Periscope®/ Remote for DOS

- Introduction and Overview
- Installing and Using Periscope in Remote Mode
- The Remote DOS Stub Program (PSR)
- The Remote DOS Utility Program (RX)
- Using PSR with Models other than Model IV
- Using PSR with Periscope Model IV

This addendum describes Periscope/Remote for DOS and how you can use it with Periscope to debug in environments not otherwise possible. You'll find detailed instructions for installing and using Periscope in the Periscope version M54 manual. The purpose of this addendum is to document what's different when you install Periscope on a host computer and Periscope/Remote for DOS on a target computer to debug software running on the target system.

INTRODUCTION AND OVERVIEW

Periscope/Remote for DOS (PSR) enables you to debug DOS software running in another DOS system. The target DOS system can be one with an MCA bus, such as a PS/2, or one with a standard ISA/EISA (PC) bus. The "main" Periscope software runs in the host system and the PSR program runs in the target system. You can debug the target system in *real-time* when you use Model IV hardware with PSR.

PSR (this package) contains this addendum and two floppy disks (5.25" and 3.5").

To run PSR, you need:

- two computers
- the Periscope Version 5.40 or later software
- PSR Version 5.40 or later
- a null-modem cable

If you're using Periscope Model IV, you need the items above plus:

- the Model IV hardware
- an NMI Clip
- the 48" shielded ribbon cable (optional—replaces the standard 18" unshielded cable)

Here's how Periscope/Remote for DOS works.

The *host system* is the system where the main Periscope software (PS.COM, etc.) runs. When you use the Model IV hardware, you install the Model IV Board in this system. You must run PC-DOS or MS-DOS on this system.

The *target system* is the system where the program you're debugging runs. You must run also run PC-DOS or MS-DOS on this system, which can be a PC or a PS/2. You may also install a Model I (Plus) Board on this system to keep PSR from using any of the lower 640K. When you use the Model IV hardware, you install the pod and an NMI Clip in this system.

PSR.COM is the Periscope/Remote for DOS software. It is a small (8K) program that runs in the target system. It communicates with Periscope on the host system via a serial link. This serial link requires an available COM port on both systems. On ISA- or EISA-bus systems, this can be COM ports 1 through 4, and on MCA-bus systems, it can be COM ports 1 through 8. The serial communications can occur at 9,600 baud (Slow); 38,400 baud (Medium); or 115,200 baud (Fast).

You use a null-modem cable to link the host system to the target system. If you have version 3.00 or later of the Brooklyn Bridge or LapLink, you can use the cable that comes with either of these products. (We do not supply a null-modem cable. If you want to make your own cable, see the specifications in the NOTES.TXT file in the Periscope directory.)

The utility program RX.COM runs on the host system. You use it to download files to and upload files from the target system. You can also use it to run programs on either the host or target system, which is handy for remote file maintenance.

When you run the PSR.COM program in the target system and link the host and target systems via the serial link described above, Periscope runs in *active remote mode*. In this mode, you use the host system to interactively read and write memory, I/O ports, registers, and so forth on the target system. When you use Periscope Model IV in active remote mode, the target system stops when a hardware breakpoint occurs and you can examine the state of the target as desired.

⇨ If your target system is a DOS system, PSR runs in the target system and Periscope runs in the host system. If the target system is a Windows 3.x or OS/2 2.x system, you'll need Periscope/32 for Windows or Periscope/32 for OS/2.

When you use Model IV, you can also run in *passive remote mode*, which does not require this package. In passive mode, you do not run the PSR program in the target system nor do you link the two systems with the null-modem cable nor do you install the NMI Clip in the target system. You just install the Model IV Board in the host system and the pod in the target system. You can set hardware breakpoints from the host system on events in the target system, but when a hardware breakpoint occurs, the target system does not stop. You can examine the information captured in the real-time trace buffer, however. See the Periscope Model IV manual for details on using Model IV in passive remote mode.

To use Periscope in *active remote mode*, you need these items in the host system:

- Periscope Version 5.40 or later software
- your program's source and symbol files
- an available COM port connected to a null-modem cable
- the Model IV Board if you're using Periscope IV

You need these items in the target system:

- PSR.COM, Version 5.40 or later
- your program's executable code
- an available COM port connected to a null-modem cable
- a Model IV pod and NMI Clip if you're using Periscope IV
- a Model I or Plus Board if you want to keep the stub program out of the lower 640K

The remainder of this addendum provides detailed instructions for installing and using Periscope/Remote.

INSTALLING AND USING PERISCOPE IN REMOTE MODE

When you run Periscope in *active remote mode*, it runs on the host system and communicates with PSR.COM on the target system over a serial link connecting the two systems. It loads into normal DOS memory. It does not load into the Model I Board's memory, even if a board is present. The default symbol table size is 80H (128) KB.

Periscope's screen indicates that Periscope is in remote mode by displaying the prompt as `/2>` instead of `>`.

Periscope/Remote implements some commands on the host system, others on the target system. The commands it implements on the host include file-oriented commands such as `LB`, `WB`, `LD`, `WD`, `LS`, `WS`, `V`, `VS`, `UB`, `US`, `/E`, `/K`, `HR`, `HT`, `HU`, `HW`, and `/X`. The commands it implements on the target include `LA`, `WA`, `LF`, `WF`, and `N`.

See Chapter 5 in the Periscope manual for general installation options and *passive remote mode* options. The only installation option for *active remote mode* is:

/2:px—Set COM port.

The port **p** is **1** through **4** or **1** through **8** if you're using a PS/2 system. The speed **x** is **S** (slow—9600); **M** (medium—38400); and **F** (fast—115,200). Both PSR and RX are dedicated remote programs that default to port 1 at the Fast speed, but PS has no default. You must specify the **/2:px** option for PS to run in remote mode. If a port or speed mismatch occurs when PS.COM is loaded, you'll need to reload PSR.COM and then reload PS.COM.

Periscope in a single-machine environment and in the remote environment are very similar, but there are some significant differences. They are:

- Normally, Periscope uses the DOS memory allocation chain to display the owner of the current disassembly address. Periscope does not do this in remote mode.
- A suffix for the Quit command lets you execute the command for the host only or for both systems. For example, `QB H` boots the host system and `QB` boots both systems. When you use the `QR` command in remote mode, the host system returns to the DOS prompt and the target system executes a `QC` command.
- The PS4TEST program does not work when the Model IV Board and pod are not in the same system, so it does not work in remote mode.
- Periscope/Remote does not allow you to use the `/A`, `/AK`, and `/AV` installation options. It assumes that you're using a single-monitor system and that you're using the COM port to connect the host to the target rather than to an alternate PC.
- When you use commands that have an optional address, such as `LF` and `WF`, be sure to specify the address. The default on the host will not generally be the correct value for the target system.
- The SYMLOAD.COM program does not work in remote mode. Periscope/Remote processes symbols on the host system. SYMLOAD is executed by your program, which

runs on the target system in remote mode.

- Periscope in remote mode shows **`PSR Version 5.xx`** as item 7 in the Watch window instead of the normal **`PERISCOPE 5.xx`**.
- When you execute RUN.COM with no arguments in remote mode, the target system does not show an INT 20 instruction. If you want to show an INT 20, create a 2-byte COM file that contains just an INT 20 and run it.
- The USEREXIT program supports remote mode. It can read and write memory on the target system. It can also download and execute code on the target system and upload a string from the target. See USEREXIT.ASM for details.

THE REMOTE DOS STUB PROGRAM (PSR)

PSR.COM is the Periscope/Remote for DOS program. It communicates with Periscope over a serial link to read/write memory, I/O ports, system registers, and load/execute programs on the target system.

⇨ We have documented the interface to Periscope so that you may license the source code for PSR (*DOS Remote Stub Source*) and port it to other real-mode environments if you wish.

If you're debugging system-level software running in a Windows 3.x or OS/2 2.x environment, the *Periscope/32 for Windows* and *Periscope/32 for OS/2* packages will be very helpful. If your target environment is any other protect-mode environment, you may license the *Periscope 32-bit Toolkit*.

PSR is less than 8K in size and less than 6K when you make it memory-resident. You can make it even smaller by trimming back on messages and eliminating optional functions. Also, you can load it into the memory on a Periscope Model I or Model I/MC Board. See the **`/M`** and **`/P`** installation options described below.

The options for PSR are:

/2:xy—Set COM port.

The port **x** is **1** through **4** or **1** through **8** if you're using a PS/2 system. The speed **y** is **S** (slow—9,600); **M** (medium—38400); and **F** (fast—115,200). The default setting is **/2:1f**, which is port 1 at the fast speed.

/3—Do not allow 386 debug register use.

Use this option to disable use of the 386 debug registers.

/D—Do not allow DOS use.

Use this option to keep PSR from using DOS services.

/H:nnnn—Set a dummy port for Model IV use.

The dummy port is the lower of two I/O addresses used to synchronize Model IV in a remote link. On entry, PSR reads dummyport+1 and on exit PSR reads dummyport+0.

/K:nn [nn]—Mask hardware interrupt request (IRQ) lines when Periscope's screen is displayed.

The value of **nn** may be from zero to FFH. The first value corresponds to the first 8259 interrupt controller. The optional second value corresponds to the second 8259 used on AT-class machines. The default value is zero, meaning that no IRQ lines are masked. A one bit in **nn** masks the corresponding IRQ line.

For example, to mask the timer (IRQ 0), use **/K:1**. To mask the keyboard (IRQ 1), use **/K:2**. To mask IRQ 1 and IRQ 2, use **/K:6** (bits 1 and 2 on).

/M:nnnn—Set the protected memory segment.

Parameter **nnnn** is the segment. Use this option if you want to load PSR into a Model I or Plus Board. Use either this option or the `/P` option to notify PSR that a Model I or Plus Board is in the system. If the target system is a PS/2, you may need to use both options, since PSR does not have the logic to automatically detect which memory and ports to use. When PSR loads into the memory on the Model I or Plus Board, it does not write-protect the memory.

/N—Do not intercept interrupt 2 (NMI).

Do not use this option unless you really need to, since it keeps PSR from handling parity errors, the break-out switch, and hardware breakpoints.

/P:nnnn—Set the protected memory port.

Parameter **nnnn** is the base port. Use this option if you want to load PSR into a Model I Board. See the `/M` option above.

/Q—The target system has a PC (8088) motherboard with an 80286/80386 turbo card.

Use this option only if the target is a PC or XT with a turbo card.

/R—Make PSR memory-resident.

If you do not specify this option, PSR remains in the foreground where it loads and executes programs using RUN.COM. When you use this option, RUN cannot load and execute programs nor is the `QR` command available.

/W:nn—Set the wait count (as a hex number) for Model IV use.

Use this option to help synchronize Periscope Model IV's trace buffer when the host and target systems run at different speeds. The default wait count is 30H, but you may adjust it up or down to synchronize the trace buffer. If you set a hardware breakpoint that does not fill the trace buffer and entries at the top of the buffer are missing or you miss hardware breaks that are very close to the starting point, increase the wait time. If you see a large number of loop instructions executing inside PSR.COM at the top of the trace buffer, decrease the wait time.

If the host is faster than the target, set the wait count to 1. If the host is slower than the target, set the wait count higher than 1.

/X—Do not intercept exception interrupts 6 and DH.

Generally, you'll find it useful to have PSR intercept these exception interrupts. However, if another program running on the target system is using them, you can specify this option to tell PSR to leave them alone.

/Y—Remove PSR from memory.

Use this option only when PSR has been installed with the `/R` option. Be sure that PSR is the last thing in memory before removing it.

THE REMOTE DOS UTILITY PROGRAM (RX)

Use the RX utility program (on the host) to execute programs on the target system from the host system, and to download files to and upload files from the target system. The only command-line option is:

/2:px—Set COM port.

The port **p** is **1** through **4** or **1** through **8** if you're using a PS/2 system. The speed **x** is **S** (slow—9600); **M** (medium—38400); and **F** (fast—115,200). The default setting is **/2:1f**, which is port 1 at the fast speed. If a port or speed mismatch occurs when RX.COM is loaded, you'll need to reload PSR.COM and then reload RX.COM.

Once you start RX, it prompts you for commands until you press **Ctrl-C**. The prompt `RX>` indicates that remote execution is occurring.

RX requires that PSR is running on the target system. Generally, RX will not work if PSR was loaded with the `/R` option or if the target system is in the middle of executing a program. Any screen output or keyboard input done by the target program remains local to the target system.

The most common use of RX is for file maintenance on the target system. For example, if you need to create a new directory, delete files, or display the contents of the directory, you can easily do these things by executing the appropriate DOS command via RX. You can execute any program using RX, since it adds COMMAND/C to the front of your command input. When you're using RX to execute a program on the target system, you cannot break into Periscope on the host system, since PSR is busy EXEC-ing the program for RX and cannot be re-entered.

To copy a file from the host to the target, enter:

```
@DN <hostfile> [<targetfile>]
```

at the RX prompt. To copy a file from the target to the host, enter:

```
@UP <targetfile> [<hostfile>]
```

at the RX prompt. The second file name is optional.

⇨ RX does not support wildcards in filenames for the `@UP` and `@DN` commands.

To execute a program on the host system, enter the command with a leading dollar sign. For example, `$dir` displays the host system's directory and `dir` displays the target system's directory.

RX Error Messages

Error 260—Not enough memory

Insufficient memory is available for RX to execute the program specified. Unless you are executing a program on the host system using `$`, the lack of memory is on the target system. Check the amount of available memory using CHKDSK and re-boot as needed.

Error 261—Exec failure

An error occurred when RX attempted to Exec the desired program. This can occur if PSR was run with the `/R` option or if the target system is executing another program.

Error 262—File not found

The specified file was not found. Check to make sure that COMMAND.COM is available on the target system and that the program you are trying to execute is in the specified directory.

Error 263—Remote timeout

The remote system did not respond. Check the null-modem cable using `PSTERM/T` on both systems. Also check the port and speed settings used on both systems.

Error 264—Host file read/write error

RX received an error trying to read or write the specified file on the host system. Check the file name and try again.

Error 265—DOS not available on target system

PSR was loaded with the `/D` option that disables its use of DOS. Reload PSR without the `/D` option and try again.

Error 266—Target file read/write error

DOS is busy on the target system and is not available to read or write the file.

Error 267—Invalid/missing argument

The `@UP` or `@DN` command is missing one or more arguments. Check the syntax and try again.

Error 268—xxxx timeout (Function= yy, byte= zz)

A read or write timeout occurred. Enter retry to retry the operation a few times. If errors persist, restart both PSR and RX and try again. If you are still having problems, please call Tech Support.

Error 269—Unable to initialize COM port

An error occurred while initializing the COM port. Make sure that you are specifying the correct port number and speed. Check to make sure that PSR is running on the target system and that the null-modem cable is correctly installed.

If errors persist, run `PSTERM/T` on both systems to verify the installation.

USING PSR WITH MODELS OTHER THAN MODEL IV

See the Periscope manual for detailed instructions on installing Periscope *without* Periscope/Remote for DOS. The following instructions provide information specific to installing Periscope *with* Periscope/Remote for DOS.

Step 1. Install the Periscope hardware.

If you have a Periscope Model I Board, install it in the target machine to keep the stub program out of low DOS memory. Periscope assumes you're using the host system only to run Periscope, so it does not recognize a Model I Board on the host system.

If you have a Break-out Switch, install it in either the host or the target system.

Step 2. Configure Periscope on the host system.

See Chapter 3 in the Periscope manual for details.

Step 3. Load PSR.COM on the target system.

Do not use the `/R` option if you want to use RUN.COM to load your program. See the 'The Remote DOS Stub Program (PSR)' section of this addendum for details.

Step 4. Install the main Periscope program, PS.COM, on the host system.

Enter `PS /2:px`, where `p` is the com port and `x` is the speed (S, M, or F). See the Periscope manual and 'Installing and Using Periscope in Remote Mode' section of this addendum for details.

Step 5. Copy the symbol (.PSS), definition (.DEF), and

source files for the program you want to debug to the host system.

See Chapter 7 in the Periscope manual for information on creating symbol (TS.COM) and definition (RS.COM) files.

If you want to use RUN.COM to run your program, make the necessary files available in similar directory structures on both machines. For example, when you enter `RUN C:\FOO\BAR` to load BAR.EXE on the target system, Periscope looks for the symbols in the file C:\FOO\BAR.PSS on the host system. If it is not possible to maintain similar directories on both systems, use the current directory on each system for all the files that are needed.

Step 6. Copy the .COM or .EXE file you want to debug to the target system if it isn't already there.

You can download it from the host using the RX utility.

Step 7. Run the program you want to debug on the target system.

Enter `RUN <program name>` on the host system to run and debug your program. See Chapter 7 of the Periscope manual for details on using RUN.COM.

USING PSR WITH PERISCOPE MODEL IV

To use Periscope Model IV in remote mode, you must have a host system with a 286 or later CPU because the Model IV code uses some 286-specific instructions. Your target system can have a 286, 386SX, 386DX, 486DX, or 486DX/2 CPU.

Since you're physically hooking two systems together, be sure that both systems are powered from the same wall outlet and have the same polarity. If the computers do not have a 3-connector plug, double-check the polarity before applying power!

To use Model IV in *active remote mode*, you need a null-modem cable to connect your host and target systems and an

NMI Clip to connect the pod to the NMI line in the target system. We supply two types of NMI Clips: ISA/EISA for standard bus machines and MCA for PS/2 systems.

⇨ Do not use the NMI Clip when running in passive remote mode. Also, do not attempt to make your own NMI Clip, since the clip requires a special diode for proper operation.

To use Model IV in either active or passive remote mode, you may need a longer-than-standard ribbon cable to connect the Model IV Board in the host system to the pod in the target system. The standard ribbon cable is 18 inches long, long enough to reach from the host to the target in some situations. The optional 48-inch shielded cable gives you more room between the pod and the Model IV Board.

⇨ If you have an early 386 or 486 Pod, a standard 18-inch ribbon cable may be permanently attached to your pod. If your cable isn't long enough to connect your host and target systems, please call Tech Support.

To use Periscope IV in *passive remote mode*, you do not need the null-modem cable or the NMI Clip. See the Periscope Model IV manual for information on passive remote mode.

See the Periscope Model IV manual for detailed instructions on installing Periscope IV without PSR. The following instructions provide information specific to PSR.

Before you begin, connect your null-modem cable to an available COM port on both the host and target systems.

Step 1. Install the board(s), ribbon cable, and pod.

The target system, where you install the pod, is the speed-sensitive system. You can use a Model IV with target systems running at any speed up to your Model IV's speed rating. Your host system can run at any speed, regardless of the Model IV speed rating.

Turn off the power on both systems. Install the Model IV

Board in the host system. Install the pod in the target system. Connect the board and pod with the 18-inch standard ribbon cable or with the 48-inch shielded ribbon cable. If you have a Plus Board, install it in the target system. It will keep the stub program out of the lower 640K.

Step 2. Install the NMI Clip.

Both AT-style and PS/2-style NMI Clips connect to the pod at pin J6 (see Figures 3-3 and 3-5 in the Model IV manual) and to the target system bus.

If you have the AT-style NMI Clip, connect the probe to the bus at pin A1. If you have the PS/2-style NMI Clip, connect the mini-grabber hook at its termination to pin B32 of the bus. See the instructions in the NMI Clip package for details.

Step 3. Start PSR.

Before you start PSR for the first time after installing the NMI Clip, double-check all connections in both systems. Be sure the NMI Clip is correctly connected to the target system's bus and is not short-circuited to any other connector. Then follow the steps below.

⇨ Always follow these steps exactly to run Periscope Model IV in active remote mode.

Step 3a. Power up the host system, then enter `PS /2:1f` to begin the installation of Periscope.

Do not turn the target system on before you complete this step.

Step 3b. When Periscope (host system) prompts you, power up the target system and run PSR with the appropriate option(s).

See 'The Remote DOS Stub Program (PSR)' section of this addendum for details.

Step 3c. Press return on the host system to complete the installation of Periscope.

Once you have performed Steps 3a through 3c, you can turn the target system on and off at will. Before you turn the host system off, power down the target system. Otherwise, the host system generates an NMI into the target system. To help you remember the sequence, think of the host at a party who must be there first and cannot leave until last.

Step 4. Copy the symbol (.PSS), definition (.DEF), and source files for the program you want to debug to the host system.

See Chapter 7 in the Periscope manual for information on creating symbol (TS.COM) and definition (RS.COM) files.

If you want to use RUN.COM to run your program, make the necessary files available in similar directory structures on both machines. For example, when you enter `RUN C:\FOO\BAR` to load BAR.EXE on the target system, Periscope looks for the symbols in the file C:\FOO\BAR.PSS on the host system. If it is not possible to maintain similar directories on both systems, use the current directory on each system for all the files that are needed.

Step 5. Copy the .COM or .EXE file you want to debug to the target system if it isn't already there.

You can download it from the host using the RX utility.

Step 6. Run the program you want to debug on the target system.

Enter `RUN <program name>` on the host system to run and debug your program. See Chapter 7 in the Periscope manual for details on using RUN.COM.

Notes on Running Model IV in Active Remote Mode.

If you use PSKEY.COM, add a `3` option to the command line. Periscope ignores a software INT 2 from the host when you run Periscope IV in active remote mode.

If the target system is a PS/2 Model 50, see the description of the probe hardware control (`HC P` command) in Chapter

5 of the Periscope Model IV manual.

Due to the different CPU speed combinations possible when doing remote hardware debugging, we've added some synchronization capabilities to Periscope and PSR.COM. One of these is the `PSR /W:nn` option. Use this value to slow down PSR's exit code so that the hardware is active on the host before the target system returns to your program. See 'The Remote DOS Stub Program (PSR)' section of this addendum for details.

When Model IV triggers on a hardware breakpoint, the trace buffer stops capturing information immediately, leaving the trigger event as the last item in the buffer (assuming the default hardware control `HC TB` was set). If something other than a hardware breakpoint (such as a code breakpoint or an exception interrupt) activates Periscope, the board continues capturing information until Periscope disarms it. In the normal single system setup this happens quite quickly, leaving only a few 'garbage' cycles in the trace buffer. In remote mode however, the time span can be quite long, since PSR gets the wake up call and passes the information back to Periscope over the serial link. This can cause most of the trace buffer to be filled with code of PSR waiting on the serial link.

Fortunately, we can use the hardware breakpoints to quit capturing information much earlier. Doing this relies on PSR's access to some dummy I/O ports and on the corresponding hardware breakpoints you set. The `PSR /H` option sets the value of the dummy ports. PSR reads the second port on entry and reads the first port on exit. Using this information, we can set some hardware breakpoints with Periscope Model IV's sequential triggers to capture just what we want. Assuming a command-line option of `/H:F300`, the command sequence would be:

Command 1: `HC X0`

This command excludes capturing trace buffer information when Model IV is in state 0.

Command 2: `HP F300 I (0,1)`

This command switches Periscope Model IV from state 0 to state 1 when the dummy port is read on exit from PSR. This starts capturing information in the trace buffer.

Command 3: `HP F301 I (1,!)`

This command causes a hardware breakpoint on entry into PSR when port F301 is read, and stops capturing information in the trace buffer.

Any hardware breakpoints you set would have a start state of 1. If you need multiple states, change the start state for Command 3 and the end state for Command 2 to the appropriate values.⌖

A D D E N D U M

Periscope/32 for Windows

This addendum describes how you can use *Periscope/32 for Windows* to debug system software at the source level in a Windows 3.x enhanced mode environment. Please refer to the Periscope manual for information on Periscope/32, the host software included in the *Periscope/32 for Windows* package. If you're using the Periscope Model IV hardware, see also the Model IV manual for additional information.

INTRODUCTION AND OVERVIEW

Periscope/32 for Windows enables you to debug Windows virtual device drivers (VxD), Windows device drivers (DRV), DOS device drivers (SYS), programs running in the DOS box, and other system software running in a Windows 3.x enhanced mode system.

Two systems are required: a host DOS system where Periscope/32 runs, and a target Windows 3.x system where Periscope/Remote for Windows runs.

Periscope/32 for Windows (this package) contains:

- a disk with both the Periscope/32 software and the Periscope/Remote for Windows software
- an optional Break-out (NMI) Switch
- the Periscope manual and quick-reference card
- this addendum

To run *Periscope/32 for Windows*, you need:

- two 80386 or later computers
- the Periscope/32 software (Version 5.40 or later if you're using Model IV hardware)
- the Periscope/Remote for Windows software (Version 5.40 or later if you're using Model IV hardware)
- a null-modem cable
- (optional) a Model IV Board, Pod, NMI clip, and shielded cable for Model IV hardware support

How Periscope/32 for Windows works.

The *host system* is the DOS system where the Periscope/32 software runs. If you're using the Model IV hardware, you install the Model IV Board in this system. The *target system* is the Windows 3.x system where the Periscope/Remote for Windows software and the program you're debugging runs. If you're using the Model IV hardware, you install the Pod and NMI Clip in this system.

PSRW.386 (PSRW) is the Periscope/Remote for Windows software. It is a small (9K when resident) virtual device driver that runs in the target system. It communicates with Periscope/32 on the host system via a serial link. This serial link requires an available COM port on both systems. On ISA- or EISA-bus systems, this can be COM ports 1 through 4, and on MCA-bus systems, it can be COM ports 1 through 8. The serial communications can occur at one of three speeds: 9,600 baud (Slow); 38,400 baud (Medium); or 115,200 baud (Fast).

You use a null-modem cable to link the host system to the target system. If you have version 3.00 or later of the Brooklyn Bridge or LapLink III, you can use the cable that comes with either of these products. If you want to make your own cable, see the specifications in the NOTES.TXT file in the Periscope directory.

If you're using Model IV, you connect the Model IV Board in the host system with the Pod in the target system using the 48-inch Shielded Ribbon Cable.

When you run PSRW in the target system and link the host and target systems via the serial link described above, Periscope/32 runs in *active remote mode*. In this mode, you use the host system as a debugging terminal to interactively debug software running on the target system.

To use *Periscope/32 for Windows*, you need these items in the host system:

- DOS 3.1 or later
- Periscope/32 software (Version 5.40 or later if you're using Model IV hardware)
- your program's source and symbol files
- an available COM port connected to a null-modem cable
- (optional) a Model IV Board for Model IV hardware support

You need these items in the target system:

- Windows 3.x running in enhanced mode
- Periscope/Remote for Windows (Version 5.40 or later if you're using Model IV hardware)
- your program's executable code
- an available COM port connected to a null-modem cable
- (optional) a Pod and NMI Clip for Model IV hardware support

INSTALLING PERISCOPE/32 FOR WINDOWS

The following instructions provide details for installing *Periscope/32 for Windows.* You will install and load symbols for the sample driver VXD as part of the installation process.

Step 1—HOST AND/OR TARGET: Install the hardware.

If you're using the Model IV hardware, install it per the instructions in the Model IV manual.

If you're not using the Model IV hardware, you can install the Break-out (NMI) Switch in either the host or the target system. See Chapter 4 in the Periscope manual for installation instructions.

If you have a Model I Board, you can install it in either the host or target system so you can use the plug-in Break-out Switch. *Periscope/32 for Windows* does not use the memory on the board, however.

Step 2—HOST AND TARGET: Hook up the null-modem cable to connect port 1, then test the serial link.

Be sure that the cable meets the specifications described in the file NOTES.TXT. Note that the motherboard COM port on some IBM PS/2 systems does not run reliably at the fast speed. Use the medium speed option as needed.

To test the serial link, insert the distribution disk in drive A and enter (on both host and target systems):

```
COPY A:\PSRW\PSTERM.COM C:
PSTERM /T
```

PSTERM should display `No errors detected` after running its tests.

Step 3—HOST: Configure Periscope/32.

Place the distribution disk in drive A and run the setup program by entering `A:SETUP.` You will be prompted for a directory name for Periscope. You may enter a name or press enter to create a directory named PERI32 to contain the needed files. Once the files have been created, SETUP will display the configuration screen. This screen looks like the configuration screen shown in Chapter 3 of the Periscope manual.

If you're not using the Model IV hardware, enter either `1` or `2` for this option. Enter `3` if you are using the Model IV hardware.

In remote mode, Periscope/32 loads into conventional memory on the host. It uses neither EM memory nor the Model I Board's memory regardless of which model you choose.

See the Periscope manual for details on the other options. Press **F10** when you have completed all options.

Step 4—HOST: Copy the source and .MAP files for the sample driver.

So that you can debug the sample driver at source-level, place the distribution disk in drive A and enter

```
COPY A:\PSRW\VXD.ASM C:\PERI32
COPY A:\PSRW\VXD.MAP C:\PERI32
```

to copy the source and map files to the Periscope directory (substitute the directory name you entered in Step 3 for PERI32 if different) on the host system.

Step 5—HOST: Run the symbol table program (TS) to extract the symbol information from the .MAP or .EXE file and write a Periscope symbol (.PSS) file.

Enter `TS VXD/V/32` to get verbose mode and 32-bit offset support. (Although we've converted TS to support 32-bit CodeView information, there are problems with local code and data variables as generated by MASM 5.10B and MASM 6.00, so you should use a .MAP file instead of an .EXE file if possible.)

Step 6—HOST: Enter `PSKEY LR` to set up the left and right shift keys as hotkeys. If you're using the Model IV hardware, enter `PSKEY LR3` instead.

Step 7—TARGET: Setup Periscope/Remote for Windows (PSRW).

Create a directory named PERI32 on drive C on the target system. Place the distribution disk in drive A and enter `COPY A:\PSRW\*.386 C:\PERI32` Next, use an editor to add the following lines to the Windows SYSTEM.INI file, in the [386Enh] section (none of the following text or options is case sensitive):

```
DEVICE=C:\PERI32\PSRW.386
PSRWOPTIONS=/2:1F
DEVICE=C:\PERI32\WINKEY.386
DEVICE=C:\PERI32\VXD.386
```

The file PSRW.386 is the Periscope/Remote for Windows driver. Its options can be set using the `PSRWOPTIONS` line.

The PSRW options shown above indicate the use of COM port 1 at the fast speed. See the section later in this addendum titled 'Periscope/Remote for Windows' for details on the options available. Be sure to match speeds on both host and target systems.

The file WINKEY.386 is a hotkey program that is used to activate Periscope by pressing the **Alt-Ctrl-Esc** key combination on the target system.

The file VXD.386 is a sample driver that can be deleted after you've become familiar with the use of *Periscope/32 for Windows.*

Step 8—HOST: Load Periscope/32.

Load Periscope/32 on the host system by entering `PS /2:1F`. Once loaded, Periscope/32 will display the message `Waiting for remote system ...`

The installation options for Periscope described in Chapter 5 of the Periscope manual also apply to Periscope/32. The following option (**/2:px** in Section 5.2 of the Periscope manual) applies specifically to using Periscope/32 in active remote mode:

/2:px—Set COM port.

The port **p** is **1** through **4** or **1** through **8** if you're using a PS/2 system. The speed **x** is **S** (slow—9600); **M** (medium—38400); or **F** (fast—115,200). PSRW (in the target) defaults to COM port **1** at the **Fast** speed, but Periscope/32 (in the host) has no default for COM port or speed. You must specify the **/2:px** option for Periscope/32 to run in remote mode. Be sure to match speeds on both host and target systems. If a port or speed mismatch occurs when Periscope/32 is loaded, you'll need to restart both systems.

Step 9—TARGET: Start Windows in enhanced mode on the target system.

Please note that there is no support for standard or real modes.

When Windows is ready to load PSRW, it will prompt you whether to load the driver. To skip the loading of PSRW, answer `N` to this prompt. Before answering `Y`, be sure that Periscope/32 is loaded and waiting for the target system to respond (Step 8 above).

Step 10—HOST: Load symbols for the sample driver.

After loading PSRW, Windows will load the sample driver VXD.386. We've embedded an INT 2 in VXD's initialization phase to wake up Periscope. Either an INT 2 (software NMI) or an INT 3 (code breakpoint) can be used to easily wake up Periscope.

When Periscope/32's screen is displayed, the processor will be at the instruction after the INT 2, but you won't see symbols or source code because Periscope/32 doesn't know what symbol file to use or where to locate it. Using two Periscope/32 commands, you can load the symbol file on the host system, then relocate it based on the location of the current instruction (CS:EIP). Once this has been done for a particular driver, the command sequence can be set up as a keystroke macro, since it would change only as a result of major rework to the structure of the driver.
To load the symbols for VXD.386, enter `LS 0 VXD`, specifying any needed path name for the third argument. To relocate the symbols for this segment group, enter `/S PHASE3 CS:EIP`. This relocates the CS and EIP (yes, both are needed) of the current symbol, PHASE3, to the current values of CS:EIP. Along

with this relocation, all other symbols in this segment group are similarly relocated.

At this point, you'll have full symbol and source-level support for this segment group. You can step through this driver at the source level using the `T` (trace) and `J` (jump) commands, and display program variables using symbol names instead of absolute memory addresses.

Now, enter `G` (go command) to resume execution. You'll soon stop at another point within VXD.386. You won't see any source code since you're now in a different segment group. The symbol table is already loaded, so you can just enter `/S CONTROL CS:EIP` to relocate the symbols for this portion of the driver.

Another technique for symbol relocation is to put the offset of the beginning of the current segment group in one register (such as EAX) and put the segment group number in another register (such as BX) before stopping in Periscope with an INT 2 or INT 3. Then enter `/S BX CS:EAX` to relocate the symbols for the segment group.

For best symbol support, be sure to structure your VxD in the order used by the sample program VXD.ASM to get data references to follow code references. Then when using the `/S` command, your code and data symbols will be relocated correctly.

To debug the real-mode portion of a VxD, use Periscope/32 in a single system. It is not possible for Periscope/32 to gain control during Phase 1 (SYS_CRITICAL_INIT) of a VxD. The only method we can offer to debug in this mode is to use Model IV in passive remote mode. Periscope/32 works fine in all modes following SYS_CRITICAL_INIT, including V86 mode in a DOS box.

PERISCOPE/REMOTE FOR WINDOWS

PSRW.386 (PSRW) is the Periscope/Remote for Windows driver. It communicates with Periscope/32 over a serial link to read/write memory, I/O ports, and system registers on the target system.

By default, PSRW hooks interrupts 1 (single step), 2 (NMI), 3 (breakpoint), 6 (illegal instruction), 8 (double fault), C (stack fault), D (general protection fault), and E (page fault).

To load PSRW, see Step 7 in "Installing and Using Periscope/32 for Windows" earlier in this addendum. The options that can be entered in the Windows SYSTEM.INI file (at the end of the line starting with `PSRWOPTIONS`) are:

/2:px—Set COM port.

The port **p** is **1** through **4** or **1** through **8** if you're using a PS/2 system. The speed **x** is **S** (slow—9,600); **M** (medium—38400); or **F** (fast—115,200). The default setting is **/2:1f**, which is port **1** at the **F**ast speed.

When using the debug version of Windows, do not try to use COM port 1 for PSRW, since Windows grabs this port for use by WDEB386. Use COM port 2 instead.

/3—Enable ring 3 debugging.

Use this option if you want to have Periscope debug ring 0 programs and an application-level debugger, such as CodeView for Windows (CVW), debug applications running in other than ring 0. When this option is used, interrupts 1 and 3 from protect mode ring 0 and from V86 mode are handled by Periscope, and these same interrupts from protect mode rings 1 through 3 are handled by the application debugger.

When using CVW, do NOT run the standard Periscope under DOS, since this will cause problems when CVW exits. To activate Periscope while your ring 3 application is running, be sure to use the `/3` command before CVW is started, and use only a break-out switch to activate Periscope. Also, use only the QC command to exit Periscope when the application is in control. This will assure that Periscope does not attempt to use interrupts 1 or 3, which are currently 'owned' by CVW.

Be sure not to enable ring 3 debugging unless an application debugger is present.

/D—Do not allow 386 debug register use.

Use this option to disable Periscope's support of the 386 debug registers.

/H:nnnn—Set a dummy port for Model IV use.

The dummy port is the lower of two I/O addresses used to synchronize Model IV in a remote link. On entry, PSRW reads dummyport+1 and on exit PSRW reads dummyport+0.

/K:nn [nn]—Mask hardware interrupt request (IRQ) lines when Periscope's screen is displayed.

The value of **nn** may be from 0 (zero) to FFH. The first value corresponds to the first 8259 interrupt controller. The optional second value corresponds to the second 8259. The default value is zero, meaning that no IRQ lines are masked. A one bit in **nn** masks the corresponding IRQ line.

For example, to mask the timer (IRQ 0), use **/K:1**. To mask the keyboard (IRQ 1), use **/K:2**. To mask IRQ 1 and IRQ 2, use **/K:6** (bits 1 and 2 on).

/M—The target computer is a Micro Channel system.

Use this option if the target system is an IBM PS/2 or compatible. This option is needed only if you want to use COM ports 5 through 8.

/Q—The target system has a PC (8088) motherboard with an 80386 or later turbo card.

Use this option only if the target system is a PC or XT with a turbo card.

/W:nn—Set the wait count (as a hex number) for Model IV use.

Use this option to help synchronize Periscope Model IV's trace buffer when the host and target systems run at different speeds. The default wait count is 30H, but you may adjust it up or down to synchronize the trace buffer. If you set a hardware breakpoint that does not fill the trace buffer and entries at the top of the buffer are missing, or you miss hardware breaks that are very close to the starting point, increase the wait time. If you see a large number of loop instructions executing inside PSRW at the top of the trace buffer, decrease the wait time.

If the host is faster than the target, set the wait count to 1. If the host is slower than the target, set the wait count higher than 1.

/X:n—Do not hook the indicated interrupt.

Use this option to keep PSRW from hooking any or all of the following interrupts: 2 (NMI), 6 (illegal instruction), 8 (double fault), C (stack fault), and/or D (general protection fault). Try to minimize the use of this option, since the trapping of these interrupts by Periscope can be very valuable.

PSRW must be able to hook interrupts 1 (single step), 3 (code breakpoint), a COM interrupt, and interrupt E (page fault). The first three are presumed to be owned by PSRW, but a page fault not caused by PSRW is passed on through to the original interrupt service routine.

PSRW needs exclusive control over the COM port it is using to communicate with the host system. Since there is no ironclad way to keep Windows from using our COM port, we have implemented a solution suggested by Microsoft — to zap the two bytes in the BIOS data area at 40:0 that indicate the base address of the COM port. This value is restored on exit from Windows. If you know of a more elegant solution, please call Technical Support.

TIPS ON USING PERISCOPE/32 FOR WINDOWS

- To use local variables, run TS.COM with the `/E`, `/32`, and `/V` switches, presuming CodeView symbol information. Once in Periscope, load the symbols using `LS 0 <file>`. Then relocate the segments as needed using Periscope's `/S` command. To generate a .PSS file for use with Periscope/32, be sure to enter the `/32` option when using

TS.

- Avoid the use of local variables for important global data, since you have to be within the scope of the function to access locals.
- Periscope/32 has no support for debugging .RTLink, PLINK, or Microsoft Windows application programs.
- If you're using MASM 5.10B, use the .MAP file for your symbols, since variables from different segment groups will be incorrectly put together in the .EXE file's symbol table. For best results, use PUBLIC declarations for all important variables and procedures. At link time, generate a .MAP file and specify the `/LI` and `/M` switches to the linker.
- If you're using MASM 6.00, you can use CodeView symbols, but note that there are still problems with 32-bit local data variables getting jumbled by the assembler. For best results, use a .MAP file as described above.
- You can use a Break-out Switch on either system to get into Periscope at any time. Also, you can use PSKEY on the host system or WINKEY on the target system to activate Periscope.
- If you have problems with the host system timing out (indicated by an asterisk alternating with an up or down arrow in the upper left hand corner of the screen on the host system), press the Break-out Switch on the target system. Since the Break-out Switch uses NMI, it can get in when something has disabled serial communications. If problems persist, try reducing the speed of the serial link.
- The **Alt-3** and **Alt-R** keystrokes have the same effect—toggling the vertical 386 register display.
- Once the host system has been set up, you can terminate and restart Windows on the target system without restarting Periscope/32 on the host system. If you change the options set by `PSRWOPTIONS` in SYSTEM.INI, you'll need to restart the host system.
- SYMLOAD.COM is not usable in remote mode, only in local (DOS) mode.
- USEREXIT.ASM has changed to support the 32-bit registers. If you use this program, see the latest source code for details. Note that the code that is downloaded to a remote system runs in 32-bit mode.
- To do source level debugging of Windows drivers (.DRV) or DOS device drivers (.SYS), first embed an INT 2 or INT 3 in

the program and compile and link the program with as much symbol information as possible. Then enter `TS <FILENAME> /V/32` to generate a PSS file. When the driver stops in Periscope, use the `LS 0 <FILENAME>` command to load the symbols. Then use the `/S` command to relocate the symbols to the current load address.

USING PERISCOPE/32 FOR WINDOWS WITH PERISCOPE MODEL IV

Beginning with Version 5.4, Periscope/32 (and therefore *Periscope/32 for Windows*, *Periscope/32 for OS/2 2.x*, and the *Periscope 32-bit Toolkit*) supports the Periscope Model IV hardware. Most of the changes made to Periscope/32 to provide this support convert the linear addresses used by the software to the physical addresses used by the hardware, and vice versa.

This section describes how using Model IV is different when you're using Periscope/32 software instead of the "regular" Periscope/EM software as described in the Model IV manual.

Overview

Protect mode operating systems use a memory management scheme that enables them to run several programs at the same time. They use segmentation to keep each application in its own private address space, and they use paging to allow programs to operate as if more memory exists than is physically available. These concepts affect the operation of Periscope/32 and its use of the Model IV Board.

Programs address memory using a linear address. In a paged environment, the CPU converts a 16:32 linear address (selector:offset32) using the page tables to determine the actual physical address. Periscope/32 performs this same task on linear addresses used in conjunction with Model IV operations (specifically, the `HM` breakpoint command and the `/A` trace buffer command).

While a linear address can be converted to just one physical address, a physical address may be represented by many linear addresses. The CPU events captured by the Model IV Board contain physical addresses. In the hardware trace buffer display, these physical addresses are converted into linear addresses using the following rules as a basis.

1. Search for a valid linear address using non-4 gigabyte selectors that are valid for the CPU event (fetch, read, or write) using global selectors first, then local selectors. Privilege level information is not used in determining whether a selector should be searched. Only application selectors are used as a part of the search. Of course, the selector must be present (currently active) to be included in the search.

2. If no match is found, use a 4-gigabyte selector that is valid for the CPU event to determine the linear address.

3. If no match is found, display the address as a physical address.

Because there is not a one-to-one relationship, you may see an address displayed in the hardware trace buffer that may have used a different 16:32 memory address at the time it was executed.

Periscope/32 uses the following styles of addresses:

`0:yyyyy`	V86 address, where `yyyyy` is the real-mode address
`xxxx|yyyy`	real-mode style address conversion, where `xxxx` is the segment and `yyyy` is the offset
`yyyy,yyyy`	physical address, where `yyyy,yyyy` is the 32-bit physical location
`xxxx:yyyy,yyyy`	16:32 linear address, where `xxxx` is the selector and `yyyy,yyyy` is the32-bit offset

Command differences

`GH, GM` — converts the breakpoint linear address range into the current physical memory locations.

`HD DW xxxx:yyyy` — becomes `HD DW xxxx,yyyy`

`HM` — accepts address styles shown above; must use the same selector for the beginning and ending address; must use the same address format for the beginning and ending address (or use an `L` value for the ending address); ending addresses that exceed the size of the selector will be limited to the maximum valid address.

`HR, HT, HU <file>` — uses the current selector settings to decode the physical addresses found in the trace buffer.

Hardware trace buffer differences

The individual columns in the hardware trace buffer displayed by Periscope/32 are the same as displayed by Periscope/EM except that the address column is larger to accommodate the 16:32 memory model. The trace buffer does not attempt to display the reading or writing of interrupt address locations.

The hardware trace buffer display decodes instructions as 16-bit or 32-bit based upon the selector used as the decoded linear address. This should eliminate many cases where you had to tell the trace buffer display which mode to use. There will still be a need for manually setting the disassembly mode, usually when a 4-gigabyte selector (32-bit) is used to decode instructions that were a part of a V86 process (16-bit). When you use `@16` or `@32` to force 16-bit or 32-bit disassembly, the selector-based decoding of instructions is disabled.

Trace buffer command differences

`/A` — accepts address styles shown above; must use the same selector for the beginning and ending address; must use the same address format for the beginning and ending address (or use an `L` value for the ending address); ending addresses that exceed the size of the selector will be limited to the maximum valid address; attempts to display a "found" address using the selector given if possible.
In a protect mode memory management scheme, you can have physical memory mapped into multiple selectors of different types (code and data). When you use `/A` to search for an address range, the address range is converted to a physical address range(s). When a match is found in the hardware trace buffer, the physical address is displayed with a linear address using the selector used in the `/A` command, if possible. However, if the selec-

tor used has a type which is invalid for the CPU event in the trace buffer, a valid selector will be chosen to display the address of the CPU event. Therefore, it can appear that the `/A` search stopped at an incorrect address.

`E` — Not available in Periscope/32 (toggle extended memory display).

`F` — Not available in Periscope/32 (fixup addresses).

`S, S+` — displays addresses using the 16:32 memory model.

S- displays physical addresses.

`S <segment>` can be used to temporarily override the selector used to display addresses in the trace buffer.

`S+ <segment>` and `S- <segment>` are no longer valid.

Error differences

03—Missing segment

The `HM` command and the `/A` trace buffer command require the beginning and ending addresses to use the same selector.

13—Too many breakpoints

The `HM` command converts each breakpoint linear address range into one or more contiguous physical address ranges. A breakpoint is set for each contiguous physical address range. If this error occurs, try limiting the linear address range.

15—Address range is too big

The `HM` command and the `/A` trace buffer command convert a contiguous linear address range into one or more contiguous physical address ranges. The number of contiguous physical address ranges exceeds the capacity of the Model IV board or the `/A` internal buffers. If this error occurs, try limiting the linear address range.

36—Linear to physical error

Either no physical address currently exists for the linear address specified or a selector was specified that is not currently in use.

Tips and caveats

- Memory paging can cause a linear address range to be fragmented into multiple physical memory pages. Both the `HM` breakpoint and `/A` trace buffer commands consolidate contiguous memory pages into a single range. However, it is possible to exceed the limits of both commands. When this happens, you will receive Error 15 or Error 13 (see above). If this happens, decrease the size of your linear range by at least 4K (the size of a page) until the error disappears.
- Because a physical address may be represented by multiple virtual addresses, you may see an address in the trace buffer that uses the "wrong" selector. This "wrong" selector may indicate that code is 16-bit when in fact it should be 32-bit code. This can cause the disassembled instructions to appear to be non-sensible since the opcodes are decoded differently in 16-bit and 32-bit. If this happens, use the `S <segment>` command if you know the address of the correct selector. The trace buffer will redisplay using virtual addresses and decoded instructions based on the selector you have chosen (if possible). This method is preferable to the use of `@16` or `@32` to override the instruction decoding because the entire trace buffer will be decoded using 16-bit or 32-bit instructions.
- When Model IV triggers on a hardware breakpoint, the trace buffer stops capturing information immediately, leaving the trigger event as the last item in the buffer (assuming the default hardware control `HC TB` was set). If something other than a hardware breakpoint (such as a code breakpoint or an exception interrupt) activates Periscope, the board continues capturing information until Periscope disarms it. In a single-system setup, this happens quite quickly, leaving only a few 'garbage' cycles in the trace buffer. In remote mode, however, the time span can be quite long, since PSRW gets the wake up call and passes the information back to Periscope over the serial link. This can cause the trace buffer to be filled with code of PSRW waiting on the serial link.

Fortunately, we can use the hardware breakpoints to quit capturing information much earlier. Doing this relies on PSRW's access to some dummy I/O ports and on the corresponding hardware breakpoints you set. The `PSRW /H` option sets the value of the dummy ports. PSRW reads the second port on entry and reads the first port on exit. Using this information, we can set some hardware breakpoints with Model IV's sequential triggers to capture just what we want. Assuming a command-line option of `/H:F300,` the command sequence would be:

Command 1: `HC X0`
This command excludes capturing trace buffer information when Model IV is in state 0.

Command 2: `HP F300 I (0,1)`
This command switches Model IV from state 0 to state 1 when the dummy port is read on exit from PSRW. This starts capturing information in the trace buffer.

Command 3: `HP F301 I (1,!)`
This command causes a hardware breakpoint on entry into PSRW when port F301 is read, and stops capturing information in the trace buffer.

Any hardware breakpoints you set would have a start state of 1. If you need multiple states, change the start state for Command 3 and the end state for Command 2 to the appropriate values.

A D D E N D U M

Periscope/32 for OS/2

This addendum describes how you can use *Periscope/32 for OS/2* to debug system software at the source level in an OS/2 2.x or later (including Warp) environment. Please refer to the Periscope manual for information on Periscope/32, the host software included in the *Periscope/32 for OS/2* package. If you're using the Periscope Model IV hardware, see also the Model IV manual for additional information.

INTRODUCTION AND OVERVIEW

Periscope/32 for OS/2 enables you to debug OS/2 physical and virtual device drivers and other system software running in an OS/2 2.x or later system.

Two systems are required: a host DOS system where Periscope/32 runs, and a target OS/2 system where Periscope/Remote for OS/2 runs.

Periscope/32 for OS/2 (this package) contains:

- a disk with both the Periscope/32 software and the Periscope/Remote for OS/2 software
- an optional Break-out (NMI) Switch
- the Periscope manual and quick-reference card
- this addendum

To run *Periscope/32 for OS/2,* you need:

- two 80386 or later computers
- the Periscope/32 software (Version 5.40 or later if you're using Model IV hardware)
- the Periscope/Remote for OS/2 software (Version 5.40 or later if you're using Model IV hardware)
- a null-modem cable
- (optional) a Model IV Board, Pod, NMI clip, and shielded cable for Model IV hardware support

How Periscope/32 for OS/2 works.

The *host system* is the DOS system where the Periscope/32 software runs. If you're using the Model IV hardware, you install the Model IV Board in this system. The *target system* is the OS/2 system where the Periscope/Remote for OS/2 software and the program you're debugging runs. If you're using the Model IV hardware, you install the Pod and NMI Clip in this system.

PSR2.SYS (PSR2) is the Periscope/Remote for OS/2 software. It is a small device driver that runs in the target system. It communicates with Periscope/32 on the host system via a serial link. This serial link requires an available COM port on both systems. On ISA- or EISA-bus systems, this can be COM ports 1 through 4, and on MCA-bus systems, it can be COM ports 1 through 8. The serial communications can occur at one of three speeds: 9,600 baud (Slow); 38,400 baud (Medium); or 115,200 baud (Fast).

You use a null-modem cable to link the host system to the target system. If you have version 3.00 or later of the Brooklyn Bridge or LapLink, you can use the cable that comes with either of these products. If you want to make your own cable, see the specifications in the NOTES.TXT file in the Periscope directory.

If you're using Model IV, you connect the Model IV Board in the host system with the Pod in the target system using the 48-inch Shielded Ribbon Cable.

When you run PSR2 in the target system and link the host and target systems via the serial link described above, Periscope/32 runs in *active remote mode*. In this mode, you use the host system as a debugging terminal to interactively debug software running on the target system.

To use *Periscope/32 for OS/2*, you need these items in the host system:

- DOS 3.1 or later

You cannot use the OS/2 DOS box—it is not compatible enough. See the README.TXT file for details.

- Periscope/32 software (Version 5.40 or later if you're using Model IV hardware)
- your program's source and symbol files
- an available COM port connected to a null-modem cable
- (optional) a Model IV Board for Model IV hardware support

You need these items in the target system:

- OS/2 2.x or later (including Warp)
- Periscope/Remote for OS/2 (Version 5.40 or later if you're using Model IV hardware)
- your program's executable code
- an available COM port connected to a null-modem cable
- (optional) a Pod and NMI Clip for Model IV hardware support

INSTALLING PERISCOPE/32 FOR OS/2

The following instructions provide details for installing *Periscope/32 for OS/2.* You will install and load symbols for the sample driver PAUSE.SYS as part of the installation process.

Step 1—HOST AND/OR TARGET: Install the hardware.

If you're using the Model IV hardware, install it per the instructions in the Model IV manual.

➲ Be sure to install the NMI Clip in the target system per Step 9 of the installation instructions in Section 3.4.5 of the Model IV manual.

If you're not using the Model IV hardware, you can install the Break-out (NMI) Switch in either the host or the target system. See Chapter 4 in the Periscope manual for installation instructions.

➲ If you have a Model I Board, you can install it in either the host or target system so you can use the plug-in Break-out Switch. *Periscope/32 for OS/2* does not use the memory on the board, however.

Step 2—HOST AND TARGET: Hook up the null-modem cable to connect port 1, then test the serial link.

Be sure that the cable meets the specifications described in the file NOTES.TXT. Note that the motherboard COM port on some IBM PS/2 systems does not run reliably at the fast speed. Use the medium speed option as needed.

To test the serial link, insert the distribution disk in drive A and enter (on both host and target systems):

```
COPY A:\PSR2\PSTERM.COM C:
PSTERM /T
```

PSTERM should display `No errors detected` after running its tests.

Step 3—HOST: Configure Periscope/32.

Place the distribution disk in drive A and run the setup program by entering `A:SETUP`. You will be prompted for a directory name for Periscope. You may enter a name or press enter to create a directory named PERI32 to contain the needed files. Once the files have been created, SETUP will display the configuration screen. This screen looks like the configuration screen shown in Chapter 3 of the Periscope manual.

If you're not using the Model IV hardware, enter either `1` or `2` for this option. Enter `3` if you are using the Model IV hardware.

In remote mode, Periscope/32 loads into conventional memory on the host. It uses neither EM memory nor the Model I Board's memory regardless of which model you choose.

See the Periscope manual for details on the other options. Press **F10** when you have completed all options.

Step 4—HOST: Copy the source and .MAP files for the sample driver.

So that you can debug the sample driver at source-level, place the distribution disk in drive A and enter

```
COPY A:\PSR2\PAUSE.ASM C:\PERI32
COPY A:\PSR2\PAUSE.MAP C:\PERI32
```

to copy the source and .MAP files to the Periscope directory (substitute the directory name you entered in Step 3 for PERI32 if different) on the host system.

Step 5—HOST: Run the symbol table program (TS) to extract the symbol information from the .MAP or .EXE file and write a Periscope symbol (.PSS) file.

Enter `TS PAUSE/V/32` to get verbose mode and 32-bit offset support. (Although we've converted TS to support 32-bit CodeView information, there are problems with local code and data variables as generated by MASM 5.1 0B and MASM 6.00, so you should use a .MAP file instead of an .EXE file if possible.)

Step 6—HOST: Enter `PSKEY LR` to set up the left and right shift keys as hotkeys. If you're using the Model IV hardware, enter `PSKEY LR3` instead.

Step 7—TARGET: Setup Periscope/Remote for OS/2 (PSR2).

Create a directory named PERI32 on drive C on the target system. Place the distribution disk in drive A and enter `COPY A:\PSR2\*.SYS C:\PERI32`. Next, use an editor to add the following lines to the OS/2 CONFIG.SYS file (none of the following text or options is case sensitive):

```
DEVICE=C:\PERI32\COMBLOCK.SYS 1
DEVICE=C:\PERI32\PSR2.SYS /2:1F/I
DEVICE=C:\PERI32\PSR2A.SYS
DEVICE=C:\PERI32\PAUSE.SYS
```

The file COMBLOCK.SYS blocks the use of a COM port by OS/2 by blocking use of the name of the port (COM1, COM2, etc.). Its use is mandatory. The two possible command-line arguments are a COM port number from `1` to `8` (the default is `1`) and `/S` for silent operation (in which case no messages are displayed).

The file PSR2.SYS is the Periscope/Remote for OS/2 driver. The options shown above indicate the use of COM port 1 at the fast speed and that PSR2.SYS will inquire if it is to be loaded. See the section later in this addendum for details on the options available. Be sure to match speeds on both host and target systems.

The file PSR2A.SYS is used to complete the installation of PSR2. It should immediately follow the PSR2 driver. The two possible command-line arguments are /S for silent operation (in which case no messages are displayed) and /! to stop Periscope as soon as possible.

The file PAUSE.SYS is a sample driver that illustrates the use of source-level and symbolic debugging. It can be deleted after you've become familiar with the use of *Periscope/32 for OS/2*.

Step 8—HOST: Load Periscope/32.

Load Periscope/32 on the host system by entering `PS /2:1F`. Once loaded, Periscope/32 will display the message `Waiting for remote system...`

The installation options for Periscope described in Chapter 5 of the Periscope manual also apply to Periscope/32. The following option (**/2:px** in Section 5.2 of the Periscope manual) applies specifically to using Periscope/32 in active remote mode:

/2:px—Set COM port.

The port **p** is **1** through **4** or **1** through **8** if you're using a PS/2 system. The speed **x** is **S** (slow—9600); **M** (medium—38400); or **F** (fast—115,200). PSR2 (in the target) defaults to COM port 1 at the Fast speed, but Periscope/32 (in the host) has no default for COM port or speed. You must specify the **/2:px** option for Periscope/32 to run in remote mode. Be sure to match speeds on both host and target systems. If a port or speed mismatch occurs when Periscope/32 is loaded, you'll need to restart both systems.

Step 9—TARGET: Start OS/2 on the target system.

When OS/2 is ready to load PSR2, it will prompt you whether to load the driver. To skip the loading of PSR2, answer `N` to this prompt. Before answering `Y`, be sure that Periscope/32 is loaded and waiting for the target system to respond (Step 8 above).

Step 10—HOST: Load symbols for the sample driver.

After loading PSR2, OS/2 will load the sample driver PAUSE.SYS. We've embedded an INT 3 in PAUSE.SYS to wake up Periscope. Either an INT 2 (software NMI) or an INT 3 (code breakpoint) can be used to easily wake up Periscope.

When Periscope/32's screen is displayed you won't see symbols or source code yet since no symbol table is loaded.

To load the symbols for PAUSE.SYS, enter `LS 0 PAUSE`, specifying any needed path name for the third argument. To relocate the symbols for the code selector, enter `/S S.CSTART CS`. To relocate the symbols for the data selector, enter `/S S.DSTART DS`.

At this point, you'll have full symbol and source-level support for PAUSE.SYS. You can step through this driver at the source level using the `T` (trace) and `J` (jump) commands, and display program variables using symbol names instead of absolute memory addresses.

When you're done tracing through PAUSE.SYS, enter the `G` (go) command to continue loading OS/2.

You can activate Periscope at any time by pressing the Break-out Switch or by pressing the hotkeys on the host system.

PERISCOPE/REMOTE FOR OS/2

PSR2.SYS (PSR2) is the Periscope/Remote for OS/2 driver. It communicates with Periscope/32 over a serial link to read/write memory, I/O ports, and system registers on the target system.

By default, PSR2 hooks interrupts 1 (single step), 2 (NMI), 3 (breakpoint), 6 (illegal instruction), 8 (double fault), C (stack fault), D (general protection fault), and E (page fault).

To load PSR2, see Step 7 in 'Installing and Using Periscope/32 for OS/2' earlier in this addendum. The options that can be entered in the OS/2 CONFIG.SYS file (at the end of the line starting with `device=psr2.sys`) are:

/2:px—Set COM port.

The port **p** is **1** through **4** or **1** through **8** if you're using a

PS/2 system. The speed **x** is **S** (slow—9,600); **M** (medium—38400); or **F** (fast—115,200). The default setting is **/2:1f**, which is port **1** at the **F**ast speed.

When using the OS/2 debugging kernel, be aware that it will use COM port 1 if only one COM port is in the system. If two or more COM ports are available, the debugging kernel will use COM port 2.

/3—Enable ring 3 debugging.

Use this option if you want to have an application-level debugger such as IBM's IPMD debug applications running in ring 3 and Periscope debug everything else (code running in rings 0, 1 and 2 as well as V86 mode). When this option is used, interrupts 1, 3, 6 and 0dh occurring in protect mode ring 3 are handled by the application debugger. All other interrupts are handled by Periscope.

To activate Periscope while your ring 3 application is running, be sure to use the `/3` command before the application debugger is started, and use only a Break-out Switch or PSKEY's hotkeys to activate Periscope. Also, use only the `QC` command to exit Periscope when executing code in ring 3. This will assure that Periscope does not attempt to use interrupts 1 or 3, which are currently 'owned' by the application debugger.

➔ Be sure not to enable ring 3 debugging unless an application debugger is present. There is no way for Periscope to determine whether an application debugger is present, so it can't provide a safety net.

/D—Do not allow 386 debug register use.

Use this option to disable Periscope's support of the 386 debug registers.

/H:nnnn—Set a dummy port for Model IV use.

The dummy port is the lower of two I/O addresses used to synchronize Model IV in a remote link. On entry, PSR2 reads dummyport+1 and on exit PSR2 reads dummyport+0.

/I—Inquire as to whether PSR2 should install itself.

/K:nn [nn]—Mask hardware interrupt request (IRQ) lines when Periscope's screen is displayed.

The value of **nn** may be from 0 (zero) to FFH. The first value corresponds to the first 8259 interrupt controller. The optional second value corresponds to the second 8259. The default value is zero, meaning that no IRQ lines are masked. A one bit in **nn** masks the corresponding IRQ line.

For example, to mask the timer (IRQ 0), use **/K:1**. To mask the keyboard (IRQ 1), use **/K:2**. To mask IRQ 1 and IRQ 2, use **/K:6** (bits 1 and 2 on).

/M—The target computer is a Micro Channel system.

Use this option if the target system is an IBM PS/2 or compatible. This option is needed only if you want to use COM ports 5 through 8.

/Q—The target system has a PC (8088) motherboard with an 80386 or later turbo card.

Use this option only if the target system is a PC or XT with a turbo card.

/S—Silent operation.

Use this option to keep PSR2.SYS from displaying any messages on the screen. This option is typically used to keep PSR2 from using video services when you're debugging a video device driver. Since both COMBLOCK.SYS and PSR2A.SYS also display messages on the screen, you'll want to use this same option for those two drivers as well. When silent operation is in effect, PSR2.SYS uses beeps to indicate any errors. The beep codes are as follows:
2 beeps—invalid command-line option
3 beeps—enable to allocate GDT entries
4 beeps—unable to initialize COM port

/W:nn—Set the wait count (as a hex number) for Model IV use.

Use this option to help synchronize Periscope Model IV's trace buffer when the host and target systems run at different speeds. The default wait count is 30H, but you may adjust it up or down to synchronize the trace buffer. If you set a hardware breakpoint that does not fill the trace buffer and entries at the top of the buffer are missing, or you miss hardware breaks that are very close to the starting point, increase the wait time. If you see a large number of loop instructions executing inside PSR2 at the top of the trace buffer, decrease the wait time.

If the host is faster than the target, set the wait count to 1. If the host is slower than the target, set the wait count higher than 1.

/X:n—Do not hook the indicated interrupt.

Use this option to keep PSR2 from hooking any or all of the following interrupts: 2 (NMI), 6 (illegal instruction), 8 (double fault), C (stack fault), and/or D (general protection fault). Try to minimize the use of this option, since the trapping of these interrupts by Periscope can be very valuable.

PSR2 must be able to hook interrupts 1 (single step), 3 (code breakpoint), a COM interrupt, and interrupt E (page fault). The first three are presumed to be owned by PSR2, but a page fault not caused by PSR2 is passed on through to the original interrupt service routine.

TIPS ON USING PERISCOPE/32 FOR OS/2

- To use local variables, run TS.COM with the `/E`, `/32`, and `/V` switches, presuming CodeView symbol information. To generate a .PSS file for use with Periscope/32, be sure to enter the `/32` option when using TS.
- Avoid the use of local variables for important global

data, since you have to be within the scope of the function to access locals.

- If you're using MASM 6.00, you can use CodeView symbols, but note that there are still problems with 32-bit local data variables getting jumbled by the assembler. For best results, use a .MAP file.
- You can use a Break-out Switch on either system to get into Periscope at any time. Also, you can use PSKEY on the host system to activate Periscope.
- If you have problems with the host system timing out (indicated by an asterisk alternating with an up or down arrow in the upper left hand corner of the screen on the host system), press the Break-out Switch on the target system. Since the Break-out Switch uses NMI, it can get in when something has disabled serial communications. If problems persist, try reducing the speed of the serial link.
- The **Alt-3** and **Alt-R** keystrokes have the same effect—toggling the vertical 386 register display.
- Once the host system has been set up, you can terminate and restart OS/2 on the target system without restarting Periscope/32 on the host system. If you change the options for PSR2.SYS in CONFIG.SYS, you'll need to restart the host system.
- SYMLOAD.COM is not usable in remote mode, only in local (DOS) mode.
- USEREXIT.ASM has changed to support the 32-bit registers. If you use this program, see the latest source code for details. Note that the code that is downloaded to a remote system runs in 32-bit mode.
- To do source level debugging of OS/2 physical or virtual device drivers, first embed an INT 2 or INT 3 in the program and compile and link the program with as much symbol information as possible. Then enter `TS <file> /V/32` to generate a .PSS file. When the driver stops in Periscope, use the `LS 0 <file>` command to load the symbols. Then use the `/S` command to relocate the symbols to the current load address.
- Periscope can be used after fatal faults that shut down OS/2. For example, if you get one of the 'system is stopped' types of faults, just press the Break-out Switch or hotkeys to activate Periscope. Then you can look around to see what caused the fault.
- To debug a COM file running in the DOS box, enter `LS 0 <file>` to load the program's symbol table. Then

enter `/S 0 CS|0` to relocate the symbols to the V86-style (0:xxxxx) addressing. Finally, use the `UV` command to toggle the disassembly display to the V86-style addressing. If you prefer the standard CS:IP display, you can use `LS CS <file>` to load the program's symbol table relative to CS and omit the use of the `UV` command. For an .EXE file, load the symbols relative to the program's PSP+10 instead of CS.

USING PERISCOPE/32 FOR OS/2 WITH PERISCOPE MODEL IV

Beginning with Version 5.40, Periscope/32 (and therefore *Periscope/32 for OS/2*, *Periscope/32 for Windows*, and the *Periscope 32-bit Toolkit*) supports the Periscope Model IV hardware. Most of the changes made to Periscope/32 to provide this support convert the linear addresses used by the software to the physical addresses used by the hardware, and vice versa.

This section describes how using Model IV is different when you're using Periscope/32 software instead of the "regular" Periscope/EM software as described in the Model IV (M50b) manual.

Overview

Protect mode operating systems use a memory management scheme that enables them to run several programs at the same time. They use segmentation to keep each application in its own private address space, and they use paging to allow programs to operate as if more memory exists than is physically available. These concepts affect the operation of Periscope/32 and its use of the Model IV Board.

Programs address memory using a linear address. In a paged environment, the CPU converts a 16:32 linear address (selector:offset32) using the page tables to determine the actual physical address. Periscope/32 performs this same task on linear addresses used in conjunction with Model IV operations (specifically, the `HM` breakpoint command and the `/A` trace buffer command).

While a linear address can be converted to just one physical address, a physical address may be represented by many linear addresses. The CPU events captured by the Model IV Board contain physical addresses. In the hardware trace buffer display, these physical addresses are converted into linear addresses using the following rules as a basis.

1. Search for a valid linear address using non-4 gigabyte selectors that are valid for the CPU event (fetch, read, write) using global selectors first, then local selectors. Privilege level information is not used in determining whether a selector should be searched. Only application selectors are used as a part of the search. Of course, the selector must be present (currently active) to be included in the search.

2. If no match is found, use a 4-gigabyte selector that is valid for the CPU event to determine the linear address.

3. If no match is found, display the address as a physical address.

Because there is not a one-to-one relationship, you may see an address displayed in the hardware trace buffer that may have used a different 16:32 memory address at the time it was executed.

Periscope/32 uses the following styles of addresses:

`0:yyyyy`	V86 address, where `yyyyy` is the real-mode address
`xxxx|yyyy`	real-mode style address conversion, where `xxxx` is the segment and `yyyy` is the offset
`yyyy,yyyy`	physical address, where `yyyy,yyyy` is the 32-bit physical location
`xxxx:yyyy,yyyy`	16:32 linear address, where `xxxx` is the selector and `yyyy,yyyy` is the 32-bit offset

Command differences

`GH`, `GM` – converts the breakpoint linear address range into the current physical memory locations.

`HD DW xxxx:yyyy` – becomes `HD DW xxxx,yyyy`

`HM` – accepts address styles shown above; must use the same selector for the beginning and ending address; must use the same address style for the beginning and ending address (or use an `L` value for the ending address); ending addresses that exceed the size of the selector will be limited to the maximum valid address.

`HR`, `HT`, `HU <file>` – uses the current selector settings to decode the physical addresses found in the trace buffer file.

Hardware trace buffer differences

The individual columns in the hardware trace buffer displayed by Periscope/32 are the same as displayed by Periscope/EM except that the address column is larger to accommodate the 16:32 memory model. The trace buffer does not attempt to display the reading or writing of interrupt address locations.

The hardware trace buffer display decodes instructions as 16-bit or 32-bit based upon the selector used as the decoded linear address. This should eliminate many cases where you had to tell the trace buffer display which mode to use. There will still be a need for manually setting the disassembly mode, usually when a 4-gigabyte selector (32-bit) is used to decode instructions that were a part of a V86 process (16-bit). When you use `@16` or `@32` to force 16-bit or 32-bit disassembly, the selector-based decoding of instructions is disabled.

Trace buffer command differences

`/A` — accepts address styles shown above; must use the same selector for the beginning and ending address; must use the same address style for the beginning and ending address (or use an `L` value for the ending address); ending addresses that exceed the size of the selector will be limited to the maximum valid address; attempts to display a "found" address using the selector given if possible.

In a protect mode memory management scheme, you can have physical memory mapped into multiple selectors of different types, code and data. When you use `/A` to search for an address range, the address range is converted to a physical address range(s). When a match is found in the hardware trace buffer, the physical address is displayed with a linear address using the selector used in the `/A` command, if possible. However, if the selector used has a type which is invalid for the CPU event in the trace buffer, a valid selector will be chosen to display the address of the cpu event. Therefore, it can appear that the `/A` search stopped at an incorrect address.

`E` — Not available in Periscope/32 (toggle extended memory display).

`F` — Not available in Periscope/32 (fixup addresses).

`S`, `S+` — Display addresses using the 16:32 memory model.

`S-` — Displays physical addresses.

`S <segment>` — Can be used to temporarily override the selector used the display addresses in the trace buffer.

`S+ <segment>` and `S- <segment>` — Not valid.

Error differences

03—Missing segment

The `HM` command and the `/A` trace buffer command require the beginning and ending addresses to use the same selector.

13—Too many breakpoints

The `HM` command converts each breakpoint linear address range into one or more contiguous physical address ranges. A breakpoint is set for each contiguous physical address range. If this error occurs, try limiting the linear address range.

15—Address range is too big

The `HM` command and the `/A` trace buffer command convert a contiguous linear address range into one or more contiguous physical address ranges. The number of contiguous physical address ranges exceeds the capacity of the Model IV board or the `/A` internal buffers. If this error occurs, try limiting the linear address range.

36—Linear to physical error

Either no physical address currently exists for the linear address specified or a selector was specified that is not currently in use.

Tips and caveats

- Memory paging can cause a linear address range to be fragmented into multiple physical memory pages. Both the `HM` breakpoint and `/A` trace buffer commands consolidate contiguous memory pages into a single range. However, it is possible to exceed the limits of both commands. When this happens, you will receive Error 15 or Error 13 (see above). If this happens, decrease the size of your linear range by at least 4K (size of a page) until the error disappears.
- Because a physical address may be represented by multiple virtual addresses, you may see an address in the trace buffer that uses the "wrong" selector. This "wrong" selector may indicate that code is 16-bit when in fact it should be 32-bit code. This can cause the disassembled instructions to appear to be non-sensible since the opcodes are decoded differently in 16-bit and 32-bit. If this happens, use the `S <segment>` command if you know the value of the correct selector. The trace buffer will redisplay using virtual addresses and decoded instructions based on the selector you have chosen (if possible). This method is preferable to the use of `@16` or `@32` to override the instruction decoding because the entire trace buffer will be decoded using 16-bit or 32-bit instructions.
- When Model IV triggers on a hardware breakpoint, the trace buffer stops capturing information immediately, leaving the trigger event as the last item in the buffer

(assuming the default hardware control `HC TB` was set). If something other than a hardware breakpoint (such as a code breakpoint or an exception interrupt) activates Periscope, the board continues capturing information until Periscope disarms it. In a single-system setup, this happens quite quickly, leaving only a few 'garbage' cycles in the trace buffer. In remote mode, however, the time span can be quite long, since PSR2 gets the wake up call and passes the information back to Periscope over the serial link. This can cause the trace buffer to be filled with code of PSR2 waiting on the serial link.

Fortunately, we can use the hardware breakpoints to quit capturing information much earlier. Doing this relies on PSR2's access to some dummy I/O ports and on the corresponding hardware breakpoints you set. The `PSR2 /H` option sets the value of the dummy ports. PSR2 reads the second port on entry and reads the first port on exit. Using this information, we can set some hardware breakpoints with Model IV's sequential triggers to capture just what we want. Assuming a command-line option of `/H:F300`, the command sequence would be:

Command 1: `HC X0`
This command excludes capturing trace buffer information when Model IV is in state 0.

Command 2: `HP F300 I (0,1)`
This command switches Model IV from state 0 to state 1 when the dummy port is read on exit from PSR2. This starts capturing information in the trace buffer.

Command 3: `HP F301 I (1,!)`
This command causes a hardware breakpoint on entry into PSR2 when port F301 is read, and stops capturing information in the trace buffer.

Any hardware breakpoints you set would have a start state of 1. If you need multiple states, change the start state for Command 3 and the end state for Command 2 to the appropriate values.

- Tracing of interrupts in V86 mode can result in erratic behavior. Evidently, OS/2's trap 0dh handler (which gets control on an interrupt from V86 mode) doesn't work correctly when the single-step flag is on. We're pursuing resolution on this problem with IBM. For now, you can't trace into an INT 21h in V86 mode since OS/2 sets it up as a trap at 0:0. Also, don't trace an interrupt that's not in the IDT (0-1fh, 50h-57h, and 70h-77h). Use Periscope's `J` or `TI` commands to step over the interrupt.

- The `GA` and `GT` commands do not work well with OS/2 because OS/2 tends to generate spurious INT 2's sometimes after the execution of INT 1. Since these commands use lots of INT 1's, you're likely to get a false stop in Periscope, since it interprets an INT 2 as a 'stop now' interrupt. Even when INT 2 is not hooked by PSR2.SYS, there are still problems, likely related to the trap 0dh problems mentioned above.
- We cannot certify Periscope or Periscope/32 for use in the OS/2 DOS box, due to significant incompatibilities with a true DOS. Please see the README.TXT file for details.