...]

Description: This command is used to force tracing of interrupts when the GT command is used.

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 the GT command is used, 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 Periscope is first installed, these interrupts are flagged for forced tracing. Using this command, you can change the interrupts that are to be traced when GT is used. The possible command arguments are *, #, ?, and numbers from 0 to FF (always presumed hex). The * clears all traps and # sets all traps. A ? displays the current trace list. A 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 function 4BH (Exec) is used or if Borland's SIDEKICK is in the system.

- When an interrupt is not traced, Periscope becomes dormant until the second instruction after the INT XX instruction.

- If a GT command is used 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: This command is used to perform user-written code from Periscope.

To use this command, a program similar to USEREXIT.ASM as described in Chapter IX must be installed and PS.COM must be installed with the /I option. The number entered after the /U command must be from nine to FFH. It is passed to the user-written program in register AH. Other information is passed, including the optional address entered on the command line. See Chapter IX for more information.

USEREXIT.COM has a status display for the 8087 and 80287 numeric processors. To use USEREXIT, load it before PS.COM is loaded. Then, use /I:60 when installing PS.COM. From within Periscope, enter '/U 87' to display the status of the numeric processor.

Example:

Assuming that a user-written interrupt handler has been installed using INT 60H and that PS.COM had the /I:60 installation option, /U 9 performs user exit number 9.

Command: Option W (Window setup)

Syntax: /W [D<:byte>] [R] [S<:byte>] [U<:byte>] [V<:byte>]

Description: This command is used to change Periscope's windows from within Periscope. Its use and syntax are identical to the /W installation option. (See Chapter VI.)

Periscope can window Data, Stack, Register, Unassembly and/or View information. Once windows are established, the windowed data is displayed at a constant location on the screen and is updated after each command (the View window is updated only when the View command is used).

The tokens D, R, S, U and V indicate the type of data to be windowed. The tokens are optional and may be in any order. If a token is omitted, the corresponding type of information will not be windowed. The windows are displayed in the same order as the tokens are

encountered on the input line, except for the stack window, which is always on the right-hand side.

The D window shows data in any of the display formats. The window continues to show the same address until another display command is used. The output of the Display Record command is not shown in this window. Duplicate lines are not suppressed for windowed data. When RUN.COM is used to enter Periscope, the display address is set to DS:100.

Up to four data windows may be used, 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. If the start address of a data window matches a symbol, the symbol name is displayed in the separator line at the end of the window. After a display command is used, the PgDn, PgUp, PadPlus, and PadMinus keys may be used to browse forward and backward through memory.

The R window shows register and flag information. The length is fixed at two lines. The effective address of any memory reads or writes is shown in the separator line following this window.

The S window is vertical, on the right-hand side of the screen. 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! An arrow in the left margin of the window indicates the current value of BP. Since the stack window can overlay the output of the byte and register display, you may turn the stack window on and off using the Alt-S key. If you choose not to use a stack window, you can always view the stack using DW SS:SP. Note that the stack is read from the upper right-hand corner of the screen downwards.

The U window shows disassembled instructions. The address used initially for the disassembly defaults to CS:IP and is reset to CS:IP each time a G, J, R, or T command is used. Any area of memory can be disassembled by using the U command with the desired address. The disassembly window shows the current instruction in reverse colors. The current high-level source line and/or procedure labels are also shown in reverse. 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. When tracing, Periscope minimizes the regeneration of the disassembly

window and just moves the reverse video bounce bar when possible. After any command other than a display command has been used, the PgDn, PgUp, PadPlus, and PadMinus keys may be used to browse forward and backward through memory.

In the separator line following the disassembly window, the current location is shown. For DOS 2.x, the message 'In DOS' is shown if the current CS is less than A000H, otherwise the message 'In BIOS' is shown. For DOS 3.x, Periscope uses the DOS memory allocation blocks to get the actural program name when possible. Two caveats—the lookup is based on CS only and 'In RUN.COM' will be shown when in a program loaded by RUN.COM.

The V window is used to view a text file. After establishing a View window, the View command uses the space reserved by the window. See the description of the View command for more information.

The byte parameter defines the length of the window in hex. If no length is specified, a default is used. The maximum length for any one window and the total area that can be windowed is four lines less than the screen length, including a separator line following each window. When a length specification is used, at least one space must follow the number. If you're using 43-line mode, the windows can get quite large. Since the windows are regenerated after each command, large windows can slow Periscope's response time.

The default and minimum number of lines for each of the five window types are:

	Default	Minimum
Data	4	1
Register	2	2
Stack	(Total of other windows)	
Unasm	4	4
View	4	1

The colors of the windows can be individually set, using a hex number from 1 to FF (the numbering scheme is the same as that used by the /C installation option described in Chapter VI.). To set the colors, append the hex number to the line length, separated by a '.'. For example, Periscope's 'standard' window setting is /W D:4.74 R.47 S.71 U:8.17, which sets a four line data window with color 74 (red on white); a register window with color 47 (white on red); a vertical stack window with color 71 (blue on white); and an eight-line disassembly window with color 17 (white on blue). The current instruction is shown in reverse (blue on white). This default window

setting is available by pressing Ctrl-F10. If you're using a monochrome monitor, try using Ctrl-F9 instead.

If you want to turn off all windowing, enter '/W' with no arguments.

Examples:

'/W D:8 R'—Window data in the first 8 lines of the screen, followed by two lines of register information. A total of 12 lines are used for windows, including the two separator lines.

'/W SRU'—Window register information in the first two lines of the screen, followed by four 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: Exit to DOS, presuming DOS is not busy and memory has been freed.

This command requires DOS 3.00 or later. DOS must not be busy and memory must have been freed using DOS function call 4AH. No arguments are allowed. To return to Periscope, enter 'EXIT' at the DOS prompt. During the exit, don't execute RUN.COM, PS.COM, or change the state of the system (e.g. change monitors).

Example:

'/X'—Assuming memory has been freed and that DOS is not busy, this command exits Periscope and displays the DOS prompt. When you're ready to return to Periscope, enter 'EXIT' at the DOS prompt.

VIII—RUNning Your Program

• Loading Your Program With RUN

Use the program RUN.COM to load COM or EXE files and enter Periscope. (Periscope must be installed.) For help, enter 'RUN ?' when the DOS prompt is displayed.

RUN can also be used to load data files or no file at all. If no file is loaded, the first instruction is set to INT 20H, the DOS return, to prevent accidental execution of meaningless data. If a data file is loaded, be sure to use the QR command to quit Periscope and return to DOS. Using QC or G(o) would have unpredictable results.

RUN is started by entering 'RUN filename.ext command-line' at the DOS prompt, where filename.ext is the path, file name, and extension (EXE, COM or other) of the file to be loaded. The command line is the same one used when the program is started from DOS. RUN adjusts the FCBs and command line in the PSP to look like the target program had been started directly from DOS.

RUN resets Periscope's display address to the PSP segment at offset 100H. It also clears some of Periscope's tables, including the software trace buffer, source file buffer, screen buffers, record definition table, and symbol table. If any breakpoints are set, they are disabled to avoid possible interference with the current program being debugged.

If a file is specified, the specified directory is searched for a file of the form filename.DEF. If this file is found, it is presumed to be a record definition file. If it is not found, the file PS.DEF is used if available. The DEF file is then used to load Periscope's record, alias, and keyboard definition tables. If a DEF file is not found, the record and alias definitions are cleared, but any keyboard definitions are not cleared. To set the Periscope path, enter 'SET PS=xxx', where xxx is the path required to

find the DEF file. If an error is found in the DEF file, the record definition table will be partially loaded. See Chapter IX for more information.

If the file extension is COM or EXE, the specified directory is searched for a file of the form 'filename.PSS'. If this file is found, it is used instead of the MAP file for the program's symbols. If the PSS file is not found, the directory is searched for a file of the form filename.MAP. This file is then used to load Periscope's symbol table with address and line references. If a MAP file is not found, the symbol table is cleared. If an error is found in the MAP file, the symbol table is partially loaded. If you're using the IBM/Microsoft linker, either the MAP file or a PSS file may be used. Other linkers must use a PSS file. See Chapter IX for more information.

RUN then 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. Registers BX and CX are set to the size in bytes of the target file. Other registers are set according to the rules for loading COM and EXE files (see the DOS 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, this area shows RUN.COM as the loaded program rather than the target 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, consider loading it normally and then loading symbols using the 'LS' command or SYMLOAD.COM (see Chapter IX).

Note: RUN.COM has an option that can be used to have the DOS EXEC function load your program. This option should be used if you experience problems using RUN. When the EXEC option is used, the program is loaded approximately 7KB higher in memory than otherwise, but the program name in the environment space is correct. To use the EXEC function, add '/X' immediately after RUN. For example, enter 'RUN/X FTOC.EXE'. If an EXEC error occurs, error 82 is displayed. This error usually indicates a bad path name.

Your program is loaded exactly where it would be if DOS were to load it under the same conditions. This feature allows RUN to be used to load memory-resident programs. Until RUN is used again, the record definition, alias, keyboard definition, and symbol tables are preserved.

Finally, control is passed to the resident portion of Periscope. When you've finished debugging your program, you can exit Periscope in one of three ways—use a Go 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.

IX—Using The Periscope Utilities

- CLEARNMI.COM—A memory-resident program that ensures NMI is available

- CONFIG.COM—Configures Periscope for your system

- INT.COM—Display, save, and compare the interrupt vectors

- PS3TEST.COM—Used to test the Periscope III breakpoint and trace buffer functions

- PSKEY.COM—Sets hotkey(s) used to activate Periscope

- PSTEST.COM—Used to test the memory on the Periscope I and III boards

- PUBLIC.COM—Generates public statements for assembler programs

- RS.COM—Used to verify and size record, alias, and key definitions

- SYMLOAD.COM—Loads Periscope's symbol tables from within your program

- SYSLOAD.SYS—Loads COM files at CONFIG.SYS time, allowing you to debug device drivers

- TS.COM—Verify and size MAP files and generate PSS files from the output of various linkers

- USEREXIT.ASM and USEREXIT.COM—A sample program to perform user exits and user breakpoints from Periscope

---

Periscope supports DOS 2.00 pathnames for all file I/O. Programs that perform file I/O accept command lines

containing any legal DOS pathname. Note that a pathname must be terminated by a carriage return (end of command line), a space, or a slash.

CLEARNMI.COM

Despite the name, the non-maskable interrupt (NMI) used by the break-out switch can indeed 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.COM 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 file. The only installation option is /Q, which should be used only if the /Q installation option is used with PS.COM to indicate a hybrid machine.

This program can be installed multiple times, but each install allocates more memory and does not clear the effects of the previous install.

CONFIG.COM

This program is used to configure Periscope for your system. To run this program, boot the system so that no memory-resident programs or device drivers are installed. (You may omit this boot step with no memory-resident programs or device drivers installed only if you're using an IBM or Compaq system.) Place the Periscope distribution disk in drive A: and enter 'CONFIG'. To use a disk drive other than A:, enter the drive on the command line. The program then shows a full-screen display (see Chapter II) that prompts you to answer four questions.

The first field is for the Model of Periscope to be used. Enter the number corresponding to the Model desired. **Note:** To run Model I and Model III, you'll need the appropriate Periscope board installed. Models II and II-X do not require any hardware. If you're planning to run Model I without the board, you can either configure it as a Model II or you can use the /N installation option at run time (see Chapter VI).

The second field indicates the target drive. Normally, in a hard disk system, this would be drive C:. Any legal drive id from A: through Z: may be used. If the subdirectory \PERI\ does not exist on the target drive, it is created. You can copy the Periscope files to another directory later if you like.

The third field indicates whether the ancillary files are copied from the distribution disk to the target disk. A reply of 'N' causes only PS.COM to be written to the target drive.

The fourth field indicates the minimum command length that is added to the circular command buffer. The default value of zero may be changed to any value from one to nine. If you choose a value of two, commands up to two characters in length are not added to the circular buffer.

After entering your responses to the four prompts, press F10 to configure Periscope. Note any warning or error messages displayed as the configuration occurs. If you get a message of the form 'Segment for interrupt xxH is not F000H—Use /V option for this interrupt', be sure that these /V installation option(s) are used each time PS.COM is run. (See Chapter VI.)

If CONFIG.COM finds that an interrupt vector does not have the expected entry point, it will patch the entry point in PS.COM, unless the system is an IBM or Compaq computer. To force patching of an IBM or Compaq system, enter 'CONFIG 0' from the DOS prompt. To suppress patching of a known 100% compatible system, use 'CONFIG 1'. Generally, you should not have to suppress or force patching if you've followed the instructions above.

INT.COM

This program is used to display, save, or compare the interrupt vectors. The three usage modes are:

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

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

INT <filename> /C—compare 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.COM, do the following:

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'.

PS3TEST.COM

This utility program is used to perform breakpoint and trace buffer tests on the Periscope III board. (PSTEST /3 is used to test the Periscope III protected memory—see PSTEST.COM later in this chapter for more information.)

The options available for PS3TEST are:

/B—Test the break-out switch
/C:nnnn—Run tests multiple times, where nnnn is a hex number from 0 to FFFF
/D—Run DMA test using a DOS disk in drive A
/E—Set error exit mode, activating Periscope if an error is found
/I—Set inverted mode, suppressing error messages
/M:nnnn—Use protected memory segment other than D000H
/P:nnn—Use write protect port other than 300H
/S—Set silent mode, showing only error messages
/V—Set verbose mode, placing each message on a separate line
/W—Write the contents of the hardware trace buffer to the file PSBUF.DAT

PS3TEST performs many different tests to validate the correct operation of a Periscope III board. If no errors occur, the message 'No errors detected' is displayed. During the test of 8-bit memory, the program uses a section of the monochrome screen as a scratch pad—don't worry about the characters that flash across the end of the fifth line.

If an error occurs, a message with possible sub-errors is displayed and the next test is begun. If you get any errors, please check the following items before calling Tech Support.

- Is the computer an IBM PC, XT, AT or 100% compatible machine? This means no zero wait-state machines, no systems with 'turbo' processor cards, and no 80386-based systems. Many machines that are compatible at the software level are not compatible at the hardware level. Known incompatible machines include the IBM XT/286, the IBM System/2 machines, the Compaq 8086 and 80386

Deskpros, the AT&T 6300 and 6300 Plus, the Sperry PC and IT, and the Zenith Z248.

- Check the placement of the umbilical socket and cable. Make sure that the socket is correctly installed and firmly seated. Make sure that the cable is plugged in to the four-position connector near the mounting bracket.

- Check the settings of the DIP switches. If they've been changed from the standard settings, be sure to specify the appropriate '/M' and/or '/P' options.

- What speed is the system running? If the CPU speed is less than or equal to 8 MHz, the standard Periscope III board should suffice. If you're running between 8 MHz and 10 MHz, you need the high-speed (10 MHz) board. If you're running faster than 10 MHz, slow the system down and run PS3TEST again.

If the board has previously worked in this machine, call Tech Support for help. Otherwise, please test the board in a known compatible system (IBM PC, XT, AT or Compaq 8088 or 80286 system) before calling Tech Support.

PSKEY.COM

PSKEY.COM is a memory-resident utility program that is used to set hot keys. It replaces the /J and /K installation options available in previous versions of Periscope, plus it lets you select your own hot key combination to activate Periscope. The options available are:

P - Use Shift-PrtSc to activate Periscope (via INT 5)
S - Use Sys Req to activate Periscope (via INT 15H)
3 - Use INT 3 instead of INT 2 to activate Periscope
A - Alt (combined with other shift keys)
C - Ctrl (combined with other shift keys)
L - Left shift (combined with other shift keys)
R - Right shift (combined with other shift keys)
I - Insert (combined with other shift keys—must be first key pressed if used!)

For example, PSKEY LRS would activate Periscope when the Sys Req key is pressed or when the Left and Right shift keys are simultaneously pressed.

Note: If you have configured Periscope as Model II-X, the 3 option must be used. This is necessary because Model II-X does not support INT 2 (NMI). If the 3 option is not used, PSKEY will beep, indicating that it was not able to

activate Periscope via INT 2. Unless you need to use NMI, configure Periscope as Model II. If the 3 option is used, you'll need to modify IP after a stop—use R IP IP+1 to skip over the INT 3 instruction.

When using the P or S keys, Periscope comes up in the keyboard handler, far away from your code. On the other hand, since the shift key combinations 'back-end' the keyboard interrupt, a single Trace command puts you into the code that was interrupted by the key press! The shift key combinations can be defined as needed to avoid conflicts with other memory-resident programs.

This program can use interrupts 2 or 3, 5, 9 and 15H, depending on the command line options. It can be installed multiple times per DOS session, but each install allocates more memory and does not clear the effects of the previous install.

If PSKEY cannot find Periscope via interrupts 2 or 3 when the hot keys are pressed, it beeps the speaker and does not invoke Periscope. Since PSKEY is loaded into normal memory and is dependent on hardware interrupts being enabled, use the break-out switch for maximum dependability.

PSTEST.COM

This program is used to perform memory tests of the Periscope I and III protected memory. The options available are:

/0—Test original Periscope I board with 16K of memory
/1—Test Periscope I board with 56K of memory (default)
/3—Test Periscope III board with 64K of memory
/C:nnnn—Run tests multiple (nnnn) times, where nnnn is a hex number from 0 to FFFFH
/M:nnnn—Use protected memory segment other than D000H
/P:nnn—Use write protect port other than 300H

The memory tests performed are: writing zeroes; writing FF; writing a rotating bit pattern; continuous copying of a repeating pattern; and a memory write-protect test. Note that the memory tests overwrite the contents of the protected memory. Be sure to use the /N installation option with PS.COM if Periscope is installed to prevent a conflict.

If an error occurs during the test, seven or eight columns of numbers are displayed. Each column reflects the

number of failures for the corresponding memory chip (U1 thru U7 for the /1 option, U1 thru U8 for the /0 option, and U29 thru U32 and U38 thru U41 for the /3 option). If you notice errors for a single IC, check the chip to be sure it is firmly seated in its socket and has no bent pins. If the memory passes the test, the display is overwritten by the next test to conserve space on the display. If the memory fails, the error messages are not overwritten.

PUBLIC.COM

This program is used to generate public statements for assembler programs. With Periscope, the more publics you have, the more symbols you have, and the more symbols you have, the easier it is to debug your program! To run this program, enter 'PUBLIC filename' from the DOS prompt. If no extension is specified, ASM is assumed. The program reads your source and writes a file of public statements to the file filename.PUB. You can then include or merge this file into your program.

The options available are:

/C - do not generate code references (PROC, LABEL, or near labels)
/D - do not generate data references (DB, DD, DW, DQ, and DT)
/E - do not generate equate references (EQU or =)
/I - generate references found inside conditional statements (IF, IFE ...)
/L - force the public definitions to lower case
/U - force the public definitions to upper case

For help, enter 'PUBLIC ?'. The program generates public statements for all data variables and procedures, subject to the rules below.

If the multi-line COMMENT statement is used, it must be the first word found in the source line. Nothing is generated for a name that starts with the numbers zero through nine. Any public statements generated for equates are absolute references and are not relocated in memory.

The source line is skipped if the first word found in the line is PUBLIC, EXTRN, END, ASSUME, ORG, INCLUDE, EVEN, NAME, TITLE, SUBTTL, PAGE, ELSE, WIDTH, %OUT, NOT, OR, AND, XOR, or MASK. A public statement is generated for the first word in a line if the first word ends in a colon or if the second word found in a line is DB, DD, DW, DQ, DT, PROC, LABEL, EQU, or =. When a STRUC definition is

encountered, no publics are generated until an ENDS is found.

PUBLIC recognizes macros and conditional statements—any items found inside these types of statements are ignored. Since these items can be nested, the program keeps track of the nesting level and generates public statements only when the nesting level is zero. The following items increment the nesting level—MACRO, IFDEF, IFIDN, IFDIF, REPT, IRPC, IFNB, IRP, IFE, IFB, IF1, IF2, IF, and IFNDEF. ENDIF and ENDM decrement the nesting level.

PUBLIC does not suppress EQU statements that refer to memory (e.g., XXX EQU [BP+2])—use the /E option or IF/ENDIF statements to suppress these.

RS.COM

This program is used to verify and size a record definition file. It reads a DEF file containing record, alias, and keyboard definitions and displays the number of definitions found and the total record table size required for the file. These definitions are loaded by RUN.COM to provide support for Periscope's DR command, keyboard assignment using the Ctrl-Fn keys, and commands that use the aliases.

To run this program, enter 'RS progname' from the DOS prompt. The file extension is presumed to be DEF. The DEF file is presumed to be in the same format as the sample file PS.DEF. Use the size shown by RS.COM for the PS.COM /R installation option. For help enter 'RS ?'.

A section of the PS.DEF file is shown below.

```
\f1=k;dr cs:Ø .psp;
\f2=dr cs:5c .fcb;
\FCB            ; File Control Block
Drive,b,1       ; Drive Ø=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
```

The first two lines of this file contain keyboard definitions for function keys F1 and F2. While in Periscope, these keyboard definitions may be called by pressing Ctrl and Fn at the same time. The keyboard

definition must contain a back-slash, the function key number, an equal sign, and the desired keystrokes. 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. No comments are allowed on keyboard definition lines. The maximum length of a keyboard definition's text is 64 characters for each of the ten function keys.

The third line of the file starts a record definition. The record name is limited to 16 characters and must be preceded by a back-slash. No embedded spaces are allowed in the record name.

Until another back-slash is found at the start of a line or the end of the file is reached, each of the following lines defines a field within the record. Each of the lines contains a field name, a field type, and a field length, separated by commas. The field name may be up to 10 characters long and may have embedded spaces. The field type may be any of the display formats, except E (effective address). The field length is the total number of bytes required by the field. This number is in hexadecimal notation.

If long real formatting is used, the length must be a multiple of eight. If double word or short real formatting is used, the length must be a multiple of four. If word, integer, or number formatting is used, the length must be a multiple of two. The length of any one field and the total length of the record may be from one to FFFFH. Each field line may be commented using a semi-colon preceding the comments. A field type of + skips over the indicated number of bytes without displaying anything.

There are five different aliases currently supported by Periscope. An alias is a two-character mnemonic that represents a name of up to 16 characters. An alias may be entered as a line in a DEF file (see FTOC.DEF for definitions of MP and MX) or may be entered using the EA command (see Chapter VII).

The supported aliases are:

MP—The module path name for source-level debugging
MX—The module extension for source-level debugging
X1—The command executed on entry to Periscope
X2—The command executed after each Periscope command
X3—The command executed on exit from Periscope

Note: An invalid command entered as alias X1, X2, or X3 can cause Periscope to issue an unexpected error message.

SYMLOAD.COM

This program is used to load Periscope's symbol tables from within your program. This approach can be used when your program manages overlays or is not loaded by RUN.COM. The LS command may also be used to load symbols on the fly.

SYMLOAD is a memory-resident routine that is run once per DOS session (it can be rerun if needed). It attaches itself to an interrupt vector so your program can access it as desired. The default interrupt used by SYMLOAD is 67H, but this can be changed if needed. 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 to be used to access SYMLOAD. Be sure that Periscope has already been installed, since it is required for SYMLOAD to work. The /I:nn command-line entry is needed only when SYMLOAD is 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 SYMLOAD has been installed, it may be accessed 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.

CL—If this register is a binary 1, the new symbols are added to the end of the existing symbol table, otherwise the symbol table is cleared and the new symbols are loaded.

DS:DX—Points to a PSS file name in ASCIIZ format. An extension of PSS is required. For example, to load C:SAMPLE.PSS, DS:DX would point to the string C:SAMPLE.PSS followed by a binary zero. See the description of the program TS.COM for information on creating a PSS file.

On return, register AH contains the status of the operation. Register AL is used to return additional error

information if a read error occurred. 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 the status returned is zero, the symbol table has been loaded successfully. Note that all addresses are relocated relative to the segment address passed in register BX, except for absolute references and for symbols whose segment was already in the range of F000H to FFFFH.

By setting register CL to 1, you can load multiple symbol tables. One PSS file is loaded into the symbol table per subroutine call, but the register setting can be used to append the new symbols to the end of the symbol table.

SYSLOAD.SYS

SYSLOAD.SYS is a utility program written by Bob Smith. Bob has licensed us to distribute this powerful program with Periscope. It allows you to load any of the .COM programs in the Periscope package as a device driver. By being able to load PS.COM and RUN.COM as device drivers, 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:

'/C=c:command.com' is an optional argument that specifies the location of the command processor in case the resident program needs it. This text is used as the argument to COMSPEC in the pseudo-environment created when calling the program to be loaded. Note that the full drive, path, filename, and extension of the command processor must be specified. This argument is not required by any of the programs in the Periscope package.

'/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 /t:8 /c:17 /a /f' in CONFIG.SYS.

TS.COM

This program is used to verify and size a MAP file and optionally generate a Periscope symbol file. It reads a MAP file as produced by the linker and displays informational messages and the total symbol table size required for the file.

To run this program, enter 'TS progname' from the DOS prompt. The file extension is presumed to be MAP. The MAP file is presumed to be in the same format as the sample file FTOC.MAP. Use the size shown by TS.COM for the /T installation option. For help, enter 'TS ?'.

The options available are:

/C - Read a DeSmet .MAP file as produced by CWare's C compiler
/D - Read a LINK86 .SYM file as produced by Digital Research's LINK86
/Fx - Filter a single leading character x from public symbols
/M - Read an Aztec .SYM file as produced by Manx's C compiler
/P - Read a PLINK .MAP file as produced by Phoenix's PLINK
/Q - Read a PLINK .EXE file as produced by Phoenix's PLINK
/S - Write a Periscope .PSS symbol file

Microsoft and IBM have made some subtle changes in the format of the MAP file from time to time. If you encounter problems reading a MAP file, please contact us as soon as possible.

To generate address references in the MAP file, you'll need to specify both a MAP file and the /M option at link time. Entries are generated in the MAP file for names

defined as PUBLIC by your compiler or assembler. For ASM programs, a PUBLIC statement is used to generate an address reference in the MAP file. For C programs, variables defined outside the MAIN and external references to other modules will generate address references in the MAP file. For Pascal programs, variables defined as PUBLIC and external references to other modules will generate address references in the MAP file.

Only the first 16 characters of a public name are used by Periscope. Any characters beyond the 16-character limit are discarded. The programs TS.COM and RUN.COM read the first group of address entries in the MAP file—the one sorted by name. Any absolute references found in the MAP file are included, but are not relocated to the program's location.

To generate line references in the MAP file, you'll need to specify a MAP file and the /LI option at link time. Not all compilers support the line number option. The ones that are known to support this option are: Computer Innovations C, IBM C, IBM Pascal, Lattice C, Mark Williams C, Microsoft C, Microsoft Pascal, and Microsoft FORTRAN. (Using TMAP—part of tDebugPlus by TurboPower Software, you can do source-level debugging of compiled Turbo Pascal programs.) Recent versions of the linker put the module name in the line number section of the MAP file. If this information is available, Periscope extracts up to 12 characters of a module name and extension for use with the MP and MX aliases. See Chapter V for more information.

The line references generated by TS.COM and RUN.COM have a single-character alphabetic prefix, followed by the actual line number. The alphabetic prefix starts at A and is incremented for each module found in the MAP file. For example, line 10 of the first module is referenced as A10, and line 20 in the second module is referenced as B20. The first 26 modules are referred to as A through Z. Subsequent modules are referred to as AA through AZ, BA through BZ, etc.

If you have some symbols that you don't want to see, edit the MAP file and insert braces as desired to turn off symbol generation. A left brace ({) turns symbol generation off, and a right brace (}) turns it back on. Be careful when saving the MAP file—don't let any TABs or high bits into the file.

Large MAP files are relatively slow to load, since RUN.COM must extract the symbols each time the corresponding program is executed. To reduce the load time for large symbol tables, enter 'TS progname/S' to

analyze the MAP file and create a file of the form progname.PSS that is a memory image of the symbol table for the program.

When RUN is executed, it first looks for the PSS file. If the PSS file is found, it is used for the symbol table. If no PSS file is found, the MAP file is used. Be careful—if you have created a PSS file and then re-link the program without creating a new PSS file, the old PSS file will be used. It's a good idea to create the new PSS file from the batch file used to compile and link your program.

If the PSS file is too big to fit in the space allocated by Periscope, no symbols will be loaded. This is in contrast to the MAP file, where the symbol table may be partially loaded before an error occurs.

Periscope supports Digital Research's LINK86 (version 1.3) and Phoenix's PLINK (version 1.4x). Since RUN knows nothing about the formats used by these two linkers, TS.COM must be used to generate a PSS file for both of them.

For LINK86, enter 'TS progname/D/S'. The /D option tells TS that the input file was generated by DRI's linker. The presumed input file extension for this option is SYM, not MAP! The /S option tells TS to write the PSS file to disk, for later use by RUN. At link time, be sure to specify a SYM file. Line numbers are not available as symbols for LINK86.

For PLINK, enter 'TS progname/P/S'. The /P option tells TS that the input file was generated by Phoenix's linker. The presumed input file extension for this option is MAP. The /S option tells TS to write the PSS file to disk, for later use by RUN. At link time, be sure to specify a MAP file and the G report (all symbol information is read from the G report). This method can be used starting with PLINK 1.30, but does not support line numbers.

To get all possible symbols with PLINK, enter 'TS progname/Q/S'. The /Q option tells TS to use the EXE file for all symbols. The /S option tells TS to write the PSS file to disk, for later use by RUN. At link time, be sure to specify 'SYMTABLE' as a linker directive. This method can be used starting with PLINK 1.40.

TS.COM also supports the CWare (DeSmet) and the Manx (Aztec) C compilers. Use the /C option to read a CWare MAP file. The CS segment is set to zero and the DS segment is set to F000H. You'll need to use the Periscope /S command to relocate DS symbols to the

appropriate location. To read a Manx SYM file, just use the /M option.

The 'filter' option is used to filter out a single leading character from symbol names. For example, Microsoft C uses leading underscores on public symbols. If you use '/F_', TS.COM will remove the first leading underscore found on each symbol.

USEREXIT.ASM and USEREXIT.COM

This sample program illustrates Periscope's ability to perform user-written code. User-written code can be used to perform breakpoint tests (see the BU command) and user exits (see the /U command).

The user-written code is installed as a memory-resident program using an available interrupt from 60H to FFH. The program must be installed before PS.COM is run. Also, the PS.COM installation option /I:nn must be used, where nn is the interrupt vector used to access the user-written code. A signature of 'PS' must be present 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. This routine is accessed via a far call. Register AH is used to indicate the function desired. When using this routine, all registers are preserved. The functions available are:

- **AH=1**—Display nul-terminated string starting at DS:SI. This function uses the standard Periscope display handler to display a string on the screen. The maximum length of the string is 83 characters, including a carriage return, line feed, and nul. The string must end with a nul (binary zero)!

Your suggestions for additional routines are welcomed.

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 semi-colon 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 described above—the cursor is no longer moved to the bottom of the screen on return from a user exit.

Do not attempt to perform DOS functions from user-written code—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.COM, run the program from DOS. Then install PS.COM 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 an 8087 or an 80287, try /U 87 to display the numeric processor status.

X—Technical Notes

- **Debugging Theory**
- **NMI Use**
- **CPU Differences**
- **DOS Notes**
- **Debugging Techniques**
- **Debugging Device Drivers And Non-DOS Programs**
- **Debugging Hardware Interrupts**
- **Periscope Internals**
- **The Periscope I Board**
- **The Periscope III Board**
- **The IBM Enhanced Graphics Adapter**

Debugging Theory

The 8086 processor family provides two built-in functions that aid the debugging process. These are the breakpoint and single-step capabilities.

The breakpoint capability uses a special single-byte code to indicate that a breakpoint is to be taken. This opcode causes the system to perform an Interrupt 3 when the first byte of an instruction equals CCH. This is the facility used by the Go command in Periscope, for both the temporary and sticky code breakpoints.

When Periscope sets a code breakpoint, the original byte is saved in an internal table and a CCH is inserted in its place. For this reason, it is not possible to set a code breakpoint in ROM or other unmodifiable memory.

When the breakpoint is taken, Periscope is entered through a special entry point. The use of this entry point signals Periscope to reverse out any code breakpoints that are currently set and then decrement the instruction pointer (IP) by one to show the correct instruction.

If an unexpected instruction contains a CCH in the first byte, Periscope is unable to reset the instruction to its prior value and will disassemble the instruction as INT 3. You will need to manually alter the byte or modify the IP register to continue execution of the program being debugged.

Single-step is the other type of 8086 breakpoint. It is set by modifying the trap flag to indicate that every instruction should be trapped. If this flag is set, the system performs an Interrupt 1 before the execution of each instruction, allowing you to single-step through a program. The trap flag is used by Periscope for the GA, GT, J, and T commands.

If an instruction outside Periscope clears the trap flag, it causes any tracing currently underway to be turned off. If an instruction external to Periscope sets the trap flag unexpectedly, Periscope ignores it.

Since Periscope cannot be used to trace itself, it is not possible for it to trace the execution of Interrupts 1, 2, or 3 or any other interrupts that point to Periscope.

Some programs may use Interrupts 1, 2, or 3 for their own purposes. Periscope normally refreshes these vectors each time it is activated, but you can override this refresh procedure. Call Tech Support at 404/256-3372 for more information.

NMI Use

All models of Periscope except Periscope II-X make use of Interrupt 2, the non-maskable interrupt (NMI). This interrupt is used by the break-out switch to gain control of the system and enter Periscope. Some systems do not support the use of NMI, since it is either not available on the system bus or is used by some other system function.

Three computers that don't support NMI on the bus are the Tandy 1000, Cordata PC, and the IBM Convertible. The most common use of NMI is to emulate a 6845 CRT controller on some of the non-standard display adapters. For example, EGAs made by Paradise Systems, Quadram, and Video 7 allow emulation of various display modes. This is achieved using software that is activated via an NMI. Other boards by Paradise Systems and Sigma Designs also use NMI. Luckily, the use of NMI by most of these boards can be turned off so that there is no conflict with Periscope's break-out switch.

Generally, you can load Periscope after the emulation code is loaded, or disable the emulation. If the device reasserts NMI periodically, you may have problems using the break-out switch. If you're programming the CMOS RAM on an AT, configure Periscope as Model II-X so Periscope won't use port 70H.

If you're using a Compaq Deskpro 386, note that the original version of the CEMM device driver intercepts INT 2 (NMI) during its execution. If you press the break-out switch during this period, the driver displays a message that says "Press any key to reboot". Compaq is working on a solution to this problem.

CPU Differences

If an 8087 is used with interrupts enabled, an error will cause an NMI. Since Periscope uses the NMI, the debugger screen is displayed. Since the 8087 may interrupt the 8088 at any point, CS:IP may contain any value. The 80287 does not use the NMI, so an error will not invoke Periscope.

Many PCs and XTs have an early version of the 8088 CPU that has a serious bug. This version can be identified by the copyright date of 1978 shown on the chip. The defect in these CPUs is that the instruction after an instruction that changes the stack segment is not protected from being interrupted. The defined method for changing the stack is to change the stack segment and then immediately change the stack pointer. If this process is interrupted, the stack may be in no man's land—the beginning of a system hang. The fix is to get the later chip—identified by copyright dates of 1978 **and** 1981. At install time, Periscope checks for a defective 8088—if one is found, Error 65 is displayed.

On a correctly functioning 8088, any instruction that modifies any of the segment registers protects the next instruction from being interrupted. This is a bit of

overkill, since changes to DS, ES, and CS do not need the protection that stack changes require. This is why you'll notice that Periscope skips instructions while tracing through an instruction that modifies a segment register—the instruction was actually executed, but it was invisible to Periscope. Be careful not to use Periscope's Go command to stop in the middle of a stack changeover, since this can cause the same problem as the defective 8088.

The NEC V20 and the 80C88 go even further than the 8088—they also protect the instruction after a read of the segment registers, except for POP instructions. For example, the V20 protects the instruction after MOV AX,ES, while the 8088 does not.

On the 80286, the instruction protection for changes to segment registers applies only to the stack segment—instructions that change DS, ES, or CS do not protect the next instruction. Still, be careful not to use the Go command to stop in the middle of a stack change.

For the 80286, Periscope may be used in real (8086) mode only. The exception interrupts 6 (invalid opcode) and 0DH (segment overrun) are intercepted by Periscope with CS:IP pointing to the offending instruction. If the code segment of these interrupts points to memory below PS.COM when it is installed, no change is made to the interrupt since it is already in use. Avoid use of hardware interrupt IRQ 5 (INT 0DH) on an AT for such things as a mouse, since a segment overrun that occurs when this interrupt does not point to Periscope will hang your system. If you get an exception interrupt, chances are good that the system is severely corrupted—a reboot is generally recommended.

DOS Notes

If you're using DOS 2.00 or 2.10, you should be aware of bugs in DOS that can cause problems. The bugs involve improper changes to DOS's stack where the SP register is modified before the SS register. This can cause problems when the break-out switch is pressed at just the right time or when you attempt to trace through DOS. PC Tech Journal published the patch for DOS 2.10 in the November 1984 issue. The same principles apply to DOS 2.00, although the addresses are different. If possible, use a later version of DOS—the bugs are fixed in DOS 3.00.

Under some versions of DOS (e.g., 3.10), DOS decides whether a program is an EXE based on the first two bytes in the file, not the file extension. This can cause

problems with RUN.COM, since it assumes that a file with a .COM extension is really a COM file. If the extension is .EXE, RUN confirms that the first two bytes indicate an EXE file.

If you're using DOS 3.2, note that DOS modifies INT 2 (NMI) after CONFIG.SYS time. To prevent the modification of INT 2, patch IBMBIO.COM at offset B0DH.

So that Periscope can perform file I/O safely, it checks the undocumented, but reliable, in-DOS flag. This byte contains zero if DOS is not busy. Periscope also checks to see that interrupts are enabled, to be sure that DOS was interrupted at a safe point. The location of the in-DOS flag can be found by performing INT 21H with AH=34H. ES:BX returns the address of the flag. If you want to perform file I/O and Periscope is telling you that DOS is busy, get CS:IP back to your code and try again. Do not attempt to modify the in-DOS flag or the interrupt flag in order to fool Periscope—you can get a garbled disk directory very easily. To make DOS not busy, you can use the user breakpoint in the sample program USEREXIT.ASM (See Chapter IX for more information).

Starting with DOS 3.20, Interrupts 1, 2, and 3 are changed by DOS on a short boot. This can complicate matters if you're trying to use the break-out switch across a short boot. You'll need to patch DOS or better yet, reset the vectors to point to Periscope.

Debugging Techniques

While Periscope is active, the BIOS interrupt vectors it uses are reset to point to BIOS unless the /V installation option was used. To access some memory-resident programs while Periscope is active, you may have to use some of these options. For example, a program that displays the time may use interrupt 1CH. Unless you specify /V:1C when PS.COM is run, the clock program won't be active when Periscope is. Be aware that each /V option used reduces Periscope's dependability, since the interrupt vector is left pointing to RAM that can be corrupted. If you're having problems running some software with Periscope, check the interrupt vectors using INT.COM (see Chapter IX) and see the known conflicts in the description of the /V installation option (Chapter VI).

Avoid debugging with ill-behaved resident programs in the system. While you can debug with these types of programs in the system, they can often muddy the waters—making it much harder for you to see just what your program is doing.

When you press the break-out switch to stop the execution of a program, chances are very good that you'll stop the machine in either BIOS or DOS. If you want to get back to your program, use the Go command to execute to a known point in your program. If that's not possible, try using the Register breakpoint. Enter 'BA *' to clear any breakpoints currently set. If you know your program's Code Segment, enter 'BR CS EQ nnnn' to set a Register breakpoint when CS equals the desired value. If you aren't sure, use 'BR CS NE CS' to set a breakpoint when the Code Segment changes from its current value. Then enter 'GT' to continue execution with the Register breakpoint set. This will usually get you back to the program, or at least from BIOS to DOS or vice-versa. If you're debugging a program that has line numbers as symbols, use the BL breakpoint to get back to your code.

To repetitively trace an instruction, enter the Go command once and then repeat it using F4. For example, if you want to watch the execution of the instruction at offset 110H, enter 'G 110' and press return. Then press F4 to repeat the Go instruction as many times as desired.

To debug a memory-resident program, use RUN to load the program and its symbol table. The program will be loaded in the same location as if it were run directly from the DOS prompt. Enter 'G' to install the program and return to the DOS prompt. Until RUN is used to load another program, the symbol table will remain available—ready for you at a push of the break-out button!

If you're programming in assembler, use the PUBLIC program described in Chapter IX to get symbolic access to as much of your program as possible. The more symbols you use, the easier it is to debug your programs.

Microsoft compilers can place data (pseudo-code) after interrupts 34H through 3DH to emulate the 8087 numeric processor. When disassembled by Periscope, the data following the interrupt causes the disassembly to be garbled. To prevent execution of data, the Jump command traces interrupts in the range from 34H to 3DH.

If you're programming in C using a compiler by Computer Innovations, CWare, IBM, Lattice, Manx, Mark Williams, or Microsoft, you can get debugging information such as line numbers and address references in your MAP file by using compile-time options provided with these compilers. By defining variables outside the MAIN, you can cause address references to be generated for program variables. At link time, be sure to specify a MAP file and the /LI and/or /M options for full symbol support. No source-

level support is available for the CWare or Manx compilers.

If you're programming in Microsoft BASIC, FORTRAN, or Pascal, the addresses in the MAP file that reference data variables will be incorrect. These compilers generate false segments for DGROUP data. The actual segment used depends on the amount of memory available at run time. To correct the false segments, do the following:

- Use RUN to load the program, then execute the program until DS is modified. The best method is to go to the first line in the source program, using the symbol for the line number. The value of DS at this point is the correct segment.

- Display a known data symbol using the Display command. The segment associated with the symbol is the invalid segment.

- Enter '/S xxxx yyyy' where xxxx is the invalid (old) segment and yyyy is the correct (new) segment. This will change all occurrences of segment xxxx in the symbol table to yyyy.

Note: This problem is corrected in version 3.3 of Microsoft Pascal and FORTRAN.

If you're calling assembly-language subroutines from a high-level language, Periscope can be used to trace through the execution of the subroutine to verify that it is operating correctly. If the subroutine is linked to a compiled program, simply use G SUBNAME, where SUBNAME is the name of the subroutine. If the subroutine is being called from an interpretive language such as BASIC, modify the subroutine so that the first byte contains CCH. Then when the subroutine is executed, the breakpoint (CCH) will activate Periscope. At that point, you can modify the instruction to be a NOP (no operation) by using E CS:IP 90 or skip to the next instruction by using R IP IP+1.

Avoid debugging packed EXE files—you'll have to trace through the header before your program is unpacked and available. If you must debug a packed EXE, use RUN to load the program, use G to start execution, press the break-out switch, and then start debugging.

If the debugger screen is garbled, use K or KI to clear the screen or initialize the screen, respectively.

Debugging Device Drivers And Non-DOS Programs

If you're debugging device drivers, see the description of SYSLOAD.SYS in Chapter IX. This utility lets you load Periscope or any other COM file at CONFIG.SYS time.

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, and leaves NMI (INT 2) intact, except when DOS 3.20 is used (see DOS Notes above). 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.

Periscope I and III use RAM external to the Periscope board when the protected memory overflows. When performing a short boot, be aware that any external tables are no longer an extension of DOS and may be used by another program or garbled during the boot process. For best results, use either the /Z installation option to suppress external tables or use the /L installation option to place the external tables in the middle of memory.

➧ For Periscope II and II-X, all code and data areas are in normal RAM and the /Z installation option is not available—use the /L installation option instead.

Debugging Hardware Interrupts

To debug hardware interrupts, 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 in the service code 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 III to be able to see just what happens when your code runs.

If Periscope must issue a non-specific End of Interrupt (EOI) to clear interrupts, a message of the form 'EOI

issued for IRQ x' is displayed, 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 control is returned to your program, so you may want to use a semaphore mechanism to keep your code from being re-entered while being debugged.

On AT-class machines, EOI is used for IRQ 0 and 1 only, but on a PC-class machine, Periscope may issue an EOI for any IRQ level.

Periscope Internals

Periscope uses Interrupts 1 (single-step), 2 (NMI), and 3 (breakpoint). If the system is an AT-class machine, Periscope may also intercept Interrupts 6 (invalid opcode) and DH (segment wrap-around). Interrupts 5, 9, and 15H can be used by the PSKEY.COM program (see Chapter IX).

The data fields used by Periscope are located at the beginning of the protected memory. The record definition PSDATA in the file PS.DEF contains the most useful of these data fields. The source file contains comments describing the various fields in the record definition. To display Periscope's data area assuming the default memory address of D000:0000, enter 'DR D000:0 PSDATA'.

➧ For Periscope II and II-X, enter 'DR xxxx:100 PSDATA', where xxxx is the starting segment for Periscope's tables. (The segment used by INT 3.)

The Periscope I Board

The current board (Rev 2) uses 56K of memory and two consecutive I/O ports. The memory is configured as seven chips of 8K-by-8 static RAM, with a cycle time of 150 NS or less. The starting address of the memory is switch-selectable to any 64K boundary. The starting I/O port is switch-selectable to any 4-byte boundary. The board draws approximately 2 watts of power. See Chapter III for information on the switch settings. The original 16K board (Rev 1), where the memory is configured as 8 chips of 2K-by-8 static RAM, cannot be used with software after version 2.1x.

The memory is write-enabled when the value DBH is output to the first of the two ports. Any other value output to this port write-protects the memory. When the break-out switch is pressed, an NMI is generated. Periscope detects that the switch was pressed by

checking the high bit read from the second I/O port. If this bit is on, the switch was pressed, otherwise the switch was not pressed. To clear the switch and NMI, output any value to the second port. Clear the switch and NMI near the end of your program using the code shown below.

```
mov dx,301h     ; assumes Periscope port is 300h
out dx,al       ; clear the break-out switch - any value in al is ok
in al,61h       ; enable nmi
jmp short $+2   ; delay for 286
mov ah,30h      ; use 30h if pc or xt, 0ch if at
or al,ah
out 61h,al
jmp short $+2   ; delay for 286
not ah
and al,ah
out 61h,al
```

The Periscope III Board

The current board (Rev 1) uses 64K of memory and four consecutive I/O ports. The memory is configured as eight chips of dynamic RAM, with a cycle time of 150 NS or less. The starting address of the memory is switch-selectable to any 64K boundary. The starting I/O port is switch-selectable to any 4-byte boundary. The board draws approximately 7 watts of power. See Chapter III for information on the switch settings.

Since the Periscope III board does not latch NMI, the first two lines of code shown above may be omitted when clearing NMI.

The IBM Enhanced Graphics Adapter

The /V:10 installation option is no longer needed if an EGA is present. Since the EGA uses memory from A000:0 through C000:3FFF, don't try to use this range of memory as protected memory when an EGA is present.

Periscope supports the EGA's new video modes, including those using segments other than B000H and B800H. For single-monitor systems, a maximum of 32K of the screen buffer is saved and restored by Periscope. If you're using the new graphics modes of the EGA, you'll need a dual-monitor system to save the full graphics image.

If you plan to use the EGA's extended palette capabilities on a single-monitor system, be sure to use the extended save area (via SAVE_PTR as described in the EGA BIOS listing). This RAM data area is updated when the palette or overscan is set, allowing Periscope to restore the original palette when returning to your program. There are

two problems with using this save area, however. First, the save/restore of the palette registers is incredibly slow. Second, the overscan or border color is not saved in a consistent location. If you use the 'set overscan' function (INT 10H, AH=10H, AL=1), the color is saved in a different location than if you use the 'set all palettes and overscan' function (INT 10H, AH=10H, AL=2). For use with Periscope, the latter call should be used. The best bet for using an EGA is to have a two-monitor system. If you use a single-monitor system, remember that the maximum screen size that can be saved and restored by Periscope is 32K.

Periscope has limited support of the EGA's 43-line mode. See the description of the /43 installation option in Chapter VI.

XI—Using Periscope III

- **Introduction to Periscope III**
- **Capabilities**
- **Compatibility And Other Caveats**
- **Hardware Breakpoint Examples**

Introduction to Periscope III

Periscope III is a powerful hardware-assisted debugger for use in IBM PC, XT, and AT-class machines. Its commands are a superset of the normal Periscope commands, so it provides both the capabilities of existing models of Periscope plus its own special hardware capabilities.

The Periscope III board monitors the system bus, watching for user-specified breakpoints to occur while the test program is running at full speed. These breakpoints can be set on memory reads and writes, code prefetch, I/O port reads and writes, DMA activity, and may be further qualified by data values and/or a pass counter. When the board detects that a breakpoint has been reached, it generates an NMI which activates the resident Periscope software.

The board requires a full-length slot. If it is installed in an AT-class machine, the slot must be a 16-bit slot. An umbilical socket is provided that must be installed in the 8087 or 80287 socket. (This umbilical socket does not preempt the use of a numeric processor.) Periscope III works on the IBM PC, XT, and AT, and the Compaq 8088 and 80286 machines at processor speeds up to 8MHz. (**Note:** a 10MHz version is also available.) Machines that the board does not support are the IBM XT/286, the IBM System/2 series, the Compaq 8086 Deskpro, the Compaq

80386 Deskpro, the AT&T 6300 and 6300 Plus, systems using an add-in 'turbo' card, and zero wait-state machines.

Capabilities

The Periscope III board has three major modules:

- Protected program memory
- Hardware breakpoints
- Real-time trace buffer

The protected program memory is 64KB of write-protected RAM. This memory is like the protected memory in Periscope I; it is used for Periscope's code and data. As much of Periscope's tables as will fit are located in this memory. Any tables that do not fit in the protected memory are located as low as possible in DOS memory. The protected memory is normally addressed at segment D000H, but can be moved to another address if needed. If there is a conflict with other memory in the system, Periscope's program memory can be disabled via jumper J3. (When the shunt block connects the top two pins of J3, the program memory is disabled; when it connects the lower two pins of J3, the memory is enabled.)

The hardware breakpoints available include:

- Memory access (read, write, DMA, code prefetch)
- I/O Port access (in or out)
- Data values and/or data bit mask (on the high and/or low bus)
- Pass counter

Memory breakpoints can be set on up to 16 ranges from 0000:0000 to F000:FFFFH. For an AT-class machine, breakpoints can also be set on memory ranges beyond the first megabyte. The qualifiers available for memory breakpoints include memory read, memory write, DMA, and code prefetch. The memory breakpoints are bit-mapped (using 128KB of local memory on the Periscope III board)—allowing every byte in the first megabyte to be individually addressed.

Port breakpoints can be set on up to 16 ranges from zero to FFFFH, although the ports fully supported by the PC are from zero to 3FFH. The qualifiers available for port breakpoints include I/O read and I/O write. The port breakpoints are bit-mapped, allowing every port in the 64K range to be individually addressed.

Data breakpoints can be set on the low (8-bit) bus and on the high (16-bit) bus if an AT-class machine is used. Data breakpoints are used to qualify a memory or port breakpoint—no breakpoints are possible on data values only. Data breakpoints can be set on up to 16 ranges from zero to FFH. A bit mask may be also be used to indicate desired data values. Bit values of zero, one, or 'don't care' are all supported.

The pass counter is used to interrupt the executing program after the indicated number of breakpoints has occurred. The default pass count is 1, which causes an interrupt on the first breakpoint. The value of the pass counter may be from one to FFFFH. If the memory, port and/or data values found on the current bus cycle would cause a breakpoint, the internal pass counter is decremented. When the pass counter reaches zero, an interrupt is generated.

The real-time trace buffer is a circular buffer that captures up to 8,192 (8K) bus events. Each bus event 'record' is 48 bits wide, which includes the 20- to 24-bit bus address, 8- or 16-bit data for the bus cycle, bus status bits (I/O read, I/O write, memory read, memory write, code prefetch, and DMA), and an external probe bit. (When the two pins of jumper J2 are connected, the 'Probe' indicator is shown in the trace buffer.) The buffer can be displayed in three formats—a raw dump that shows the address, data, and status information for each cycle, a disassembly mode that uses the prefetch cycles found in the buffer to show only instructions, and a combination of the above two, which shows a mix of instructions and the data accesses performed by the instruction stream.

Once the Periscope software is loaded, the buffer is updated after each bus cycle, even if you're not debugging. This means that you can press the break-out switch any time to see what's been happening. Also, if you should ever get an exception interrupt, just look in the trace buffer to see where things went awry.

A breakpoint option is available to stop program execution when the trace buffer is filled, allowing you to examine the execution flow of a program starting at a known point and stopping after 8,192 bus events have occurred. Also, a selective trace capability is available that lets you capture just trigger events in the trace buffer — useful when you need to capture multiple widely-spaced events in real-time.

The trace buffer can be set to show the events leading up to a breakpoint in three different methods: up to 8K

events before the breakpoint; up to 4K events before the breakpoint and 4K events after the breakpoint; or 8K events after the breakpoint. For real-time applications, these capabilities are extremely valuable, since you can see how your program got where it did plus where it went afterwards!

Using the above breakpoints, it is possible to trap complex events. For example, if you need to watch for the fifth time the character '%' is displayed anywhere on the monochrome display, set a breakpoint for: memory writes to the screen (B000:0000 to B000:0FFF); data equaling the percent sign; and the pass counter equal to five.

Since some conditions are not visible from the system bus, we've implemented a hardware/software combination breakpoint in Periscope III. Using this capability, you can execute at full speed until a hardware breakpoint is reached, then drop into software to evaluate a condition. If the software condition gives a 'hit', Periscope's screen is displayed, otherwise full speed execution continues. For example, if your program is zapping low memory, set a memory breakpoint to trap any writes to the desired range. This breakpoint alone will probably get some false hits, since DOS and other programs could be legitimately writing into the specified range. If you set a software test that the code segment must be equal to your program's code segment, then all writes other than those done by your code are filtered out. This technique runs your program at something less than 100% full speed, but since the software is activated only when there's a hardware hit, the effective speed is rarely less than 99%. See the description of the GM command in Chapter VII for more information.

Compatibility And Other Caveats

When designing a product as complex as Periscope III, some tradeoffs are required. To make the board work on both PC- and AT-class machines, the board has to get the majority of its signals from the system bus. The only signal used by the board that is not available on the bus is the line that indicates whether a memory read is an instruction fetch. This signal is the one picked up by the umbilical socket from the numeric processor (using pin 26 on the 8087 and pin three on the 80287). Accessing signals from the bus works fine as long as the bus is used per IBM's design. Some computers, notably the 8086 and 80386 Deskpros and the AT&T 6300 and 6300 Plus, have a non-standard bus or don't put all signals on the bus, and therefore do not work with Periscope III.

Periscope III is necessarily more sensitive to the bus structure than other add-in cards since its job in life is to monitor bus transactions—if some of these transactions are not present on the bus or are otherwise different from the IBM standard, Periscope III cannot work properly.

One area to watch is the setting of data breakpoints. If you use a PC-class machine, just remember that all data breakpoints use low, or 8-bit memory. If you use an AT-class machine, things are much more complicated.

An AT-class machine has both 8-bit and 16-bit memory. Eight-bit memory includes such things as display adapters, the protected memory in Periscope, and other cards that work in PC-class machines as well as AT-class machines. For these devices, all memory access is via the low or 8-bit bus, a byte at a time.

The other memory in an AT-class machine, including all DOS memory and ROM BIOS, is 16 bits wide. This memory can be accessed either a byte or a word at a time. How it is accessed affects how the data appears on the bus. The most efficient method is to access a word at an even address. This will read two bytes in one bus cycle, with the low (even) byte appearing in the low bus and the high (odd) byte appearing in the high bus.

If a word is accessed starting at an odd address, the read or write is split into two bus cycles, with the first cycle accessing the low (odd) byte on the high bus and the second cycle accessing the high (even) byte on the low bus. If a byte is accessed starting at an odd address, the high bus is used. If an even-addressed byte is accessed, the low bus is used.

Just remember this: 8-bit accesses are always on the low bus. For 16-bit accesses, even addresses are on the low bus and odd addresses are on the high bus.

The table below shows the possible permutations:

Environment	Low bus	High bus
PC or XT	X	
AT 8-bit memory	X	
AT 16-bit memory		
even byte	X	
even word	LO	HI
odd byte		X
odd word	HI(2)	LO(1)

On an AT-class machine, a breakpoint set at an odd address in 16-bit memory causes Periscope to issue the message 'Warning - Odd low address may cause missed breakpoint', since a word access of the preceding (even)

word would not indicate the affected address! You can safely ignore this warning when you're sure that access will start at the odd address—otherwise you should adjust the breakpoint address to the next lower even address.

To determine which memory in an AT-class machine is 8-bit and which is 16-bit, Periscope tests memory at 4KB intervals at install time and builds a control table containing this information. Unfortunately, it is not possible to test the I/O ports—Periscope must assume that all I/O ports are 8-bit devices.

When a byte access is made to an odd address in 16-bit memory, the low data bus will usually contain data from the previous bus cycle.

There is a phenomenon called 'breakpoint overrun' that occurs with all hardware breakpoint devices. Between the time a breakpoint is detected and the time the processor is stopped with an NMI, one or more instructions are executed. When looking in the traceback buffer with the 'HR' or 'HS' commands, the last bus cycle shown is the one that caused the interrupt to occur. The 'HT' and 'HU' commands show the instructions being executed or about to be executed when the breakpoint occurred. In any event, expect that several instructions will have been executed since the breakpoint occurred. For this reason, it is pointless to set a hardware breakpoint watching for an instruction that overwrites the NMI vector — use Periscope's BM command for this job.

Note: Don't reboot the system using Ctrl-Alt-Del when the board is armed (i.e., a GH or a GM command is in use)—if a breakpoint occurs during the boot, your system may hang. Go into PS and use any exit other than GH or GM to disarm the board.

Hardware Breakpoint Examples

• To break in BIOS, use 'HM * F000:xxxx L1 X;GH', where xxxx is the desired IP in ROM. This will trap on a prefetch of the specified location—usually stopping when the instruction is about to be executed.

• To trap set cursor calls, use 'HM 0:10*4 L3 R;BR AH EQ 1;GM'. This traps access to the video interrupt (INT 10H) and then tests register AH. If the register is equal to one, Periscope's screen is displayed, otherwise full-speed execution resumes.

• To trap writes to interrupt vectors by your program, use 'HM 0:0 0:3FF W;BR CS EQ CS;GM'. If you use IBM's VDISK on an AT-class machine, avoid breakpoints on memory from 0:380 to 0:3FF since this range is used as the stack while the machine returns from protected mode.

• If you suspect your program is underflowing its stack, use 'HM SS:0 L3 W;GH' to trap writes to the bottom of the stack, assuming that the lower bound of the stack is at SS:0.

• If you use Microsoft Windows ramdisk on an AT-class machine, set a breakpoint on reads of memory beyond the first megabyte using 'HC * M1 TC' and 'HM 0:0 1000:0 R'. Then enter 'GH' and do a DIR of the ramdisk. Periscope will show a very interesting code sequence in the center of the trace buffer.

• To debug across a short boot, arm the Periscope III board with GH and then restore all BIOS vectors (8, 9, 10, 15, 16, 17, 1C) to ROM and issue an 'INT 19'. Assuming NMI is left pointing to Periscope and Periscope is not corrupted by the boot, the breakpoint will re-activate Periscope.

• To monitor interrupt usage, enter 'HM * 0:0 3FF RW' and then 'GH;HS'.

Appendix A—Error Messages

- **Error Messages**

The error messages generated by programs in this package 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 Periscope (PS.COM)
40 through 69—transient or installation portion of Periscope (PS.COM)
70 through 89—RUN.COM
90 through 99—RS.COM
100 through 109—TS.COM
110 through 119—INSTALL.COM
120 through 129—PUBLIC.COM
130 through 139—SYMLOAD.COM
140 through 149—INT.COM
150 through 159—PSKEY.COM
160 through 189—PSTEST.COM
190 through 199—SYSLOAD.SYS
200 through 229—PS3TEST.COM

A list of the possible error messages and an explanation of each follows:

01 Invalid command
An unknown debugger 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. The address may be entered as a symbol (optionally preceded by a period) or a one- to four-digit segment, a colon, and a one- to four-digit offset. A register name may be substituted for the segment or offset.

The segment and colon may be omitted from most commands. The offset must be present for all commands requiring an address.

03 Missing segment
Some commands that modify memory (Enter, Fill, and Move) 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 entered as 'L nnnn', the number nnnn must be greater than zero. If entered as an offset, the number must be greater than or equal to the first offset. If entered as a symbol, the symbol's segment must equal the first segment entered and the symbol's offset must be greater than or equal to the first offset.

Note that the length argument is optional for the Display and Unassemble commands. The default length for these commands is 80H and 20H bytes respectively.

05 Unexpected input
After completion of a command, an unexpected entry was found. If multiple commands are desired, place a semi-colon 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 Missing operator
If the Hex arithmetic command was used, this error indicates the absence of an arithmetic operator between the two numbers. The valid operators are: +, -, *, and /.

If the EA command was used, the alias name must be exactly two characters, followed by the one- to 16-character alias, or followed by nothing to clear the alias.

If the Byte breakpoint (BB), Register breakpoint (BR), or Word breakpoint (BW) command was used, this error indicates the absence of a test. The valid tests are: LT, LE, EQ, NE, GE, and GT, in upper or lower case.

If the Line breakpoint (BL) or eXit breakpoint (BX) command was used, this error indicates the absence of a plus or minus sign to enable or disable the breakpoint.

If the BM, BP, HD, HM, or HP commands were used, this error indicates the absence of an operator after the range. For example, BP 300 303 is missing the In/Out operator.

09 Number is not decimal
The number entered when the translate decimal (XD) command is used must be in decimal format, 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 and a register name cannot be used.

11 Invalid/missing register
The register name must be AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, SS, CS, or IP. The 8-bit registers AH, AL, BH, BL, CH, CL, DH, and DL may also be used. 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 Chapter VII for the limits for the command used.

14 Invalid sub-function
For commands using sub-functions, the sub-function must be entered immediately after the function. Enter '?' to display the commands available.

15 Cannot trace INT 3
An attempt was made to trace interrupt 3 using Periscope. This interrupt is off-limits, since it points to Periscope and any attempts to have Periscope trace itself would result in total confusion.

16 Cannot 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 used with the BM, BP, HM, or HP commands was 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 16 characters. Lower case input is converted to upper case before the symbol or record definition table (if DR is used) is searched.

Line number symbols are of the form Xnn, where X is the module prefix (A for the first module in the link map, B for the second, etc.) and nn is the decimal line number. The line number does not have leading zeroes. For example, .A1 may be a valid line number, but .A01 is not. **Note:** Use a leading period if a symbol name could be interpreted as a hex number.

To display the symbol names and addresses from the symbol table, use F8. To display the record definition table, use F7.

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 table is invalid, chances are good that it has been garbled. For the symbol table, use the LS command to clear and reload symbols.

20 DOS busy
DOS function calls are used by the Load, Name, View, Unassemble Source, Write, Echo and DOS exit commands. Since DOS is not re-entrant, Periscope tests to be sure that DOS is available (not busy). This is done by

checking a flag set by DOS 2.00 and later versions. This flag must be zero, interrupts must be enabled, and the interrupt vector must equal its saved address for Periscope to allow DOS functions. If you receive this message, use the Go command to get back to your program's code and try the DOS function again. If you're in DOS, execute RUN.COM and try the DOS function again. For the LA and WA commands, Interrupt vectors 25 and 26 must contain the same address as when PS.COM or RUN.COM was run.

21 Not enough memory
Insufficient memory is available to perform the Load or eXit to DOS command. Periscope checks the amount of memory to be sure enough memory is available for disk I/O.

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 Cannot open file
Periscope was unable to open a file for input or output. If you're loading a file into memory, check the name as 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 Incorrect 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 and V in any order, in upper or lower case. If a number is entered, it must be of the form X:nn, where X is the token, and nn is the number of lines desired. For the R token, the number is ignored and presumed to be two. A number must be followed by a space, a slash (indicating the start of another installation option), a carriage return, or a period and a color setting in the format cc. The total number of windowed lines, including a separator line for each window, must be 21 or less. The minimum size of the U window is four lines. If 43-line mode is used, the window sizes may be 18 lines longer than for 25-line mode.

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.

26 Function not available
This error indicates that the desired command is not available.

This error can occur under several conditions: when an IR command is used and no IS command has been previously used to save the interrupt vectors; when a RR command is used and no RS command has been previously used to save the registers; when a TB, TR, or TU command is used and /B:0 was used when PS.COM was installed; when a UB, US or V command is used and /E:0 was used when PS.COM was installed; when LS is used and /T:0 was used when PS.COM was installed; or when a BU or /U command is used and no /I:nn was used when PS.COM was installed.

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 B, W, D, Q or T 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. The instruction would be entered as MOV B [SI],1 or MOV W [SI],1, respectively. Note that Periscope's disassemble command shows the B as byte ptr and the W as word ptr.

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 Line symbol not found

For the JL command, the current instruction (CS:IP) must be a line symbol. If you're not currently at a source line, use the Go command or the BL breakpoint to get to the next source-code line and then use the JL command.

32 PSP not found
The Name command was not able to locate the PSP. This error can be ignored 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.

33 Cannot write echo file
A file error has occurred while writing the echo file.

34 DOS 3.0 or later required
The /X command requires DOS 3.0 or later to be able to retrieve the PSP.

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 DE mode set
A D command was entered after a DE command. Use a sub-function to change the display mode.

37 Range conflicts with Periscope
The range specified by an HM command crosses into Periscope's memory. Specify another range that does not interfere with Periscope.

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, /R, /S, /T, and /V options, the number must be one or two hex digits.

41 Not enough memory
Insufficient memory is available to install Periscope. Check the amount of available memory using CHKDSK. Boot the system or reduce the space Periscope requires in RAM 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 above.

44 Unable to modify protected memory
Periscope was not able to install itself in the protected memory. Check the port setting on the board and the port number specified with the /P installation option, if any. Also, check that you're not using a 16K board. More memory is now required! If the problem persists, run PSTEST.COM (see Chapter IX).

45 Unable to protect memory
After protecting the memory on the board, Periscope was able to modify the supposedly protected memory. Check the memory setting on the board and the memory address specified with the /M installation option, if any. Also, check that you're not using a 16K board. More memory is now required! If the problem persists, run PSTEST.COM (see Chapter IX).

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 memory board is properly seated in the expansion slot and that the chips on the board are properly seated in their sockets. Also, check that you're not using a 16K board. More memory is now required! If the problem persists, run PSTEST.COM (see Chapter IX).

47 Screen size must be from 0 to 20H (32) KB
The size of the program's screen specified with the /S option must be from zero to 20H K. Note that the number is in hex!

48 Symbol table size must be from 0 to 3FH (63) KB
The size of the symbol table specified with the /T option must be from zero to 3FH K. Note that the number is in hex!

49 DOS 2.00 or later required
Periscope requires DOS 2.00 or later.

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 zero to 20H K. Note that the number is in hex!

51 /Q option invalid for this system
The /Q option can be used only on a PC- or XT-class machine with an 80286 CPU.

52 Unable to read Help or Comment file

An error occurred reading PSHELP.TXT or PSINT.TXT. Restore these files from your backup disk.

53 Port number must be from 100H to 3FCH
The port number specified with the /P option must be from 100H to 3FCH. 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 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 response file
An error occurred when PS.COM tried to read the response file. Check the file name and try again. Note that any installation options entered after the response file name are ignored. 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 use PS @c:std /c:17, the color attribute is not used.

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 zero to 3FH K. 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, for more information.

60 Load segment for Periscope tables must be from xxxx to yyyy
The load segment as 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 1000H paragraphs. If the PSP is C00H and the top of memory is 5000H, then the allowable range for the load segment is C10H through 4000H.

61 Source buffer size must be from 0 to 10H (16) KB
The size of the source buffer specified with the /E option must be from zero to 10H K. Note that the number is in hex!

62 Unable to write response file
Periscope is unable to write the response file on the default drive. Check the disk and try again.

63 Periscope not configured-run CONFIG.COM
Periscope must be configured before it can be run in your system. See Chapter II.

64 Unable to use 43-line mode
An Enhanced Graphics Adapter is required for 43-line mode. If the EGA is driving a color display, the Enhanced Color Display is also required.

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 hardware
An attempt was made to run the Periscope I software on the Periscope III board or vice-versa. The Periscope I software may be run without the Periscope I board in the system, but it may not be run on the Periscope III board. The Periscope III software requires that the Periscope III board be in the system.

67 Unable to read trace buffer
Periscope was unable to read the hardware trace buffer. Check the items listed under the description of the utility program PS3TEST.COM and try again. This error may also occur if a parity error occurs as memory in the first megabyte is read by Periscope.

70 File not found
RUN was not able to find the specified file. Check the file name and restart RUN.

71 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.

72 Unable to read xxxxxxxxxxx
An error occurred reading the indicated file. Check to be sure the disk is ready and that the file size shown by DIR indicates the true file size.

73 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.

74 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.

75 Periscope not installed correctly
RUN was unable to modify the protected memory. Reload Periscope and try again.

76 Format error in MAP file
A MAP file was found for a COM or EXE file, but RUN was unable to load it correctly. Check the format of the MAP file and the size required for the MAP file using TS.COM. The MAP file must be in the format as produced by LINK—some editors may cause subtle reformatting of the file.

77 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.COM.

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 F7 to see the records that were loaded. Note that the last record will usually be only partially defined.

78 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, otherwise check the file and try again.

79 Logical error in symbol table

A logical error was found in the symbol table during final processing. If a PSS file is used, regenerate it and try again. If a MAP file is used, please report the error immediately.

80 MAP/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 three minutes later than that of the MAP or PSS file. This is to be expected if the program file has been patched. Otherwise, it indicates that an obsolete MAP or PSS file is being used. Proceed with caution!

81 MAP file larger than symbol table-some symbols loaded
The space required by the MAP file is greater than the symbol table size. As much of the MAP file is loaded into the symbol table as possible. Use F8 to see the symbols that were loaded. To correct the problem, reinstall Periscope with a larger symbol table.

90 DEF file not found
RS.COM 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 drive and file 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.COM in Chapter IX.

93 Not enough memory
Insufficient memory is available for RS.COM to load the DEF file. Check the amount of available memory using CHKDSK and re-boot as needed.

94 DOS 2.00 or later required
Periscope requires DOS 2.00 or later.

100 MAP file not found
TS.COM was not able to find a file of the specified name with an extension of MAP. If the options so indicate, a SYM or EXE file is used instead of a MAP file.

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. The format as produced by the linker may have changed—try using another version of LINK. If this is the problem, please advise us of the situation as soon as possible.

103 Not enough memory
Insufficient memory is available for TS.COM to load the MAP file. Check the amount of available memory using CHKDSK and re-boot as needed.

104 Unable to write PSS file
A disk error occurred when TS attempted to write the PSS file. Check the disk and try again.

105 DOS 2.00 or later required
Periscope requires DOS 2.00 or later.

106 Unknown PLINK symtable type-x
TS.COM encountered an unknown record type in the symbol table at the end of the PLINK file. Please notify us immediately if you encounter 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.

110 Unable to read PS.PGM or PSH.PGM
An error occurred reading the file PS.PGM or PSH.PGM. Check the disk and file and try again.

111 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 at 404/256-3372 for assistance.

112 DOS 2.00 or later required
Periscope requires DOS 2.00 or later.

113 Wrong version of Periscope
The versions of CONFIG.COM and PS.PGM do not agree. Use the same version of both programs and try again.

114 Unable to write file
An error occurred when CONFIG.COM attempted to write a file on the target disk. Check the disk and try again.

115 File too large
An internal error has occurred. Reboot and try again. If the problem persists, please notify us.

116 Unable to read file
An error occurred when CONFIG.COM attempted to read a file from the distribution disk. Make sure the disk is a diskcopy of the original distribution disk and try again.

120 DOS 2.00 or later required
Periscope requires DOS 2.00 or later.

121 File not found
The PUBLIC program was not able to find a file of the specified name. If no extension is specified, ASM is used. If an extension is specified, it is used.

122 Not enough memory
Insufficient memory is available for PUBLIC.COM to load the program file. Check the amount of available memory using CHKDSK and re-boot as needed.

123 Unable to read program file
An error occurred reading the input file. Check the file and try again.

124 Unable to write .PUB file
A disk error occurred when PUBLIC attempted to write the output file. Check the disk and try again.

125 Invalid option
An invalid command-line option was used. Enter 'PUBLIC ?' to display the valid options.

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.

131 Invalid option
The only valid command-line option for SYMLOAD is '/I:nn' where nn is the interrupt vector to be used by SYMLOAD. The hex number must be from zero to FF.

140 File not found
INT.COM 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 DOS 2.00 or later required
Periscope requires DOS 2.00 or later.

143 Invalid option
An invalid command-line option was used. Enter 'INT ?' to display the valid options.

150 No hotkeys set
No hotkeys were specified. Enter 'PSKEY ?' to display the possible hotkeys.

160 Invalid option
An invalid command line option was used. Enter 'PSTEST ?' to display the valid options.

161 Zero test failed
PSTEST was unable to write zeroes to the protected memory. If all memory chips fail, check the memory and port switch settings. If one or two chips fail, check the indicated memory chips.

162 Foxfox test failed
PSTEST was unable to write FFH to the protected memory. If all memory chips fail, check the memory and port switch settings. If one or two chips fail, check the indicated memory chips.

163 Rotate test failed
PSTEST was unable to write a rotating bit pattern to the protected memory. If all memory chips fail, check the memory and port switch settings. If one or two chips fail, check the indicated memory chips.

164 Write protect failed
PSTEST was unable to write protect the memory. Check the port switch settings.

165 Copy test failed
PSTEST was unable to copy a block of memory to the protected memory. If all memory chips fail, check the memory and port switch settings. If one or two chips fail, check the indicated memory chips.

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 Chapter IX.

200 Invalid option
An invalid command-line option was used. Enter 'PS3TEST ?' to display the valid options.

201 Unable to write PSBUF.DAT
An error occurred writing the hardware trace buffer to disk. Check the disk and command-line options used and try again.

202 through 225
These errors indicate a potentially serious diagnostic failure. Check the items listed under the description of the utility program PS3TEST.COM and try again. If any errors persist, call Tech Support for assistance.

Index

Special Characters

A

B

C

D

E

F

N

O

P

Q

R

S

T

U

V

W

X