



US005764892A

United States Patent [19]

Cain et al.

[11] **Patent Number:** 5,764,892
[45] **Date of Patent:** Jun. 9, 1998

[54] SECURITY APPARATUS AND METHOD

[75] Inventors: **Fraser Cain; Christian Cotichini**, both of Vancouver; **Thanh Cam Nguyen**, New Westminster, all of Canada

[73] Assignee: **Absolute Software**, Vancouver

[21] Appl. No.: **558,432**

[22] Filed: **Nov. 15, 1995**

Related U.S. Application Data

- [63] Continuation-in-part of Ser. No. 339,978, Nov. 15, 1994.
[51] Int. Cl.⁶ **H04M 3/00; G08B 13/22**
[52] U.S. Cl. **395/200.3; 340/825.34; 340/825.36; 340/571; 340/572; 364/222.5; 379/106**
[58] **Field of Search** 364/514 R, 222.5, 364/286.4, 709.05; 379/37, 39, 40, 45, 56, 57, 58, 95, 93, 106, 142; 342/450, 451; 307/10.2; 326/8; 180/287; 116/5, 6, 75; 340/426, 506, 503, 571, 572, 568, 825.44, 825.36, 825.34, 825.54, 825.48, 825.52; 380/4, 23, 25; 395/200.3, 200.31, 200.47, 200.49, 200.57

[56] References Cited

U.S. PATENT DOCUMENTS

- 3,925,763 12/1975 Wadhvani et al. 340/164 R
4,999,621 3/1991 Loeb 340/825.34

- 5,077,788 12/1991 Cook et al. 379/142
5,511,109 4/1996 Hartley et al. 379/40
5,566,339 10/1996 Perholtz et al. 395/750

FOREIGN PATENT DOCUMENTS

0588519 8/1993 European Pat. Off. .

Primary Examiner—James P. Trammell

Assistant Examiner—Tony M. Cole

Attorney, Agent, or Firm—Irell & Manella LLP

[57] ABSTRACT

A system for locating and monitoring electronic devices utilizing a security system that is secretly and transparently embedded within the software, firmware, or hardware of a computer. This security system causes the client computer to periodically and conditionally call a host system to report its serial number via an encoded series of dialed numbers. A host monitoring system receives calls from various clients and determines which calls to accept and which to reject. This determination is made by comparing the decoded client serial numbers with a predefined and updated list of numbers corresponding to reported stolen computers. Only calls from clients on the predefined list are accepted. The host also concurrently obtains the caller ID of the calling client to determine the physical location of the client computer. The caller ID, indicating the physical location of the stolen device, and the serial number are subsequently transmitted to a notifying station in order to facilitate the recovery of the stolen device. The security system remains hidden from the user, and actively resists attempts to disable it.

47 Claims, 18 Drawing Sheets

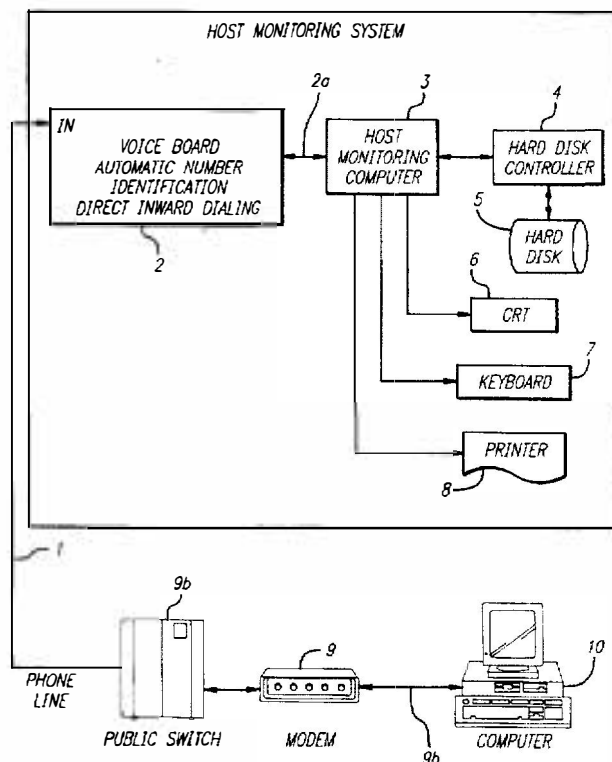
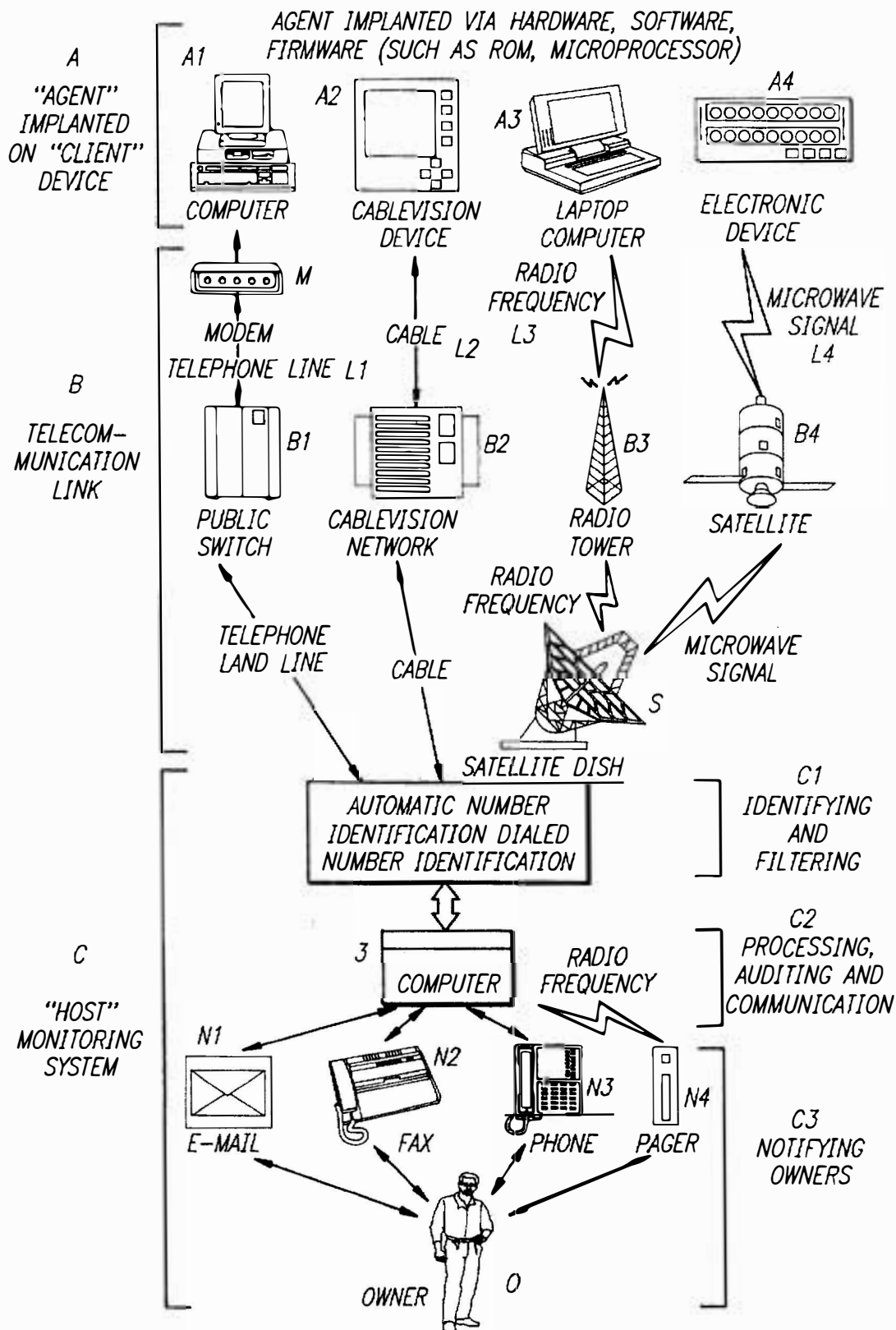


FIG. 1



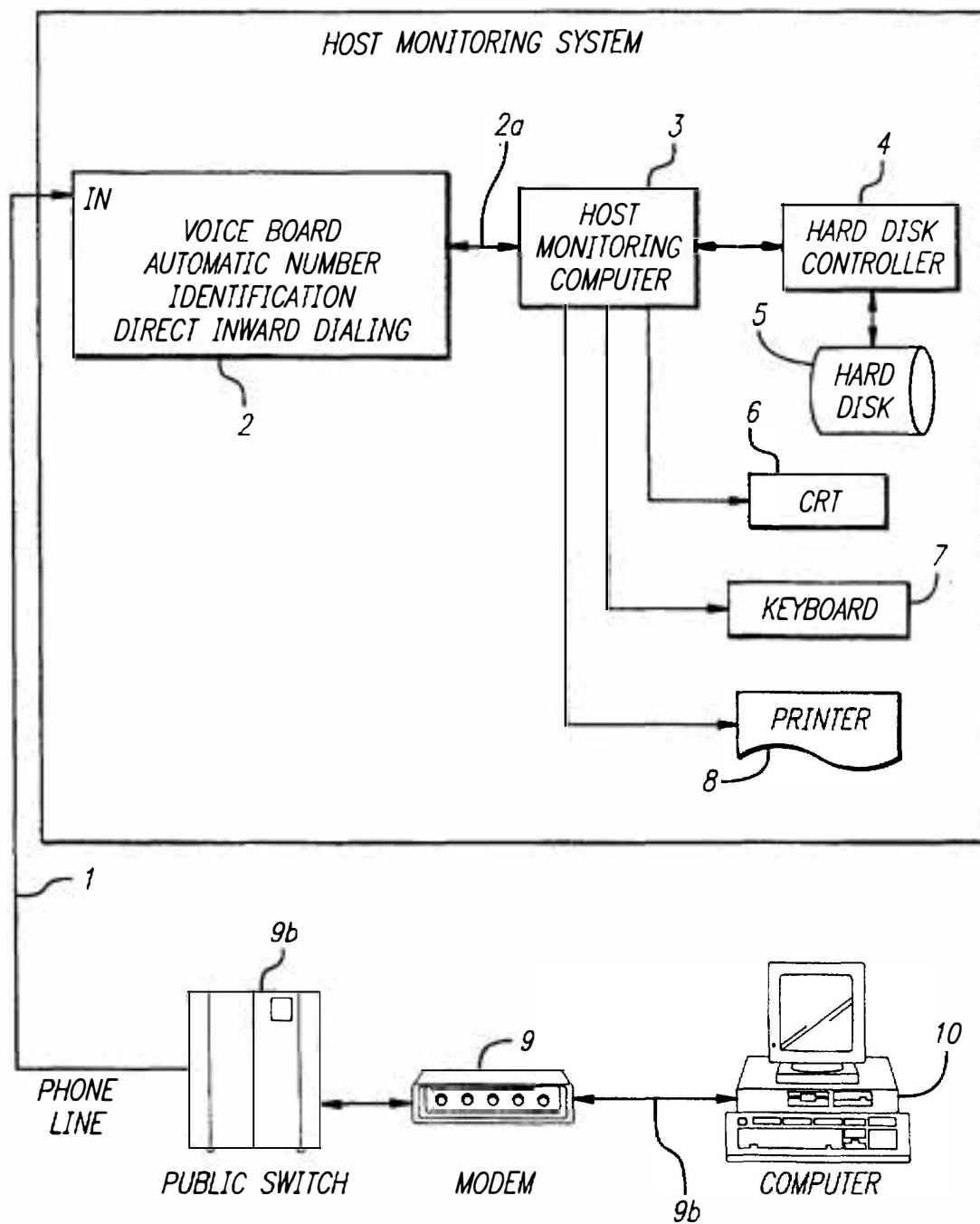


FIG. 2

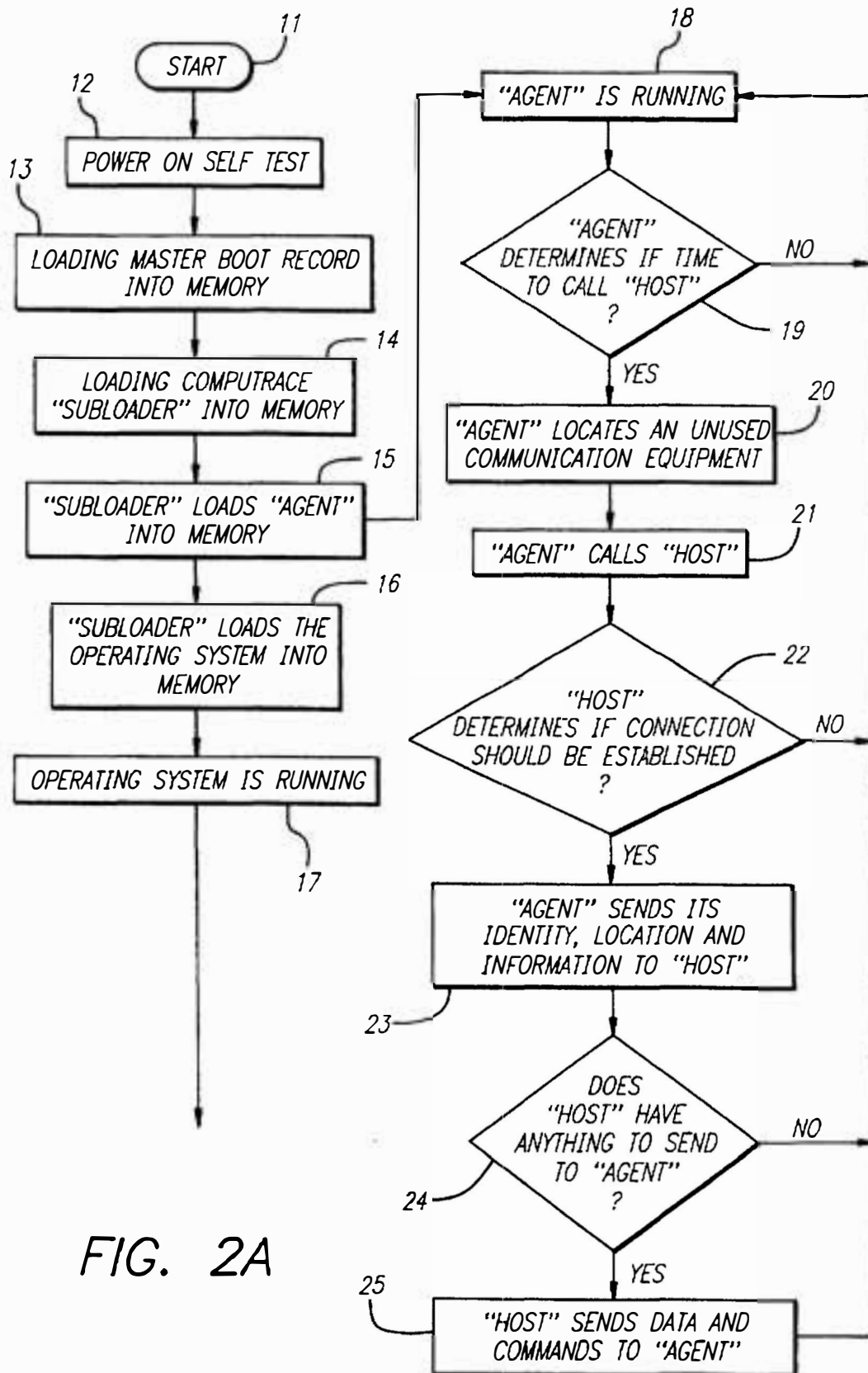


FIG. 2B

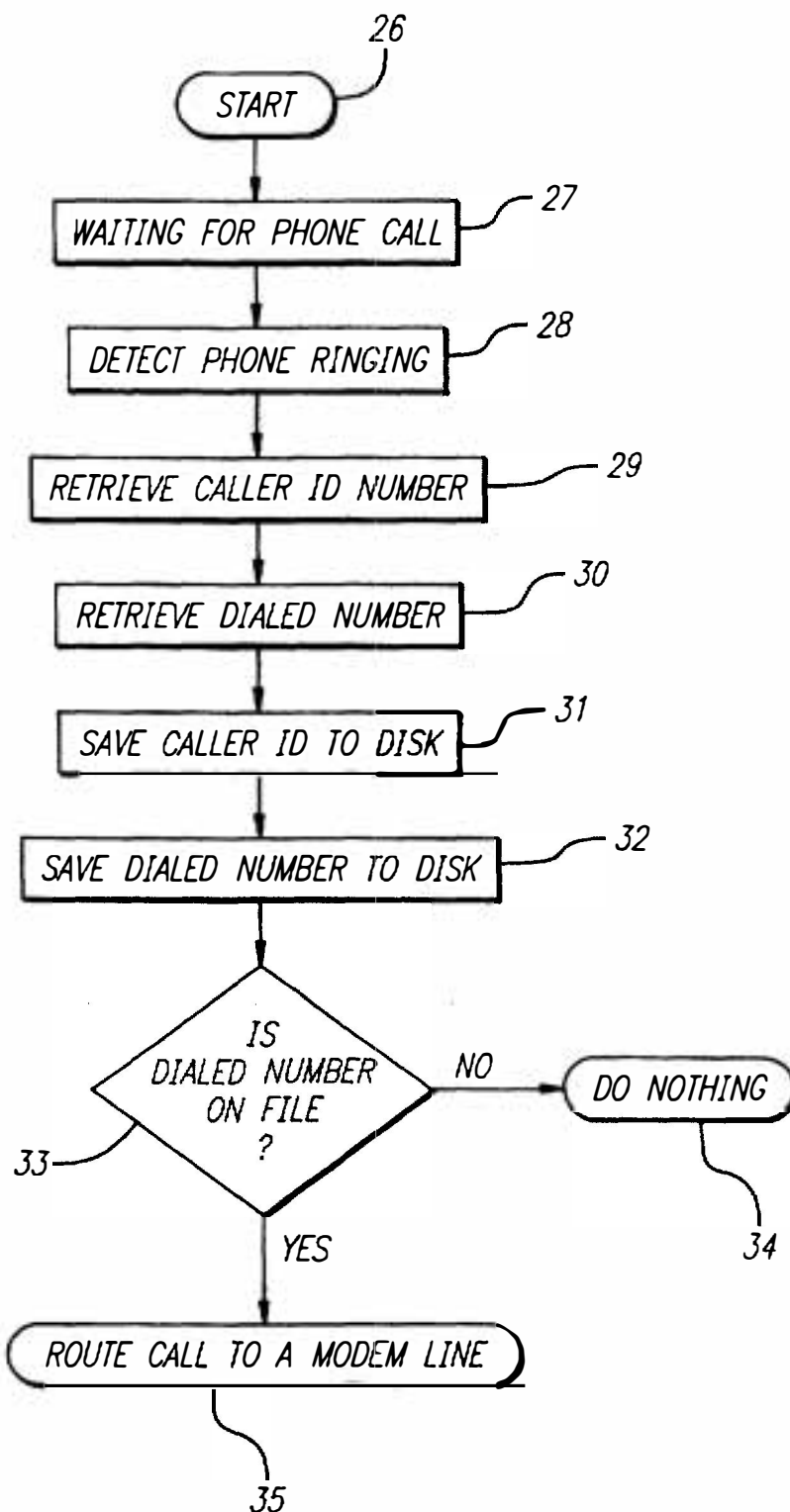
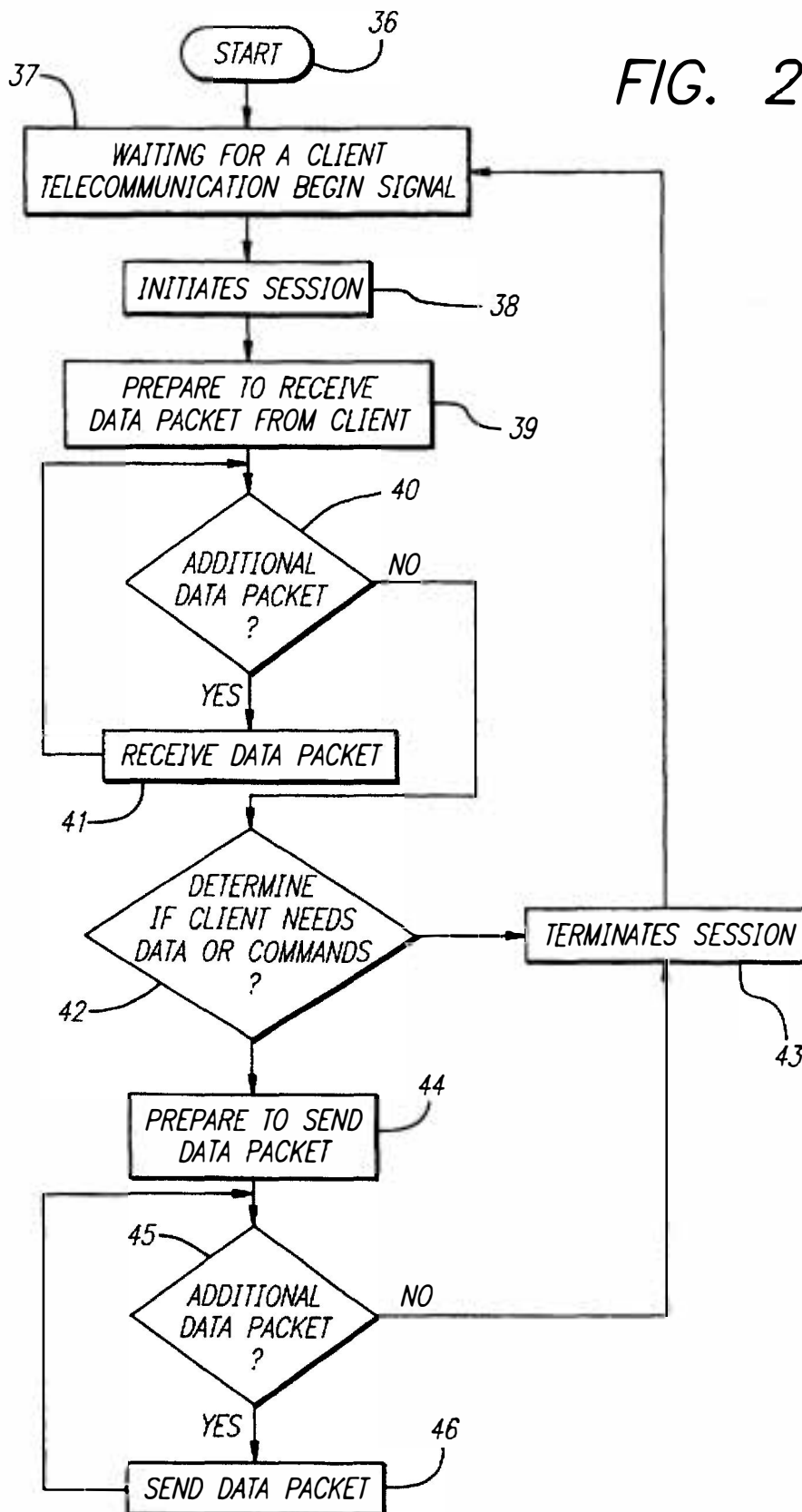


FIG. 2C



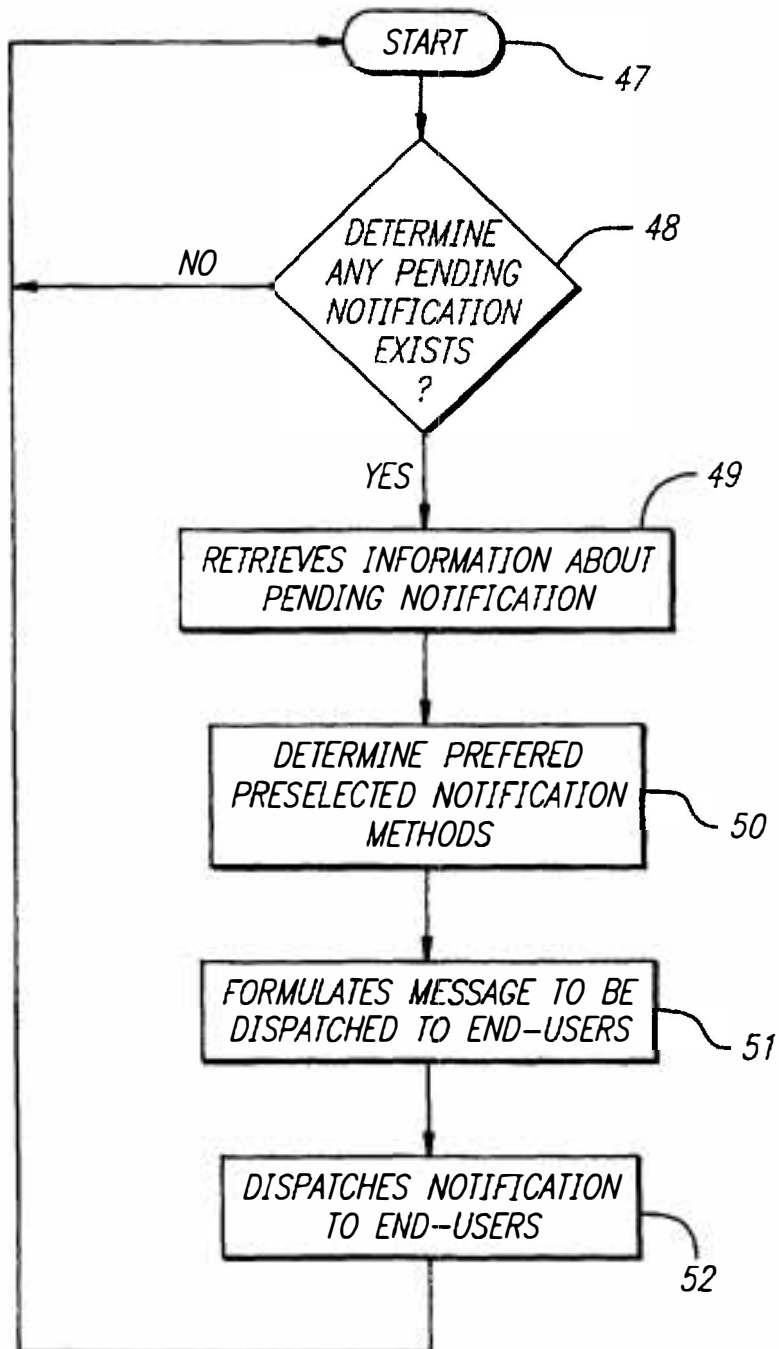


FIG. 2D

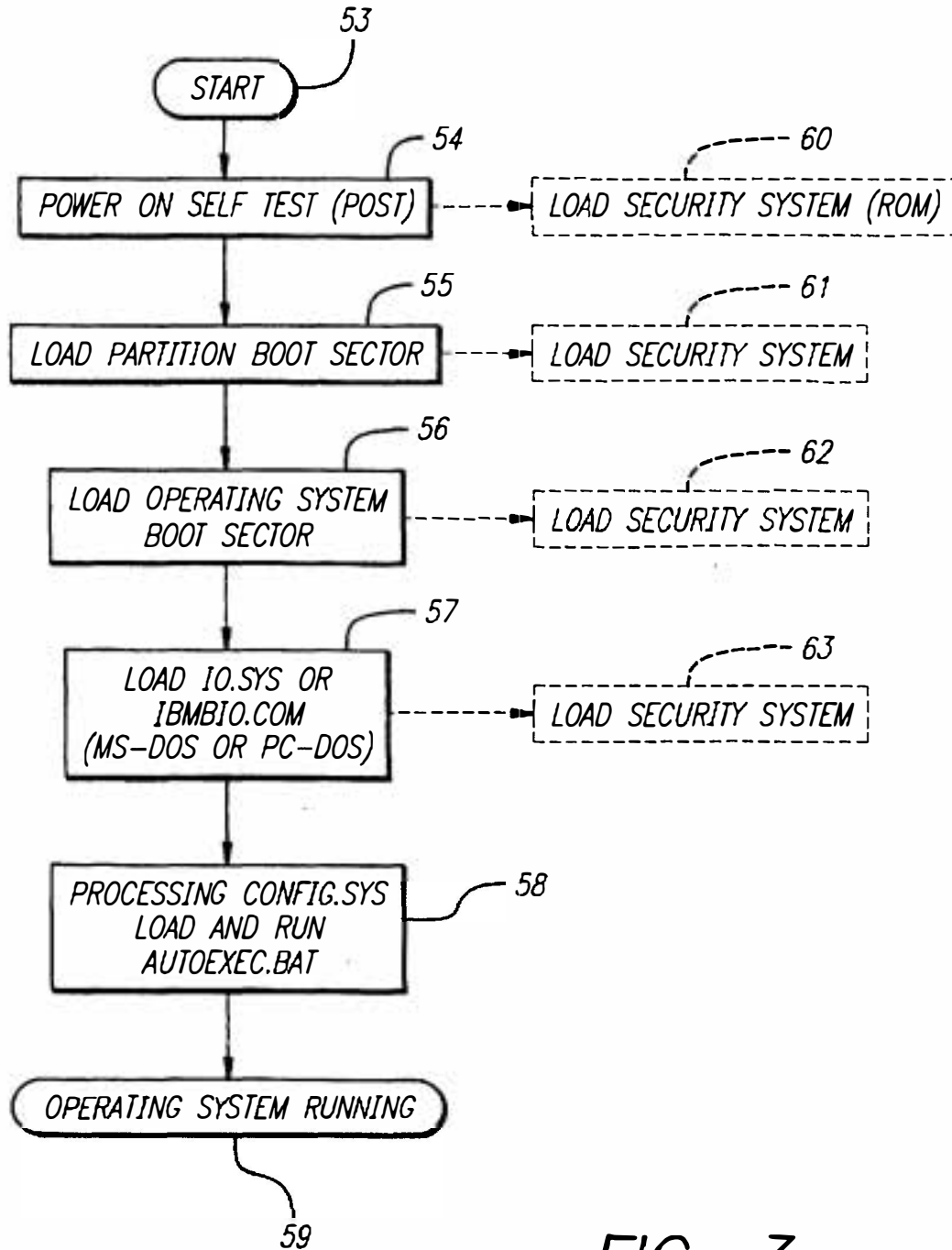


FIG. 3

FIG. 3A

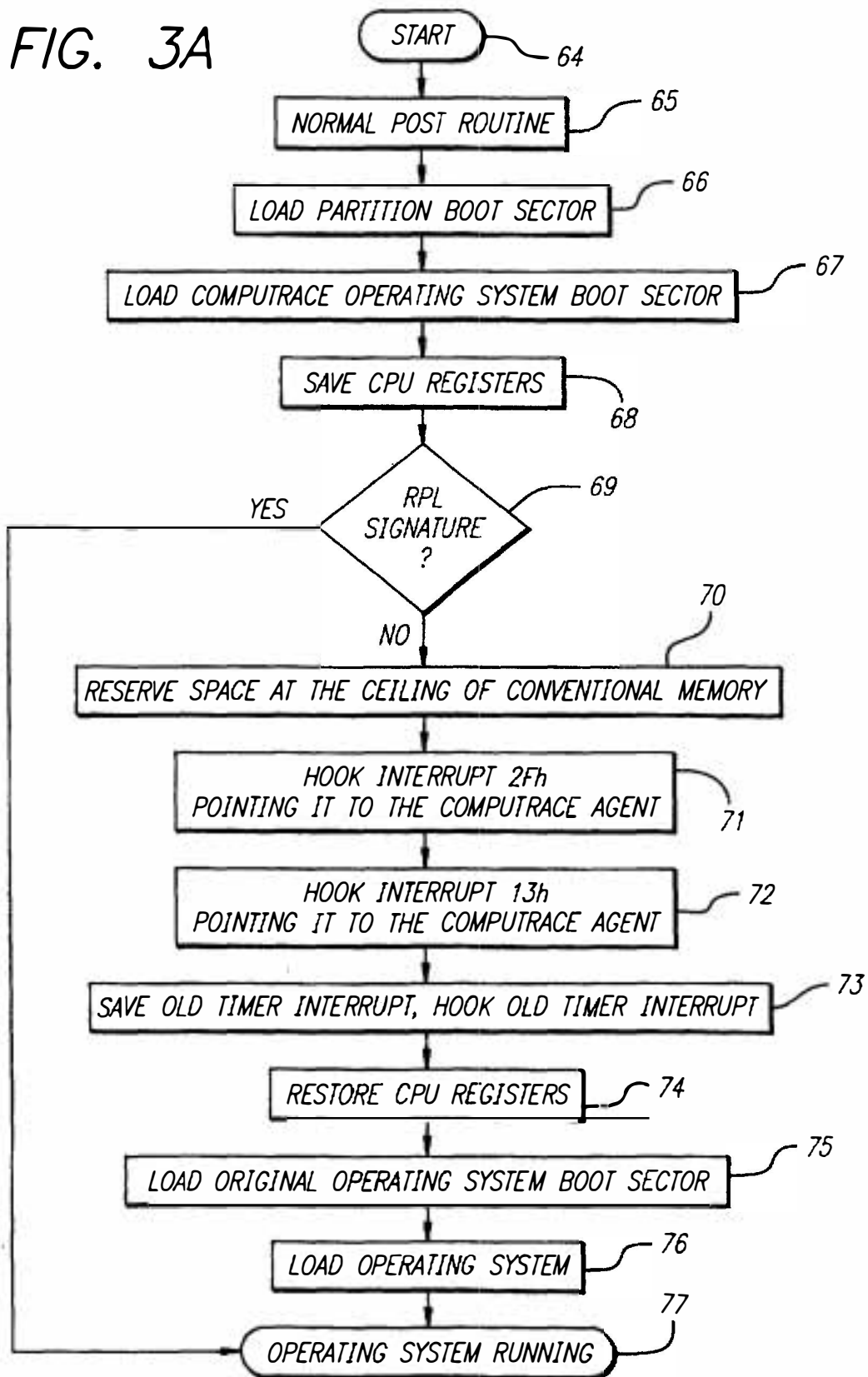


FIG. 3B

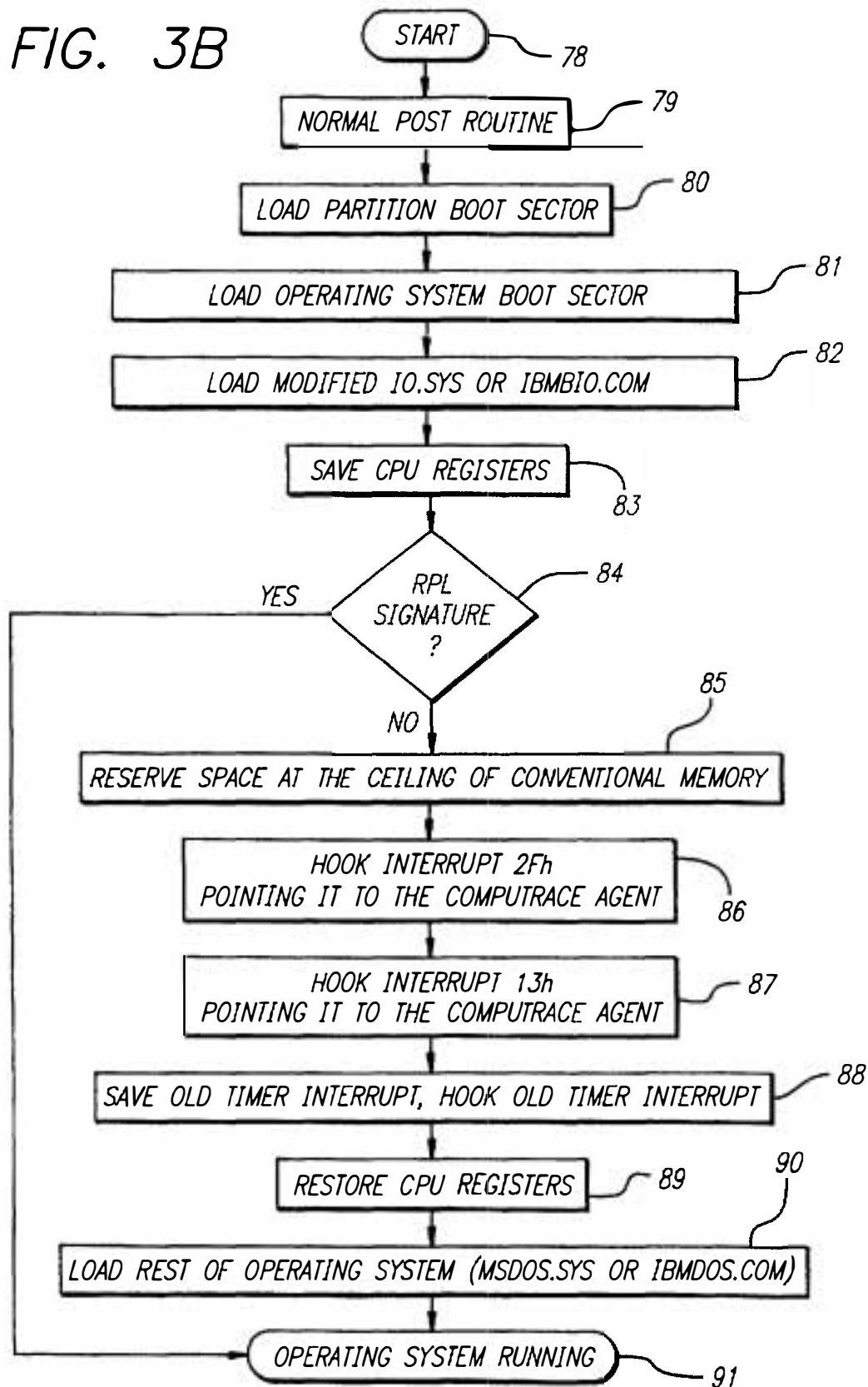


FIG. 3C

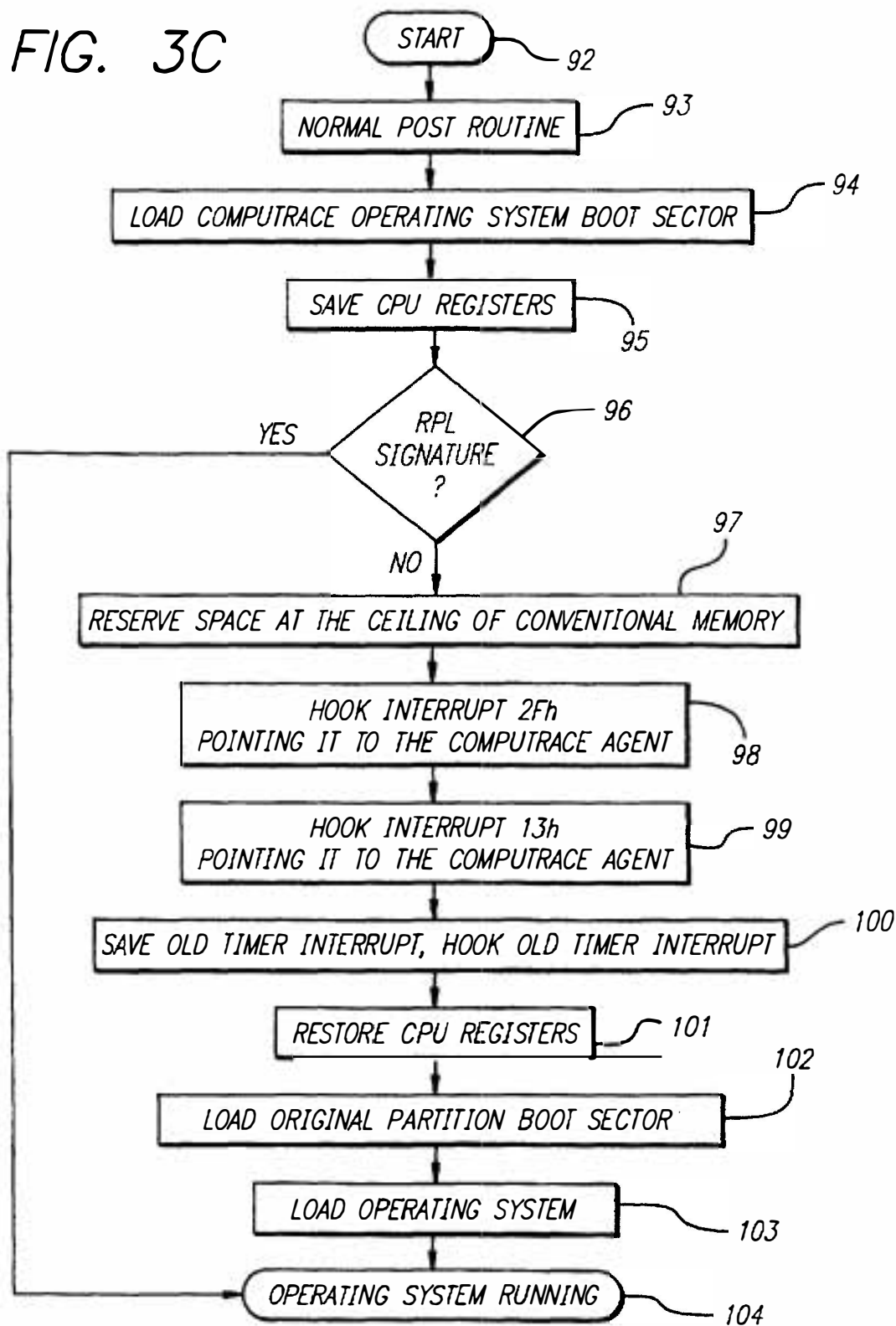


FIG. 3D

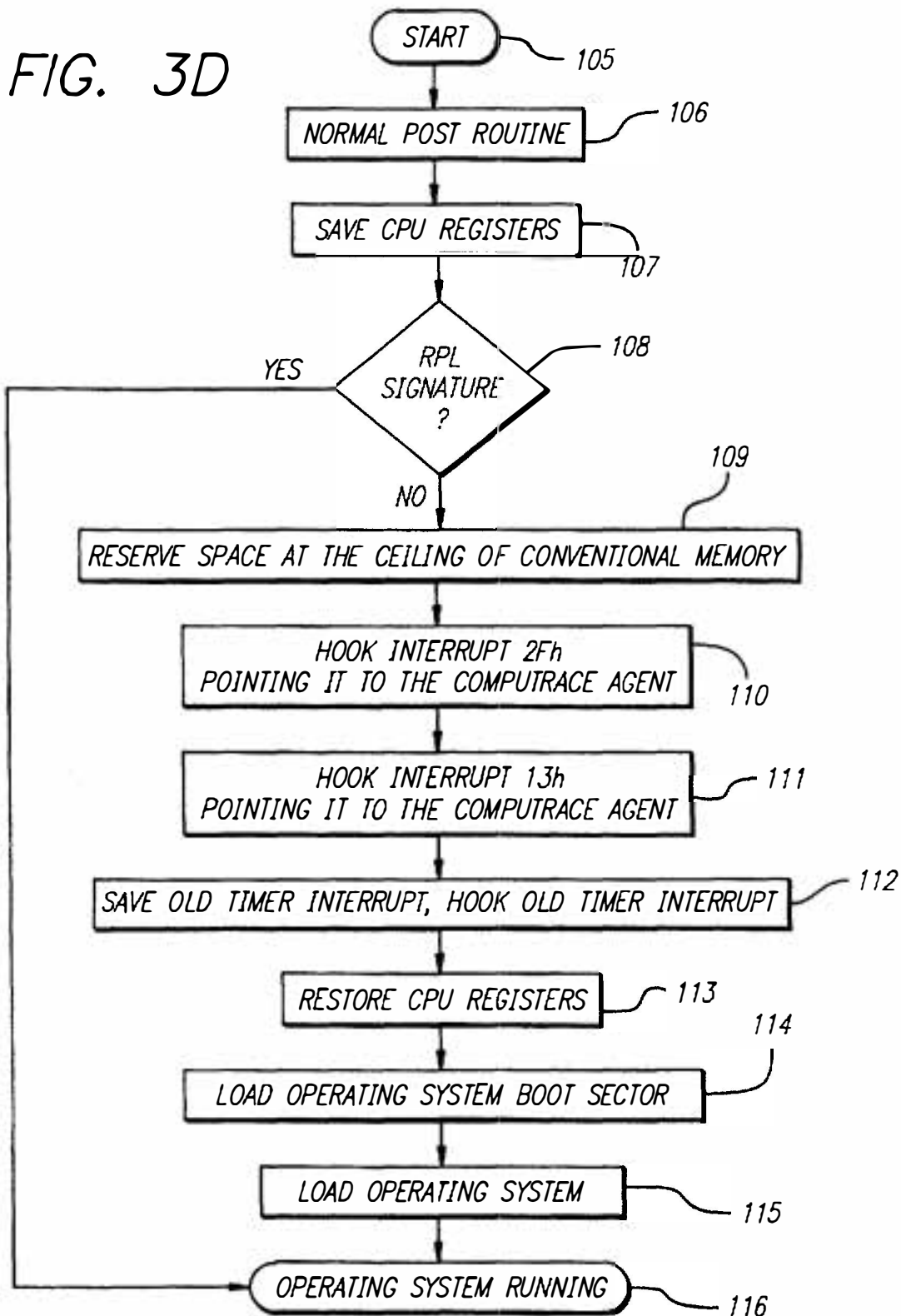


FIG. 3E(1)

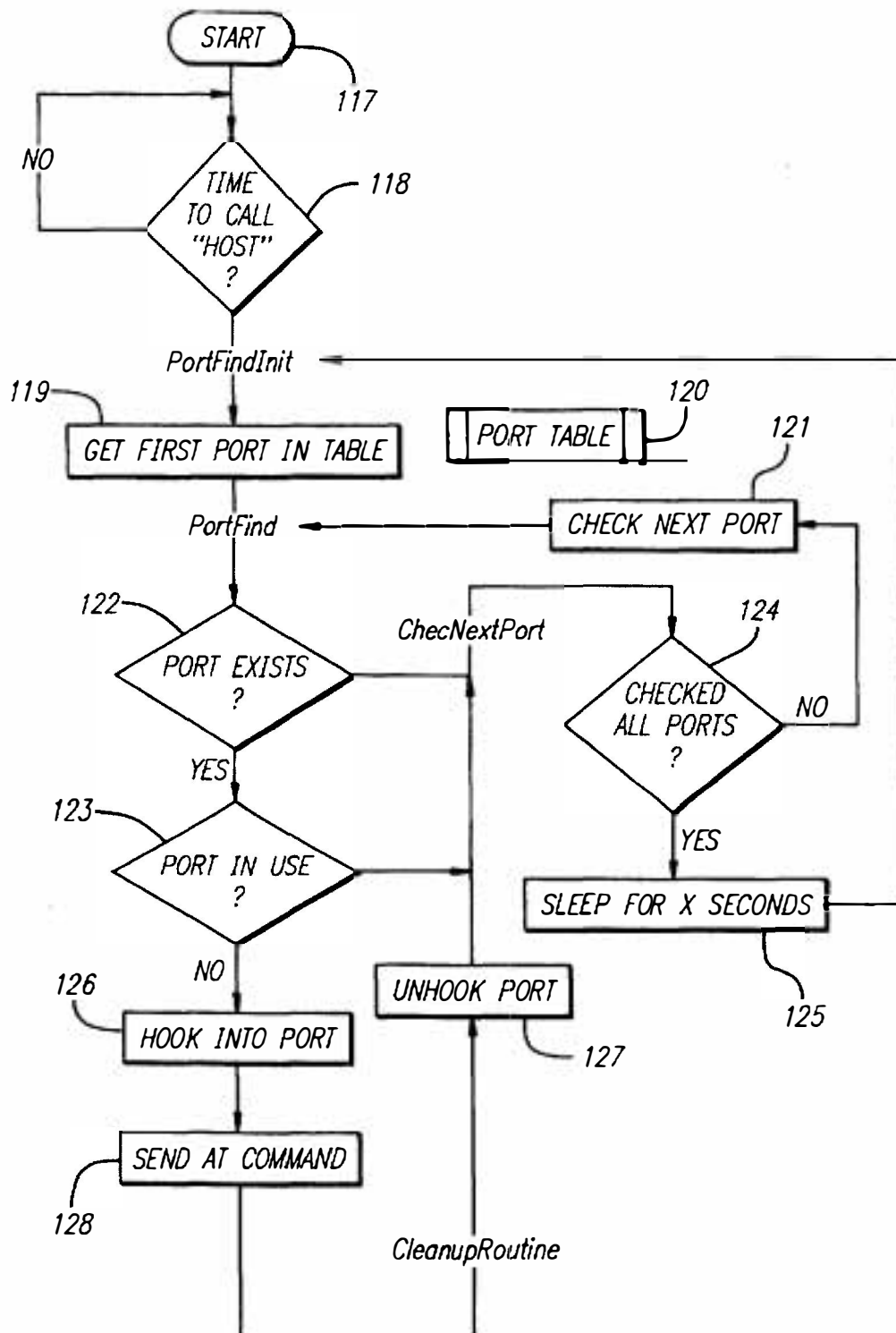
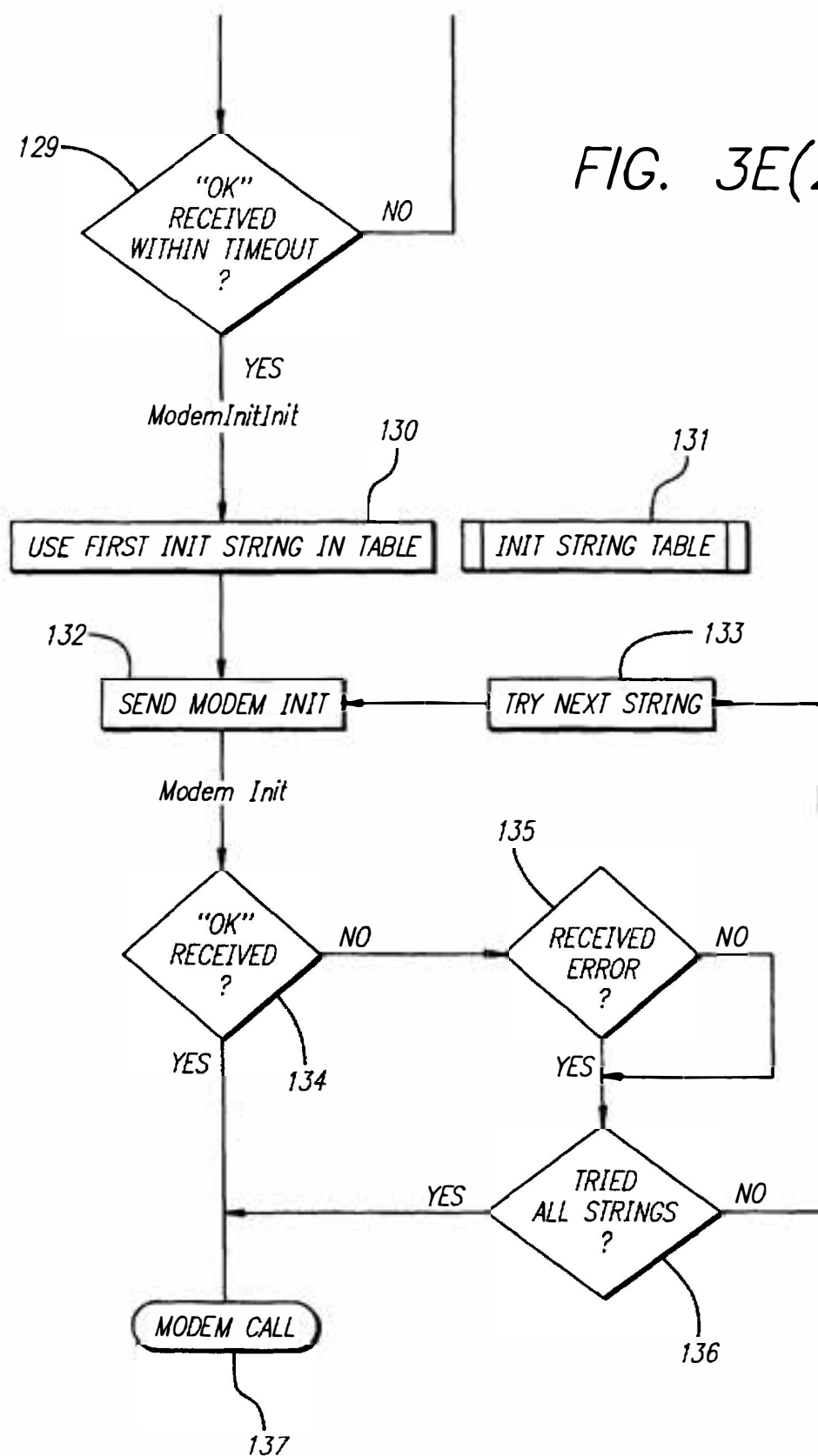


FIG. 3E(2)



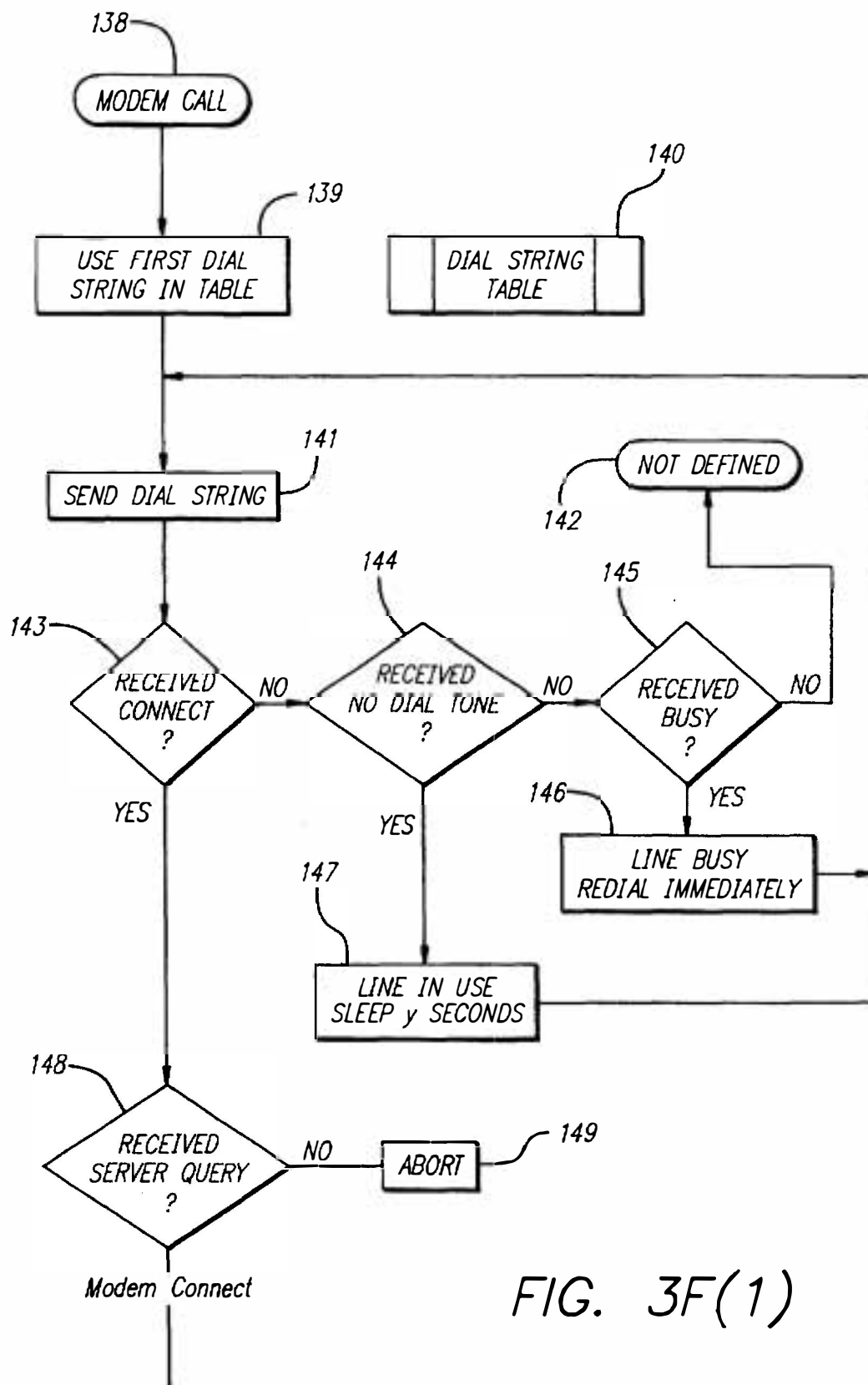


FIG. 3F(1)

FIG. 3F(2)

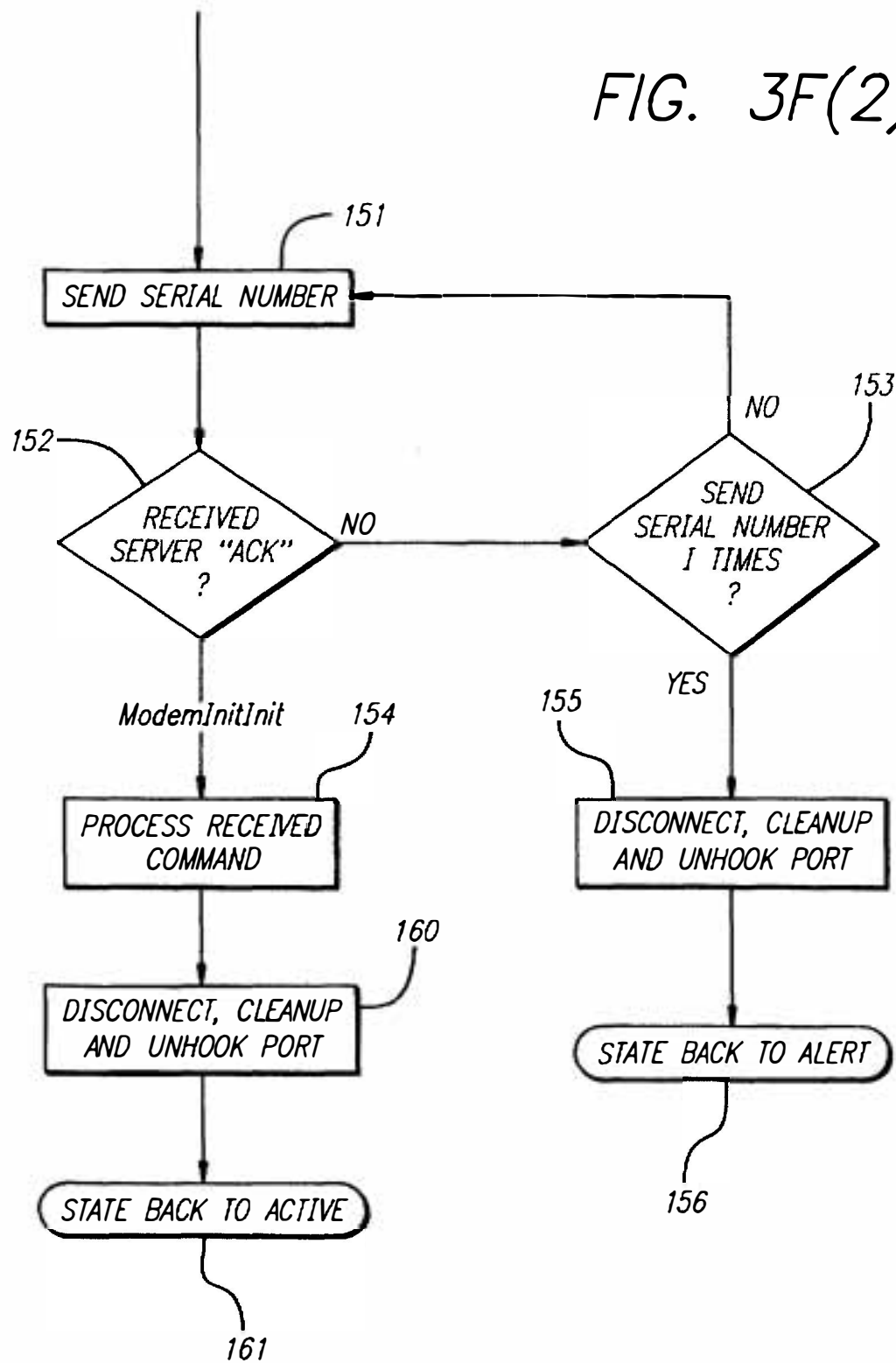


FIG. 3G

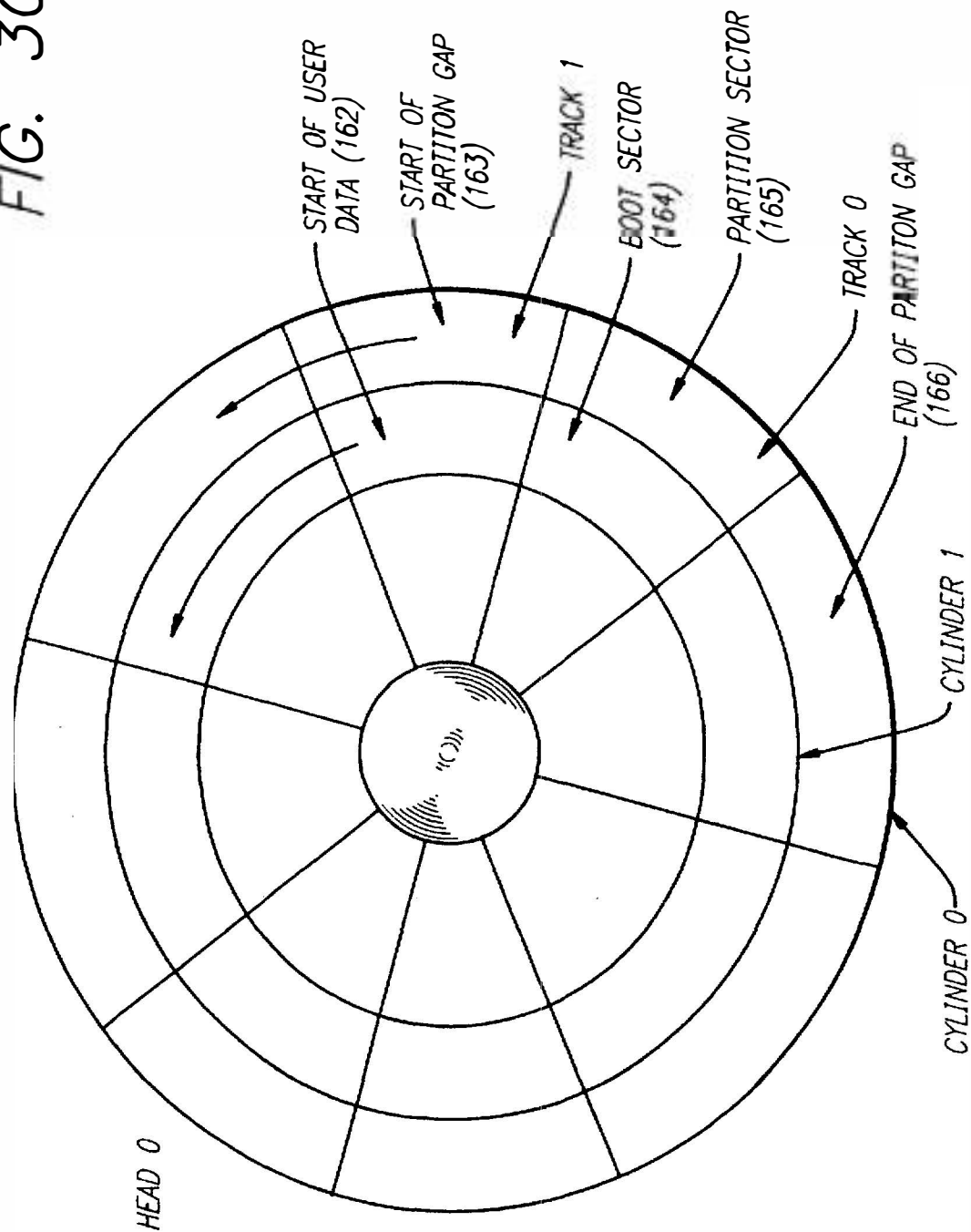


FIG. 4

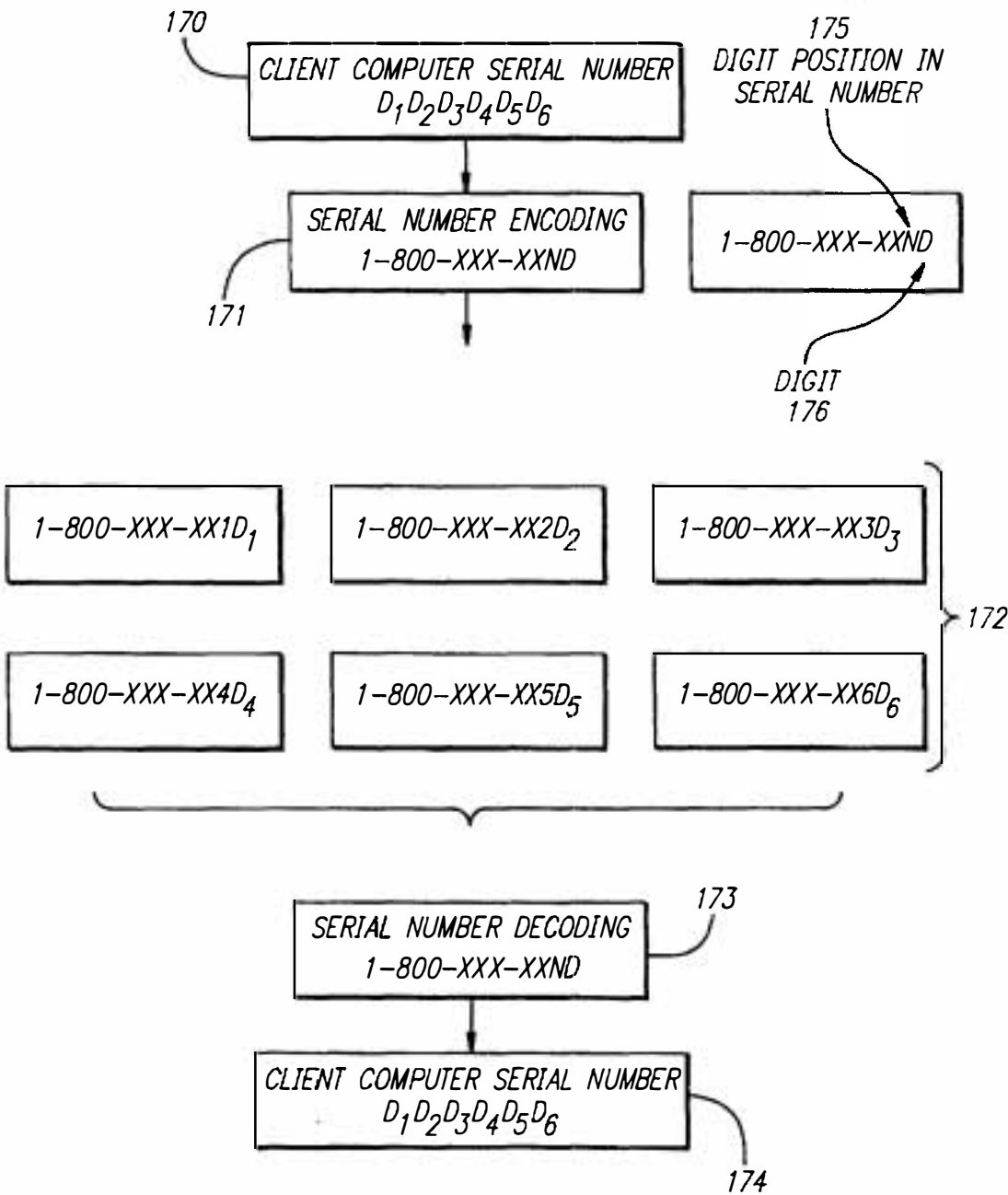
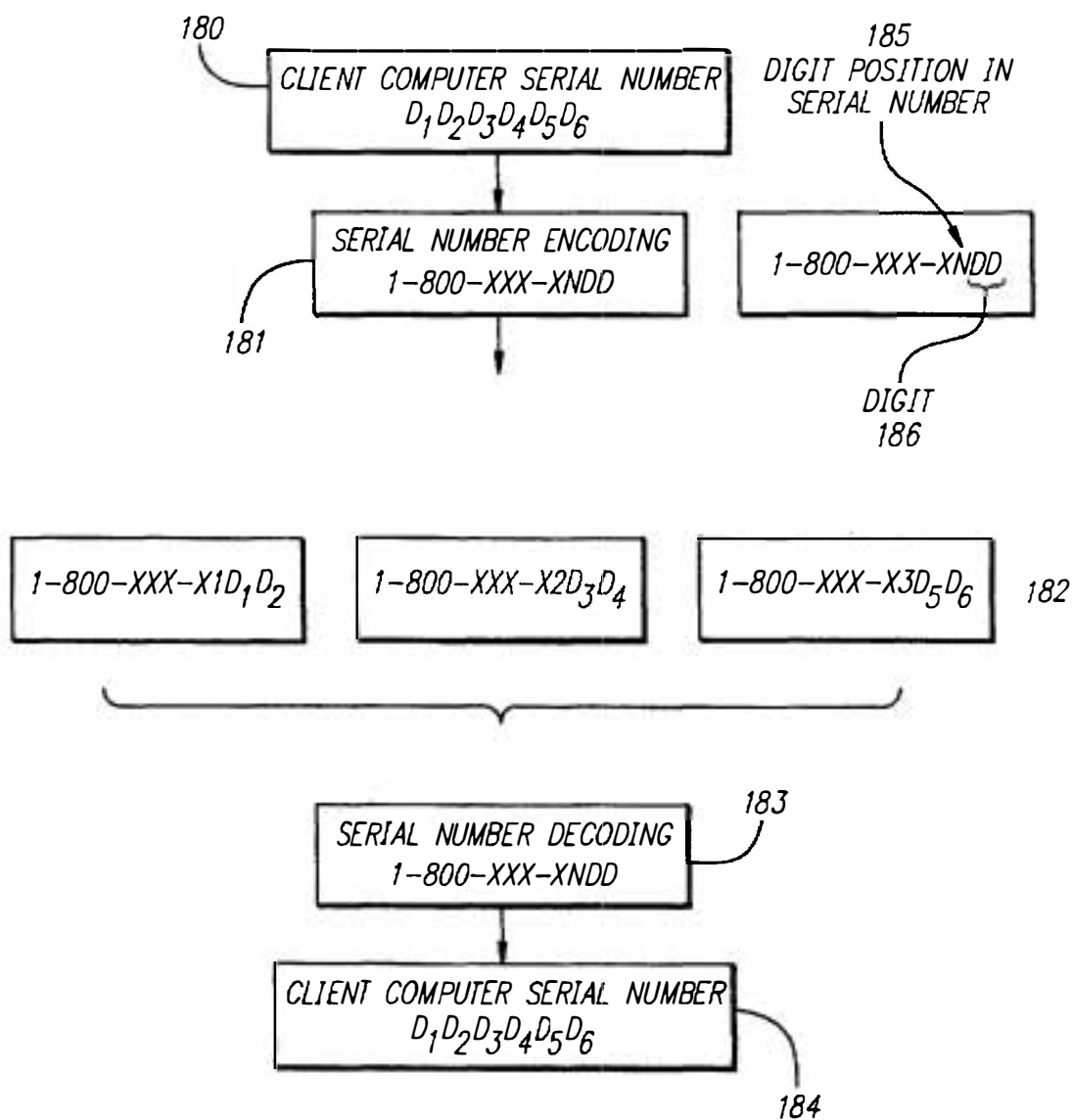


FIG. 4A



SECURITY APPARATUS AND METHOD

This application is a continuation-in-part of application Ser. No. 08/339,978, filed Nov. 15, 1994.

BACKGROUND OF THE INVENTION

Many electronic devices, such as laptop computers and cellular telephones, are becoming more compact and portable. While such portability is extremely convenient for the user, it has given rise to an increased risk of theft. These electronic devices are often very expensive and are easily lost or stolen.

Previously, attempts have been made to provide means for retrieving lost or stolen items of various types. The simplest approach is marking the item with the name and the address of the owner, or some other identification such as a driver's license number. If the item falls into the hands of an honest person, then the owner can be located. However, this approach may not deter a thief who can remove visible markings on the device.

Password protection schemes are of dubious value in discouraging theft or retrieving an item. Although the data can be protected from theft, the computer hardware cannot be found or retrieved.

Another approach has been to place a radio transmitter on the item. This has been done in the context of automobile anti-theft devices. The police or a commercial organization monitors the applicable radio frequency to try to locate a stolen vehicle. This method is not suitable for smaller items such as cellular telephones or laptop computers. First, it is inconvenient to disassemble such devices in order to attempt to install a transmitter therein. Second, there may not be any convenient space available to affix such a transmitter. Furthermore, a rather elaborate monitoring service, including directional antennas or the like, is required to trace the source of radio transmissions.

It is therefore an object of the invention to provide an improved means for tracing or locating smaller lost or stolen objects, particularly laptop computers, cellular telephones, desktop computers and other small, portable electronic devices or expensive home and office electronic equipment.

It is also an object of the invention to provide an improved means for tracing such electronic devices which can be installed without disassembly or physical alteration of the devices concerned.

It is a further object of the invention to provide an improved means for locating lost or stolen items, this means being hidden from unauthorized users in order to reduce the risk of such means being disabled by the unauthorized user.

It is a still further object of the invention to provide an improved means for locating lost or stolen items which actively resist attempts to disable the means by an unauthorized user.

It is a still further object of the invention to provide an improved means for inexpensively and reliably locating lost or stolen items.

The invention overcomes disadvantages associated with the prior art by yielding a security device for small computers, cellular telephones or the like which can be programmed onto existing memory devices such as ROM devices, hard disks or the like. Accordingly, no physical alteration is necessary or apparent to a thief. The existence of the security device is well cloaked and it cannot be readily located or disabled even if the possibility of its existence is suspected. Apparatuses and methods according to the inven-

tion can be very cost effective, requiring relatively inexpensive modifications to software or hardware and operation of relatively few monitoring devices.

SUMMARY OF THE INVENTION

This invention, Electronic Article Surveillance System, relates to a security apparatus and method for retrieving lost or stolen electronic devices, such as portable computers. This invention enables electronic articles to be surveyed or monitored by implanting an intelligent Agent with a predefined task set onto an electronic device. This Agent communicates with a preselected Host Monitoring System which is capable of multiple services including: tracing location, identifying the serial number, and electronically notifying the end user/owner of its location. The Agent hides within the software/firmware/hardware of the electronic device, and operates without interfering with the regular operation of the device. The Agent is designed to evade detection and resist possible attempts to disable it by an unauthorized user.

According to one aspect of the invention there is provided an electronic device with an integral security system. The security system includes means for sending signals to a remote station at spaced apart intervals of time. The signals including identifying indicia for the device. Preferably, the means for sending signals includes a telecommunications interface connectable to a telecommunications system, and means for dialing a preselected telecommunications number. The remote station includes a telecommunications receiver having said preselected telecommunications number.

Where the electronic device is a computer, the means for sending signals includes means for providing signals to the telecommunication interface to dial the preselected telecommunication number and send the identifying indicia. The telecommunication interface may include a modem. The means for providing signals may include security software programmed on the computer.

The Agent security system may be recorded on the boot sector of a hard disk or, alternatively, on a hidden system file such as IO.SYS, MSDOS.SYS, IBMBIO.COM or IBMDOS.COM.

There is provided according to another aspect of the invention a method for tracing lost or stolen electronic devices whereby a telecommunications interface is connectable to a telecommunications system at a first telecommunications station. The method includes providing the electronic device with means for sending signals to the telecommunications interface. The means is instructed by the program to send first signals to the telecommunications interface which dials a remote telecommunications station. These first signals contain the encoded identification (serial number) of the sending computer. The telecommunications interface then dials a remote telecommunications station corresponding to the intended receiving computer. Upon detecting a ring signal, the remote computer retrieves the caller phone number and the identification of the sending computer from the telephone company. The remote computer decodes the serial number of the sending computer, and compares it with a predefined listing of serial numbers of lost or stolen computers. The call will only be answered if the sending computer is on the predefined list.

In an alternative embodiment, if the remote computer answers the ring then the means for sending signals automatically sends second signals to the telecommunications interface, which transmits to the remote telecommunications station identifying indicia for the device as well as any other pertinent information.

There is provided according to another aspect of the invention a method for encoding the serial number of the sending computer within a sequential series of dialed numbers. In this method, a predetermined digit within the dialed number sequence corresponds to one of the digits of the serial number. The preceding digit within the encoded signal indicates which digit within the serial number sequence that the predetermined digit represents.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects and advantages will become apparent by reference to the following detailed description and accompanying drawings, in which:

FIG. 1 is a functional block diagram of the Electronic Article Surveillance System in accordance with the teachings of this invention.

FIG. 2 is a simplified illustration of FIG. 1 for the purpose of showing an illustrative embodiment of the present invention.

FIG. 2A is a flowchart of the process by which the operating system and Agent are able to start up and run simultaneously.

FIG. 2B is a flowchart of the process by which the Host Identification and Filtering Subsystem identifies and filters out unwanted calls from Agents.

FIG. 2C is a flowchart of the process by which the Host Processing, Auditing and Communication Subsystem, contained within the host computer, exchanges data with an Agent.

FIG. 2D is a flowchart of the process by which the Host Notification Subsystem, contained within the host computer, notifies end-users of the status of monitored devices.

FIG. 3 is a flowchart showing the conventional method of booting up a personal computer with alternative loading points for the Agent security system shown in broken lines.

FIG. 3A is a flowchart showing a method for startup loading of an Agent security system according to an embodiment of the invention wherein the operating system boot sector is loaded with the Agent.

FIG. 3B is a flowchart similar to FIG. 3A wherein the hidden system file IO.SYS or IBMBIO.COM is modified to be loaded with the Agent.

FIG. 3C is a flowchart similar to FIGS. 3A and 3B wherein the partition boot sector is modified to be loaded with the Agent.

FIG. 3D is a flowchart similar to FIGS. 3B and 3C wherein the Agent security system is ROM BIOS based.

FIGS. 3E, 3F are portions of a flowchart showing the Agents' work cycle apparatus and method according to an embodiment of the invention. [Are these schematics discussed?]

FIG. 3G is an isometric view, partly diagrammatic, of the physical structure of a computer disc.

FIG. 4 is a schematic showing the encoding/decoding method whereby the monitoring service would have to subscribe to 60 telephone numbers.

FIG. 4A is a schematic showing the encoding/decoding method whereby the monitoring service would have to subscribe to 300 telephone numbers.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

System Overview

Referring to FIG. 1, the Electronic Article Surveillance System is comprised of three main components: (1) Client device A consisting of any electronic device which has been implanted with the Agent; (2) A telecommunication link B such as a switched communications system, cable networks, radio/microwave signal; and (3) The host monitoring system C which controls the communications between the client device A and the host monitoring system C.

Referring to FIG. 1, the client device can be a cablevision device A2, laptop computer A3, or other type of electronic device A4. However, for illustrative purposes, the client device consists of a computer A1 attached to modem M. The host monitoring system C sends and receives data packets from the client computer 10 over a suitable bi-directional transmission medium, such as a common telephone line L1. Telephone line L1 couples the client device C to the host monitoring system C, and the host computer 3, through Public Switch B1 (telephone company). The host monitoring system C notifies the appropriate parties C3 (owner O, law enforcement agency, or monitoring company) of the status of the client device A via suitable communication means such as electronic mail N1, fax N2, telephone N3 or pager N4. Host monitoring system C also identifies and filters incoming calls C1, and also provides processing, auditing and communication functions C2.

In another embodiment of the invention cablevision device A2 is connected to cablevision network B2 via cable L2. This cable L2 further connects cablevision network L2 to the host monitoring system C.

In another embodiment of the invention laptop computer A3 is connected to radio tower B3 via radio frequency (RF) transmissions L3. These RF transmissions are received by satellite dish S at the host monitoring system C.

In yet another embodiment of the invention electronic device A4 is connected to satellite B4 via microwave signal L4. Microwave signal L4 further connects satellite B4 to satellite dish S at the host monitoring system C.

Referring to FIG. 2, the Host Monitoring system C is comprised of a Voice Board 2, Host Monitoring Computer 3, Hard Disk Controller 4, Hard Disk 5, CRT 6, Keyboard 7, and Printer 8. The host monitoring computer 3 is coupled to a suitable display device, such as a CRT monitor 6, keyboard 7, and to printer 8. The keyboard 7 permits the operator to interact with the Host Monitoring System C. For example, the operator may use keyboard 7 to enter commands to print out a log file of the clients that have called into the system. The host computer 3 illustratively takes the form of an IBM personal computer. The source codes for the host monitoring system C, in Visual C++ by MicroSoft, are attached in Appendix I.

Telephone line 1 is connected to the computer 3 by a voice board 2 adapted to receive and recognize the audible tones of both caller ID and dialed numbers transmitted via the telephone line 1. Client computer 10 is connected to modem 9 via serial ports 9a. Host computer 3 is connected to voice board 2 via serial port 2a. The modem 9 and voice board 2 are connected to telephone line 1 which is routed through public switch 9b in accordance with a conventional telephone system. Computer 10 and modem 9 form a first telecommunication station, while computer 3 and voice board 2 form a second, or remote telecommunications system. The Host Monitoring System C sends and receives data packets from client computer 10.

Ring signals are received on phone line 1 as an input to voice board 2. In an illustrative embodiment of the invention, voice board 2 may take the form of the DID/120, DTI/211 and D/12X Voice boards manufactured by Dialogic Corporation. The voice board 2 is coupled to host computer 3 via data bus 2a. The voice board 2 is operative to recognize the ring signal. Then it receives the caller ID and dialed numbers and converts them into corresponding digital signals. Host computer 3 uses these signals for comparison against a list stored in hard disk 5.

In an illustrative embodiment of the invention, the hard disk controller 4 may comprise memory control boards manufactured by Seagate Tech under the designation Hard Disk Controller. The hard disk controller 4 is particularly suitable to control the illustrative embodiment of the hard disk memory 5 manufactured by Seagate Tech under their designation ST-251.

The Agent is a terminated and stay resident program which is installed on hardware, software, or firmware. The alternative methods of installation are described in detail in FIGS. 3A, 3B, 3C, and 3D. Once the Agent is installed it will report its identity and its location to the host after specified periods of time have elapsed, and upon the occurrence of certain predetermined conditions. This is further illustrated in FIG. 2A. Client source codes are disclosed, in Tazam Assembler Code by Borland, in the Appendix.

Installing and Loading the Agent

The Agent is installed during a typical boot up sequence to the operating system of a computer. FIG. 3 shows a boot-up process for a typical personal computer. The details of the boot up process are discussed below. It should be understood that this invention is applicable to other types of computers and electronic devices presently available or as marketed in the future with suitable modifications. The aspect of the invention described below is the process of installing the security software onto a portable computer such as client computer 10. The method of installation is crucial because the software must remain undetectable once installed. Furthermore, the software should be as difficult as possible to erase. In summary, the invention achieves these objects by installing the software in such a manner that it remains hidden to the operating system, such as MS-DOS.

Three alternative ways of installing the Agent security system during the disk boot are illustrated in FIGS. 3A-3C respectively. A conventional boot up method is described in detail below. A fourth alternative, installing via ROM, is shown in FIG. 3D. The system can also be installed with MS.SYS or IBMDOS.COM, but these are more difficult and less preferred than the three alternatives set out below. The loading program TENDER (further described in the Appendix) can be used to install the Agent by one or more of these alternative installation methods. Thus, the Agent may be installed in a variety of locations whereby second and third Agents can provide back up support for the primary Agent. The three locations where the Agent can be installed on the client device are as follows:

1. The operating system boot sector—See FIG. 3A.
2. A hidden system file such as IO.SYS for MS-DOS or IBMBIO.COM for PC-DOS—See FIG. 3B.
3. The partition boot sector—See FIG. 3C.

Referring to FIG. 3A, the Agent loading sequence is described for loading the Agent on the operating system boot sector. The computer 10 is powered on and the loading sequence begins 64. As is well known in the art, the computer 10 performs an initial testing routine to assure that all components are working properly 65. Illustratively, the

program incorporated is the IBM-PC compatible Power-On Self Test (POST) routine. The partition boot sector is loaded 66. Next the operating system boot sector with the installed Agent is loaded 67. In an effort to maintain the transparency of the Agent, the CPU registers (corresponding to the current state of the computer) are saved 68. Before the Agent is installed there is a check for a Remote Procedure Load (RPL) signature 69. If the signature is present this indicates that the Agent is already in memory and will not be loaded again. However, if there is no RPL signature then preparation is made to load the Agent. First, space is reserved for the Agent at the ceiling of conventional memory 70. Next, Interprocess Communication Interrupt (2Fh) is hooked 71 which enables communication with other programs. Interrupt 13h, which is the disc input/output handler, is hooked 72. The old timer interrupt is saved, and new hook timer interrupt is put into place 73. Now the CPU registers are restored 74 in order to maintain the transparency of the system. The original operating system boot sector is loaded 75. The original operating system had been moved to accommodate the Agent installation. Finally, the operating system is loaded 76 and running 77 again.

Referring to FIG. 3B, the Agent loading sequence is described 78-91 for loading the Agent on a hidden system file such as IO.SYS for MS-DOS or IBMBIO.COM for PC-DOS. The sequence is analogous to that disclosed above for the operating system boot sector. However, instead of loading the Agent with the operating system boot sector, the Agent is loaded with the operating system file 82 (load modified IO.SYS or IBMBIO.COM).

Referring to FIG. 3C, the Agent loading sequence is described 92-104 for loading the Agent on the partition boot sector. The sequence is analogous to that disclosed above for the operating system boot sector. However, instead of loading the Agent with the operating system boot sector, the Agent is loaded with the operating system partition boot sector 94.

Referring to FIG. 3D, the Agent loading sequence is described 105-116 for loading the Agent via ROM BIOS. This schematic illustrates an embodiment of this invention on firmware. The sequence is analogous to that disclosed above for the operating boot sector. However, the Agent is loaded from the ROM after the CPU registers are saved 107. At that time the ROM can take control of the system and load the Agent. Once the CPU registers are restored 113, the ROM can no longer load the Agent.

FIG. 2A is a flow chart of the Agent Work Cycle. This Work Cycle describes the method by which the Agent is loaded when the computer 10 is initially turned on, and the manner in which the operating system and the Agent run simultaneously. Once the client computer 10 is powered on 11, it performs a power on self-test (POST) 12. The POST tests the system hardware, initializes some of the devices for operation, and loads the master boot record (MBR) 13. Since the MBR was installed with an Agent Subloader, the Subloader is loaded into memory 14 and executed. The Subloader's first task is to load the Agent 15 into memory. Then the Subloader loads the operating system (OS) into memory 16 and returns control to the operating system. Now both the operating system 17 and the Agent 18 are running simultaneously.

Functions of the Agent

Referring to FIG. 2A, the Agent's primary job is to determine the appropriate time for it to call the Host Monitoring System (Host) 19 to report its status (such as identity, location and other information). Secondly, like any ter-

minated and stay resident program, the Agent will not interfere with any running applications unless designed to interfere. Thus, the Agent can avoid being detected. The Agent will determine if it should call the Host 18 times per second. The Agent will only call the host when a pre-defined time period has elapsed, or a pre-determined event has occurred which triggers the client to contact the host. The Agent compares the current date and time with the date and time corresponding to the next time that the Agent is due to call the host. If the Agent determines that it is time to call the Host, it will do a thorough search within the computer 10 to find free (not currently being used by any running application) communication equipment 20. In an illustrative embodiment, the communication equipment is a modem 9. If the agent fails to find any free equipment, then it will abort its attempt to call the Host and repeat the cycle 18. However if the Agent locates free communication equipment, it will call the Host 21. Upon receiving a call from the client 10, the Host examines the Agent identity and determines if a connection should be established 22. If the Host does not accept the call then the Agent will not call back until the next appropriate time (after predetermined time period has elapsed) 18. If the Host accepts the call, then the Agent will send the Host its encoded identity (serial number), location (caller ID) and any other pertinent information such as local date and time 23. The Agent then checks if the Host has any data or commands for the client 24. If the Host has no data or commands to be sent, then the Agent will terminate the call and repeat the cycle 18. Otherwise, the client will receive the data or commands from the Host before it terminates the call and repeats the cycle 18. This Work Cycle is described in much greater detail in FIGS. 3E and 3F and is described in the Detailed Operation section.

The system remains transparent to an unauthorized user via implementation of well known deflection methods. Attempts to read or write to the location where the Agent has been installed are deflected in order to prevent discovery of the Agent. When read attempts are made to the Agent location the system generates meaningless bytes of data to be returned to the user. When write attempts are made to the location where the Agent is installed, the client computer 10 accepts the input data and informs the user that the write has been successful. However, the data is not really stored, and thus the Agent is preserved. In the Appendix, the source code for the disk deflection routines are disclosed within file SNTLI13V.ASM.

Detailed Operation of Agent Work Cycle

Referring to FIG. 3E, the following is a description of what happens during the period of time when the Agent security system is in "active" mode 117, 118.

Once the system is powered on, the timer interrupt will occur 18.2 times per second 117. Every 18 timer interrupts, the complementary metal-oxide semiconductor (CMOS) real-time clock will be accessed, and the time and date will be stored for comparison with the previous real-time clock access. If the date and/or time changes towards the future, no action will be taken to track the time displacement. In this way the Agent determines whether it is time to call the host 118. Thus if the current date has advanced far enough into the future (past the date and time to call the host), the Agent security system will change its mode of operation from active to alert 119 whereby calls will be regularly attempted until a call is made and a transaction with the host server has been completed. If the system time has been backdated, this will also cause a modal change from active to alert.

Referring to FIGS. 3E and 3F, the following is a description of what happens when the Agent security system is in "alert" mode 119-161.

The communications ports are checked 119-125 (via a port address table 120) to see if they exist. If the first one encountered is not in use 123, it will be dynamically hooked 126 into by swapping the appropriate interrupt handler and unmasking the appropriate interrupt request line. If an error occurs, the next port will be checked 124 until either a valid port is found or the port address table has been exhausted 125. Appropriate cleanup routines restore "swapped" ports to their initial settings.

If the communications port responds properly, the system will then attempt to connect to a modem via issue of the Hayes compatible AT command 128. If the modem does not exist, then the next port will be checked 124. If the modem responds with an 'OK' to the AT command 129, the system will attempt to initialize the modem by sending it a modem initialization string 130, 132 (from a table of initialization strings 131). If the modem does not respond with an "OK" 134, this indicates that the initialization attempt failed 135. If the initialization attempt failed, then the next string in the table will be tried 136, and so on until a valid initialization string is found 134, or the modem initialization string table is exhausted 136 (at which point, the routine will delay for some seconds then try again from the start, using the first initialization string 130).

Once a valid and available communications port has been found, and it has been verified that a functional modem is associated with that port, the system will attempt to dial out to the remote host server 137, 138.

A dial string table 140 is used 139 to attempt the call since a PBX or switchboard etc. may need to be exited via a dialing prefix. If successful 141-143, the CONNECT result code (numeric or letters) from the remote host server will be received by the client 143. The host will send a signal ("Query") to the client requesting its serial number. If the client does not receive the query signal 148 it will abort 149 and repeat the cycle 119. If the client receives the "Query" signal, then the serial number is sent 151. At this point, telecommunications have been established and the client-server transaction begins. If the transaction succeeds, the resultant state will be "active", otherwise "alert". If, for some reason, a "NO DIALTONE" event happens 144, a delay will occur 147 and the next dial string 141 will be attempted. If the line is "BUSY" 145, then a redial attempt 146 will occur using the same dial string for a predefined number of attempts or a telecommunications connection is made, whichever comes first.

The client to remote host server transaction involves the sending of the computer serial number 151 via the telephone company or carrier service. The "Caller ID" is implicitly received by the remote server (typically during the initial telecommunications event known as "RING"). Upon the telecommunications event called "CONNECT", the remote host server sends the Agent security system client a vendor specific message called "QUERY" 148 which in effect tells the client to send the serial number. The sending of this serial number 151 involves the server acknowledging that it has indeed received 152 and processed 154 the serial number (validating it). The client computer will attempt to send this serial number a predefined number of times 153 before it gives up (disconnect, cleanup, unhooks port 127, 155 and returns to "alert" mode 156). At this point, the modem disconnects 160. Any other cleanup necessary (such as changing the date of the last call to the present) will also be done here 160. Finally, the resultant state will be reset to active 161.

If the computer that called in was not reported stolen, no further action with regard to the computer system that called

in will be taken. If, however, the serial number transmitted to the remote host server matches one of the serial numbers on a currently valid list of stolen computers, further processing will occur to facilitate the recovery of the missing equipment. Such processing includes, but is not limited to, placing either an automatic or manual call to the local authorities in the vicinity of the missing equipment or the owner of such equipment.

Host Identification and Filtering System

The Host Identification and Filtering System identifies and filters out unwanted calls from Agents. FIG. 2B is a flow diagram of the Host Identification and Filtering program executed by host computer 3. Once the security program is executed 26, the voice board waits 27 for the ring signal on the telephone line 1. When a ring signal is detected 28, the voice board 2 acknowledges the incoming call by sending a signal to the telephone company 9B via telephone line 1 requesting that the caller ID and the dialed numbers be sent to it. The voice board then waits until these numbers are received 29, 30.

Once the caller ID and the dialed numbers have been received, they are saved to the hard disk 31, 32. The security program then compares the dialed numbers 33, which provide a coded version of the serial number of the client computer 10 (coding scheme explained in detail below), against a list of serial numbers stored on the hard disk 4. If no match is found, the program lets the phone ring until the client computer 10 hangs up the telephone line 1. In the preferred embodiment, the client computer is programmed to hang up after 30 seconds of unanswered ringing. However, if a match is found, the security program routes the call to an appropriate receiving line connected to a modem 35, which answers the call.

Encoding of the client computer serial number

Referring to FIG. 4, the serial number of client computer 10 is encoded within the dialed numbers it sends to the host 3. In the preferred embodiment of the invention, the client computer transmits its six digit serial number 170 to the host via a series of six complete dialed phone numbers 172. The first eight dialed digits after the first "1" are meaningless. The ninth dialed digit "N" 175, indicates which digit position within the serial number that the tenth dialed number corresponds to. The tenth dialed digit "D" provides the Nth digit of the serial number. The host computer 3 receives the six complete dialed phone numbers 172 and decodes them 173 by looking at only the ninth and tenth dialed digits. The client computer serial number 174 is thus reproduced.

For example, in the sequence "800-996-5511", the only relevant digits are the "11" portion. The first "1" indicates that the digit immediate to its right (1) is the first digit in the serial number. Similarly, in the sequence "800-996-5526", the "2" indicates that the number immediate to its right (6) is the second number in the serial number. The client 10, in total, dials six numbers 172 in order to convey its six-digit serial number to the host.

In order to accommodate this method of serial number coding, the host monitoring system needs to subscribe to sixty different phone numbers. All sixty numbers should have the same first eight digits, and only vary from one another with respect to the last two digits. The ninth digit need only vary from "1" through "6" corresponding to the six digits within a serial code. However, the last digit must vary from "0" to "9".

Referring to FIG. 4A, the coding system can alternatively be modified such that the client computer 10 need only call the host three times to convey its serial number 180.

According to this coding method, two digits of the serial number 186 would be transmitted in each call. Thus, the eighth dialed digit 185 would vary from "1" to "3", corresponding to the three packets of two digits 186 that make up the serial number 180. The ninth and tenth dialed digits 186 would vary from "0" through "9". However, this would require the operator of the monitoring system to subscribe to three hundred different phone numbers.

Host Processing, Auditing and Communication Subsystem

Referring to FIG. 2C, the Host Processing, Auditing and Communication Subsystem receives and transmits information to and from clients. FIG. 2C is a flow diagram of the Host Communication program executed by host computer 3. After the host computer 3 is powered on 36, communication equipment is instructed to wait 37 for the telecommunication begin signal from the client computer 10. The telecommunication equipment acknowledges the begin signal by initiating a session to communicate with the client computer 38. The program first checks the client computer 39 to establish that it is sending data packets 40, and then receives the packets 41. Next, the program determines if the client has any data or commands to be sent to the host 42. If not, the session is terminated 43, and the cycle is repeated 37. When all data packets have been received, the program permits the host to send data packets to the client computer. The program prepares to send data packets 44, and then establishes that there are more data packets to be sent 45 before sending each packet 46. Once all data packets have been sent, the program terminates the session 43, hangs up the phone, and prepares to repeat the entire cycle 37. Host-side source codes are disclosed in the Appendix in Visual C++ (Microsoft) Code.

Host Notification Subsystem

The Host Notification Subsystem notifies the end-users regarding the status of their electronic devices. In FIG. 1, various methods of notification such as; electronic mail N1, fax N2, paging N4, and telephone call N3, are depicted. FIG. 2D is a flow diagram of the Host Notification program executed by host computer 3. The Host Notification program determines whether there are any pending notification instructions or commands 48. If there are pending notifications, the information is retrieved 49. The program then determines the preferred preselected notification method 50, and formulates the message to be dispatched 51 according to the preselected notification method. This message is dispatched to the end-user 52. After dispatching the message, the program repeats the entire cycle 47. Host-side source codes are disclosed in the Appendix in Visual C++ (Microsoft) Code.

Variations and Alternatives

The above description relates to the Agent security system installed and operating in a conventional PC with an Intel 80X86 microprocessor or equivalent and with a conventional MS-DOS or PC-DOS operating system. It will be recognized that the system can be modified to fit other types of computers including, for example, those sold under the trademark Macintosh. The system can easily be modified to suit other types of operating systems or computers as they develop in this rapidly advancing art.

The above system is also intended to be added to existing computers without physical alteration. Another approach is to modify the ROM of such computers to contain the Agent security system as shown in FIG. 3D. This is generally not considered to be feasible for computers sold without the security feature, but is a theoretical possibility. More likely

11

is the possibility of incorporating the Agent security system into the ROM of portable computers, cellular telephones or other such items when they are manufactured. FIG. 3D above describes the loading of the system from such a modified ROM.

The description above also assumes that the computer device has a modem connected thereto or includes an internal modem. In the future it is likely that telephone systems will be digitized, thus obviating the need for a modem.

The system could also be included in the ROM of a cellular telephone. In this case, the program should be designed to hide the outgoing calls from the user by silencing audio signals and maintaining a normal screen display. It is also conceivable that portable computers can be supplied with integral cellular telephones modified in this manner or with some other telecommunication device. It is not clear at the time of this invention exactly which direction the field of telecommunications will likely go in the immediate future. The main telecommunication criteria for this Agent security system is that the outgoing transmission (wire, radio signal or otherwise), be received by a switching mechanism, and contain information that causes the switching mechanism to forward the information received to a remote station. Presently, this information is a telephone number. But other indicia of the remote station may be substituted in alternative switchable communications systems.

Attached hereto are descriptions relating to the following: (1) Description of the conventional boot up method; (2) Details of agent installation; (3) Brief description of the routines; (4) Copy of the source code (Appendix I) of both the client-side and host-side. This host-side source code is in Visual C++ (Microsoft). The client-side source code is in Tazam Assembler Code by Borland.

It will be understood by someone skilled in the art that many of the details described above are by way of example only and are not intended to limit the scope of the invention which is to be interpreted with reference to the following claims.

Conventional Boot Up Method

Referring to FIG. 3G, an isometric view of a computer disc is shown. This figure illustrates the location of the start of user data 162, partition gap 163, boot sector 164, partition sector 165, and partition gap 166.

Referring to FIG. 3, upon hitting the on switch of a personal computer (PC) 53, the computer first goes through a conventional power-on self-test (POST) 54. At this point the Agent could be loaded if ROM-BIOS loading is used 60. POST ensures that all hardware components are running and that the central processing unit (CPU) and memory are functioning properly. Upon completion of the POST, the next task is to load software onto the random access memory (RAM) of the computer. Conventionally, there is a read-only memory (ROM) device which contains a boot program. The boot program searches specific locations on the hard disk, diskette or floppy disk for files which make up the operating system. A typical disk is shown in FIG. 3G. Once these files are found, the boot program on the ROM reads the data stored on the applicable portions of the disk and copies that data to specific locations in RAM. The first portion of the disk boot sector to be loaded is the partition boot sector 55 shown in FIG. 3G as 165. At this point the load partition boot sector method could be used 61. The partition boot sector 165 then loads the remaining boot sector 164 from the

12

disk, namely the operating system boot sector 56. Now the Agent could be loaded according to the load operating system boot sector method 62. The operating system boot sector 164 loads into memory a system file, normally named IO.SYS on personal computers or IBMBIO.COM on IBM computers 57. Now the Agent could be loaded according to the IO.SYS or IBMMIO.COM methods. Each of these files is marked with a special file attribute that hides it from the DOS Dir. The IO.SYS or equivalent then loads the rest of the operating system, conventionally called MSDOS.SYS on MS-DOS systems, and IBMDOS.COM for PC-DOS systems. Next the AUTOEXEC.BAT is processed and run 58. Now the operating system is running 59. The Agent security system according to the invention is loaded during the boot up process and accordingly is transparent to the operating system.

Details of Agent Installation

Once the TENDER program, which enables the Agent to be installed, has been run and the Agent has been determined to be loaded via one, two or three of these alternatives, the system is primed and proceeds to attempt to install the Agent security system according to the present state of the computer memory and the instructions given by the programmer. The SNTLINIT routine initializes the Agent security system and is passed one of three possible loading options via the AX microprocessor register by the calling program (SUBLOADR), which itself was loaded on any one of the three enumerated locations described above. The SUBLOADR program reads the configuration file (which may be encrypted) that was generated for user input. The validity of the configuration file is checked at this point to see if it is corrupted or not. If for some reason it cannot read the configuration file, it initializes the Agent security system from a table of default settings.

The SUBLOADR program then checks to see if the Agent security system is in memory by looking for the "RPL" signature. SUBLOADR saves the application programmer interface (API) entry point and then determines which version of the security program, if any, is in memory. If not in memory, the SUBLOADR program searches the disk for the SNTLINIT routine. Depending upon the version of the SUBLOADR program, it may perform a validity check on the SNTLINIT routine. This routine may be a cyclical redundancy check (CRC) of 16 or 32 bits, a checksum check or a hash count.

The TENDER program checks the partition boot sector, the operating system boot sector, and the IO.SYS (or IBMBIO.COM on PC-DOS systems) to see if any of them have been modified to contain the SNTLINIT code. A comparison to the configuration file is made to determine if the Agent has already been installed in any of the alternative locations. If the Agent has already been installed, the TENDER program takes no action. It then tracks the level of modification that was requested by the user (i.e. whether one, two or three areas were to be modified). Each of these areas has all the modem related information written to it amongst other user selected settings. At this point it writes the current configuration file to disk.

The TENDER program then takes a system snapshot of the partition boot sector, the operating system boot sector and the IO.SYS or IBMMIO.COM file, validating them, determines and then writes this file to disk. It then checks the partition gap between the partitions, calculating the number of unused sectors between the valid boot sectors (be they partition or operating system boot sectors).

There is almost certainly at least 8K of space in the partition gap 163. The Agent security system requires only 4K. The SNTLINIT module is usually stored here. If for some reason there is not enough space in the partition gap, or if the data area is physically unusable, the TENDER program will pick a suitable cluster of sectors, mark the data area logically as being unusable, then store SNTLINIT in the cluster of sectors. The TENDER program sets out the attributes to system, hidden etc in order to hide the program image. It then calculates the physical coordinates of the cluster that was used and writes this information to the configuration file. At this point the system is ready to proceed and will be loaded prior to the completion of the loading of the operating system regardless of what strategy the programmer has chosen.

In a manner similar to how viruses reinfect the boot sector 164 of the hard disk drive, the Agent security system according to the invention uses such technology to help protect against theft of the computer. Other technologies such as system timer programming and communications programming are bound to this virus like technology to create a new technology. It should also be understood that a security company which handles incoming calls from clients may readily redefine the time period between successive calls from a client to its host.

The system is typically in one of two modes of operation: (1) Waiting until it is time to call/report into the server—"active mode"; (2) Calling or attempting to call the server—"alert mode". When the Agent security system changes its mode of operation from active to alert mode, the activation period is reduced to a minimal period such that the Agent calls the host eighteen times per second until a successful connection is made. The activation period in active mode is predetermined, and likely to be days if not weeks. This shortened activation period (time between successive calls) is necessary to prevent busy signals and other temporal error conditions from precluding transaction attempts. The system will stay in this alert mode until a valid transaction has been completed.

Since MS-DOS and PC-DOS were designed to be single-user, single-tasking operating systems, the timer interrupt is used to run the system unattended and automatically in the background to provide multi-tasking. Neither the user nor a potential thief would notice this background process although registered owners will be aware of its existence.

In a standard personal computer, routine housekeeping tasks are performed periodically and automatically by the CPU without instructions from the user. There is a timer routine which is called 18.2 times per second to perform such tasks as turning off the floppy disk motor after a certain period of inactivity. The Agent security system hooks into this timer routine. The total timer routine takes about 55 milliseconds and the Agent security system utilizes a small portion of CPU time during that period; this is limited to less than 0.5% of the total timer routine. This is not sufficient time to run the entire security program. Accordingly, the security program is run in small increments with each timer routine. It is important that the security program not "steal" enough computer time to be noticed. Otherwise the computer would be noticeably slowed and the existence of the program might be suspected.

Serial port and modem setup routines must be called by the timer interrupt. Once this is done, the serial interrupt handler that is being used will handle the details of data transfer between the client and host systems. Once the system is set up, the serial port interrupt handler does most

of the work with the timer interrupt acting as a monitor watching the transaction when it happens between the client and the server. It analyzes the receive buffer and takes the appropriate actions as necessary. The communication portion of the system can handle outgoing and incoming data transfers on its own since it has its own access to the CPU via its own interrupt request (IRQ) line, typically IRQ3 or IRQ4. Therefore the system can handle the data flow between the client machine and the server unattended.

At the start of its time-slice, the timer interrupt checks the flag, which is set when a user uses the modem, in the Agent security system, the InComISR flag byte (In Communications Interrupt Service Routine). If the flag is set, the timer interrupt exits immediately so as not to interfere with the progress of any serial communications that may be occurring, therefore not disrupting any transaction in progress. If the flag is not set, the timer interrupt routine will check to see if the Agent security system is in an error state. If not in error, a flag called TimerISR count is set to indicate that a timer interrupt is in progress.

A deferred execution function pointer is used to point to the upcoming routine to be executed. Just before the timer interrupt routine finishes, it points to the next routine to be executed. When the next timer interrupt occurs the routine that was pointed to will be executed. The routine must complete in less than 55 milliseconds so that the next timer interrupt does not occur while the routine is still executing.

Attached to the PC's system bus are communications ports, all of which are optional and typically called COM1, COM2, COM3, COM4 for the first four ports. It is unusual to have more than four serial ports in a PC that is using only MS-DOS or PC-DOS as its operating system. The Agent security system also requires that a modem be connected to one of these serial ports so that calls can be made to a remote host server using normal telephone lines or dedicated telecommunications lines. When alerted 118, the Agent security system needs to be able to find an available serial port 119-122, once it does so it checks to see if a modem is attached 128-129 and tries to initialize it by sending it an initialization string 132. If successful, it checks for a dialtone, then tries to make a quiet call to a remote host server 141. Once the server has been connected, the client machine attempts to initiate a data transaction with the server so it can send its serial number and other data defined to be part of the transaction 151. The server is configured to connect at 2400 bps with no parity, 8 data bits and 1 stop bit. Thus the client matches this configuration. This allows a high connection reliability.

Description of Routines

SNTLINIT:

After this routine has been loaded high into conventional memory 67 and execution has been passed to it, the machine state is saved 68. Conventional memory is the first 640 kilobytes (655,360 bytes) of memory on an Intel 80X86 compatible computer for example. Registers 15 that are affected by this routine are saved on the stack, "saving the machine state". The stack referred to is a LIFO structure, where the LIFO stands for "last in first out". It is where you can temporarily save the contents of CPU registers so that you can restore their initial values.

The microprocessor register AX is used to pass one of three values to the SNTLINIT routine. Depending upon which of the three values are passed to this routine, three different courses of action will be taken. Each course of action describes how the program will initialize itself. To

15

summarize, this routine initializes the Agent security system from either the partition boot sector 55, the operating system boot sector 56 or the input/output module of the operating system 57.

If the microprocessor register AX contains the value 0:

The partition sector 165 is loaded into memory (which has been overwritten on the disc with the boot sector version of the SUBLOADR module). On execution of this code, the SNTLINIT is called.

If the microprocessor register AX contains the value 1:

The boot sector 55 of the hard disk (which has been overwritten on the disc with the boot sector version of the SUBLOADR module) is loaded into memory. On execution of this code, the SNLINIT routine is called.

If the microprocessor register AX contains the value 2:

The first sector of IO.SYS/IBMBIO.COM 57 (which has been overwritten on the disk with the IO version of the SUBLOADR module) is loaded into memory.

This routine then tests to see if it is in memory already by checking for the 'RPL' signature 69, 84, 96, 108 located at the start of the address for Interrupt 2FH. If it is in memory, this routine exits 77 (to avoid loading more than one copy of the program into memory). If it is not already in memory, then it points (hooks) Interrupt 2FH to an internal routine 71, and does the same with Interrupt EAH 72. It then hooks Interrupt 8 after saving the original Interrupt 8 vector to an internal memory location (internal to the Agent security system).

The machine state is restored 74 and the routine exits by jumping to memory location 0000:7C00H for the partition table and boot sector execution paths or 0070:0000H for the IO execution path 75, 76.

SNTLAPI:

This API is for use by an external program. It has three functions as follows:

1. Get state of Agent security system. (Checks to see if Agent is already installed.)
2. Set state of Agent security system.
3. Set serial number of system.

SWAPINT:

SwapInt stores the existing interrupt vector by replacing the vector for the interrupt number in the CPU register BX with the new vector pointed to by the CPU register pair DS:CX after it stores the current vector at a location pointed to by the register pair DS:DI. If the CPU register DI contains 0 then the vector for the interrupt number contained in the CPU register BX is not stored.

DELAYFUNC:

This is a delay function used for hardware timing purposes. This routine is used in FIG. 3F, block 125.

TIMERISR:

Interrupt 8h/1Ch is the System Timer Interrupt which executes 18.2 times per second 117 and is used to do the following:

1. Call the old system timer interrupt.
2. Check to see if a communications interrupt is occurring, exiting immediately if so.
3. Save affected CPU registers.
4. Check for an internal state error, exiting immediately if so.
5. Call the state routine.
6. Restore the saved CPU registers.

ACTIVEROUTINE:

The ActiveRoutine checks to see if the activation period has been exceeded 118. By activation period we mean a

16

period of time that has elapsed since the last valid security call. This period of time is set during the transaction to the server, but is initially set to approximately 7 days.

CHECKNEXT PORT:

This is a check for valid serial ports, and involves checking a table of serial port addresses 120 and then testing them to ensure their functionality 122. If a valid serial port cannot be found, a sleep state is entered 125. Upon awakening, this routine is repeated 119.

DELAYLOOP:

This delay is used for communications delays due to busy signals or no dial-tone and other problems that can affect the communications link.

PORTFINDINIT:

This procedure calls the previously described CHECKNEXTPORT function 118, 119 in its quest for a valid serial port to initialize. On finding a valid serial port, it stores the ports address, and its corresponding interrupt vector.

PORTFIND:

This is a check to see if the serial communications port is in use 123 by dynamically testing the registers in the universal asynchronous receiver-transmitter (UART) that is associated with the current serial port address. Specifically, it tests the Interrupt Enable Register of the UART. This UART register is read into the AL register of the CPU, and if any of the bits are set (logical 1), then the port is in use, otherwise the port is idle. It also tests the interrupt enable bit of the modem control register in the UART. If the bit is not set (logical 1) then the port is idle and available for use.

Each serial port in the port table 120 is checked until either a valid one is found 123, or the routine goes to sleep 125. If a serial port is found 123, this routine will decide whether or not to initialize the UART using the system BIOS. Interrupt 14H routine, or bypass this routine, programming the UART registers directly. If an error occurs during this process, the routine is exited, and CHECKNEXT PORT is invoked.

If the serial port is successfully initialized 128, 129 to the predefined bit rate, parity, word size, number of stop bits etc., the UART is cleared of any pending errors. The serial port buffer is flushed (emptied), so there is no chance of old data being picked up a second time. The state flag that the timer interrupt checks on each clock tick is cleared, as interrupt driven communications have not yet been set up. The appropriate interrupt number is selected and the old interrupt vector is swapped with the new one by calling SWAPINT. The statuses RTS (Request to Send) and DTR (Data Terminal Ready), are enabled in the UART. The 8259 PIC is then unmasked, interrupts are enabled in the UART, then the hardware interrupts for the CPU are enabled. Then this routine exits.

MODEMFINDDELAY:

This procedure sets the [state-routine] function pointer to point to the MODEMFINDINIT routine, delaying execution until the next interrupt.

MODEMFINDINIT:

This routine points to a string to send to the modem, then calls the COMTRANSINT routine.

MODEMINTINIT:

This procedure tries to initialize the modem 130 with an appropriate initialization string from a table of initialization strings 131, and will try until either the modem is initialized or there are no more initialization strings in the table to try. The COMTRANSINT routine is called from within this procedure 132-136.

MODEMINT:

This procedure checks the state of the transmission, and checks for incoming data by calling the COMTRANS and COMTRANSCHECK routines 132. This procedure ends by jumping to a jump table which points to the next appropriate routine.

MODEMCALLINT:

This routine attempts to place a call 137, 138 by selecting a telephone number 139 (and its appropriate prefix if necessary) from a table of dial strings 140. It will continue to do so until either a call is completed 148 or there are no more initialization strings in the table to try. If a call could not be made 144 then the CLEANUPROUTINE and ERRORROUTINE procedures are to be run during the next state(s) (Interrupt 8 system timer ticks) 155.

MODEMCALLINT2:

This routine checks the state of the transmission, ending if it is complete. This procedure is called from within the MODEMCALLINT routine. It in turn calls the MODEMCALL procedure.

MODEMCALL:

This routine checks the state of the transmission, ending if it is incomplete. It also checks to see if data has been received yet or not.

MODEMCONNECTINT:

This procedure waits for a query from the host server 148 (at the other end of the communications link), and sends the serial number 151 of the computer. If a call could not be made then the CLEANUPROUTINE and ERRORROUTINE procedures 155 are to be run during the next state(s) (Interrupt 8 system timer ticks).

MODEMCONNECT:

This routine checks the state of the transmission, ending if the transmission is incomplete.

CLEANUPROUTINE:

This routine resets the Agent security system 155, 156 (sometimes referred to as Sentinel in the source code) back to a known state (ACTIVE), zeroes the transmission state flags, flushes the UART buffer. Then it disables all interrupts, restores the old communications interrupt service routine via the SWAPINT procedure. It then sets the state routine function pointer to the CLEANUPROUTINE (to be run during the next Interrupt 8).

ERRORROUTINE:

The Agent security system state is set to SNTL STATEERROR (the Agent security system is put in an error state).

COMISR:

The interrupt service routine used to control one of the systems serial communications ports (and one of the Interrupt Request lines) in order to provide telecommunications services to the Agent security system. It calls the SEND-BYTE and BUT PUTCHAR procedures. It handles the low-level details of sending and receiving data during the transmission when it happens.

SENDBYTE:

This procedure attempts to send a byte of data to the referenced serial communications port (a variable containing the port address). This routine is used in 141, 151.

COMTRANSINT:

This procedure initializes a transaction between the Agent security system and the modem. A transaction involves

sending a string of data 151 to the modem to be sent via telecommunications link to a host server, which after receiving the string of data, in return, sends back a string of data to the client machine 152 containing the Agent security system. The returned string can then be analyzed by the Agent security system to determine what action should be taken next.

COMTRANS:

This procedure handles much of the technical details regarding the maintenance of the transaction between the Agent security system and the host server 129, 134, 135, 143, 144, 145, 152, 157. It is primarily responsible for error handling such as incomplete transactions and stalled transmissions.

COMTRANSCHECK:

Checks the results of a completed transaction between the host server, and the client security system against a table of strings. Three possible outcomes are allowed for:

1. If the incoming data has not been completely received, the carry flag of the client CPU is set (logical 1).
2. If the function timed out (exceeded a time threshold value) and no Agent security system internal string matched the string received from the host server, the carry flag of the client CPU is set, and the AH register is zeroed.
3. If a matching string was found, the carry flag on the client CPU is reset (local 0), and the AL register contains a value that matches the internal table entry.

BUF_FLUSH:

Flushes the internal serial port communications receive buffer on the client machine (containing Agent security system).

- The buffer is a circular queue. A circular queue is a data structure that has what is called a head pointer and a tail pointer where the head pointer chases the tail pointer around the queue, never really catching it, but processes each byte of the data stored in it. As a byte of data is received by the serial port, it is latched and must be put into a buffer (an area of memory reserved for this purpose) before the next byte arrives (which overwrites the existing latched byte).

Whenever a communications session starts, it is important that both the input and output buffers are flushed so that all new incoming and outgoing data are not contaminated by old data still sitting in the buffer.

BUF_GETCHAR:

Gets a character from the internal serial port communications receive buffer, removing it from the buffers as it does so.

BUF_PUTCHAR:

Adds a character to the internal serial port communications receive buffer. Increments the head pointer, checking to see if the buffer is full, and setting the carry flag if it is.

- 55 **BUF_INC_PTR:**

Increments the receive buffer pointer assigned to the client CPU register SI, and wraps it if necessary.

INT2FVECT:

- 60 **Reserves the required space at the top of conventional memory for the RAM resident portion of the Agent security system. The undocumented Interrupt 21 H, Function 4AH, SubFunction 06 is used to do this.**

APPENDIX ~~III~~^V - SOURCE CODES

Electronic Article Surveillance System
Source Code for Client-side
(Tazam Assembler Code by Borland)

```

*****
; * Copyright (c) Absolute Software 1994, 1995
; *
; * SENTINEL.INC - Sentinel definition file
; *
; * PURPOSE:
; *   This file contains or INCLUDEs all constants, macros, and directives
used
; *   by the Sentinel Module.
; *
; * HISTORY:
; *   1995.09.05  CCOTI
; *               New source file taken from build 63a.
; *               See the subdirectory OldFiles for the original
; *               SENTINEL.INC
; *
; * NOTES:
; *
*****

IDEAL                ; Set parsing mode.
JUMPS                ; Allow local jumps.
P286N                ; Allow 286
instructions only.

INCLUDE "UART.INC"   ; RS232 UART
constants.

; Enable Debugging.
;   Debug = 0

; Sentinel Signature.
;   SNTL_SIG1      = 0FD FEh
;   SNTL_SIG2      = 0E FC Dh

; Sentinel Version number.
;   SNTL_VERSION = 0 * 256 + 100

; Conditional compilation switches.
;   EMIT_ON = 0          ; enables debugging.
;   IODELAY_ON = 1       ; enables io delays.
;   TWODSKHKS = 0       ; to maintain deflection with 32-bit disk
access

; Timing & Delays.
;   PORT_LOOP_DELAY = 18      ; 1 second delay
;   DIAL_LOOP_DELAY = 18 * 5  ; provide an inter-dial delay of 5 seconds
;   PREINT13_TIMEOUT = 18 * 120 ; Timeout before sentinel hook the system.

; Magic Numbers and Fixed Offsets.
;   DATA_SECTOR_OFFSET = 130h ; MUST Be Sector aligned for disk write
;                               ; (see Int13ISR)

; Debug macros.
MACRO EMIT ch
IF EMIT_ON
    PUSH    AX
    MOV     AL, ch
    CALL    PutChar
    POP     AX

```

```

ENDIF
ENDM

MACRO IODELAY
  IF IODELAY_ON
    CALL DelayFunc
  ENDIF
ENDM

;*****DO NOT CHANGE WITHOUT UPDATING SUBLOADR.H*****
; Sentinel State constants.
SNSTACTIVE      = 0
SNSTALERT       = 1
SNSTCALLING     = 2
SNSTCONNECT     = 3
SNSTERROR       = 4 ; Check for error: >= SNTL_STATE_ERROR

;*****

; Bit flag settings for <xmit state flags>.
XMIT_RECEIVE_BIT = 00000001b
XMIT_SEND_BIT    = 00000010b
XMIT_SENT_AWK_BIT = 00000100b
XMIT_RECEIVE_AWK_BIT = 00001000b

IFDEF Testing
  RECEIVE_TIMEOUT = 0FFFFh ; test timeout huge
ELSE
  RECEIVE_TIMEOUT = 18 * 40 ; timeout ~= 40 seconds.
ENDIF
; timer values (based on 18

ticks/second)
TM1SEC      = 18 * 1
TM2SEC      = 18 * 2
TM3SEC      = 18 * 3
TM4SEC      = 18 * 4
TM5SEC      = 18 * 5
TM6SEC      = 18 * 5
TM2SEC      = 18 * 5
TM10SEC     = 18 * 10
TM30SEC     = 18 * 30
TM40SEC     = 18 * 40
TM1MIN      = 18 * 60
TM2MIN      = 18 * 60 * 2

SNMDMFINDTO = 18 * 5 ; timeouts
SNMDMINITIO = 18 * 5 ; modem find timeout ~5 seconds
; modem initialization timeout ~5
seconds
SNMDMDLTO   = 18 * 40 ; modem dial out timeout ~40 seconds
SNRESPONSETO = 18 * 40 ; server response timeout ~40 seconds
SNPWRUPDLYTO = 18 * 10 ; power-up delay before hooking int 2F
-10 seconds

SNCALLNA    = 0 ; call status
SNPRTSRCH   = 1 ; no attempt yet
SNPDMSRCH   = 2 ; searching for an available port
SNMDMINIT   = 3 ; searching fo a modem on the port
SNMDMPD     = 4 ; initializing modem
SNMDMDL     = 5 ; sending predial string to modem
SNWTCON     = 6 ; sending dial string to modem
server      = 6 ; waiting for modem to connect to
SNWTENQ     = 7 ; waiting for ENQ from server

```

```

SNWTACK      = 8      ; waiting for ACK from server
SNWTNCD      = 9      ; waiting for next-call-date from
server
SNCALLPASS   = 10     ; call passed
SNCALLFAIL   = 11     ; call failed

STRUC RXZCM
    rxstate   DW ?      ; receiver structure
    rxctmr    DW ?      ; receiver state
    rxlrc     DB ?      ; receive timer
    rxpkrlen  DW ?      ; received packet running-sum LRC
    rxbytcnt  DW ?      ; packet length to receive
    rxtype    DB ?      ; received bytes in current packet
    rxstype   DB ?      ; packet type
    rxbufp    DW BYTE PTR ? ; packet subtype
                                ; pointer to receive buffer
ENDS RXZCM

STRUC TXZCM
    txstate   DW ?      ; transmit structure
    txxtst    DW ?      ; current transmitter state
    txctmr    DW ?      ; next transmitter state
    txpkrtyp  DB ?      ; transmit timer
    txxtxing  DB 0      ; packet type to transmit
    txmakcnt  DB 0      ; transmission in progress flag
    txenqcnt  DB 0      ; transmit NAK count
    txlrc     DB ?      ; transmit ENQ count
    txpkrlen  DW ?      ; transmit packet running-sum LRC
    txdatcnt  DW ?      ; remaining data bytes to transmit
    txtype    DB ?      ; index of next data byte to transmit
    txstype   DB ?      ; packet type
    txbufp    DW BYTE PTR ? ; packet subtype
                                ; pointer to transmit buffer
ENDS TXZCM

CMTXDATPKT   = 0      ; transmit packet types:
CMTXMDMPKT   = 1      ; data packet
CMTXDLENQ    = 2      ; modem packet
CMTXDLENAK   = 3      ; datalink ACK
CMTXDLENAK   = 3      ; datalink NAK
CMTXDLENAK   = 4      ; datalink ENQ
CMTXDLENAK   = 5      ; datalink EOT

DLSTX        = 2h      ; protocol control characters
DLETX        = 3h      ; STX character
DLEOT        = 4h      ; ETX character
DLENO        = 5h      ; EOT character
DLACK        = 6h      ; ENQ character
DLNAK        = 15h     ; ACK character
                ; NAK character

SNSERVER      = 80h     ; protocol message types
                ; message from the server

SNNEXTCALL    = 0h      ; protocol message subtypes
SNDISABLE     = 1h      ; next call packet
                ; disable sentinel packet

SNSNTLSIZE    = 11      ; Sentinel sector size

```

```

*****
;
; * Copyright (c) Absolute Software 1994, 1995
;
; * SNTLAPI.INC
;
; * Contains global labels for the api module.
;
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
;
*****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

    GLOBAL SntlAPI                : FAR
    GLOBAL SwapInt                : NEAR

IF IODELAY_ON
    GLOBAL DelayFunc              : NEAR
ENDIF

    GLOBAL CmpDates               : NEAR

ENDS

```

```
*****
*
* Copyright (c) Absolute Software 1994, 1995
*
* SNTLBUFF.INC
*
* Contains global labels for the buffer module.
*
* HISTORY:
*   1995.09.05 - CCOTI
*               Created.
*
*****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

GLOBAL buf_flush          NEAR
GLOBAL buf_getchar        NEAR
GLOBAL buf_putchar        NEAR
GLOBAL buf_inc_ptr        NEAR

ENDS
```

```
*****
;
; * Copyright (c) Absolute Software 1994, 1995
; *
; * SNTLCOMM.INC
; *
; * Contains the global labels for the comm module.
; *
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
; *****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

GLOBAL cmftxnak          : NEAR
GLOBAL cmftxenq          : NEAR
GLOBAL cmfprpmdm         : NEAR
GLOBAL cmftx             : NEAR
GLOBAL cmfpack           : NEAR

ENDS
```

```
*****
*
* Copyright (c) Absolute Software 1994, 1995
*
* SNTLCOMV.INC
*
* Contains global lable for the Comm ISR.
*
* HISTORY:
*   1995.09.05 - CCOTI
*               Created.
*
*****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

    GLOBAL cmfisir
    FAR

ENDS
```

```

*****
;
; * Copyright (c) Absolute Software 1994, 1995
;
; * SNTLDATA.INC
;
; * PURPOSE:
; *   Contains the global labels for the data segment.
;
; * HISTORY:
; *   1995.09.05 - CCOTI
; *   Created.
;
*****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

    GLOBAL sngstftn          : WORD
    GLOBAL Sentinel_state   : BYTE

;Scatch vars to store the current port info being used.
    GLOBAL sngmdmprt         : WORD
    GLOBAL sngmdmprtint     : WORD
    GLOBAL sngmdmprtadd     : WORD

;Previous ISR vectors.
    GLOBAL sngprvtmr        : DWORD
    GLOBAL sngprvcom        : DWORD
    GLOBAL sngprvdsk1       : DWORD

IF TWODSKHKS
    GLOBAL sngprvdsk2       : DWORD
    GLOBAL sng2dskhks       : BYTE
    GLOBAL sngdskskip       : BYTE
ENDIF

    GLOBAL sngprvint2f      : DWORD

;ROR'd to limit updating the real-time clock once every 16 ticks (see
ActiveRoutine).
    GLOBAL cycle_var        : WORD

    GLOBAL win_flag         : BYTE
    GLOBAL win_vm           : BYTE

    GLOBAL sngincmisr       : BYTE

    GLOBAL send_buf_len     : WORD
    GLOBAL send_buf_ptr     : WORD

    GLOBAL sngcomcnt        : WORD
    GLOBAL sngcomerr        : BYTE
    GLOBAL TimerISR_count   : WORD
    GLOBAL sent_count       : WORD
    GLOBAL received_count   : WORD
    GLOBAL sngflcnt         : BYTE
    GLOBAL sngclst          : BYTE
    GLOBAL sngcomhk         : BYTE
    GLOBAL sngsuspend       : BYTE
    GLOBAL sngdlytmr        : WORD
    GLOBAL sngint2ftmr       : WORD
    GLOBAL sngprtdlytmr     : WORD

```

```

GLOBAL sngdeflect      : BYTE
GLOBAL dkgcyl          : WORD
GLOBAL dkgctr          : BYTE
GLOBAL sngapifl        : BYTE
GLOBAL sngpwd1         : WORD
GLOBAL sngpwd2         : WORD

;Sentienl Settings.

GLOBAL modem_default_port : WORD

GLOBAL port_table        : WORD
PORT_TABLE_SIZE = 4

; Disk location of data sector.

GLOBAL data_cyl_sect    : WORD
GLOBAL data_head_drive  : WORD
GLOBAL sngdskwrt        : BYTE

; Output strings.

GLOBAL init_str_num      : WORD
GLOBAL init_str_table    : WORD : 5
INIT_STR_TABLE_SIZE = 6

GLOBAL dial_str_num      : WORD
GLOBAL dial_str_table    : WORD : 4
DIAL_STR_TABLE_SIZE = 5

GLOBAL dial_number       : BYTE

GLOBAL sn_packet_start   : UNKNOWN
GLOBAL stx_byte          : BYTE
GLOBAL lsb_length_byte   : BYTE
GLOBAL msb_length_byte   : BYTE
GLOBAL sn_text_start     : UNKNOWN
GLOBAL text_type         : BYTE
GLOBAL text_sub_type     : BYTE
GLOBAL sn_data_start     : UNKNOWN
GLOBAL sngsernum         : BYTE : 6
GLOBAL now_date          : UNKNOWN
GLOBAL now_year          : BYTE
GLOBAL now_month         : BYTE
GLOBAL now_day           : BYTE
GLOBAL now_hour          : BYTE
GLOBAL now_minute        : BYTE
GLOBAL sn_data_end       : UNKNOWN
GLOBAL etx_byte          : BYTE
GLOBAL lrc_byte          : BYTE
GLOBAL sn_packet_end     : UNKNOWN
GLOBAL sngsernum_str     : UNKNOWN
GLOBAL sngsernum_str_len : BYTE
GLOBAL sngdatalen        : BYTE

GLOBAL rx                : RXZCM

GLOBAL tx                : TXZCM

; Result tables.
GLOBAL command_result_table_len : BYTE
GLOBAL command_result_table     : UNKNOWN

```

```

GLOBAL mdm_init_result_table_len    BYTE
GLOBAL mdm_init_result_table        UNKNOWN

GLOBAL dial_result_table_len        BYTE
GLOBAL dial_result_table            UNKNOWN

GLOBAL connect_result_table_len     BYTE
GLOBAL connect_result_table         UNKNOWN

; Modem and result string pool.
GLOBAL string_pool                  BYTE : 127

GLOBAL modem_find_str               UNKNOWN

; next call date
GLOBAL next_call_date               UNKNOWN
GLOBAL next_call_year               BYTE
GLOBAL next_call_month              BYTE
GLOBAL next_call_day                BYTE
GLOBAL next_call_hour               BYTE
GLOBAL next_call_minute             BYTE

GLOBAL sngrxbufhd                   WORD
GLOBAL sngrxbufhl                   WORD
GLOBAL sngrxbufst                   UNKNOWN
GLOBAL sngrxbuf                    BYTE
GLOBAL sngrxbufend                  UNKNOWN

GLOBAL nextcall_text                BYTE : 5

GLOBAL sngtxindex                   BYTE
GLOBAL sngtxbufst                   UNKNOWN
GLOBAL sngtxbuf                    BYTE
GLOBAL sngtxbufend                  UNKNOWN

; Result jump tables.
; Table for ModemFind
GLOBAL find_jump_table              CODEPTR

; Table for ModemInit.
GLOBAL init_jump_table              CODEPTR

; Table for dial results.
GLOBAL dial_jump_table              CODEPTR

GLOBAL cnct_jump_table              CODEPTR

ENDS

```

```
*****
;
; * Copyright (c) Absolute Software 1994, 1995
; *
; * SNTLI13V.INC
; *
; * PURPOSE:
; *   Contains INT 13 ISRs and disk deflection routines.
; *
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
*****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

    GLOBAL load_time      WORD
    GLOBAL Int13ISR       FAR

ENDS
```

```

*****
;
; * Copyright (c) Absolute Software 1994, 1995
; *
; * SNTLI2FV.INC - SNTLI2FV.ASM global lables.
; *
; * PURPOSE:
; *
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
; * NOTES:
; *
; *****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

    GLOBAL Int2FVect      : FAR
    GLOBAL snfint2f       : FAR

ENDS

```

```

;*****
;
;* Copyright (c) Absolute Software 1994, 1995
;*
;* SNTLJTBL.INC
;*
;* Contains the global labels for the jump table.
;*
;* HISTORY:
;*   1995.09.05 - CCOTI
;*             Created.
;*
;*****
;
SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

    GLOBAL JumpTable      : NEAR
    GLOBAL cleanup        : NEAR
    GLOBAL find_ok        : NEAR
    GLOBAL find_timeout   : NEAR
    GLOBAL init_ok        : NEAR
    GLOBAL init_error     : NEAR
    GLOBAL dial_server    : NEAR
    GLOBAL dial_busy      : NEAR
    GLOBAL dial_error     : NEAR
    GLOBAL dial_no_carr   : NEAR
    GLOBAL dial_no_tone   : NEAR
    GLOBAL cnct_ack       : NEAR
    GLOBAL cnct_enq       : NEAR
    GLOBAL cnct_error     : NEAR
    GLOBAL cnct_eot       : NEAR
    GLOBAL cnct_nak       : NEAR
    GLOBAL cnct_resend    : NEAR
    GLOBAL cmrxpktto      : NEAR

ENDS

```

```
*****
;
; * Copyright (c) Absolute Software 1994, 1995
; *
; * SNTLSTRT.INC
; *
; * Contains global labels for the string table module.
; *
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
; *****

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

    GLOBAL ComTransCheck    : NEAR

ENDS
```

```

;*****
;*
;* Copyright (c) Absolute Software 1994, 1995
;*
;* SNTLTIMR.ASM
;*
;* Contains the global labels for the TimerISR.
;*
;* HISTORY:
;*   1995.09.05 - CCOTI
;*               Created.
;*
;*****
;
SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

GLOBAL tmfiser
GLOBAL ActiveRoutine
GLOBAL snfsnrst
GLOBAL ModemInitInit
GLOBAL ModemCallInit
GLOBAL ModemFindInit
GLOBAL snftxchkin
GLOBAL snfgetpkt

FAR
NEAR
NEAR
NEAR
NEAR
NEAR
NEAR
NEAR
NEAR
NEAR

ENDS

```

```

UART.INC -- Asm header file for programming the UART chip.

UART memory port base addresses
COM1_ADDRESS equ 3F8h
COM2_ADDRESS equ 2F8h
COM3_ADDRESS equ 3E8h
COM4_ADDRESS equ 2E8h

UART port interrupts
COM1_INTERRUPT equ 04h
COM2_INTERRUPT equ 03h
COM3_INTERRUPT equ 04h
COM4_INTERRUPT equ 03h

UART memory port offsets
THR equ 0 ; Transmitter holding register (out).
RDR equ 0 ; Receiver data register (in).
BRDL equ 0 ; Low byte, baud rate divisor (alternate port).
IER equ 1 ; Interrupt enable register.
BRDH equ 1 ; High byte, baud rate divisor (alternate port).
IIR equ 2 ; Interrupt ID register.
LCR equ 3 ; Line control register.
MCR equ 4 ; Modem control register.
LSR equ 5 ; Line status register.
MSR equ 6 ; Modem status register.

UART memory bit masks

Interrupt enable register.
IER_RDR_FULL equ 00000001b
IER_THR_EMPTY equ 00000010b
IER_DATA_ERR equ 00000100b
IER_MSR_CHANGED equ 00001000b

Interrupt ID register.
IIR_MULT_INT equ 00000001b

IIR_INT_ID_MASK equ 00000110b
IIR_MSR_CHANGED equ 00000000b
IIR_THR_EMPTY equ 00000010b
IIR_RDR_FULL equ 00000100b
IIR_DATA_ERR equ 00000110b

Line control register.
LCR_CHAR_MASK equ 00000011b
LCR_CHAR_5 equ 00000000b
LCR_CHAR_6 equ 00000001b
LCR_CHAR_7 equ 00000010b
LCR_CHAR_8 equ 00000011b

LCR_STOP_BIT_MASK equ 00000100b
LCR_1STOP_BIT equ 00000000b
LCR_2STOP_BIT equ 00000100b

LCR_PARITY_MASK equ 00111000b
LRC_NO_PARITY equ 00000000b
LRC_ODD_PARITY equ 00100000b
LRC_EVEN_PARITY equ 00110000b
LRC_MARK_PARITY equ 00101000b
LRC_SPACE_PARITY equ 00111000b

```

```

LCR_BREAK_MASK      equ    01000000b
LCR_BREAK_OFF       equ    00000000b
LCR_BREAK_ON        equ    01000000b

LCR_PORT_MASK       equ    10000000b
LCR_NORMAL_PORT     equ    00000000b
LCR_ALT_PORT        equ    10000000b

; Modem control register.
MCR_DTR_ON          equ    00000001b
MCR_RTS_ON          equ    00000010b ;NOT CONFIRMED!!!
MCR_USER_OUT_1      equ    00000100b
MCR_ENABLE_INT      equ    00001000b
MCR_UART_TEST       equ    00010000b

; Line status register.
LSR_RDR_FULL        equ    00000001b
LSR_OVER_ERR        equ    00000010b
LSR_PARITY_ERR       equ    00000100b
LSR_FRAMING_ERR     equ    00001000b
LSR_BREAK           equ    00010000b
LSR_THR_EMPTY       equ    00100000b
LSR_TSR_EMPTY       equ    01000000b

; Modem status register.
MSR_CTS_CHANGED     equ    00000001b
MSR_DSR_CHANGED     equ    00000010b
MSR_RI_CHANGED      equ    00000100b
MSR_DCD_CHANGED     equ    00001000b
MSR_CTR_ACTIVE      equ    00010000b
MSR_DSR_ACTIVE      equ    00100000b
MSR_RI_ACTIVE       equ    01000000b
MSR_DCD_ACTIVE      equ    10000000b

; BIOS services.
BIOS_INIT_PORT      equ    00h
BIOS_WRITE_PORT     equ    01h
BIOS_READ_PORT      equ    02h
BIOS_STATUS_PORT    equ    03h

; BIOS initialization values.
BIOS_7BITS          equ    00000010b
BIOS_8BITS          equ    00000011b
BIOS_1STOP          equ    00000000b
BIOS_2STOP          equ    00000100b

BIOS_PARITY_MASK     equ    00011000b
BIOS_NO_PARITY       equ    00000000b
BIOS_ODD_PARITY      equ    00001000b
BIOS_EVEN_PARITY     equ    00011000b

BIOS_BAUD_MASK       equ    11100000b
BIOS_110_BAUD       equ    00000000b
BIOS_150_BAUD       equ    00100000b
BIOS_300_BAUD       equ    01000000b
BIOS_600_BAUD       equ    01100000b
BIOS_1200_BAUD      equ    10000000b
BIOS_2400_BAUD      equ    10100000b
BIOS_4800_BAUD      equ    11000000b
BIOS_9600_BAUD      equ    11100000b

```

55

5.764.892

56

59

```

*****
;
; * Copyright (c) Absolute Software 1994, 1995
; *
; * SENTINEL.ASM - Sentinel Initialization and TSR Code
; *
; * PURPOSE:
; *   This is the main build file for the Sentinel module.
; *
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Source taken from build 63a and broken up into separate source
; *               files. See the subdirectory OldFiles for the original
; *               SNTLINIT.ASM
; *
*****

IDEAL
include "SENTINEL.INC"

;
; *****
; *
; * SNTL_SEG - Resident segment.
; *
; *****
SEGMENT SNTL_SEG PARA PUBLIC 'CODE'

;=====
include "SNTLJTBL.ASM"
;=====
include "SNTLCOMV.ASM"
;=====
include "SNTLSTRT.ASM"
;=====
include "SNTLBUFF.ASM"
;=====
include "SNTLI2FV.ASM"
;=====
include "SNTLI13V.ASM"
;=====
ENDS

END

```

```

;*****
;
; Copyright (c) Absolute Software 1994, 1995
;
; SNTLAPI.ASM
;
; Contains the sentinel API routine and general purpose routines used by all
; modules.
;
; HISTORY:
;   1995.09.05 - CCOTI
;               Created.
;*****
;
IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLAPI.INC"
include "SNTLDATA.INC"
include "SNTLTIMR.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
;
; SNTLAPI
;
; PURPOSE:
;   This function provides an external API for the Ward and Tender modules,
;   as well as development software tools, to gain access to the Sentinel.
;
;   The following functions are supported:
;
;   Function 0 - Get Sentinel State
;     returns AL = Sentinel_state
;            BX = sngstftn
;
;   Function 1 - Set Sentinel State to ALERT
;     returns CF = 0 if successful
;            CF = 1 if failed
;
;   Function 2 - Get Sentinel Version Number
;     returns AH = major version number
;            AL = minor version number
;
;   Function 3 - Get Sentinel Serial Number
;     returns ES:DI = pointer to serial number
;
;   Function 4 - Cancel Sentinel ALERT
;     returns CF = 0 if successful
;            CF = 1 if failed
;
;   Function 5 - Set next-call date and time
;     returns ES = Sentinel data segment
;            DI = offset of next_call_date
;            SI = offset of sngdskwrt
;
;   Function 6 - Get call status

```

```

; *      returns AL = sngclst: SNCALLNA    = 0  no call attempt yet
; *      SNPRTSRCH    = 1  searching for an available port
; *      SNMDMSRCH    = 2  searching fo a modem on the
port
; *      SNMDMINIT    = 3  initializing modem
; *      SNMDMPD      = 4  sending predial string to modem
; *      SNMDMDL      = 5  sending dial string to modem
; *      SNWTCON      = 6  waiting for modem to connect to
server
; *      SNWTENQ      = 7  waiting for ENQ from server
; *      SNWTACK      = 8  waiting for ACK from server
; *      SNWTNCD      = 9  waiting for next-call-date from
server
; *      SNCALLPASS   = 10 call passed
; *      SNCALLFAIL   = 11 call failed
; *
; *      Function 7 - Disable Sentinel disk deflection
; *      returns CF = 0 if successful
; *      CF = 1 if failed
; *
; *      Function 8 - Enable Sentinel disk deflection
; *      returns CF = 0 if successful
; *      CF = 1 if failed
; *
; *      Function 9 - return data segment pointers
; *      returns ES:DI = Sentinel Data Segment (SntlDataSeg in sentinel.h)
; *      ES:SI = Sentinel Settings (SntlSettings in sentinel.h)
; *
; *      PARAMETERS:
; *      None
; *
; *      Registers destroyed: none
; *
; *      Globals referenced:
; *      Sentinel_state
; *
; *      Globals modified:
; *      Sentinel_state - set to SNSTALERT by function 1
; *      sngstftn - set to
; *
; *      BIOS calls: none
; *
; *      DOS calls: none
; *
; *      proc calls: none
; *
; *      hardware access: none
; *
; *****
;
;      ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING
PROC SntlAPI FAR
@@check0:
; Return the state.
    CMP     AH,0
    JNE     @@check1
    MOV     AL,[Sentinel_state]
    MOV     BX,[sngstftn]
    RET

@@check1:
; Attempt to set the state to ALERT.
    CMP     AH,1
    JNE     @@check2

```

```

    CMP     [Sentinel_state], SNSTACTIVE
    JNE     @@exit_w_error
    MOV     [Sentinel_state], SNSTALERT
    MOV     [sngstftn], OFFSET snfsnrst
    CLC
    RET

@@check2:
    CMP     AH, 2
    JNE     @@check3
    MOV     AX, SNTL_VERSION
    RET

; Return the version number.
; MOD CCOTI 48:95.01.27

@@check3:
    CMP     AH, 3
    JNE     @@check4
    PUSH    CS
    POP     ES
    MOV     DI, OFFSET sngsernum
    RET

; Return the serial number.

@@check4:
    CMP     AH, 4
    JNE     @@check5
    CMP     [Sentinel_state], SNSTACTIVE
    JE      @@check4_done
    MOV     [Sentinel_state], SNSTACTIVE
    MOV     [sngstftn], OFFSET snfsnrst
@@check4_done:
    RET

@@check5:
    CMP     AH, 5
    JNE     @@check6

; test for function 5
; not detected, continue

    PUSH    CS
    POP     ES
    MOV     DI, OFFSET next_call_date
    MOV     SI, OFFSET sngdskwrt
    RET

; prepare to copy string
; get ES = CS
; ES:DI points to next_call_date
; ES:SI points to data_write flag
; exit

@@check6:
    CMP     AH, 6
    JNE     @@check7

; test for function 6
; not detected, continue

    MOV     AL, [sngclst]
    RET

; get the call status into AL
; exit

@@check7:
    CMP     AH, 7
    JNE     @@check8
    MOV     [sngdeflect], 0
flag
    CLC
    RET

; clear the carry flag
; exit

@@check8:
    CMP     AH, 8
    JNE     @@check9

; test for function 8
; not detected, exit with error

```

```

        MOV     [sngdeflect], 1          ; set the Sentinel disk deflection
flag
;This is commented out to maintain the data segment offset with the CTM.EXE
(See CCOTI)
        CLC
        RET                                ; clear the carry flag
                                           ; exit

@@check9:
        CMP     AH, 9                    ; test for function 9
        JNE     @@exit_w_error          ; not detected, exit with error
        PUSH    CS                       ; get ES = CS
        POP     ES                       ; ES:DI points to data segment
        MOV     DI, OFFSET sngstftn     ; ES:SI points to sentinel settings
        MOV     SI, OFFSET modem_default_port
        CLC
        RET                                ; clear the carry flag
                                           ; exit

@@exit_w_error:
        STC

@@exit:
        RET

ENDP SntlAPI
ASSUME NOTHING

;*****
;
;Routine: SwapInt
;
;Descript: SwapInt stores the existing vector
;          replaces the vector for the interrupt in BX with the new vector DS:CX
after
;          it stores the current vector at [DS:DI]. If DI = 0 the current vector
is
;          not stored.
;
;Arguments:
;          BX = the interrupt to hook into
;          DS:DI = address to save the existing vector; if DI = 0 the existing
vector
;          if not stored.
;          DS:CX = the new vector to install
;
;Registers destroyed: AX, BX, ES, FLAGS
;
;Returns: nothing
;
;BIOS calls: none
;
;DOS calls: none
;
;proc calls: none
;*****
        ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC SwapInt
        XOR     AX,AX
        MOV     ES,AX
        SHL     BX,2                      ; BX = address of
vector
; load the existing vector and save it to DS:DI (if requested).

```

```

        OR      DI,DI
        JZ      @@no_store
        MOV     AX,[ES:BX]
        MOV     [DS:DI],AX
        MOV     AX,[ES:BX+2]
        MOV     [DS:DI+2],AX
@@no_store:
; install the new vector
        CLI
        MOV     [ES:BX],CX
        MOV     [ES:BX+2],DS
        STI
        RETN
ENDP SwapInt
        ASSUME NOTHING

;*****
;Routine: DelayFunc
;
;Descript: DelayFunc - introduces an delay.
;
;Arguments: none
;
;Registers destroyed: none
;
;Returns: nothing
;
;BIOS calls: none
;
;DOS calls: none
;
;proc calls: none
;*****
IF IODELAY_ON
        ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC DelayFunc
        PUSH    CX
        MOV     CX,1
@@loop_start:
        LOOP    @@loop_start
        POP     CX
        RETN
ENDP DelayFunc
        ASSUME NOTHING
ENDIF

;*****
;Routine: CmpDates
;
;Descript: CmpDates - compares two dates and sets the CF=1 if date1 < date2.
;
;Arguments: [SI] -> date1
           [DI] -> date2
;
;Registers destroyed: SI, DI, CX, ES
;
;Returns: CF = 1 if date1 < date2
;
;BIOS calls: none
;

```

```
DOS calls: none
proc calls: none
*****
      ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC CmpDates
      PUSH     DS
      POP      ES
      CLD
      MOV      CX,5
@@cmp_loop:
      CMPSB
      JB       @@cmp_exit      CMP [SI], [DI]
      LOOPE    @@cmp_loop      CF = 1?
@@cmp_exit:
      RETN
ENDP CmpDates

      ASSUME NOTHING
ENDS

      END
```

```

;*****
;
; * Copyright (c) Absolute Software 1994, 1995
; *
; * SNTLBUFF.ASM
; *
; * Contains the circular buffer access routines.
; *
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
;*****

IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLBUFF.INC"
include "SNTLDATA.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'
    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

;*****
;
; *
; *
; * BUF_FLUSH - flush receive buffer
; *
; * PURPOSE:
; *   This function flushes the receive buffer by setting the tail index
; *   equal to the head index.
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * REGISTERS DESTROYED:
; *   None
; *
; * GLOBALS REFERENCES:
; *   None
; *
; * GLOBALS MODIFIED:
; *   sngrxbufhd
; *   sngrxbuftl
; *
; * BIOS CALLS:
; *   None
; *
; * DOS CALLS:
; *   None
; *
; * PROCEDURE CALLS:
; *   None
; *
; * HARDWARE ACCESS:
; *   None

```

```

NOTES:
*****
*

PROC buf_flush NEAR
    MOV     [sngxbufhd], OFFSET ngxbuf
    MOV     [sngxbuftl], OFFSET ngxbuf
    RET
    exit
ENDP buf_flush
*****
*

BUF_GETCHAR - get a character from receive buffer

PURPOSE:
    This function returns the next available character in the receive buffer
    and increments the tail pointer.

;Arguments: none
;Registers destroyed: AL, SI
;Globals referenced:
    sngxbuftl
    sngxbufhd
;Globals modified:
    received_buf_tail - moved to the location of the next character
;Returns: AL = the character taken, CF=0
    If the buffer is empty CF=1
;BIOS calls: none
;DOS calls: none
;proc calls: buf_inc_ptr
;hardware access: none
*****

PROC buf_getchar NEAR

    MOV     SI, [sngxbuftl]    ; get the tail pointer
    CMP     SI, [sngxbufhd]    ; is it the same as the head
    JE      @@empty            ; yes, exit with status

    MOV     AL, [SI]           ; no, get the next byte
    CALL    buf_inc_ptr        ; increment tail pointer
    MOV     [sngxbuftl], SI    ; set new tail pointer position
    CLC
    RET                                ; set status
                                ; exit

@@empty:
    STC
    RET                                ; set status
                                ; exit

ENDP buf_getchar

```

```

;*****
;
;Routine: buf_putchar
;
;Descript: Adds a character to the buffer.
;
;Arguments: AL = the character to add
;
;Registers destroyed: SI
;
;Globals referenced:
;    sngrxbufhl
;    sngrxbufhd
;
;Globals modified:
;    received_buf_head - moved to the location of the next free space
;
;Returns: CF=0 if the character is stored correctly.
;         CF=1 if the buffer is full.
;
;BIOS calls: none
;
;DOS calls: none
;
;proc calls: buf_inc_ptr
;
;hardware access: none
;*****
;
PROC buf_putchar NEAR
    MOV     SI, [sngrxbufhd]    ; point to the head of the buffer
    MOV     [SI], AL           ; store the received character
    CALL    buf_inc_ptr        ; increment the head
    MOV     [sngrxbufhd], SI   ; set new head pointer

    CLC
    RET                                ; set return status
                                ; exit

ENDP buf_putchar

;*****
;
; * BUF_INC_PTR - increment bffer pointer
;
; * PURPOSE:
; *   This function increments the head or tail pointer associated with the
; *   receive buffer.
;
; * PARAMETERS:
; *   SI = the pointer to increment
;
; * RETURNS:
; *   SI = the next location in sngrxbuf
;
; * REGISTERS DESTROYED:
; *   SI
;
; * GLOBALS REFERENCED:
; *   OFFSET sngrxbuf
; *   OFFSET sngrxbufend

```

```

* GLOBALS MODIFIED:
*   None
;
* BIOS CALLS:
*   None
;
* DOS CALLS:
*   None
;
* PROCEDURE CALLS:
*   None
;
* HARDWARE ACCESS:
*   None
;
* NOTES:
;
*****
*

PROC buf_inc_ptr NEAR
    INC     SI                ; increment SI
    CMP     SI,OFFSET sngrxbufend ; check if the pointer has wrapped
    JNE     @@no_buf_wrap     ; no, continue
    MOV     SI,OFFSET sngrxbuf ; yes, set back to beginning of buffer

@@no_buf_wrap:
    RET                                ; exit
ENDP buf_inc_ptr

ENDS

END

```

```

*****
* Copyright (c) Absolute Software 1994, 1995
*
* SNTLCOMM.ASM
*
* Contains comm routines.
*
* HISTORY:
*   1995.09.05 - CCOTI
*               Created.
*
*****

IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLCOMM.INC"
include "SNTLDATA.INC"
include "SNTLTIMR.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
;
;
; * CMFTMOUT - transmit a NAK
;
; * PURPOSE:
; *   This function transmits a NAK. If 3 NAK's have already been
transmitted,
; *   the transaction is terminated with an EOT.
;
; * PARAMETERS:
; *   DX = UART Transmit Holding Register
;
; * RETURNS:
; *   Nothing
;
; * NOTE:
;
;*****

ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC cmftxn timer NEAR

    CMP     [tx.txnackcnt], 3           ; only send 3 NAK's before aborting
    JE      @@aborttx

    MOV     AL, DLNAK                   ; send another NAK
    OUT     DX, AL
    INC     [tx.txnackcnt]
    MOV     [tx.txnxtst], OFFSET snfgetpkt ; set state function following tx
    MOV     [rx.rxtmr], TM10SEC         ; set response to NAK timeout
    JMP     @@exit

@@aborttx:
    MOV     AL, DLEOT                   ; send EOT to terminate transaction

```

```

OUT    DX, AL
MOV    [tx.txxxtst], OFFSET snfsnrst    ; set state function following tx

@@exit:
MOV    [sngstftn], OFFSET cmftx        ; set next state function
MOV    [tx.txxstate], OFFSET CS:cmtxcomp; set transmitter state: tx complete
MOV    [rx.rxxstate], OFFSET cmfpstx    ; reset receiver
RETN

ENDP cmftxnak

ASSUME NOTHING

```

```

;*****
;*
;* CMFTXENQ - transmit an ENQ
;*
;* PURPOSE:
;*   This function transmits a NAK. If 3 NAK's have already been
transmitted,
;*   the transaction is terminated with an EOT.
;*
;* PARAMETERS:
;*   DX = UART Transmit Holding Register
;*
;* RETURNS:
;*   CF = 0 if not timed out
;*   CF = 1 if timed out
;*
;* NOTE:
;*
;*****

```

```

ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING

```

```

PROC cmftxenq NEAR

```

```

CMP    [tx.txxenqcnt], 3                ; only send 3 NAK's before aborting
JE     @@aborttx

MOV    AL, DLENQ                        ; send another ENQ
OUT    DX, AL
INC    [tx.txxenqcnt]                   ; increment transmitted ENQ count
MOV    [tx.txxxtst], OFFSET snfgetpkt   ; set state function following tx
MOV    [rx.rxxtmr], TM10SEC              ; set response to ENQ timeout
JMP    @@exit

@@aborttx:
MOV    AL, DLEOT                        ; send EOT to terminate transaction
OUT    DX, AL
MOV    [tx.txxxtst], OFFSET snfsnrst    ; set state function following tx
MOV    [rx.rxxstate], OFFSET cmfpstx    ; reset receiver

@@exit:
MOV    [sngstftn], OFFSET cmftx        ; set next state function
MOV    [tx.txxstate], OFFSET CS:cmtxcomp; set transmitter state: tx complete
RETN

ENDP cmftxenq

```

ASSUME NOTHING

```

*****
*
* CMFPRPMDM - prepare to transmit modem string
*
* PURPOSE:
*   This function prepares the transmit structure before initiating
*   transmission of a string to the modem.
*
* PARAMETERS:
*   BX => the string to transmit (see note below)
*
* RETURNS:
*   Nothing
*
* REGISTERS DESTROYED:
*
* GLOBALS REFERENCED:
*
* GLOBALS MODIFIED:
*
* BIOS CALLS:
*   None
*
* DOS CALLS:
*   None
*
* PROCEDURE CALLS:
*   None
*
* HARDWARE ACCESS:
*   None
*
* NOTE:
*   BX points to the length of the string to transmit, which is preceeded in
*   memory by the string (eg. AT<CR>3)
*****

```

ASSUME CS:SNL_SEG, DS:SNL_SEG, ES:NOTHING

PROC cmfprpmdm

```

MOV     AL, [BX]                ; get the length of the packet
MOV     [BYTE LOW tx.txpktlen], AL
MOV     [BYTE HIGH tx.txpktlen], 0
SUB     BX, [tx.txpktlen]       ; set pointer to start of string
MOV     [tx.txbuf], BX
MOV     [tx.txpkttyp], CMTXMDMPKT ; transmitting modem packet
MOV     [tx.txxtmr], TM1SEC      ; set maximum transmit time
MOV     [tx.txxtxng], 0         ; clear transmission in progress
flag

MOV     [sngstftn], OFFSET cmftx ; next state: transmit
MOV     [rx.rxxtmr], TM6SEC      ; wait 5 seconds after tx for rx

RETN

ENDP cmfprpmdm

```

ASSUME NOTHING

```

*****
;
; CMFTX - transmit state machine
;
; PURPOSE:
;   This function acts as the transmitter state machine performing all
;   packet transmissions and data-link ACK's, NAK's, and ENQ's.
;
; PARAMETERS:
;   None
;
; RETURNS:
;   Nothing
;
; REGISTERS DESTROYED:
;
; GLOBALS REFERENCED:
;
; GLOBALS MODIFIED:
;
; BIOS CALLS:
;   None
;
; DOS CALLS:
;   None
;
; PROCEDURE CALLS:
;   None
;
; HARDWARE ACCESS:
;   UART (IN LSR, OUT THR)
;
; NOTE:
;
*****

```

ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

PROC cmftx

```

    CMP     [tx.txxtmr], 0      ; has the transmitter been on too
long?                          ;
    JE      cmtxrst            ; yes, reset transmitter and
Sentinel                       ;
                                ; no, continue
                                ; ensure THR is empty.
    MOV     DX, [sngmdmptadd]   ; get DX = LSR
    ADD     DX, LSR             ; and determine if the THR is empty
    IN      AL, DX              ; and load another byte if it is
    TEST    AL, 00100000b       ; not needy for DOS world where TX
ISR                             ;
    JZ      @@exit              ; works fine but in WINDOWS world
the                             ; TX ISR chokes and this routine is
                                ; called by ComTrans()

    MOV     DX, [sngmdmptadd]   ; THR empty, get THR address

```

```

CMP     [tx.txxtxing], 1           ; transmission in progress?
JE      cmcont                     ; yes, continue
MOV     [tx.txxdatcnt], 0          ; no, clear data bytes tx'd count
MOV     [tx.txxtxing], 1          ; set transmission in progress flag
CMP     [tx.txxpkttyp], CMTXDLNAK ; transmitting a NAK?
JE      cmtnak                     ; yes
CMP     [tx.txxpkttyp], CMTXDLACK  ; transmitting an ACK?
JE      cmtxack                     ; yes
CMP     [tx.txxpkttyp], CMTXDLENQ  ; transmitting an ENQ?
JE      cmtxenq                     ; yes
CMP     [tx.txxpkttyp], CMTXDLEOT  ; transmitting an EOT?
JE      cmtxeot                     ; yes
CMP     [tx.txxpkttyp], CMTXMDMPKT ; transmitting modem packet?
JNE     cmprpdata                  ; no, must be data packet
MOV     [tx.txxstate], OFFSET CS:cmtxdata ; yes, just transmit data segment
JMP     cmcont

cmprpdata:
MOV     [tx.txxstate], OFFSET CS:cmtxstx ; transmitting data packet
cmcont:
JMP     [tx.txxstate]

cmtxstx:
MOV     AL, DLSTX
OUT     DX, AL
MOV     [tx.txxlrc], 0             ; clear LRC checksum
MOV     [tx.txxstate], OFFSET CS:cmtxlenlsb
JMP     @@exit

cmtxlenlsb:
MOV     AL, [BYTE LOW tx.txxpktlen]
OUT     DX, AL
XOR     [tx.txxlrc], AL
MOV     [tx.txxstate], OFFSET CS:cmtxlenmsb
JMP     @@exit

cmtxlenmsb:
MOV     AL, [BYTE HIGH tx.txxpktlen]
OUT     DX, AL
XOR     [tx.txxlrc], AL
MOV     [tx.txxstate], OFFSET CS:cmtxstype
JMP     @@exit

cmtxstype:
MOV     AL, [tx.txxstype]
OUT     DX, AL
XOR     [tx.txxlrc], AL
MOV     [tx.txxstate], OFFSET CS:cmtxstype
JMP     @@exit

cmtxstype:
MOV     AL, [tx.txxstype]
OUT     DX, AL
XOR     [tx.txxlrc], AL
MOV     [tx.txxstate], OFFSET CS:cmtxdata
JMP     @@exit

cmtxdata:
MOV     SI, [tx.txxbufp]           ; transmit the next byte
ADD     SI, [tx.txxdatcnt]
MOV     AL, [SI]
OUT     DX, AL
XOR     [tx.txxlrc], AL           ; update the LRC
INC     [tx.txxdatcnt]           ; increment data byte index

```

```

    DEC     [tx.txxpktlen]           ; decrement data bytes to transmit
    JNZ     @@exit                   ; and exit if more to send
                                           ; transmission complete,
    CMP     [tx.txxpkttyp], CMTXMDPKT ; transmitting modem packet?
    JNE     cmtxsetetx               ; no, data packet, set to finish tx
    MOV     [tx.txxstate], OFFSET CS:cmtxcomp ; yes, transmission complete
    JMP     @@exit

cmtxsetetx:
    MOV     [tx.txxstate], OFFSET CS:cmtxetx ; or set next state tx data packet
    JMP     @@exit

cmtxetx:
    MOV     AL, DLETX
    OUT     DX, AL
    XOR     [tx.txxlrc], AL
    MOV     [tx.txxstate], OFFSET CS:cmtxcomp
    JMP     @@exit

cmtxlrc:
    MOV     AL, [tx.txxlrc]
    OUT     DX, AL
    MOV     [tx.txxstate], OFFSET CS:cmtxcomp
    JMP     @@exit

cmtxack:
    MOV     AL, DLACK
    OUT     DX, AL
    MOV     [tx.txxstate], OFFSET CS:cmtxcomp
    JMP     @@exit

cmtxnak:
    CALL    cmftxnak
    JMP     @@exit

cmtxenq:
    CALL    cmftxenq
    JMP     @@exit

cmtxetot:
    MOV     AL, DLEOT
    OUT     DX, AL
    MOV     [tx.txxstate], OFFSET CS:cmtxcomp
    JMP     @@exit

cmtxcomp:
    MOV     [tx.txxtxing], 0           ; transmission complete
    flag    ; clear transmission in progress
    MOV     AX, [tx.txxxtst]           ; move onto the next state function
    MOV     [sngstftn], AX
    JMP     @@exit

cmtxrst:
    MOV     [tx.txxtxing], 0           ; transmitter timeout
    flag    ; clear transmission in progress
    MOV     [sngstftn], OFFSET snfsrst ; next state: reset Sentinel

@@exit:
    RET

ENDP cmftx

    ASSUME NOTHING

```

```

*****
;
; * CMFPACK - process expected ACK
;
; * PURPOSE:
; *   This functions tests for an acknowledgement from the CT Server.
;
; * PARAMETERS:
; *   None
;
; * RETURNS:
; *   Nothing
;
; * NOTE:
;
*****

    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC cmfpack NEAR

    CMP     AL, DLACK                ; ACK received?
    JNE     @@testnak               ; no, test for NAK

    MOV     [rx.rxxstate], OFFSET cmfpstx ; yes, transfer complete go
    RETN                                     ; await another packet

@@testnak:
    CALL    cmfpnak                 ; treat as potential NAK

@@exit:
    RETN

ENDP cmfpack
    ASSUME NOTHING

;
; * CMFPNAK - process NAK
;
; * PURPOSE:
; *   This functions tests for a negative-acknowledgement from the CT Server.
;
; * PARAMETERS:
; *   AL contains the character that may be a NAK
;
; * RETURNS:
; *   Nothing
;
; * NOTE:
;
*****

    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC cmfpnak NEAR

    CMP     AL, DLNAK                ; NAK received?
    JNE     @@exit                   ; no, exit

```

```

@@cont:
    MOV     BX, OFFSET sngsenum_str      ; point to string to send
    CALL    ComTransInit                ; initiate retransmission
    MOV     [sngstftn], OFFSET snftxchkin

@@exit:
    RETN
ENDP cmfpnak
    ASSUME NOTHING

;*****
;
; * CMFPSTX - process STX
; *
; * PURPOSE:
; *   This functions tests for a start-of-text character.
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *
;*****

    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC cmfpstx NEAR

    CMP     AL, DLSTX                    ; STX received?
    JE      @@cont                      ; yes, continue

    CALL    cmfstrx                      ; no, reset receiver
    RETN                                  ; exit

@@cont:
    MOV     [rx.rxxlrc], 0               ; clear LRC checksum
    MOV     [rx.rxxstate], OFFSET cmfplen1 ; set next state

@@exit:
    RETN

ENDP cmfpstx
    ASSUME NOTHING

;*****
;
; * CMFPLEN1 - process first byte of length
; *
; * PURPOSE:
; *   This functions accepts the least significant byte of the length field of
; *   a packet.
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
;*****

```

```

; *
; * NOTE:
; *
; *****
;
    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC cmfplen1 NEAR

    MOV     [BYTE LOW rx.rxxpktlen], AL    ; store LSB of length
    XOR     [rx.rxxlrc], AL               ; update LRC

    MOV     [rx.rxxstate], OFFSET cmfplen2 ; set next state

@@exit:
    RETN

ENDP cmfplen1
    ASSUME NOTHING
; *****
; *
; * CMFPLEN2 - process second byte of length
; *
; * PURPOSE:
; *   This functions accepts the most significant byte of the length field of
; *   a packet.
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *
; *****

    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC cmfplen2 NEAR

    MOV     [BYTE HIGH rx.rxxpktlen], AL   ; store LSB of length
    XOR     [rx.rxxlrc], AL               ; update LRC

    MOV     [rx.rxxstate], OFFSET cmfptype ; set next state

@@exit:
    RETN

ENDP cmfplen2
    ASSUME NOTHING
; *****
; *
; * CMFPTYPE - process packet type
; *
; * PURPOSE:
; *   This functions accepts the packet type field.
; *
; *

```

```

; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *
; *****
;
; ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC cmfptype NEAR
    MOV     [rx.rxxstype], AL           ; store packet type
    XOR     [rx.rxxlrc], AL            ; update LRC
    DEC     [rx.rxxpktlen]              ; decrement bytes remaining
    MOV     [rx.rxxstate], OFFSET cmfpstyp ; set next state
@@exit:
    RETN
ENDP cmfptype
    ASSUME NOTHING
; *****
; *
; * CMFPSTYP - process packet subtype
; *
; * PURPOSE:
; *   This functions accepts the packet subtype field.
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *
; *****
;
; ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC cmfpstyp NEAR
    MOV     [rx.rxxstype], AL           ; store packet subtype
    XOR     [rx.rxxlrc], AL            ; update LRC
    DEC     [rx.rxxpktlen]              ; decrement bytes remaining
    JNZ     @@cont                      ; continue if more data
    MOV     [rx.rxxstate], OFFSET cmfpetx ; expect ETX next if over
    JMP     @@exit
@@cont:
    MOV     [rx.rxxstate], OFFSET cmfpdata ; set next state
    MOV     [rx.rxxbytcnt], 0           ; clear the received byte count
@@exit:
    RETN

```

```

ENDP cmfpstyp
    ASSUME NOTHING

;*****
;
;
; * CMFPDATA - process packet data
;
; * PURPOSE:
; *   This functions accepts the packet data field.
;
; * PARAMETERS:
; *   None
;
; * RETURNS:
; *   Nothing
;
; * NOTE:
;
;*****

    ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfpdata NEAR

    MOV     SI, [rx.rxxbufp]           ; get offset to store data
    ADD     SI, [rx.rxxbytcnt]         ;
    MOV     [SI], AL                  ; store packet data
    XOR     [rx.rxxlrc], AL           ; update LRC
    INC     [rx.rxxbytcnt]            ; increment data byte count

    DEC     [rx.rxxpktlen]             ; decrement bytes remaining to
receive
    JNZ     @@exit                    ; and exit if more to come
    MOV     [rx.rxxstate], OFFSET cmfpctx ; or set next state if finished

@@exit:
    RETN

ENDP cmfpdata
    ASSUME NOTHING

;*****
;
;
; * CMFPCTX - process CTX
;
; * PURPOSE:
; *   This functions accepts the packet CTX delimiter.
;
; * PARAMETERS:
; *   None
;
; * RETURNS:
; *   Nothing
;
; * NOTE:
;
;*****

    ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING

```

```

PROC cmfpetx NEAR
    XOR    [rx.rxxlrc],AL           ; update LRC
    CMP    AL,DLETX                 ; test for ETX
    JE     @@cont                   ; ETX rx'd, continue

    CALL   cmfrstrx                 ; ETX not rx'd, reset rx'r
    JMP    @@exit

@@cont:
    MOV    [rx.rxxstate], OFFSET cmfplrc ; set next state

@@exit:
    RETN

ENDP cmfpetx
    ASSUME NOTHING

;*****
;
; * CMFPLRC - process LRC
; *
; * PURPOSE:
; *   This functions accepts the packet LRC checksum.
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *
;*****

    ASSUME CS:SENTL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfplrc NEAR
    IF 0
        CMP    AL,[rx.rxxlrc]       ; test for valid LRC
        JE     @@cont               ; LRC valid, continue
        CALL   cmfrstrx             ; LRC invalid, reset rx'r
        RETN                        ; and exit
    ELSE
        CMP    AL,0                 ; test for 0 for now
        JE     @@cont               ; LRC valid, continue

    @@nak:
        MOV    [sngstftn], OFFSET cmftx ; LRC invalid, send a NAK
        MOV    [tx.txpkttyp], CMTXDLNAK ; set next state: transmit
        RETN                        ; set packet type: send NAK
    ENDIF

    @@cont:
        MOV    [sngstftn], OFFSET cmftx ; set next Sentinel state function
        MOV    [tx.txpkttyp], CMTXDLACK ; transmitting an ACK
        MOV    [tx.txxtmr], TM1SEC      ; give tx one second to complete
        MOV    [tx.txnxtst], OFFSET cmfprdata ; set state fuction following tx

    @@exit:

```

```

    RETN

ENDP cmfplrc
    ASSUME NOTHING

;*****
;
; * CMFGETNEXT - get next call date
; *
; * PURPOSE:
; *   This functions extracts the next call date from a received packet.
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *
;*****

    ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfgetnext NEAR

    PUSH DS
    POP ES
    MOV DI, OFFSET next_call_date
    MOV SI, [rx.rxxbufp]
    CLD
    MOV CX, 5
    REP MOVSB
    INC [sngdskwrt]
    MOV [sngclst], SNCALLPASS
    MOV AX, [sngmdmprt]
    MOV [modem_default_port], AX

    ; get ES = DS
    ; ES:DI points to next_call_date
    ; DS:SI points to received data
    ; move up through pointers
    ; copy five bytes of BCD data:
    ;   YYMMDDHHMM
    ; copy the new date/time
    ; set the disk write flag
    ; set the call status
    ; set default modem for next call
    ; call based upon current port

@@exit:
    RETN

ENDP cmfgetnext
    ASSUME NOTHING

IF 0
;*****
;
; * CMFDISABLE - disable Sentinel
; *
; * PURPOSE:
; *   This functions disables the Sentinel based upon a packet received from
; *   the tracking server. The Sentinel is disabled by recording a call date
; *   and time of 0xFFFFFFFF.
; *
; * PARAMETERS:
; *
;*****

```

```

; *
; * None
; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *
; *****
;
; ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC cmfdisable NEAR
    PUSH    DS
    POP     ES
; get ES = DS

    MOV     DI, OFFSET next_call_date
    MOV     SI, OFFSET rx.rxxdata
; ES:DI points to next_call_date
; DS:SI points to received data

    CLD
    MOV     CX, 5
; move up through pointers
; copy five bytes of BCD data:
;       YYMMDDHHMM
    REP     MOVSB
; copy the new date/time

    INC     [sngdskwrt]
; set the disk write flag

@@exit:
    RETN

ENDP cmfdisable
    ASSUME NOTHING
ENDIF
; *****
; *
; * CMFPRSDATA - parse received data
; *
; * PURPOSE:
; *   This functions parses received data and takes appropriate action.
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *
; *****
;
; ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING
PROC cmfprsddata NEAR
    MOV     AL, [rx.rxxstype]
; test for valid data type
    CMP     AL, SNSERVER
    JNE     @@reset

    MOV     AL, [rx.rxxstype]
; test for valid subtype
    CMP     AL, SNNEXTCALL
; test for next call packet

```

```

JNE  @@nxtest1
CALL  cmfgetnext          ; extract next call date from packet
JMP   @@reset

@@nxtest1:
IF 0
CMP   AL, SNDISABLE      ; test for disable packet
JNE   @@nxtest2          ; disable Sentinel
CALL  snfdisable
ENDIF

@@reset:
CALL  cmfrstrx           ; reset receiver

@@exit:
RETN

ENDP cmfprdata
ASSUME NOTHING

;*****
;
; * CMFRSTRX - reset the receiver
; *
; * PURPOSE:
; *   This functions resets the receiver.
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * NOTE:
; *
;*****

ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfrstrx NEAR

MOV   [rx.rxxstate],OFFSET cmfpstx ; reset receiver state machine
MOV   [sngstftn],OFFSET snfsnrst  ; reset the Sentinel to active mode
MOV   [Sentinel_state],SNSTACTIVE

@@exit:
RETN

ENDP cmfrstrx
ASSUME NOTHING

ENDS

END

```

```

;*****
;
;* Copyright (c) Absolute Software 1994, 1995
;*
;* SNTLCOMV.ASM
;*
;* Contains the comm ISR routine.
;*
;* HISTORY:
;* 1995.09.05 - CCOTI
;* Created.
;*
;*****
*

IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLCOMV.INC"
include "SNTLDATA.INC"
include "SNTLBUFF.INC"
include "SNTLAPI.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
;
; CMPISR - communications interrupt service routine
;
; PURPOSE:
; This function implements the communications ISR that supports both
; receiving and transmitting data. This function hooks the system's
; communications port interrupt (IRQ 4/3).
;
; PARAMETERS:
; None
;
; RETURNS:
; Nothing
;
; GLOBALS REFERENCED:
; sngmdmptadd
;
; GLOBALS MODIFIED:
; sngincmistr - Incremented on entrance, decremented on exit.
; sngstftn - set to error handler if error is detected.
;
; BIOS CALLS:
; None
;
; DOS CALLS:
; NONE
;
; PROCEDURE CALLS:
; cmftxbyte, buf_putchar
;
; HARDWARE ACCESS:
; UART (IN IIR, I/O MCR, IN RDR)
;*****
;

```

```

        ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC cmfiser FAR

    PUSH    AX                ; save registers
    PUSH    DX
    PUSH    SI
    PUSH    DS

    PUSH    CS                ; set DS
    POP     DS
    ASSUME DS:SNTL_SEG

    INC     [sngincmisr]      ; set ISR in progress flag

IFDEF Debug
    INC     [sngcomcnt]      ; increment comm ISR count
ENDIF

@@check_iir:
; Check the reason for the call (error, ready to send, data received).
    MOV     DX,[sngmdmptadd] ; get interrupt identification
register
    ADD     DX,IIR
    IN      AL,DX

    TEST    AL,00000100b     ; test for receive interrupt
    JNZ     DataReceive     ; proceed with data reception

    TEST    AL,00000010b     ; test for transmit interrupt
    JNZ     DataSend        ; proceed with data transmission

@@error:
IFDEF Debug
    INC     [sngcomerr]
ENDIF
; Check the status of the error.
    MOV     DX,[sngmdmptadd] ; reading the register clears the
error
    ADD     DX, LSR
    IN      AL, DX
    JMP     @@end

DataSend:
    CALL    cmftxbyte
    JMP     @@end

DataReceive:
; First, turn off RTS.
    MOV     DX,[sngmdmptadd]
    ADD     DX, MCR          ; Move DX to MCR.
    IN      AL, DX
    IODelay
    AND     AL, 11111101b    ; turn off RTS
    OUT     DX, AL
    IODelay

Receive:
IFDEF Debug
    INC     [received_count]
ENDIF
    MOV     DX,[sngmdmptadd] ; DX = RDR.
    IN      AL, DX          ; AL = received byte.
    IODelay

```

```

CALL    buf_putchar          ; Put the byte into the buffer.

; Check if there is another request pending.
ADD     DX, 2                ; Move to IIR reg.
IN      AL, DX
IODELAY
TEST    AL, 00000001b
JZ      @@check_iir

@@end:
MOV     AL, 20h              ; signal end of interrupt to PIC
OUT     20h, AL

MOV     DX, [sngmdmprtadd]   ; get the modem control register
ADD     DX, MCR
IN      AL, DX
IODELAY

OR      AL, 00000010b        ; turn RTS back on
OUT     DX, AL              ; set the modem control register
IODELAY

DEC     [sngincmisr]         ; clear ISR in progress flag

ASSUME DS:NOTHING

POP     DS                  ; recover registers
POP     SI
POP     DX
POP     AX
IRET                        ; exit

ENDP cmfisir
ASSUME NOTHING

ENDS

END

```

```

;*****
;
;* Copyright (c) Absolute Software 1994, 1995
;*
;* SNTLDATA.ASM
;*
;* Contains the global data segment for the sentinel.
;*
;* HISTORY:
;*   1995.09.05 - CCOTI
;*               Created.
;*
;*****

IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLTIMR.INC"
include "SNTLJTBL.INC"
include "SNTLCOMM.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

; Transient variables *****

    sngstftn          DW NEAR PTR OFFSET ActiveRoutine    ; CCOTI
    Sentinel_state    DB SNSTACTIVE

; Scatch vars to store the current port info being used.
    sngmdmpt          DW ?
    sngmdmptint       DW ?
    sngmdmptadd       DW ?

; Previous ISR vectors.
    sngprvtmr         DD FAR PTR 0
    sngprvcom         DD FAR PTR 0
    sngprvdekl        DD FAR PTR 0
    sngprvint2f       DD FAR PTR 0

; ROR'd to limit updating the real-time clock once every 16 ticks (see
ActiveRoutine).
    cycle_var         DW 0001h

    win_flag          DB 0
    win_vm            DB 1

    sngincmisr        DB 0

    send_buf_len      DW 0
    send_buf_ptr      DW BYTE PTR 0

    sngcomcnt         DW 0 ; comm. interrupt count
    sngcomerr         DB 0 ; comm. error count
    TimerISR_count    DW 0 ; timer interrupt count
    sent_count        DW 0 ; bytes transmitted
    received_count     DW 0 ; byte received
    sngflcnt          DB 0
    sngclst           DB SNCALLNA
    sngcomhk          DB 0
    sngsuspend        DB 0

```

```

sngdlytmr          DW 0
sngint2ftmr        DW TM2MIN ; wait 2 minutes for an XMS manager
sngprtdlytmr       DW 0
sngdeflect         DB 1 ; Sentinel disk deflection
flag
  dkgcyl           DW ? ; disk access cylinder
  dkgsctr          DB ? ; disk access sector
  sngapifl         DB 0 ; API fialed request count
  sngpwd1          DW 'FO' ; API request user password1
  sngpwd2          DW 'AD' ; API request user password2
; Port info..
modem_default_port DW 0
port_table         DW 03F8h, 000Ch, \
                  02F8h, 000Bh, \
                  0000h, 0000h, \
                  02E8h, 000Dh
PORT_TABLE_SIZE = 4
; Disk location of data sector.
data_cyl_sect      DW 0
data_head_drive    DW 0
sngdskwrt         DB 0
; Output strings.
init_str_num       DW 0
init_str_table     DW 5 DUP( 0 )
INIT_STR_TABLE_SIZE = 6
dial_str_num       DW 0
dial_str_table     DW 4 DUP( 0 )
DIAL_STR_TABLE_SIZE = 5
dial_number_start  DB "18003396122", 0Dh
LABEL dial_number  BYTE
dial_number_len    DB 12
LABEL sn_packet_start UNKNOWN
stx_byte           DB 02h
lsb_length_byte    DB ?
msb_length_byte    DB ?
LABEL sn_text_start UNKNOWN
text_type          DB 0
text_sub_type      DB 0
LABEL sn_data_start UNKNOWN
sngsernum          DB 6 DUP( 0 )
LABEL now_date     UNKNOWN
now_year           DB 1
now_month          DB 1
now_day            DB 1
now_hour           DB 1
now_minute         DB 1
LABEL sn_data_end  UNKNOWN
etx_byte           DB 03h
lrc_byte           DB ?
LABEL sn_packet_end UNKNOWN
LABEL sngsernum_str UNKNOWN
sngsernum_str_len  DB sn_packet_end - sn_packet_start
sngdatalen         DB sn_data_end - sn_data_start

```

```

;END MOD

rx          RXZCM    < ; initialize receive structure
               < OFFSET cmfpack, \
               0, 0, 0, 0, 0, 0, \
               OFFSET CS:nextcall_text >

tx          TXZCM    < ; initialize transmit structure
               < 0, 0, 0, 0, 0, 0, 0, 0, \
               0, 0, 0, 0, \
               OFFSET CS:sngtxbuf >

; Result tables.
command_result_table_len DB 3
command_result_table     DW 3 DUP( 0 )

mdm_init_result_table_len DB 2
mdm_init_result_table     DW 2 DUP( 0 )

dial_result_table_len    DB 6
dial_result_table        DW 6 DUP( 0 )

connect_result_table_len DB 4
connect_result_table     DW 10 DUP( 0 )

; Modem and result string pool.
string_pool              DB 127 DUP( 0 )

modem_find_str_start     DB 'ATZ', 0Dh
LABEL modem_find_str     UNKNOWN
modem_find_str_len       DB 4

; next call date
LABEL next_call_date     UNKNOWN
next_call_year           DB OFFh
next_call_month          DB OFFh
next_call_day            DB OFFh
next_call_hour           DB OFFh
next_call_minute         DB OFFh

sngrxbufhd              DW 0 ; receive buffer
sngrxbufhl              DW 0
LABEL sngrxbufst         UNKNOWN
sngrxbuf                DB 80h DUP( 0 )
LABEL sngrxbufend        UNKNOWN

nextcall_text           DB 05h DUP( 0 )

sngtxindex              DB 0 ; transmit buffer
LABEL sngtxbufst         UNKNOWN
sngtxbuf                DB 7Bh DUP( 0 )
LABEL sngtxbufend        UNKNOWN

; Result jump tables.
; Table for ModemFind
find_jump_table         DW NEAR PTR find_timeout ; TIMEOUT
                        DW NEAR PTR find_ok       ; NO CARRIER (NOTE
1)
                        DW NEAR PTR find_timeout ; ERROR
                        DW NEAR PTR find_ok       ; OK

; NOTE 1: 29 March 1995 - DBOYD

```

```

;
;   USR modem (and maybe others) does not return <NO CARRIER>
;   when the server disconnects.  <NO CARRIER> returned when next
;   signal (command or control line) sent to modem.  Sometimes
this      response is sent before the next command, sometimes after.
When      the Sentinel receives this response to a modem query (<AT>) it
;         should interpret it as <OK>.

; Table for ModemInit.
init_jump_table      DW NEAR PTR init_error      TIMEOUT
                    DW NEAR PTR init_error      ERROR
                    DW NEAR PTR init_ok         OK

; Table for dial results.
dial_jump_table      DW NEAR PTR dial_error      TIMEOUT
                    DW NEAR PTR dial_error      ERROR
                    DW NEAR PTR dial_busy       BUSY
                    DW NEAR PTR dial_no_tone    NO DIAL TONE
                    DW NEAR PTR dial_no_carr    NO CARRIER
                    DW NEAR PTR dial_server     Server Query
(NAK)
(ENQ)                DW NEAR PTR dial_server     Server Query

cnct_jump_table      DW NEAR PTR cnct_error      TIMEOUT
                    DW NEAR PTR cnct_error      NO CARRIER
                    DW NEAR PTR cnct_eot        Server EOT
                    DW NEAR PTR cnct_enq        Server ENQ
                    DW NEAR PTR cnct_nak        Server NAK
                    DW NEAR PTR cnct_ack        Server ACK

ENDS

include "SNTLDATA.INC"

END

```

```

;*****
;
; * Copyright (c) Absolute Software 1994, 1995
;
; * SNTLI13V.ASM
;
; * Contains INT 13 ISRs and disk deflection routines.
;
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
;*****

IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLI13V.INC"
include "SNTLDATA.INC"
include "SNTLI2FV.INC"
include "SNTLTIMR.INC"
include "SNTLAPI.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'
;*****
;
; DKFDLDRD - Disk deflect read
;
; PURPOSE:
;   This function performs disk read deflections by filling up the
; destination
;   buffer with erroneous characters.
;
; PROTOTYPE:
;
; PARAMETERS:
;   AL = number of sectors to read (must be non-zero)
;   CH = low eight bits of cylinder number
;   CL = sector number 1-63 (bits 0-5)
;         high two bits of cylinder number (bits 6-7, hard disk only)
;   DH = head number
;   DL = drive number (bit 7 set for hard disk)
;   ES:BX => data destination
;
; RETURNS:
;   The flags register as set by the ROM interrupt 13 handler:
;   - CF = 0 if successful
;   AH = status
;   AL = number of sectors transferred
;
; NOTE:
;*****
;
; ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC dkfdflrd NEAR

```

```

MOV     DI, BX                ; get offset of destination buffer
PUSH    AX                   ; store disk access parameters
PUSH    DS                   ; store register
PUSH    CS                   ; set DS:SI pointer
POP      DS
MOV     SI, OFFSET fillr

@@dflloop:                    ; deflect loop
CLD                          ; set pointers to increment
MOV     CX, 100h              ; fill 256 words (512 bytes = 1
sector)

@@dfslct:                    ; single sector deflection
MOVSW                          ; copy filler to destination
DEC     SI                   ; decrement source pointer
DEC     SI                   ; by 2 for word moves
LOOP    @@dfslct

DEC     AL                   ; decrement the number of sectors to
fill   JNZ    @@dflloop

POP     DS                   ; restore register
POP     AX                   ; restore disk access parameters
MOV     AH, 0                ; set success parameters and exit
CLC
RET

fillr:
FILL    EQU    0f6f6h

ENDP    dkfdflrd
ASSUME  NOTHING

```

```

;*****
;
; DKFDLMBR - Disk deflect MBR access
;
; PURPOSE:
;   This function performs disk deflection on attempted access to MBR
;   sector.
;   Access is deflected from our subloader in the MBR to the original MBR.
;
; PROTOTYPE:
;
; PARAMETERS:
;   AH = disk function: 0x02 = disk read
;                       0x03 = disk write
;   AL = number of sectors to read (must be non-zero)
;   CH = low eight bits of cylinder number
;   CL = sector number 1-63 (bits 0-5)
;         high two bits of cylinder number (bits 6-7, hard disk only)
;   DH = head number
;   DL = drive number (bit 7 set for hard disk)
;   ES:BX => the data source (writes) or data destination (reads)
;
; RETURNS:
;   The flags register as set by the ROM interrupt 13 handler:
;   - CF = 0 if successful
;   AH = status
;   AL = number of sectors transferred

```

NOTE:

ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC dkfdflmbr NEAR

```

    CMP     AH, 02h           ; read access to MBR?
    JE      @@dfmlbrd        ; yes, deflect read
    CMP     AH, 03h           ; write access to cylinder 0?
    JE      @@dfmlbrwrt      ; yes, deflect write

@@dfmlbrd:
    PUSH    CX               ; save disk access parameters
    PUSH    AX
    MOV     CX, 0002h        ; load CX to access deflected MBR
    MOV     AL, 1            ; load AL to access a single sector
    PUSHF                    ; push flags because IRET
    CALL    [DWORD CS:sngprvdskl] ; from original handler pops flags
    JNC     @@dfldrctnt      ; error?, no, continue
    POP     CX               ; yes, recover access parameters
    POP     CX               ; discard original AX
    JMP     @@exit           ; exit

@@dfldrctnt:
    POP     AX               ; recover disk access parameters
    POP     CX
    MOV     AH, 0            ; set success indication
    CMP     AL, 1            ; all sectors read?
    JE      @@exit          ; yes, exit
                                ; no, load crap to the next 10
sectors
    PUSH    AX               ; save disk access parameters
    MOV     AX, ES           ; increment destination buffer by
    ADD     AX, 200h         ; by 512 (512 bytes = 1 sector)
    MOV     ES, AX
    POP     AX               ; recover disk access parameters

    DEC     AL               ; fill one less sector than required
    CALL    dkfdflrd        ; fill destination buffer with junk

    PUSH    AX
    MOV     AX, ES
    SUB     AX, 200h
    MOV     ES, AX
    POP     AX
    INC     AL               ; increment number of sectors read
    MOV     AH, 0            ; set success indication
    CLC
    JMP     @@exit          ; exit

@@dfmlbrwrt:
    PUSH    CX               ; save disk access parameters
    PUSH    AX
    MOV     CX, 0002h        ; load CX to access deflected MBR
    MOV     AL, 1            ; load AL to access a single sector
    PUSHF                    ; push flags because IRET
    CALL    [DWORD CS:sngprvdskl] ; from original handler pops flags
    JNC     @@dfldrctnt     ; error? no, continue
    POP     CX               ; yes, recover access parameters

```

```

        POP        CX                ; discard original AX
        JMP        @@exit            ; exit

@@dfldrwtcnt:
        MOV        AH, 2             ; read in the (possibly) modified
        MOV        CX, 0002h         ; image of true MBR
        MOV        AL, 1
        PUSHF
        CALL       [DWORD CS:sngprvdskl] ; push flags because IRET
        JC         @@exit            ; from original handler pops flags
                                        ; error? yes, exit

        PUSH       DS               ; get copy of partition table
        PUSH       ES               ; save register
        PUSH       ES               ; save disk access parameter
        POP        DS               ; set DS
        PUSH       CS               ; set ES
        POP        ES
        MOV        AX, BX
        ADD        AX, 0FCh
        MOV        SI, AX
        MOV        DI, OFFSET sngrxbuf ; get a pointer to a buffer
        MOV        CX, 100h         ; prepare to move 256 bytes
        REP        MOVSB            ; do the move

        POP        ES               ; get copy of subloader
        MOV        AH, 2             ; restore disk access parameter
        MOV        CX, 0001h         ; read subloader from the MBR
        MOV        AL, 1
        PUSHF
        CALL       [DWORD CS:sngprvdskl] ; push flags because IRET
        JC         @@exit2          ; from original handler pops flags
                                        ; error? yes, exit

        copy partition table into

subloader
        PUSH       CS               ; set DS
        POP        DS
        MOV        SI, OFFSET sngrxbuf ; DS:SI => partition table in

subloader
        MOV        AX, BX
        ADD        AX, 0FCh
        MOV        DI, AX
        MOV        CX, 100h         ; ES:DI => partition table in MBR
        REP        MOVSB            ; prepare to move 256 bytes
                                        ; do the move

        MOV        AH, 3             ; write the subloader
        MOV        CX, 0001h
        MOV        AL, 1
        PUSHF
        CALL       [DWORD CS:sngprvdskl] ; push flags because IRET
        JC         @@exit2          ; from original handler pops flags
                                        ; error? yes, exit

        MOV        AH, 2             ; read new MBR back into ES:BX
        MOV        AL, 1
        PUSHF
        CALL       [DWORD CS:sngprvdskl] ; push flags because IRET
        JC         @@exit2          ; from original handler pops flags
                                        ; error? yes, exit

        POP        DS               ; recover register
        POP        AX               ; recover disk access parameters
        POP        CX
        MOV        AH, 0             ; set success indication

```

```

        CLC
        JMP      @exit

@@exit2:
        POP      DS                ; if exiting due to disk write
                                   ; deflection error

@@exit:
        RET

ENDP dkfdflmbr
        ASSUME NOTHING

```

INT13ISR - Sentinel interrupt 13 ISR

PURPOSE:

This function provides the Sentinel's interrupt 13 hook for disk access. It serves two purposes: to store next-call information to disk after a transaction with the Sentinel server, and to prevent disk reads of the sectors that contain the Sentinel.

After a tracking transaction with the server, the Sentinel will have received a next-call-date that must be recorded to disk. The Sentinel disk access is piggy-backed onto a disk read or write to the disk that the Sentinel is installed on.

If a program (such as a Norton Disk Editor or Anit-Virus) attempts to read a section of the hard disk that the Sentinel occupies, this function will deflect the read to the original code that occupied the Sentinel's disk space.

Disk access other than read/writes is passed through to the original interrupt 13h handler.

PROTOTYPE:

PARAMETERS:

AH = disk function: 0x02 = disk read
 0x03 = disk write
AL = number of sectors to read (must be non-zero)
CH = low eight bits of cylinder number
CL = sector number 1-63 (bits 0-5)
 high two bits of cylinder number (bits 6-7, hard disk only)
DH = head number
DL = drive number (bit 7 set for hard disk)
ES:BX => the data source (writes) or data destination (reads)

RETURNS:

The flags register as set by the ROM interrupt 13 handler:

- CF = 0 if successful
- AH = status
- AL = number of sectors transferred

NOTE:

ASSUME CS:SENTL_SEG, DS:NOTHING, ES:NOTHING

```

OFFSET_TO_PREHANDLER = PreInt13Handler - JMP_REL_OFFSET
OFFSET_TO_FULLHANDLER = FullInt13Handler - JMP_REL_OFFSET

load_time      DW ?                ; the time the system loaded.

PROC Int13ISR FAR
    JMP SHORT_REL_OPCODE DB 0EBh
    Int13_RelOffset DB OFFSET_TO_PREHANDLER
JMP_REL_OFFSET:

PreInt13Handler:
    PUSH    AX
    PUSH    ES
    PUSH    DS
    PUSH    CS
    POP     DS
    ASSUME DS:SNTRL_SEG
; Check for an XMS manager.
    MOV     AX, 4300h
    INT     2Fh
    CMP     AL, 80h
    JE      @@XMS_Detected        ; XMS loaded, re-hook INT 2Fh.
; Check for timeout.
    MOV     AX, 0040h
    MOV     ES, AX                ; ES = bios segment.
    MOV     AX, [ES:006Ch]        ; Load current bios time.
    SUB     AX, [load_time]       ; Find delta since sntlinit.
    CMP     AX, PREINT13_TIMEOUT ; Check for timeout.
    JMP     @@jmp_to_full_isr
    JB      @@jmp_to_full_isr     ; If timeout, continue and hook
sentinel.

@@XMS_Detected:
    PUSH    BX
    PUSH    CX
    PUSH    DI

@@hook2F:
    MOV     BX, 002Fh
    MOV     DI, OFFSET sngprvint2f
    MOV     CX, OFFSET snfint2f
    CALL    SwapInt

@@hook1C:
    MOV     BX, 001Ch
    MOV     DI, OFFSET sngprvtmr
    MOV     CX, OFFSET tmfiser
    CALL    SwapInt
; Enable full int13 handler.
    MOV     [Int13_RelOffset], OFFSET_TO_FULLHANDLER
    POP     DI
    POP     CX
    POP     BX

@@jmp_to_full_isr:
    ASSUME DS:NOTHING
    POP     DS
    POP     ES
    POP     AX
    JMP     [DWORD CS:sngprvdsk1] ; pass control to original handler

FullInt13Handler:
    IF TWODSKHKS

```

```

test?    CMP      [CS:sngdskskip], 0      ; this invocation directed to skip
disk hook JNE      @passthru1             ; yes, pass control through to first
hook      MOV      [CS:sngdskskip], 1      ; set flag for (possible) second
ENDIF

@@dsktst1:
OR        AL, AL                          ; is the sector quantity zero?
JNZ       @@dsktst2                       ; no, continue
JMP       @passthru                       ; pass control through

@@dsktst2:
CMP       [CS:sngdeflect], 1              ; disk deflection enabled?
JNE       @piggyback                      ; no, check for piggy-back access

@@dsktst3:
CMP       DX, 0080h                       ; access to Sentinel head and drive?
JNE       @piggyback                      ; no, check for piggy-back access

@@dsktst4:
CMP       CX, 000Ch                       ; access to first 12 sectors?
location) JA      @piggyback              ; (MBR subloader and Sentinel)
; no, check for piggy-back access

PUSH     BX                              ; save important register
MOV      [BYTE LOW CS:dkgcyl], CH         ; get the cylinder
MOV      BL, CL
SHR      BL, 6
AND      BL, 03h
MOV      [BYTE HIGH CS:dkgcyl], BL
MOV      [CS:dkgsctr], CL                ; get the sector
AND      [CS:dkgsctr], 3fh
POP      BX                              ; recover important register

@@deflect:
determined ; at this point it has been
; that the system is attempting to
; access the first 12 sectors of
; cylinder 0 and we must deflect

CMP      [dkgsctr], 1                    ; access starting on MBR?
JE       @@dfmbr                         ; yes, go deflect read/write
CMP      AH, 02h                         ; read access to cylinder 0?
JE       @@dfldr                         ; yes, deflect read
CMP      AH, 03h                         ; write access to cylinder 0?
JE       @@dflwrt                        ; yes, deflect write
JMP      @passthru                       ; pass control through

@@dfmbr:
CALL     dkdfmbr                         ; deflect access from MBR sector
JMP      @return2                        ; to original MBR
; exit

@@dfldr:
CALL     dkdfldr                         ; deflect a read
JMP      @return2                        ; exit

@@dflwrt:
MOV      AH, 0                           ; deflect a write
CLC                                             ; set success parameters and exit
JMP      @return2

```

```

@@piggyback:
    CMP        [CS:sngdskwrt], 1      ; does the Sentinel need disk
access?
    JE         @@contpb               ; yes, continue piggy-back
    JMP        @@passthru             ; pass control through

@@contpb:
; write next-call-date to disk
; at this point we are ready to
; piggy-back onto a write to the
; same drive that the Sentinel is on

IF TWODSKHKS
    CMP        [sng2dskhks], 1      ; are we hooked twiced?
    JE         @@dskacc2             ; yes, execute second handler

@@dskacc1:
    PUSHF
    CALL       [DWORD CS:sngprvds1] ; execute first disk handler
    JC         @@return              ; push flags because IRET
    JMP        @@contpb2             ; from original handler pops flags
    ; exit if disk access error

@@dskacc2:
    PUSHF
    CALL       [DWORD CS:sngprvds2] ; execute second handler
    JC         @@return              ; push flags because IRET
    ; from original handler pops flags
    ; exit if disk access error
ELSE
    PUSHF
    CALL       [DWORD CS:sngprvds1] ; push flags because IRET
    JC         @@return              ; from original handler pops flags
    ; exit if disk access error
ENDIF

@@contpb2:
    PUSHAD
    PUSH       DS
    PUSH       ES
    PUSH       CS
    POP        DS
    POP        CS
    POP        ES
    ASSUME     DS:SNTL_SEG
    MOV        [sngdskwrt], 0        ; clear the Sentinel flag

    MOV        AX, 0301h
    MOV        CX, [data_cyl_sect]   ; Load registers for int13 call.
    MOV        DX, [data_head_drive] ; 03=disk write; 01=1 sector
    MOV        BX, DATA_SECTOR_OFFSET ; set cylinder and sector to write
    ; set the head and drive

    PUSHF
    CALL       [sngprvds1]           ; push flags because IRET
    JC         @@write_error         ; from original handler pops flags
    ; disk access error, jmp here for
now
    JMP        @@cleanup             ; disk write successful

@@write_error:
@@cleanup:
    ASSUME     NOTHING
    POP        ES
    POP        DS
    POPAD

@@return:

```

```

IF TWODSKHKS
    MOV     [CS:sngdskskip], 0      ; clear disk access skip flag
ENDIF
RET     2                          ; discard FLAGS from stack and
return

@@return2:
    ASSUME NOTHING
IF TWODSKHKS
    MOV     [CS:sngdskskip], 0      ; clear disk access skip flag
ENDIF
RET     2                          ; discard FLAGS from stack and
return

IF TWODSKHKS
@@passthru:
    ASSUME CS:SNTL_SEG
    CMP     [CS:sng2dskhks], 0      ; is disk access hooked twice?
    JNE     @@sechandle             ; yes, pass control to second hook
    JMP     [DWORD CS:sngprvdskl]   ; no, pass control to original
handler
@@sechandle:
    PUSHF
    CALL    [DWORD CS:sngprvdskl]
    JMP     @@cleanup

@@passthru1:
    JMP     [DWORD CS:sngprvdskl]   ; earlier disk hook handling access
                                        ; pass control to original handler
                                        ; and it will IRET
ELSE
@@passthru:
    JMP     [DWORD CS:sngprvdskl]   ; cannot piggy-back this time
                                        ; pass control to original handler
ENDIF

ENDP Int13ISR

    ASSUME NOTHING

ENDS

END

```

```

;*****
; * Copyright (c) Absolute Software 1994, 1995
; *
; * SNTLI2FV.ASM
; *
; * PURPOSE:
; *   Contains INT 2F ISRs used by the sentinel.
; *
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
;*****

IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLI2FV.INC"
include "SNTLDATA.INC"
include "SNTLTIMR.INC"
include "SNTLAPI.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
; * Unmovable code.
;*****

;*****
; Routine: Int2FVect
;
; Descript: Provides an API and RPL 2F/4A06 support
;*****

        ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

PROC Int2FVect FAR
    JMP     SHORT @@entry
    NOP
    rpl_sig    DB 'RPL'

@@entry:
    CMP     AX,4A06h
    JNE     @@next_check
    MOV     DX,CS

@@next_check:
    IF 0
        CMP     AX,SNTL_SIG1
        JNE     @@exit
        CMP     DX,SNTL_SIG2
        JNE     @@exit
        MOV     AX,OFFSET CS:SntlAPI
        MOV     DX,CS
    ELSE
        ; must be a Sentinel check
        ; proper signature provided?
        ; no, exit
        ; proper signature provided?
        ; no, exit
        ; yes, return API address
    ENDIF

@@exit:
    IRET

```

```

ENDP Int2FVect
    ASSUME NOTHING

```

```

*****
;
; * SNFINT2F - interrupt 2F hook
;
; * PURPOSE:
; *   This is the interrupt 2F hook that supports the Sentinel API request and
; *   monitors WINDOWS activation/deactivation
;
; * PARAMETERS:
; *   None
;
; * RETURNS:
; *   Nothing
;
; * REGISTERS DESTROYED:
;
; * GLOBALS REFERENCED:
;
; * GLOBALS MODIFIED:
;
; * BIOS CALLS:
; *   None
;
; * DOS CALLS:
; *   None
;
; * PROCEDURE CALLS:
; *   None
;
; * HARDWARE ACCESS:
; *   None
;
*****

```

```

    ASSUME CS:SNL_SEG, DS:NOTHING, ES:NOTHING

```

```

PROC snfint2f FAR

```

```

to load    CMP    AX, 1605h    ; check if WINDOWS is starting
           JNE     @@check1
           MOV     [BYTE CS:sngsuspend], 1    ; suspend Sentinel until
WINDOWS is loaded    PUSHF    ; push flags because IRET from
call       CALL    [DWORD CS:sngprvint2f]    ; to previous handler pops
flags      IRET    ; return from interrupt

@@check1:
           CMP     AX, 1608h    ; check if WINDOWS has
finished loading    JNE     @@check2
           MOV     [BYTE CS:sngsuspend], 0    ; allow Sentinel to resume
           MOV     [BYTE CS:sngdlytmr], 90    ; set the delay timer

           reset after delay
           MOV     [WORD CS:sngstftn], OFFSET snfsnrst

```

103

```

call        PUSHF                ; push flags because IRET from
flags       CALL        [DWORD CS:sngprvint2f] ; to previous handler pops
flag        MOV         [BYTE CS:win_flag], 1 ; set WINDOWS status flag
            MOV         [sngincmisr], 0      ; clear communications ISR
            IRET                ; return from interrupt

@@check2:   CMP         AX, 1606h           ; check if WINDOWS has exited
            JNE         @@check3
            MOV         [BYTE CS:sngsuspend], 0 ; allow Sentinel to resume
            MOV         [BYTE CS:sngdlytmr], 90 ; set the delay timer

; reset after delay
            MOV         [WORD CS:sngstftn], OFFSET snfsnrst
            PUSHF                ; push flags because IRET from
call        CALL        [DWORD CS:sngprvint2f] ; to previous handler pops
flags       MOV         [BYTE CS:win_flag], 0 ; clear WINDOWS status flag
flag        MOV         [sngincmisr], 0      ; clear communications ISR
            IRET                ; return from interrupt

@@check3:   CMP         AX, 1609h           ; check if WINDOWS is starting
exit        JNE         @@check4
            MOV         [BYTE CS:sngsuspend], 1 ; suspend Sentinel until
WINDOWS has exited
            PUSHF                ; push flags because IRET from
call        CALL        [DWORD CS:sngprvint2f] ; to previous handler pops
flags       IRET                ; return from interrupt

@@check4:   CMP         AX, 1607h           ; check if WINDOWS is testing
for 32      JNE         @@check5           ; bit disk access support
            CMP         BX, 0010h
            JNE         @@check5
            CMP         CX, 0003h
            JNE         @@check5
            MOV         CX, 0000h          ; set return value to indicate
32-bit support
            IRET                ; return from interrupt

@@check5:   ; check for API request
            CMP         AX, SNTL_SIG1      ; check for signature in AX:DX
handler     JNE         @@org              ; no match, go to previous
            CMP         DX, SNTL_SIG2
handler     JNE         @@org              ; no match, go to previous

; AX:DX match, but no access yet
requests?   CMP         [sngapifl], 10     ; more than ten failed API
handler     JAE         @@apifail         ; yes, jump to original

```

```

password      CMP     BX, [sngpwd1]      ; check for passwords in BX:CX
               JNE     @bkdr           ; no match, check for backdoor
password      CMP     CX, [sngpwd2]      ; check for passwords in BX:CX
               JNE     @bkdr           ; no match, check for backdoor
password      JMP     @@apipass         ; ok!

@bkdr:        ; check for backdoor access
               CMP     BX, [WORD sngsernum]
               JNE     @@apifail        ; no match, increment failure
count         ;
               CMP     CX, [WORD sngsernum + 2]
               JNE     @@apifail        ; no match, increment failure
count         ;

@@apipass:    MOV     AX, OFFSET CS:SntlAPI ; signature and password match
               MOV     DX, CS           ; return API entry point
               IRET                    ; return from interrupt

@@apifail:    MOV     DX, 0F0ADh        ; alert CTM to presence but
failed access ;
               INC     [sngapif1]      ;
               IRET                    ; return from interrupt

@@org:        JMP     [DWORD CS:sngprvint2f] ; pass control to previous
handler       ;

ENDP snfint2f
               ASSUME NOTHING
ENDS

END

```

```

;*****
;
;* Copyright (c) Absolute Software 1994, 1995
;*
;* SNTLINIT.ASM
;*
;* contains all initialization code that is discarded from memory.
;*
;* HISTORY:
;*   1995.09.05 - CCOTI
;*               File created from the old SNTLINIT.ASM.
;*****

SEGMENT SNTL_INIT_SEG PARA PUBLIC 'CODE'
ENDS

IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLDATA.INC"
include "SNTLI2FV.INC"
include "SNTLI13V.INC"
%LIST

;*****
;
;*
;* SNTL_INIT_SEG - Transient segment.
;*
;*****

SEGMENT SNTL_INIT_SEG PARA PUBLIC 'CODE'
    ASSUME CS:SNTL_INIT_SEG, DS:NOTHING

;*****
; Sntlinit Header

        part_sector    DB 512 DUP( 0 )
        boot_sector    DB 512 DUP( 0 )
        io_sector      DB 512 DUP( 0 )

        SntlSignature   DW SNTL_SIG1, SNTL_SIG2
        JMP             NEAR SntlInit

fdddataseg:
        fdgssihddrv     DW 0000h           ; sentinel source image parameters
        fdgssicylsec     DW 0101h           ; determined with Norton Disk Editor
        fdgssisec       DB 11             ; determined with Norton Disk Editor
                                           ; written by CTM

        fdgstihddrv     DW 0000h           ; sentinel target image parameters
        fdgsticylsec     DW 0000h           ; written by CTM
        fdgstisec       DB 00             ; written by CTM

        fdgslsihddrv     DW 0100h           ; subloader source image parameters
        fdgslsicylsec     DW 0010h           ; determined with Norton Disk Editor
        fdgslsisec       DB 1             ; determined with Norton Disk Editor
                                           ; written by CTM

```

```

        fdgsltihddrv      DW 0000h      ; subloader target image parameters
        fdgslticylsec     DW 0000h      ; written by CTM
        fdgsltisec        DB 0          ; written by CTM

        fdginstall        DB 0          ; written by CTM to activate HDD
        fdgdsckerr        DB 0          ; infection by FDD boot program
                                         ; disk access error count

        fdghddbshd        DW ?          ; HDD Boot Sector head and
drive   fdghddbscs        DW ?          ; HDD Boot Sector cylinder and
sector  fdghddid          DD ?          ; HDD volume ID written by CTM to
prevent                                         ; FDD boot program from infecting
wrong disk

;*****
*
; SntlInit entry points (stack= AX,BX,CX,DX,DS,ES)
SntlInit:
        EMIT      'S'
        PUSH     SI
        PUSH     DI
        PUSH     CS
        POP      DS
        ASSUME   DS:SNTL_INIT_SEG

        XOR      BX, BX                ; copy original MBR over the subloader
        MOV      ES, BX                ; at location 0000:7C00h
        MOV      DI, 7C00h

        MOV      AX, OFFSET part_sector
        MOV      SI, AX                ; SI = sector to copy.
        MOV      CX, 100h              ; 256 words to copy.
        CLD
        REP MOVSW
        EMIT      'M'
; Check if sntlinit is already in memory.
        XOR      BX, BX
        MOV      ES, BX
        MOV      BX, [ES:00BCh]
        MOV      ES, [ES:00BEh]
        CMP      [WORD ES:BX+3], 'PR'
        JNZ      RPL_check_fail
        CMP      [BYTE ES:BX+5], 'L'
        JNZ      RPL_check_fail
RPL_exist:
; Check if the sentinel acknowledges.
        EMIT      'R'
        MOV      AX, SNTL_SIG1
        MOV      DX, SNTL_SIG2
        INT      2Fh
        CMP      DX, SNTL_SIG2
        JNE      exit                  ; Is the sentinel installed?
        EMIT      '!'                  ; Yes, exit now.

RPL_check_fail:

        XOR      AX, AX
        MOV      ES, AX                ; ES = IVT segment.

```

```

; Calculate and assign SNTL_SEG to DS.
    MOV     AX, CS
    MOV     BX, OFFSET SNTL_SEGMENT
    SHR     BX, 4
    ADD     AX, BX
    MOV     DS, AX
    ASSUME DS:SNTL_SEG

; Hook the interrupt handlers into the system:
    CLI     ; DISABLE INTERRUPTS

; Hook 2Fh.
    MOV     [WORD ES:00BCh], OFFSET Int2FVect
    MOV     [WORD ES:00BEh], AX

; Hook 13h.
    MOV     AX, [WORD ES:004Ch] ; first hook of INT 13h to
control disk access
    MOV     [WORD sngprvdskl], AX
    MOV     AX, [WORD ES:004Eh]
    MOV     [WORD sngprvdskl+2], AX
    MOV     [WORD ES:004Ch], OFFSET Int13ISR
    MOV     [WORD ES:004Eh], DS

; *****
    MOV     AX, [WORD ES:0064h] ; hook INT 19h to track reboot
    MOV     [WORD sngprvint19], AX
    MOV     AX, [WORD ES:0066h]
    MOV     [WORD sngprvint19+2], AX
    MOV     [WORD ES:0064h], OFFSET snfint19
    MOV     [WORD ES:0066h], DS

; *****
; like
; code
    MOV     [BYTE ES:03C4h], 'N' ; QEMM requirement
;                               ; to work with QEMM DOSDATA look
    MOV     [BYTE ES:03C5h], 'e' ; a Novell NetWare RPL by loading
;                               ; this string at INT F1h and our
    MOV     [BYTE ES:03C6h], 't' ; segment (less DOS wrapper) at
;                               ; segment portion of INT F3h
    MOV     [BYTE ES:03C7h], 'w' ; Novell puts its INT 13h address
;                               ; at
    MOV     AX, DS ; INT F3h, so try that for our hook
    SUB     AX, 0001h
    MOV     [WORD ES:03CCh + 2], AX
    MOV     [WORD ES:03CCh], OFFSET Int13ISR
; *****

; Initialize runtime variables (if any).
    MOV     AX, [modem_default_port] ; set first port to attempt dial out
    MOV     [sngmdmprt], AX
; Set the load time variable for the preint12 handler.
    MOV     AX, 0040h
    MOV     ES, AX ; ES = bios segment.
    MOV     AX, [ES:006Ch] ; Load current bios time.
    MOV     [load_time], AX

    STI     ; ENABLE INTERRUPTS.

    EMIT     'H'

exit:
; Jump to io.sys
    EMIT     'X'
    POP     DI

```

```

        POP        SI
    ASSUME ES:NOTHING
        POP        ES
    ASSUME DS:NOTHING
        POP        DS
        POP        DX
        POP        CX
        POP        BX
        POP        AX
; Jump back to sector.
        jmpOpcode  DB 0EAh
        EntryPnt   DW 7C00h
        SectSeg     DW 0000h

IF EMIT_ON ;Only needed for EMIT macro =====
;
; Puts the character in AL to the console.
;
PROC PutChar NEAR
    PUSH        AX
    PUSH        BX
    MOV         AH,0Eh                ;Output a character
    MOV         BH,0
    push        bp                    ;TCN - For old BIOS
    INT         10h
    pop         bp                    ;TCN - For old BIOS
    POP         BX
    POP         AX
    RETN
ENDP PutChar

ENDIF ;EMIT_ON=====

; Padding to maintain segment offset that matches the current CTM.EXE
    padding DB 20h DUP( 90h )

;Following statments must be at the end of the SNTL_INIT_SEG.
    ALIGN 16
SNTL_SEGMENT:                ; Used to calculate the location of
SNTL_SEG.
ENDS

END

```

```

;*****
;* Copyright (c) Absolute Software 1994, 1995
;*
;* SNTLJTBL.ASM
;*
;* Contains the main jumtable code used by TimerISR.
;* HISTORY:
;* 1995.09.05 - CCOTI
;* Created.
;*
;*****
*

IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLJTBL.INC"
include "SNTLDATA.INC"
include "SNTLTIMR.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'
    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

; Enter: AL = table index, BX = table offset.
PROC JumpTable NEAR
    XOR     AH,AH          ; zero AH
    SHL     AX,1           ; multiply AX by 2 to get offset
    ADD     BX,AX          ; add offset to the table base
    JMP     [WORD BX]      ; jump the indexed address
ENDP JumpTable

cleanup:
    MOV     [sngstftn],OFFSET snfsnrst
    MOV     [cleanup_routine],OFFSET ActiveRoutine
    MOV     [Sentinel_state],SNSTACTIVE
    RETN

find_timeout:
    IF 1
        MOV     [sngstftn],OFFSET snfsnrst
    ELSE
        MOV     [sngstftn],OFFSET snfsnrst
        MOV     [cleanup_routine],OFFSET CheckNextPort
    ENDIF
    RETN

find_ok:
; Modem successfully initialized.
    MOV     [BYTE init_str_num],INIT_STR_TABLE_SIZE ;reset modem init
string table index

init_error:
    MOV     [sngstftn],OFFSET ModemInitInit
    RETN

init_ok:
    IF 0
;MOD DBOYD 55:95.04.12

```

```

; reset the dial string number when the Sentinel goes active, doing this will
; allow the system to search for another port and continue on from the last
; pre-dial string used
MOV     [BYTE dial_str_num], DIAL_STR_TABLE_SIZE
ENDIF
MOV     [sngstftn], OFFSET ModemCallInit
RETN

IF 0
;MOD DBOYD 50:95.02.22:
; to allow direct dial and PBX dial to work on opposite system, treat <BUSY>
; the same as <NO_DIAL_TONE> so that the next pre-dial string will be used
dial_busy:
DEC     [dial_str_num] ; reuse the last dial string
MOV     [sngstftn], OFFSET ModemCallInit
RETN
ENDIF

IF 1
dial_busy: ;MOD DBOYD 50:95.02.22
dial_error: ;MOD DBOYD 55:95.04.12
dial_no_carr: ;MOD DBOYD 55:95.04.12
cnct_error: ;MOD DBOYD 55:95.04.13
ENDIF
dial_no_tone:
MOV     [sngdlytmr], TM1SEC ; delay 1 second before redialing
MOV     [sngstftn], OFFSET ModemFindInit ; search for modem before
redialing
RETN

IF 0
;MOD DBOYD 55:95.04.12:
; to allow 8 prefix to work on 9 prefix PBX's and direct dial, treat
; the responses below the same as no dial tone so that the next pre-dial
; string will be used
dial_error:
dial_no_carr:
ENDIF
cmrxpktto: ;ADD CCOTI 48:95.02.07
MOV     [sngstftn], OFFSET snfsnrst
MOV     [cleanup_routine], OFFSET ActiveRoutine
MOV     [sngclst], SNCALLFAIL
RETN

IF 0 ;MOD DBOYD 55:95.04.13
cnct_error:
ENDIF
cnct_eot:
MOV     [sngstftn], OFFSET snfsnrst
MOV     [cleanup_routine], OFFSET ActiveRoutine
RETN

dial_server:
cnct_eng:
cnct_nak:
cnct_resend:
MOV     [sngstftn], OFFSET snftxchkin
RETN

cnct_ack:
MOV     [sngstftn], OFFSET snfgetpkt
RETN

```

161

5,764,892

162

```
IF 0
cnct_hold:
    MOV    [receive_tick_count],0
    RETN
ENDIF
ENDS
END
```

Reset timeout.

112

```

*****
;
; * Copyright (c) Absolute Software 1994, 1995
; *
; * SNTLSTRT.ASM
; *
; * Contains routines for using the sentinel's string tables.
; *
; * HISTORY:
; *   1995.09.05 - CCOTI
; *               Created.
; *
*****

IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLSTRT.INC"
include "SNTLDATA.INC"
include "SNTLBUFF.INC"
%LIST

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'

;*****
;
; *
; *
; * COMTRANSCHECK - check result of transmission
; *
; * PURPOSE:
; *   This function checks the result of a transmission between the Sentinel
; *   and the modem. It test modem responses against a table of strings.
; *
; * More
; *
; * PARAMETERS:
; *   BX = the beginning of the string table (more than one table is
; *   supported)
; *
; * RETURNS:
; *   CF = 1 if response is not completely received
; *   CF = 0 and AL = 0 if time-out without a match
; *   CF = 0, AL = index of match in response table (1 = last string in table)
; *
; * GLOBALS REFERENCES:
; *   receive_tick_count
; *   sngrxbuf1
; *   sngrxbufhd
; *
; * GLOBALS MODIFIED:
; *   sngrxbuf1 - if a string is found, set past the found string.
; *
; * BIOS CALLS:
; *   None
; *
; * DOS CALLS:
; *   None
; *
; * PROCEDURE CALLS:
; *   buf_inc_ptr, buf_getchar
; *
; * HARDWARE ACCESS:

```

```

None
NOTES:
*****
*

ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

PROC ComTransCheck

    CLD
    PUSH    DS
    POP     ES
    MOV     AL, [BX-1]
    XOR     CH, CH
    XOR     AH, AH
    ; set CMPSB pointers to increment
    ; get ES = DS
    ; AL = the number of strings to compare
    ; zero CH (CL is defined below)
    ; make AH zero

    @@str_loop_start:
    ; Initialize the inner loop.
    MOV     SI, [sngrxbuf1]
    MOV     DI, [BX]
    ADD     BX, 2
    MOV     CL, [DI-1]
    ; DI = the string to check
    ; CX = the string len

    @@char_loop_start:
    CMP     SI, [sngrxbufhd]
    JE      @@buffer_overrun
    CMPSB
    JNE     @@unmatched_byte
    LOOP    @@char_loop_start
    ; at the end of the buffer
    ; this increments SI and DI
    ; this string doesn't match

    @@found_match:
    MOV     [sngrxbuf1], SI

    @@clear_carry:
    CLC
    RET
    ; set return status
    ; exit

    @@buffer_overrun:
    INC     AH
    ; AH != 0 if no match has been found

    @@unmatched_byte:
    ; Have we checked all of the strings?
    DEC     AL
    JNZ     @@str_loop_start
    ; decrement string counter

    @@no_match:
    ; Check if we have timed out.
    CMP     [rx.rxtmr], 0
    JE      @@clear_carry
    ; AL = 0
    ; timed out, exit

    ; Check if we need to consume a character.
    OR      AH, AH
    JNZ     @@exit
    CALL    buf_getchar
    ; AH != 0 if no match was found

    @@exit:
    STC
    RET
    ; set return status
    ; exit

ENDP ComTransCheck
ASSUME NOTHING

```

167

5,764,892

168

ENDS

END

115

```

;*****
;
;* Copyright (c) Absolute Software 1994, 1995
;*
;* SNTLTIMR.ASM
;*
;* Contains Timer ISR and related subroutines.
;*
;* HISTORY:
;*   1995.09.05 - CCOTI
;*               Created.
;*
;*****
;
IDEAL

%NOLIST
include "SENTINEL.INC"
include "SNTLTIMR.INC"
include "SNTLDATA.INC"
include "SNTLJTBL.INC"
include "SNTLAPI.INC"
include "SNTLCOMM.INC"
include "SNTLCOMV.INC"
include "SNTLBUFF.INC"
include "SNTLSTRT.INC"
%LIST

;TCN Nov 1/95
MACRO TCN_EMIT ch
    PUSH    AX
    MOV     AL,ch
    CALL    TCN_PutChar
    POP     AX
ENDM
;TCN Nov 1/95

SEGMENT SNTL_SEG BYTE PUBLIC 'CODE'
    ASSUME CS:SNTL_SEG, DS:NOTHING, ES:NOTHING

;TCN Nov 1/95
PROC TCN_PutChar NEAR
    PUSH    AX
    PUSH    BX
    MOV     AH,0Eh                ;Output a character
    MOV     BH,0
    push    bp                    ;TCN - For old BIOS
    INT     10h
    pop     bp                    ;TCN - For old BIOS
    POP     BX
    POP     AX
    RETN
ENDP TCN_PutChar
;TCN Nov 1/95

;*****
;
;*
;* TMFISR - timer interrupt service routine
;*
;* PURPOSE:

```

```

; * This function implements the timer ISR that is hooked to the system
; * timer. This function and performs the following:
; *
; * Checks the Sentinel's state <Sentinel state>
; * Executes one of the following subroutines based on the state:
; * SNSTACTIVE
; *   ActiveRoutine:
; *
; * SNSTALERT
; *   PortFindInit:
; *   PortFind
; *   CheckNextPort:
; *
; * SNSTCALL
; *   ModemFindInit:
; *   ModemFind:
; *   ModemFindError:
; *
; *   ModemInitInit:
; *   ModemInit:
; *   ModemInitError:
; *   ModemCallInit:
; *   ModemCall:
; *   ModemDialError:
; *
; * SNSTCONNECT
; *   snftxchkin:
; *   ModemConnect:
; *   ModemConnectError:
; *
; * SNSTERROR
; *   ErrorRoutine:
; *
; * PARAMETERS:
; *   None
; *
; * RETURNS:
; *   Nothing
; *
; * REGISTERS DESTROYED:
; *   None
; *
; * GLOBALS REFERENCED:
; *   sngincmistr
; *   Sentinel_state
; *   sngstftn
; *   time_count
; *   activation_period
; *   modem_default_port
; *   port_table
; *   sngmdmptadd
; *   sngmdmptint
; *   modem_init_str
; *   init_result_table
; *
; * GLOBALS MODIFIED:
; *   sngmdmprt - set to the port currently being used.
; *   Sentinel_state - set to the now state of the Sentinel
; *   sngstftn - set to the routine that will perform the next action.
; *   sngmdmptadd - set to the address used by the current port (sngmdmprt)
; *   sngmdmptint - set to the interrupt used by the current port (sngmdmprt)
; *   sngincmistr - reset to 0 before cmfistr is hooked in.
; *   send_buf_len - reset before cmfistr is hooked in.

```

```

; *   sngprvcom - stores the old com ISR before hooking in cmfisir.
; *
; *   BIOS CALLS:
; *   None
; *
; *   DOS CALLS:
; *   None
; *
; *   PROCEDURE CALLS:
; *   buf_flush
; *   SwapInt
; *   ComTransInit
; *   ComTransCheck
; *
; *   HARDWARE ACCESS:
; *   UART (I/O MCR, OUT IER), I/O PIC
; *
; *****
;

PROC tmfisir FAR                                ; Entry point for TimerISR.
IF Debug
    INC     [CS:TimerISR_count]                ; increment debug timer
ENDIF

    CMP     [CS:sngsuspend], 0                  ; is the Sentinel suspended?
    JNE     TimerAbort                          ; yes, exit

    CMP     [CS:sngincmisr], 0                  ; is the Sentinel in the comm. ISR?
    JNE     TimerAbort                          ; yes, exit

    PUSH    DS                                  ; save registers
    PUSH    ES
    PUSH    CS
    POP     DS
    ASSUME  DS:SNTL_SEG

    CLI                                           ; halt interrupts
    CMP     [sngcomhk], 0                       ; have we hooked the comm.
interrupt?  JE     @@tmcont                      ; no, continue
; yes, determine if we still have
the hook
    MOV     BX, [sngcomprtint]                  ; the IVT entry to test
    SHL     BX, 2                               ; BX = the IVT offset to get ISR
vector
    XOR     AX, AX                              ; clear ES
    MOV     ES, AX

    MOV     AX, [ES:BX]                         ; get offset of installed vector
    CMP     AX, OFFSET cmfisir                  ; is it our routine?
    JNE     @@tmrst                             ; no, Reset sentinel
    MOV     AX, DS                              ; get our segment
    CMP     AX, [ES:BX+2]                       ; compare to installed vector
segment
    JNE     @@tmrst                             ; if not equal, reset Sentinel

```

```

        JMP      @@tmcont      ; we still have the interrupt,
continue
@@tmrst:
        MOV      [sngstftn], OFFSET ActiveRoutine      ; reset the Sentinel and continue
        MOV      [Sentinel_state], SNSTACTIVE
        MOV      [sngcomhk], 0      ; clear the the comm. hooked flag

@@tmcont:
        STI
        CMP      [win_flag], 0      ; restore interrupts
        JE       @@chktmrs      ; are we in Windows?
        MOV      AX, 1683h      ; no, go check running timers
        INT      2Fh      ; yes, determine virtual machine
        CMP      BL, [win_vm]
        JNE      TimerExit      ; should be a word!!
                                ; not in virtual machine 1, exit

@@chktmrs:
        CMP      [tx.txxtmr], 0      ; is the transmit timer running?
        JE       @@nxtmr0      ; no, continue
        DEC      [tx.txxtmr]      ; yes, decrement the transmit timer

@@nxtmr0:
        CMP      [rx.rxxtmr], 0      ; is the receive timer running?
        JE       @@nxtmr1      ; no, continue
        DEC      [rx.rxxtmr]      ; yes, decrement the port delay
timer

@@nxtmr1:
        CMP      [sngprtdlytmr], 0      ; is the port delay timer running?
        JE       @@nxtmr2      ; no, continue
        DEC      [sngprtdlytmr]      ; yes, decrement the port delay
timer

@@nxtmr2:
        CMP      [sngdlytmr], 0      ; is the Sentinel delay timer
running?
        JE       @@gostate      ; no, execute state function
        DEC      [sngdlytmr]      ; yes, decrement timer
        JMP      TimerExit      ; no, call previous timer handler

@@gostate:
        CALL     [sngstftn]      ; execute the state function

TimerExit:
        POPA
        ASSUME DS:NOTHING, ES:NOTHING
        POP      ES
        POP      DS

TimerAbort:
        JMP      [DWORD CS:sngprvtmr]

ENDP tmfiser
        ASSUME NOTHING

```

```

;*****
;
;*
;* SNFWTFORXMS - wait for XMS
;*
;* PURPOSE:

```

```

; * This function waits for the extended memory manager (XMS) to be loaded
; * and then hooks interrupt 2Fh. This hook allows the Sentinel to track
the
; * PC moving in and out of WINDOWS, and allows ASC utilities to communicate
; * with the utility.
; *
; * PARAMETERS:
; *     None
; *
; * RETURNS:
; *     Nothing
; *
; * GLOBALS REFERENCED:
; *
; * GLOBALS MODIFIED:
; *
; * BIOS CALLS:
; *     None
; *
; * DOS CALLS:
; *     None
; *
; * PROCEDURE CALLS:
; *     None
; *
; * HARDWARE ACCESS:
; *     Nothing
; *
; *****
;
; ASSUME CS:SNL_SEG, DS:SNL_SEG, ES:NOTHING
; *** STOLEN BY CCOTI ***
;
; ASSUME NOTHING
; *****
;
; * SNFWAIT - wait for timer to expire
; *
; * PURPOSE:
; *     This function waits for the delay timer, sngdlytmr, to expire before
; *     allowing the Sentinel to proceed. This function is used to delay the
; *     Sentinel from activating on power-up and when entering and exiting
; *     WINDOWS. Since the delay may be started at any time for a number of
; *     reasons, when the delay expires the Sentinel goes to snfsrst()
; *     before going back to ActiveRoutine().
; *
; * PARAMETERS:
; *     None
; *
; * RETURNS:
; *     Nothing
; *
; * GLOBALS REFERENCED:
; *
; * GLOBALS MODIFIED:
; *
; * BIOS CALLS:
; *     None
; *
; *

```

```

* DOS CALLS:
*   NONE
* PROCEDURE CALLS:
*   None
* HARDWARE ACCESS:
*   None
*****
*
    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

IF 0
PROC snfwait NEAR

    CMP     [sngdlytmr], 0           ; wait for delay timer to expire
    JNE     @@exit                   ; not yet expired, exit

@@reset:
    MOV     [sngstftn],OFFSET snfsrst ; reset the Sentinel
    MOV     [cleanup_routine],OFFSET ActiveRoutine

@@exit:
    RETN                                ; exit

ENDP snfwait

    ASSUME NOTHING
ENDIF

    ASSUME CS:SNTL_SEG, DS:SNTL_SEG, ES:NOTHING

PROC ActiveRoutine NEAR

; Check if the activation period has been exceeded.
    ROR     [cycle_var],1
    JNC     @@end

; Get current date and time.
    MOV     AH,4
    INT     1Ah                     ; Get RTC date.
    JC      @@end
    MOV     [now_year],CL           ; Store year.
    XCHG    DL,DH
    MOV     [WORD now_month],DX    ; Store month and day.
    MOV     AH,2
    INT     1Ah
    JC      @@end
    XCHG    CL,CH
    MOV     [WORD now_hour],CX     ; Store hour and minute.

; Check if next_call_date has been passed.
    MOV     SI, OFFSET next_call_date
    MOV     DI, OFFSET now_date
    CALL    CmpDates
    JNC     @@end

@@alert:
    MOV     [Sentinel_state],SNSTALERT ; Date passed, set to alert.

@@end:

```

```

; Check if we've been activated.
    CMP     [Sentinel_state], SNSTALERT    ; Check state.
    JNE     @@exit

@@activated:

; Set the Sentinel to the ALERT state.
IF 0
    MOV     AX, [modem_default_port]
    MOV     [sngmdmprt], AX                ; set first port
ENDIF
    MOV     [BYTE dial_str_num], DIAL_STR_TABLE_SIZE ; set first pre-dial
string
    MOV     [sngstftn], OFFSET PortFindInit ; set next state function

@@exit:
    RETN
ENDP ActiveRoutine

;
;
;
PROC CheckNextPort NEAR
    MOV     AX, [sngmdmprt]
    INC     AX
    CMP     AL, PORT_TABLE_SIZE
    JB      @@assign_port
    XOR     AX, AX                ; start back at first port
@@assign_port:
    MOV     [sngmdmprt], AX        ; set the modem port to check
    ; go look for modem on the port
    MOV     [sngstftn], OFFSET PortFindInit
    RETN                            ; exit
ENDP CheckNextPort

;
;
;
PROC PortFindInit NEAR
; initialize PortFind variables (based on sngmdmprt which was set previously).
    MOV     [sngclst], SNPRTSRCH    ; set call status

    MOV     BX, [sngmdmprt]
    SHL     BX, 2
    ADD     BX, OFFSET port_table
    MOV     AX, [BX]
    OR      AX, AX                ; check if port is valid

    JZ      CheckNextPort
    MOV     [sngmdmprtadd], AX      ; store current port address
    MOV     AX, [BX+2]
    MOV     [sngmdmprtint], AX      ; store current port interrupt
    MOV     [sngstftn], OFFSET PortFind ; set next state
    MOV     [sngprtdlytmr], TMSSEC  ; set port delay timer to 5 seconds
    RETN
ENDP PortFindInit

```

```

PROC PortFind NEAR
; Check if the port exists.
;
; ***
; NOT IMPLEMENTED - NEEDED FOR PCMCIA
; ***

    TCN_EMIT      '1'                ;TCN Nov 1/95

    MOV     DX,[sngmdmptadd]          ; DX = current port's address
    INC     DX                       ; DX = current port's IER
    IN      AL,DX                     ; AL = IER port status
    IODELAY
    AND     AL,0FFh                   ; if IER = 0xff, UART does not exist
    CMP     AL,0FFh
    JNE     @@chkprtavl               ; port exists, go check availability
    JMP     CheckNextPort             ; port does not exist, go check next
port
@@chkprtavl:                          ; check if the port is in use
    TCN_EMIT      '2'                ;TCN Nov 1/95

    MOV     CX,[sngmdmptint]          ; test PIC IMR first
    SUB     CL,08h                    ; get bit of interest
    MOV     BL,01h
    SHL     BL,CL                     ; bit mask ready
    IN      AL,21h                    ; get primary PIC IMR
    IODELAY
    AND     AL,BL                     ; bit set => interrupt disabled
    JNZ     @@port_idle

    TCN_EMIT      '3'                ;TCN Nov 1/95

    MOV     DX,[sngmdmptadd]          ; PIC IMR bit set, test IER next
    INC     DX                       ; DX = current port's address
    IN      AL,DX                     ; DX = current port's IER
    IODELAY
    OR      AL,AL                     ; AL = IER port status
    JZ      @@port_idle               ; Are any IER bits set?
                                        ; if no, port is idle

    TCN_EMIT      '4'                ;TCN Nov 1/95

    MOV     DX,[sngmdmptadd]          ; PIC IMR bit set, and IER bits
    ADD     DX,MCR                     ; set, test OUT2 next
    IN      AL,DX                     ; DX = current port's address
    IODELAY
    TEST    AL,08h                    ; DX = current port's MCR
    JZ      @@port_idle               ; AL = MCR port status
                                        ; is MCR OUT2 bit set?
                                        ; if no, port is idle

    TCN_EMIT      '5'                ;TCN Nov 1/95

    JMP     CheckNextPort             ; all checks failed
@@port_idle:

```

```

TCN_EMIT      '6'                                ;TCN Nov 1/95

period
  CMP          [sngprtdlytmr], 0                    ; port must be available for a set
  JNE          @@exit                                ; before a call is attempted

  MOV          DX, [sngmdmptadd]                      ; set port for no parity, eight
  ADD          DX, LCR                                ; data bits, and 1 stop bit
  MOV          AL, 00000011b                          ; get address of LCR
  OUT          DX, AL                                ; set LCR for N81
  IODELAY

  OR           AL, 80h                                ; set DLAB
  OUT          DX, AL                                ; set value in LCR
  IODELAY

  MOV          DX, [sngmdmptadd]                      ; force 9600 bps
  ADD          DX, BRDL                                ; DX = f / ( 16 * bps )
  MOV          AX, 000Ch                                ; = 1.8432 MHZ ( 16 * 9600 bps )
  OUT          DX, AX                                ; = 0x000C
  IODELAY                                              ; get address of DL LSB
  MOV          DX, [sngmdmptadd]                      ; set new divisor
  ADD          DX, LCR
  IN           AL, DX
  IODELAY                                              ; get address of DL LSB
  AND          AL, 7Fh                                ; get value in LCR
  OUT          DX, AL                                ; clear DLAB
  IODELAY                                              ; set value in LCR

@@init_ok:
; Clear any pending errors in the UART.
  MOV          DX, [sngmdmptadd]                      ; get address of LSR
  ADD          DX, LSR
  IN           AL, DX
  IODELAY

; Hook into the port, first init and install the interrupt vector.
  CALL         buf_flush                                ; flush the receive buffer
  MOV          [sngincmiser], 0
  MOV          [send_buf_len], 0
  MOV          BX, [sngmdmptint]                      ; The int to install.
  MOV          DI, OFFSET sngprvcom                    ; DS:DI = the address to store the
old vect.
  MOV          CX, OFFSET cmfiser                      ; DS:DX = the new com vector.
  CALL         SwapInt
  MOV          [sngcomhk], 1                          ; set the comm. hooked flag

  CLI
  MOV          DX, [sngmdmptadd]                      ; disable interrupts
  ADD          DX, MCR                                ; get address of MCR
  IN           AL, DX
  IODELAY
  OR           AL, 00001011b
  OUT          DX, AL                                ; interrupts enabled in the UART
  IODELAY

```

```

MOV     CX,[sngmdmprtint]      ; clear (enable) IRQ bit mask in PIC
SUB     CL,08h
MOV     BL,01h
SHL     BL,CL
NOT     BL
IN      AL,21h
IODELAY
AND     AL,BL
OUT     21h,AL                ; interrupts enabled in the PIC
IODELAY

MOV     DX, [sngmdmprtadd]     ; get address of IER
INC     DX
MOV     AL,00000001b          ; interrupt when data received
OUT     DX,AL
IODELAY

STI                                           ; enable interrupts

MOV     [Sentinel_state], SNSTCALLING
MOV     [sngdlytmr], TM1SEC      ; delay 1 second before attempting
to
MOV     [sngstftn], OFFSET ModemFindInit ; find modem

@@exit:
RETN
ENDP PortFind

```

```

PROC ModemFindInit NEAR

```

```

MOV     [sngclst], SNMDMSRCH      ; set call status

MOV     BX, OFFSET modem_find_str ; get a pointer to modem string
CALL    cmfprpmdm                 ; prepare transmit structure
MOV     [tx.txcontst], OFFSET ModemFind ; set next state after transmission

RETN

ENDP ModemFindInit

```

```

PROC ModemFind NEAR

```

```

MOV     BX,OFFSET command_result_table ; check for received data
CALL    ComTransCheck
JC      @@end                        ; data not received yet
MOV     BX,OFFSET find_jump_table     ; check for acceptable response

mov     [sngdlytmr], 9                ;TCN Nov 1/95
                                           ;According to Hayes Modem spec.
                                           ;we should wait at least 0.5 secs
                                           ;after sending the "ATZ" command

JMP     JumpTable

```

```

@@end:
    RETN

ENDP ModemFind

;
;
;
PROC ModemInitInit NEAR
; Attempt to initialize the modem (send modem_init_str).
    MOV     [sngclst], SNMDMINIT           ; set call status
    MOV     BX, OFFSET init_str_num       ; get the index of the next string
    DEC     [BYTE BX]
    JZ      @@reset                       ; wrap-around and start over

    MOV     AX, [BX]                       ; prepare transmit structure
    SHL     AX, 1                         ; get a pointer to the next string
    ADD     BX, AX
    MOV     BX, [BX]
    CALL    cmfprpmdm                     ; prepare transmit structure
    MOV     [tx.txnxtst], OFFSET ModemInit ; set state following transmission
    RETN

@@reset:
    MOV     [BYTE BX], INIT_STR_TABLE_SIZE ; retry initialization strings
    MOV     [sngstftn], OFFSET ModemCallInit
    RETN

ENDP ModemInitInit

;
;
;
PROC ModemInit NEAR
; Check for reply.
    MOV     BX, OFFSET mdm_init_result_table
    CALL    ComTransCheck
    JC      @@end                         ; data not received yet

    MOV     BX, OFFSET init_jump_table
    JMP     JumpTable

@@end:
    RETN

ENDP ModemInit

;
;
;
PROC ModemCallInit NEAR
; Attempt to dial (send modem pre-dial string).

```

```

        MOV     [sngclst], SNMDMPD                ; set call status

@@getstr:
        MOV     BX, OFFSET dial_str_num           ; get the index of the next string
        DEC     [BYTE BX]
        JZ      @@reset                          ; wrap-around and start over

        MOV     AX, [BX]
        SHL     AX, 1
        ADD     BX, AX
        MOV     BX, [BX]
        CALL    cmfprpmdm                        ; prepare transmit structure
        MOV     [tx.txxxtst], OFFSET ModemCallInit2 ; set state following
transmission

        RETN

@@reset:
        MOV     [BYTE dial_str_num], DIAL_STR_TABLE_SIZE
        JMP     @@getstr
        RETN

ENDP ModemCallInit

;
;
PROC ModemCallInit2 NEAR

        MOV     [sngclst], SNMDMDL                ; set call status
        MOV     BX, OFFSET dial_number           ; get the packet length
        CALL    cmfprpmdm                        ; prepare transmit structure
        MOV     [tx.txxxtst], OFFSET ModemCall    ; set state following transmission
        MOV     [rx.rxxxtmr], TM40SEC             ; override default response time and
                                                ; wait 40 seconds for response

        RETN

ENDP ModemCallInit2

;
;
PROC ModemCall NEAR

        MOV     [sngclst], SNWTCON                ; set call status
        MOV     BX, OFFSET dial_result_table
        CALL    ComTransCheck                    ; Check for reply.
        JC      @@end                            ; Data not received yet.

        MOV     BX, OFFSET dial_jump_table
        JMP     JumpTable                        ; attempt to parse data

@@end:
        RETN

ENDP ModemCall

```

```
PROC snftxchkin NEAR
```

```
; Query from server received by this point, send data packet
```

```

MOV AL, [sngdatalen]           ; prepare transmit structure
ADD AL, 2                      ; get the data segment length
MOV [BYTE LOW tx.txpktlen], AL ; add 2 for type and subtype
MOV [BYTE HIGH tx.txpktlen], 0
MOV [tx.txbufp], OFFSET sn_data_start
MOV [tx.txxtmr], OFFSET snfgetpkt ; set state following transmission
MOV [tx.txpkttyp], CMTXDATPKT    ; transmitting data packet
MOV [tx.txxtmr], TM3SEC          ; set transmission timeout

MOV [sngstftn], OFFSET cmftx     ; next state: transmit
MOV [rx.rxxxtmr], TM10SEC        ; wait 10 seconds for response
MOV [rx.rxxstate], OFFSET cmfpack ; receiver state: process expected
ACK

```

```
RETN
```

```
ENDP snftxchkin
```

```

;*****
;
; SNFGETPKT - collect packet data
;
; PURPOSE:
;   This functions collects packet data and determines if a receive timeout
;   has occurred.
;
; PARAMETERS:
;   None
;
; RETURNS:
;   Nothing
;
; NOTE:
;*****
;

```

```
PROC snfgetpkt NEAR
```

```

MOV [sngclst], SNWTNCD        ; set call status
CMP [rx.rxxxtmr], 0           ; test for timeout
JE @@timeout                  ; timed out

CALL buf_getchar              ; retrieve a character
JC @@exit                    ; none available, exit

CALL [rx.rxxstate]            ; run the rx state function
RETN

@@timeout:
MOV [sngstftn], OFFSET cmftx  ; set next Sentinel state function
MOV [tx.txpkttyp], CMTXDLENQ  ; set transmitter state: send ENQ

```

```

@@@exit:
    RETN

```

```

ENDP snfgetpkt

```

```

PROC snfsnrst NEAR

```

```

; Reset the Sentinel to a known state (ACTIVE), assume nothing.

```

```

    CALL    buf_flush
    CMP     [sngcomhk], 1
    JNE     @@cont
; have we hooked the comm. port
; no, continue

    MOV     BX, [sngmdmprtint]
    XOR     DI, DI
    PUSH    DS
    LDS     CX, [sngprvcom]
    CALL    SwapInt
    POP     DS
    MOV     [sngcomhk], 0
; yes, unhook the com interrupt
; the interrupt to install
; DS: CX = the com vector to install.
; clear the comm. hooked flag

```

```

@@@cont:

```

```

    MOV     DX, [sngmdmprtadd]
    INC     DX
    XOR     AL, AL
    OUT     DX, AL
    IODELAY
    ADD     DX, MCR-IER
    OR      AL, 03h
; DX = IER
; Disable all interrupts.
; DX = MCR
; leave RTS & DTR asserted to get
<NO CARRIER>
    OUT     DX, AL
    IODELAY
; MCR OUT2 bit = 0

    MOV     [sngstftn], OFFSET ActiveRoutine

```

```

    RETN
ENDP snfsnrst

```

```

ENDS

```

```

END

```

Electronic Article Surveillance System
Source Code for Host-side
Visual C++ (MicroSoft)

```

/*=====
=====*\

    Description:
        Source code for CompuTrace Server and DBServer.

5    Copyright:
        Copyright 1993-1995 Absolute Software Inc. All
        Rights Reserved.

    \*=====
    =====*/

10    #define INCL_NOPMAPI          // no PM in this program.
    #define INCL_DOS
    #define INCL_BSE
    #include <os2.h>
    #include <fstream.h>
    #include <time.h>

    #include <server.h>
15    #include <DB_Objects.HPP>
    #include <CTMessage.HPP>
    // #include <packet.h>
    #include "CT_Trans.H"

    FLAG fQueryCTIDStatus( MessagePipe &Pipe, const
    QueryCTIDStatusMsg &Status, CTIDStatusResultMsg &Result
    );
20    FLAG fStoreMonitorEvent( MessagePipe &Pipe, const
    StoreMonitorEventMsg &Store, StoreResultMsg &Result );
    FLAG fSignalQuit( MessagePipe &Pipe );

    void AssignTS( TTimestamp &ts, const SNTL_DATE &Date );
    void AssignSNTL_DATE( SNTL_DATE &Date, const TTimestamp
    &ts );
25    // Temp function.
    void ProcessClient( TPort &Port, TConnectInfo
    &ConnectInfo, CTID_TEXT *text );

    extern MessagePipe *pipe;

30    //
    // SntlConnect: called when a CONNECT comand has been
    // received, this function processes
    // a transaction between the server and a
    // Sentinel client.
    //
35    void SntlConnect( TPort &Port, MessagePipe &Pipe,
    TConnectInfo *cnct_info )
    {
        WORD msg_type;

```

```

        DosGetDateTime( &cnct_info->start_time );           //
        Fill start time.

        TPacket packet( Port );

        while (TRUE) {
5         // Get a packet.
            if (packet.rGetPacket() != TPacket::TRANS_ACK) {
                cout << "Packet Error" << endl;
                return;
            }

            // Determine packet type.
            packet.cbCopyText( &msg_type, sizeof( msg_type ) );
10         switch( msg_type ) {
            case CTID_TEXT_TYPE:
                // Create a new client object.
                // TClient Client( Port, Pipe, *cnct_info );
                // Get CTID Text and add to Client object.
                CTID_TEXT Text;
                packet.cbCopyText( &Text, sizeof( Text ) );
                // Client.SetCTID( Text );
15         // ProcessClient.
                // ProcessClient( Client );
                ProcessClient( Port, *cnct_info, &Text );
                return;
            default:
                return;
20         }

        void ProcessClient( TPort &Port, TConnectInfo
        &ConnectInfo, CTID_TEXT *text )
        {
            SNTL_DATE next_call;

25         // ENTER APPLICATION LAYER...

            // Query the Client state.
            QueryCTIDStatusMsg StatusMsg;
            StatusMsg.CTID = (ULONG)text->sn[0] + ((ULONG)text->sn[1] << 16);

            CTIDStatusResultMsg Result;

30         cout << "QueryCTIDStatus for CTID " << StatusMsg.CTID
            << " ";

            if (!QueryCTIDStatus( *pipe, StatusMsg, Result )) {
35         cout << "Error in QueryCTIDStatus!" << endl;
            }
            else {
                cout << "CTIDStatusResult Received..." << endl;

```

```

        cout << "    Status = " << (STRING)Result.Status <<
endl;
        cout << "    PeriodDays = " << Result.PeriodDays <<
endl;
        cout << "    PeriodMinutes = " <<
Result.PeriodMinutes << endl;
5      cout << "    StolenFlag = " <<
        (STRING)Result.StolenFlag << endl;
        cout << "    SpecialProcess = " <<
Result.SpecialProcess << endl;
        cout << "    Orgnum = " << Result.Orgnum_n << endl;
    }

10    // Send NextCall Message back to the Client.
        CTimestamp next_ts;
        AssignTS( next_ts, text->now_date );
        if (next_ts.usYear() < 1900) { // If date is not
valid substitute the local date instead.
            next_ts = ConnectInfo.start_time;
        }
        next_ts.AddToDate( 0, 0, Result.PeriodDays, 0,
15    Result.PeriodMinutes );
        AssignSNTL_DATE( next_call, next_ts );

        SendDatePacket( Port, next_call );
        SntlDisconnect( Port, ConnectInfo );

    // Store the Monitor Event.
20    StoreMonitorEventMsg Event;
        Event.StoreAsStolen = Result.StolenFlag;
        Event.StoreAsExpire = FALSE;

        Event.LicenseStatus = Result.Status;
        AssignTS( Event.ClientTS, text->now_date );
        Event.ServerTS = ConnectInfo.start_time;
        Event.NextCallTS_n = Event.ServerTS;
25    Event.NextCallTS_n.AddToDate( 0, 0, Result.PeriodDays,
0, Result.PeriodMinutes );
        Event.NextCallClientTS_n = next_ts;
        Event.CTID = StatusMsg.CTID;
        Event.TelcoTS_n.Assign( Event.ServerTS.usYear(),
                                ConnectInfo.cnd.month,
                                ConnectInfo.cnd.day,
30                                ConnectInfo.cnd.hour,
                                ConnectInfo.cnd.minute );

        Event.DurationSec_n = 0;
        Event.CallerID_n = (const
char(*) [CALLERID_SIZE]) ConnectInfo.cnd.number;
        Event.LineNum = 1;
35    Event.LogFlag = FALSE;
        Event.EnvironmentID = "DBS-9508";
        Event.ErrorCnt = 0;

```

```

StoreResultMsg ResultMsg;

cout << endl << "Storing the MonitorEvent... ";

if (!fStoreMonitorEvent( *pipe, Event, ResultMsg )) {
    cout << "Error in StoreMonitorEvent!" << endl;
5   }
    else {
        cout << "StoreResult = " << (ResultMsg.Result ?
"TRUE" : "FALSE") << endl;
    }
}

10 void SendDatePacket( TPort& Port, const SNTL_DATE& date )
{
    NC_PACKET packet;

    packet.header.stx = STX;
    packet.header.lsb_length = sizeof( NC_TEXT );
    packet.header.msb_length = 0;
15   packet.text.type = NC_TEXT_TYPE;
    packet.text.next_call_date = date;

    packet.footer.etx = ETX;
    packet.footer.lrc = 0;

20   Port.fWritePort( (PVOID)&packet, sizeof( packet ) );

FLAG fQueryCTIDStatus( MessagePipe &Pipe, const
QueryCTIDStatusMsg &Status, CTIDStatusResultMsg &Result )
{
    TStream in_strm, out_strm;
25   out_strm << Status;
    if (!Pipe.fTransact( out_strm, in_strm )) return
FALSE;
    in_strm >> Result;

    if (Result.eType() == CTID_STATUS_RESULT) return TRUE;
30   else return FALSE;
}

FLAG fStoreMonitorEvent( MessagePipe &Pipe, const
StoreMonitorEventMsg &Store, StoreResultMsg &Result )
{
35   TStream in_strm, out_strm;

    out_strm << Store;

```

```

        if (!Pipe.fTransact( out_strm, in_strm )) return
FALSE;
        in_strm >> Result;

        if (Result.eType() == STORE_RESULT) return TRUE;
        else return FALSE;
5    }

FLAG fSignalQuit( MessagePipe &Pipe )
{
    TStream stream;
    CliQuitMsg QuitMsg;
10    stream << QuitMsg;
    return Pipe.fSendMessage( stream );
}

void SntlDisconnect( TPort &Port, TConnectInfo
&ConnectInfo )
15 {
    // Drop DTR.
    DosSleep( 500 );    // Broc - 13 Feb 95
                        // Add delay to let modem clear xmt
buffer                    // to fix intermittent modem fault.
    Port.fDropDTR();
20    cout << "Disconnecting..." << flush;

    DosGetDateTime( &ConnectInfo.end_time );    //
    Fill end time.
    DosSleep( 200 );

    // Raise DTR.
25    Port.fRaisedDTR();
}

// *** helper functions.
UCHAR BCD2ToUChar( BYTE bcd )
{
30    // Convert a two digit bcd number to decimal.
    return (bcd >> 4) * 10 + (bcd & 0x0F);
}

BYTE UCharToBCD2( UCHAR dec )
{
35    // Convert a 8 bit decimal number to bcd.
    return (dec % 10) + (((dec / 10) % 10) << 4);
}

```

```

USHORT BCD4ToUShort( WORD bcd )
{
    // Convert a four digit bcd number to decimal.
    return (bcd >> 12) * 1000 + ((bcd & 0x0F00) >> 8) *
100 + ((bcd & 0x00F0) >> 4) * 10 + (bcd & 0x000F);
}

5  WORD UShortToBCD4( USHORT dec )
{
    // Convert a 16 bit decimal number to a 4 digit decimal.
    return (dec % 10) + (((dec / 10) % 10) << 4) + (((dec
/ 100) % 10) << 8) + (((dec / 1000) % 10) << 12);
}

10 void AssignTS( TTimestamp &ts, const SNL_DATE &Date )
{
    ts.Assign( BCD2ToUChar( Date.year ),
               BCD2ToUChar( Date.month ),
               BCD2ToUChar( Date.day ),
               BCD2ToUChar( Date.hour ),
15  BCD2ToUChar( Date.minute ) );
}

void AssignSNL_DATE( SNL_DATE &Date, const TTimestamp
&ts )
{
    Date.year   = UCharToBCD2( ts.usYear() % 100 );
    Date.month  = UCharToBCD2( ts.usMonth() );
    Date.day    = UCharToBCD2( ts.usDay() );
20  Date.hour   = UCharToBCD2( ts.usHour() );
    Date.minute = UCharToBCD2( ts.usMinute() );
}

/*
inline BYTE HiNibble( BYTE b ) { return (BYTE)((b & 0xF0)
>> 4); }
25 inline BYTE LoNibble( BYTE b ) { return (BYTE)(b & 0x0F);
}

void AddDays( SNL_DATE *next_call, int days )
{
    static BYTE days_per_month[18] = {
        0x31,
        0x28,
30  0x30,          // 0x03 - March
        0x31,
        0x30,
        0x31,          // 0x06 - June
        0x30,
        0x31,
35  0x30,          // 0x09 - Sept
        0x00,          // 0x0A
        0x00,          // 0x0B
    };
}

```

```

        0x00,      // 0x0C
        0x00,      // 0x0D
        0x00,      // 0x0E
        0x00,      // 0x0F
        0x31,      // 0x10 - Oct
        0x30,
5       0x31      // 0x12 - Dec
    };

    BYTE old_day = next_call->day;
    // Save for BCD adjust.

    // Add the days to the current date.
    next_call->day += days;
10   // Check if we passed the end of the current month.
    if (next_call->day > days_per_month[next_call->month])
    {
        // Add one to month.
        if (++next_call->month > 12)
        {
            next_call->month = 1;
            ++next_call->year;
15         next_call->day -= days_per_month[next_call->month]
            - 1;      // Roll over to proper day.

        // Adjust the day back to BCD.
        if (LoNibble( next_call->day ) > 0x9 || HiNibble(
next_call->day ) != HiNibble( old_day ))
20         next_call->day += 6;

        // Adjust the month to BCD.
        if (LoNibble( next_call->month ) > 0x9) next_call->
month += 6;

        // Adjust the year back to BCD.
        if (LoNibble( next_call->year ) > 0x9) next_call->year
25     += 6;
        if (HiNibble( next_call->year ) > 0x9) next_call->year
= LoNibble( next_call->year );
    }
    */

#define INCL_DOSNMPPIPES
30 #include <os2.h>

#include <iostream.h>
#include <fstream.h>
#include <string.h>

35 #include <server.h>

#include "DBServer.H"

```

```

#include <usertype.h>
#include <DB_Objects.HPP>
#include <CTID.H>
#include <CTIMS.HPP>
#include <CTMessage.HPP>
#include <MessagePipe.HPP>
5  FLAG fProcessClientEvent( MessagePipe &Pipe, TStream
    &MsgStream );

    FLAG fProcessQueryCTIDStatus( MessagePipe &Pipe,
    QueryCTIDStatusMsg &Status );
    FLAG fProcessStoreMonitorEvent( MessagePipe &Pipe,
    StoreMonitorEventMsg &MEvent );
10  FLAG fUpdateLicenseStatus( StoreMonitorEventMsg& );

    // Helper functions.
    FLAG _fCopyTStoDBVars( char *tsstring, short *indicator,
    CTTimestamp &ts, STRING varname = "Timestamp" );

    DataBase DB;
15  int main( int argc, char *argv[] )
    {
        if (argc != 3) {
            cout << "Usage: dbserver <database_name>
            <pipe_name>" << endl;
        }

20    DB.SetName( argv[1] );
        SvrMsgPipeFactory Factory( argv[2], 512, 10 );
        MessagePipe *pipe;

        if (!DB.fConnect()) {
            cout << "Unable to connect to " << argv[1] << "
            SQLCODE = " << (long)DB.ulSQLCode() << endl;
25    return 1;

            if (!Factory.fCreatePipe( pipe )) {
                cout << "Unable to create pipe DosErrorCode = " <<
                Factory.rcDosErrorCode() << endl;
                return 2;
            }

30    cout << "Waiting for pipe to connect to client..." <<
            endl;
            if (!pipe->fOpenPipe()) {
                cout << "Error connecting to the client
                DosErrorCode = " << pipe->rcDosErrorCode() << endl;
35    return 2;

            cout << "Pipe connected to client." << endl;

```

```

    TStream MsgStream;
    while (fProcessClientEvent( *pipe, MsgStream ))
    MsgStream.Reset();
    pipe->fClosePipe();
    return 0;
}

5
FLAG fProcessClientEvent( MessagePipe &Pipe, TStream
&MsgStream )
{
    if (!Pipe.fGetMessage( MsgStream )) {
        cout << "Error reading message from pipe
        DosErrorCode = " << Pipe.rcDosErrorCode() << endl;
10        return FALSE;
    }

    CTMessageHeader Header;
    MsgStream >> Header;
    switch (Header.eType()) {
        case QUERY_CTID_STATUS:
15        {
            QueryCTIDStatusMsg StatusMsg( Header );
            MsgStream >> *(QueryCTIDStatus*)&StatusMsg;
            if (!fProcessQueryCTIDStatus( Pipe, StatusMsg ))
                cout << "Error in fProcessQueryCTIDStatus,
SQLCODE = " << (long)ulGetSQLCode() << endl;
            }
            break;
20        case STORE_MONITOREVENT:
            {
                StoreMonitorEventMsg EventMsg( Header );
                MsgStream >> *(StoreMonitorEvent*)&EventMsg;
                if (!fProcessStoreMonitorEvent( Pipe, EventMsg
            )) {
                cout << "Error in fProcessStoreMonitorEvent,
25        SQLCODE = " << (long)ulGetSQLCode() << endl;
            }
            break;
            case CLI_QUIT:
                return FALSE;
            default:
30                cout << "Unknown Command Received!" << endl;
                return FALSE;
        }
    }
    return TRUE;
}

35
FLAG fProcessQueryCTIDStatus( MessagePipe &Pipe,
QueryCTIDStatusMsg &CTID )
{

```

```

    CTlicense Rec;
    CTIDStatusResultMsg ResultMsg;

    if (!fXlatCliCTID( CTID.CTID, CTID.CTID )) {
        cout << "Error converting client CTID to server
CTID" << endl;
5      // Process error here.
    }

    ResultMsg.QueryResult = _fQueryLicense( &Rec,
CTID.CTID );

    if (!ResultMsg.QueryResult) {
10      ResultMsg.CTID          = CTID.CTID;
        ResultMsg.Status      =
CTLicStatus::ACTIVE;
        ResultMsg.PeriodDays   = 2;
        ResultMsg.PeriodMinutes = 0;
        ResultMsg.StolenFlag   = FALSE;
        ResultMsg.SpecialProcess = 0;
        ResultMsg.Orgnum_n     = .fSetNull();
15      ResultMsg.LastCallTS_n  = .fSetNull();
        ResultMsg.NextCallTS_n = .fSetNull();
        ResultMsg.NextCallClientTS_n = .fSetNull();
        ResultMsg.ProductType  = .fSetNull();
    }
    else {
        ResultMsg.CTID          = Rec.CTID;
        ResultMsg.Status      = Rec.LicStatus;
20      ResultMsg.PeriodDays   = Rec.PeriodDays;
        ResultMsg.PeriodMinutes = Rec.PeriodMinutes;
        ResultMsg.StolenFlag   = Rec.StolenFlag ==
'Y';
        ResultMsg.SpecialProcess = Rec.SpecialProcess;
        ResultMsg.LastCallTS_n  = .Assign(
Rec.LastCallTS_N, DB_ISNULL( Rec.IsNull_LastCallTS ) );
25      ResultMsg.NextCallTS_n  = .Assign(
Rec.NextCallTS_N, DB_ISNULL( Rec.IsNull_NextCallTS ) );
        ResultMsg.NextCallClientTS_n = .Assign(
Rec.NextCallClientTS_N, DB_ISNULL(
Rec.IsNull_NextCallClientTS ) );
        if (DB_ISNULL( Rec.IsNull_Orgnum ))
            ResultMsg.Orgnum_n = .fSetNull();
30      else
        ResultMsg.Orgnum_n     = Rec.Orgnum_N;
        ResultMsg.ProductType  = Rec.ProductType;
    }

    cout << "SQLCODE = " << (long)ulGetSQLCode() << endl;
35  // Return Query results.
    TStream Stream;
    Stream << ResultMsg;

```

```

    return Pipe.fSendMessage( Stream );
}

FLAG fProcessStoreMonitorEvent( MessagePipe &Pipe,
StoreMonitorEventMsg &Msg )
5 {
    StoreResultMsg ResultMsg;

    // Prepare reply message.
    ResultMsg.Result = TRUE;

    // Prepare the monitorevent data.
10    _CTmonitorEvent Rec;

    if (!fXlatCliCTID( (ULONG&)Rec.CTID, Msg.CTID )) {
        cout << "Error converting client CTID to server
CTID" << endl;
        // Process error here.
    }

15    _fCopyTStoDBVars( Rec.ServerTS, NULL,
Msg.ServerTS, "ServerTS" );
    _fCopyTStoDBVars( Rec.ClientTS, NULL,
Msg.ClientTS, "ClientTS" );
    _fCopyTStoDBVars( Rec.TelcoTS_N, &Rec.IsNull_TelcoTS,
Msg.TelcoTS_n, "TelcoTS" );

20    Rec.DurationSec_N = Msg.DurationSec_n;
    Rec.IsNull_DurationSec = DB_NOT_NULL;

    if (!Msg.CallerID_n) {
        Rec.IsNull_CallerID = DB_NULL;
    }
    else {
        Rec.IsNull_CallerID = DB_NOT_NULL;
25    strncpy( Rec.CallerID_N, Msg.CallerID_n, sizeof(
Rec.CallerID_N ) );
    }

    Rec.LineNum = Msg.LineNum;

    if (!Msg.LogFlag) {
30    cout << "INVALID_DATA_ERROR: LogFlag is NULL,
defaulting to FALSE" << endl;
        Rec.LogFlag = 'N';
    }
    else {
35    Rec.LogFlag = ((STRING)Msg.LogFlag)[0];

    strncpy( Rec.EnvironmentID, Msg.EnvironmentID, sizeof(
Rec.EnvironmentID ) );

```

```

        Rec.ErrorCnt = Msg.ErrorCnt;

    // Update the License Record.
    if (!fUpdateLicenseStatus( Msg )) {
        if (ulGetSQLCode() != 100) {
            cout << "DB2_ERROR: Error updating License
5 Table, CliCTID = " << Msg.CTID
            << " SQLCODE = " << (long)ulGetSQLCode() <<
endl;
        }
    }

    // Perform the insert.
10 if (!fInsertIntoMonitorEvent( &Rec )) {
    ResultMsg.Result = FALSE;

    else {
        if (Msg.StoreAsStolen) {
            if (!fInsertIntoMonitorEventStolen( &Rec )) {
                ResultMsg.Result = FALSE;
            }
15         if (Msg.StoreAsExpire) {
            if (!fInsertIntoMonitorEventExpired( &Rec )) {
                ResultMsg.Result = FALSE;
            }
        }
    }

20     cout << "SQLCODE = " << (long)ulGetSQLCode() << endl;

    TStream Stream;
    Stream << ResultMsg;
    if (Pipe.fSendMessage( Stream ) && ResultMsg.Result ==
TRUE) {
25         DB.Commit();
        return TRUE;

        else {
            DB.Rollback();
            return FALSE;
        }
    }

30     FLAG fUpdateLicenseStatus( StoreMonitorEventMsg &Msg )
    {
        CTupdateLicenseStatus Rec;
        short dummy1; // Used to quiet the
35 Null validation below.

        fxlatCliCTID( (ULONG&)Rec.CTID, Msg.CTID );
    }

```

```

    strcpy( Rec.Status, Msg.LicenseStatus, sizeof(
Rec.Status ) );

    _fCopyTStoDBVars( Rec.LastCallTS_N,      &dummy1,
Msg.ServerTS,      "LastCallTS" );
    _fCopyTStoDBVars( Rec.NextCallTS_N,      &dummy1,
5  Msg.NextCallTS_n,      "NextCallTS" );
    _fCopyTStoDBVars( Rec.NextCallClientTS_N, &dummy1,
Msg.NextCallClientTS_n, "NextCallClientTS" );

    if (!Msg.NextCallTS_n) strcpy( Rec.NextCallTS_N,
"0001-01-01-00.00.00.000000" );
    if (!Msg.NextCallClientTS_n) strcpy(
10  Rec.NextCallClientTS_N, "0001-01-01-00.00.00.000000" );

    return _fUpdateLicenseStatus( &Rec );
}

FLAG _fCopyTStoDBVars( char *tsstring, short *indicator,
CTTimestamp &ts, STRING varname )
15 {
    if (!ts) {
        if (indicator == NULL) {
            cout << "INVALID DATA ERROR: " << varname << "
is NULL, forcing validation" << endl;
            ts.ForceValidate();
        }
        else {
20         *indicator = DB_NULL;
            tsstring[0] = '\x0';
            return FALSE;
        }
    }
    else if (!ts.fValidate()) {
        cout << "INVALID DATA ERROR: " << varname << " is
25  invalid, forcing validation - " << ts << endl;
        ts.ForceValidate();
    }

    if (indicator != NULL) *indicator = DB_NOT_NULL;
    ts.ToSTRING( tsstring );
    return TRUE;
30 }

#define INCL_NOPMAPI          // no PM in this program
35 #define INCL_DOS
#define INCL_BSE
#define INCL_DOSSEMAPHORES
#define INCL_DOSNMPPIPES

```

```

#include <os2.h>

#include <ctype.h>
#include <stdlib.h>
#include <iostream.h>
#include <fstream.h>
5  #include <server.h>

#include <MessagePipe.HPP>
#include <TModem.HPP>

#include "CT_Trans.H"
10 /*GLOBAL
VARIABLES*****/

HEV hQuitSem;

// Temp, move to thread.
CltMsgPipeFactory *factory;
15 MessagePipe *pipe;

/*****
**/

FLAG fLoadLineThreads( TModem&, PCSZ, PCSZ );
void _Optlink CT_CommandThread( PVOID );
FLAG fParseCmd( TPort &Port, TConnectInfo *CnctInfo,
20 STRING buffer );

TPort::ComSettings ComSetting = {
    "COM1",          // port name
    0,               // not used
    38400,           // bps
    8,               // data bits
25  TPort::NO,       // no parity
    TPort::ONE       // one stop bit
};

int main( int argc, char *argv[] )
    APIRET rc;
30  cout << "CompuTrace Server V0.99q" << endl;

// Check arguments.
if (argc != 4) {
    cout << "Usage: server <pipe_name> <port_name>
35  <init_string>" << endl << endl;
    return 0;
}

```

```

// Create quit semaphore.
    if ((rc = DosCreateEventSem( NULL, &hQuitSem, 0, FALSE
)) != 0)
        return 1;

    factory = new CltMsgPipeFactory( argv[1], 512 );
5
// Load port server threads.
    TPort Port;
    TModem Modem = Port;
    if (!fLoadLineThreads( Modem, argv[2], argv[3] ))
return 2;

    cout << "Successfully connected to local modem" <<
10 endl;

// Wait for quit signal.
    DosWaitEventSem( hQuitSem, SEM_INDEFINITE_WAIT );

    return 0;
15

//
// fLoadLineThreads: Loads the threads to operate a
server line. This function
// should be called for each server
line.
//
20 FLAG fLoadLineThreads( TModem &Modem, PCSZ port_str, PCSZ
init_str )
{
    // Start port log.
    // Port.LogOn();

    // Open port.
25 ComSetting.port_name = port_str;
    if (!Modem.Port().fOpenPort( ComSetting )) {
        cout << "Error openning port" << endl;
        return FALSE;

    // Start the port manage thread.
30 if (!Modem.Port().fStartManageThread()) {
        cout << "Thread execution error" << endl;
        return FALSE;

    // Initialize the modem.
35 STRING result = Modem.strSendCommand( init_str, -1 );
    if (strcmp( result, "OK" ) != 0) {
        cout << "Error initiallizing modem" << endl;
        return FALSE;

```

```

    }

    // Connect pipe to dbserver.
    if (!factory->fCreatePipe( pipe )) return FALSE;
    if (!pipe->fOpenPipe()) return FALSE;

5   // Start the command thread.
    if (!Modem.Port().fStartCommandThread(
CT_CommandThread, (PVOID)&Modem )) {
    cout << "Thread execution error" << endl;
    Modem.Port().KillManageThread();
    return FALSE;
    }

10   return TRUE;
}

//
// CT_CommandThread: Processes incoming data from a
server line.
15 //
void _Optlink CT_CommandThread( PVOID ptr )
{
    TModem &Modem = *(TModem*)ptr; // Alias
    (should be optimized out by the compiler).

    // Thread local variables
    STRING result;
20   TConnectInfo cnct_info;

    while (TRUE) {
        result = Modem.strGetString( -1 );
        // Parse buffer for cmd.
        if (!fParseCmd( Modem.Port(), &cnct_info, result ))

25   {
        memset( (PVOID)&cnct_info, '\x0', sizeof
cnct_info );
    }
}

#define CND_DATE_FIELD      "DATE ="
30 #define CND_TIME_FIELD    "TIME ="
#define CND_NUMBER_FIELD    "NMBER ="

#define CND_NONUM_FIELD     "REASON FOR NO NUMBER:"
#define CND_NAME_FIELD      "CALLER NAME:"
35 #define CND_NONAME_FIELD   "REASON FOR NO NAME:"

//
// fParseCmd: called when a '\n' has been received, this
function will process the string.

```

```

//          Returns TRUE if a transaction is occurring,
//          FALSE if the buffers should be cleared.
//
FLAG fParseCmd( TPort &Port, TConnectInfo *cnct_info,
STRING buffer )
5 {
    const char *index;

    // Parse command.
    if (strstr( buffer, "RING" ) != NULL) {
        cout << "Command parsed as RING" << endl;
    }
    else if ((index = strstr( buffer, CND_DATE_FIELD )) !=
10 NULL) {
        index += sizeof CND_DATE_FIELD;
        while (!isdigit( *index )) index++;
        // Grab the month.
        if (!isdigit( *index ) || !isdigit( *(index+1) ))
        return FALSE;
        cnct_info->cnd.month = (*index++ - '0') * 10;
15        cnct_info->cnd.month += *index++ - '0';
        // Grab the day.
        if (!isdigit( *index ) || !isdigit( *(index+1) ))
        return FALSE;
        cnct_info->cnd.day = (*index++ - '0') * 10;
        cnct_info->cnd.day += *index++ - '0';

20        cout << buffer << endl;
    }
    else if ((index = strstr( buffer, CND_TIME_FIELD )) !=
    NULL) {
        index += sizeof CND_TIME_FIELD;
        while (!isdigit( *index )) index++;
        // Grab the hour.
        if (!isdigit( *index ) || !isdigit( *(index+1) ))
25        return FALSE;
        cnct_info->cnd.hour = (*index++ - '0') * 10;
        cnct_info->cnd.hour += *index++ - '0';
        // Grab the minute.
        if (!isdigit( *index ) || !isdigit( *(index+1) ))
        return FALSE;
        cnct_info->cnd.minute = (*index++ - '0') * 10;
30        cnct_info->cnd.minute += *index++ - '0';

        cout << buffer << endl;
    }
    else if ((index = strstr( buffer, CND_NUMBER_FIELD ))
    != NULL) {
35        index += sizeof CND_NUMBER_FIELD;
        while (isspace( *index )) index++;
        // Grab the number.
        for (int i = 0; i < CND_NUM_MAXLEN; i++) {

```

```

        if (index[i] == '\x0' || index[i] == '\r') {
            cnct_info->cnd.number[i] = '\x0';
            break;
        }
        else {
            cnct_info->cnd.number[i] = index[i];
5         }
        cout << buffer << endl;
    }
    else if (strstr( buffer, CND_NONUM_FIELD ) != NULL) {
        index += sizeof CND_NONUM_FIELD;
        // Grab the string.
        while (isspace( *index )) index++;
10        for (int i = 0; i < CND_NUM_MAXLEN; i++) {
            if (index[i] == '\x0' || index[i] == '\r') {
                cnct_info->cnd.number[i] = '\x0';
                break;
            }
            else {
                cnct_info->cnd.number[i] = index[i];
15        }
        }
        cout << buffer << endl;
    }
    else if (strstr( buffer, CND_NAME_FIELD ) != NULL) {
        index += sizeof CND_NAME_FIELD;
        // Grab the name.
20        while (isspace( *index )) index++;
        for (int i = 0; i < CND_NAME_MAXLEN; i++) {
            if (index[i] == '\x0' || index[i] == '\r') {
                cnct_info->cnd.name[i] = '\x0';
                break;
            }
            else {
25        cnct_info->cnd.name[i] = index[i];
        }
        }
        cout << buffer << endl;
    }
    else if (strstr( buffer, CND_NONAME_FIELD ) != NULL)
30 {
        index += sizeof CND_NONAME_FIELD;
        // Grab the string.
        while (isspace( *index )) index++;
        for (int i = 0; i < CND_NAME_MAXLEN; i++) {
            if (index[i] == '\x0' || index[i] == '\r') {
35        cnct_info->cnd.name[i] = '\x0';
                break;
            }
            else {

```

```

        cnct_info->cnd.name[i] = index[i];

        cout << buffer << endl;
    }
5   else if (strstr( buffer, "CONNECT" ) != NULL) {
        cout << "Command parsed as CONNECT" << endl;

        SntlConnect( Port, *pipe, cnct_info );
        return FALSE;
    }
10  else if (strstr( buffer, "NO CARRIER" ) != NULL) {
        cout << "Command parsed as NO CARRIER" << endl;
        return FALSE;
    }
    else if (strstr( buffer, "OK" ) != NULL) {
        cout << "Command parsed as OK" << endl;
        return FALSE;
    }
15  else if (strstr( buffer, "ERROR" ) != NULL) {
        cout << "Command parsed as ERROR" << endl;
        return FALSE;
    }
    else {
        cout << "Unknown command received: " << buffer <<
endl;
        return FALSE;
20  }
    return TRUE;

#include <CTIMS.HPP>

//=====
25 //
// CTStatus friends and members.
//
CTStatus::CTStatus()
{
    memset( value, ' ', sizeof( value ) );
}
30 CTStatus::CTStatus( STRING str )
{
    ASSERT( strlen( str ) < sizeof( value ) );
    memcpy( value, str, strlen( str ) );
35 }

const char CTLicStatus::STR_SET[][CT_TOK_SIZE+1] = {

```

```

        UNUSED_TOK,
        NOTEST_TOK,
        ACTIVE_TOK,
        EXPIRED_TOK
    };

5   CTLicStatus& CTLicStatus::operator = ( STRING str )
    {
        for (int i = 0; i <= EXPIRED; i++) {
            if (strcmp( STR_SET[i], str ) == NULL) {
                setNotNull();
                value = VALUE( i );
10         return *this;
            }
        }
        ASSERT( FALSE );           // No match was found
        for the string.
        return *this;
    }

15  /*****
    FLAG CTOrgnum::fSetPrefix( STRING str )
    {
        if (strlen( str ) != ORGNUM_PREFIX_SIZE) {
            return FALSE;
        }
        else {
20         value[0] = str[0];
            value[1] = str[1];
            value[2] = str[2];
            value[3] = str[3];
            return TRUE;
        }
    }

25  FLAG CTOrgnum::fSetIndex( UINT num )
    {
        if (num > 9999) {
            return FALSE;
        }
        else {
            value[ORGNUM_PREFIX_SIZE + 0] = (num%10000) / 1000
30  + '0';
            value[ORGNUM_PREFIX_SIZE + 1] = (num%1000) / 100 +
            '0';
            value[ORGNUM_PREFIX_SIZE + 2] = (num%100) / 10 +
            '0';
            value[ORGNUM_PREFIX_SIZE + 3] = (num % 10) + '0';
35  }
    }

    FLAG CTOrgnum::fGetPrefix( char *str ) const

```

```

    {
        if (strlen( str ) != ORGNUM_PREFIX_SIZE) {
            return FALSE;
        }
        else {
            str[0] = value[0];
            str[1] = value[1];
            str[2] = value[2];
            str[3] = value[3];
            str[4] = '\x0';
        }
    }

10  FLAG CTOrgnum::fGetIndex( UINT &i ) const
    {
        i = atoi( &(value[ORGNUM_PREFIX_SIZE]) );
        return TRUE;
    }

    FLAG CTOrgnum::fGeneratePrefix( STRING org_name )
15  {
        char pre[ORGNUM_PREFIX_SIZE];

        // Grab first four alphanum characters.
        for (int i = 0, j = 0; i < ORGNUM_PREFIX_SIZE;) {
            if (isalnum( orgname[j++] )) pre[i];
        }

20  *****/

        //*****
        //
        // ostream stream operators.
        //
25  ostream& operator <<( ostream &os, const CTStatus &status
    {
        {
            return os << (STRING)status;
        }
    }

        //*****
        //
30  // TStream stream operators.
        //
        TStream& operator << ( TStream &buf, const CTStatus
        &status )
    {
35  {
        buf << *(TNull*)&status;
        if (!status) return buf;
        else return buf.Put( PVOID( status.value ), sizeof(
        status.value ) );
    }

```

```

    }

    TStream& operator >> ( TStream &buf, CTStatus &status )
    {
        buf >> *(TNull*)&status;
        if (!status) return buf;
5      else return buf.Get( status.value, sizeof(
        status.value ) );
    }

    TStream& operator << ( TStream &buf, const CTCallerID &id
    )
    {
10     buf << *(TNull*)&id;
        if (!id) return buf;
        else return buf.Put( PVOID( id.value ), sizeof(
        id.value ) );
    }

    TStream& operator >> ( TStream &buf, CTCallerID &id )
15     buf >> *(TNull*)&id;
        if (!id) return buf;
        else return buf.Get( id.value, sizeof( id.value ) );
    }

    TStream& operator << ( TStream &buf, const CTLicStatus
    &lic )
20     {
        buf << *(TNull*)&lic;
        if (!lic) return buf;
        else return buf << USHORT( lic.value );
    }

    TStream& operator >> ( TStream &buf, CTLicStatus &lic )
25     USHORT num;

        buf >> *(TNull*)&lic;
        if (!lic) return buf;
        else {
            buf >> num;
            lic.value = CTLicStatus::VALUE( num );
30         return buf;
        }

    TStream& operator << ( TStream &buf, const CTOrgnum &num
    )
35     {
        buf << *(TNull*)&num;
        if (!num) return buf;
    }

```

```

        else return buf.Put( PVOID( num.value ), sizeof(
num.value ) );
    }

TStream& operator >> ( TStream &buf, CTOrgnum &num )
{
5   buf >> *(TNull*)&num;
    if (!num) return buf;
    else return buf.Get( num.value, sizeof( num.value ) );
}

TStream& operator << ( TStream &buf, const CTMonitorEvent
10  &event )
{
    return buf << event.CTID
        << event.ServerTS
        << event.ClientTS
        << event.TelcoTS_n
        << event.DurationSec_n
        << event.CallerID_n
15  << event.LineNum
        << event.LogFlag
        << event.EnvironmentID
        << event.ErrorCnt;
}

TStream& operator >> ( TStream &buf, CTMonitorEvent
20  &event )
{
    return buf >> event.CTID
        >> event.ServerTS
        >> event.ClientTS
        >> event.TelcoTS_n
        >> event.DurationSec_n
        >> event.CallerID_n
25  >> event.LineNum
        >> event.LogFlag
        >> event.EnvironmentID
        >> event.ErrorCnt;
}

30  #include <CTMessage.HPP>

    //*****
    //
35  // TStream stream operators.
    //
    TStream& operator << ( TStream &buf, const
    CTMessageHeader &head )

```

```

    }
    return buf << head.ID << head.Type << head.Len;
}

TStream& operator >> ( TStream &buf, CTMessageHeader
&head )
5 {
    buf >> head.ID;
    buf >> head.Type;
    buf >> head.Len;

    return buf;
}

10 #define INCL_NOPMAPI           // no PM in this program
#define INCL_DOS
#define INCL_BSE
#define INCL_DOSSEMAPHORES
#define INCL_DOSNMPIPES
#include <os2.h>

15 #include "CT_Buffer.HPP"

CT_Buffer::CT_Buffer()
{
    head( 0 ),
    tail( CT_BUFFER_MAXLEN )

    // Create the mutex sem.
20 rc = DosCreateMutexSem( NULL, &hBufSem, 0, 0 );
    if (rc) {}

    // Create the event sem.
    rc = DosCreateEventSem( NULL, &hReleaseGetSem, 0, 0 );
}

CT_Buffer::~CT_Buffer()
25 {
    DosCloseMutexSem( hBufSem );
}

void CT_Buffer::Flush()
{
    ULONG post_count;

30 DosRequestMutexSem( hBufSem, SEM_INDEFINITE_WAIT );
    head = 0;
    tail = CT_BUFFER_MAXLEN;
    DosResetEventSem( hReleaseGetSem, &post_count );
    DosReleaseMutexSem( hBufSem );
35 }

FLAG CT_Buffer::fPutChar( char ch )
{

```

```

        FLAG ret_val;

// Get ownership of the semaphore.
rc = DosRequestMutexSem( hBufSem, SEM_INDEFINITE_WAIT
);
    if (rc) return FALSE;
5 // First check that the log buffer hasn't overflowed.
    if (!fIsFull()) {
        // Store the char, update head, signal the event.
        buffer[head] = ch;
        head = IncBufPtr( head );
        DosPostEventSem( hReleaseGetSem );
10         ret_val = TRUE;
    }
    else ret_val = FALSE;

// Release the semaphore.
DosReleaseMutexSem( hBufSem );

    return ret_val;
15 }

FLAG CT_Buffer::fGetChar( char &ch )
{
    ULONG post_count;
    FLAG ret_val;

20 // If empty wait for timeout.
    if (fIsEmpty()) DosWaitEventSem( hReleaseGetSem,
SEM_INDEFINITE_WAIT );

// Get ownership of the semaphore.
rc = DosRequestMutexSem( hBufSem, SEM_INDEFINITE_WAIT
);
    if (rc) return FALSE;
25     if (!fIsEmpty()) {
        // Fetch the char, update tail.
        tail = IncBufPtr( tail );
        ch = buffer[tail];
        ret_val = TRUE;
    }
30     else ret_val = FALSE;

    DosResetEventSem( hReleaseGetSem, &post_count );

// Release the semaphore.
DosReleaseMutexSem( hBufSem );
35     return ret_val;
}

```

```

#define INCL_NOPMAPI          // no PM in this program
#define INCL_DOS
#define INCL_BSE
#define INCL_DOSSEMAPHORES
#define INCL_DOSNMPIPES
#include <os2.h>

5  #include "CT_Log.HPP"

#include <fstream.h>

CT_Log::CT_Log( UINT len )
:   buf_len( len ),
10  index( 0 )
{
    if ((buffer = new BYTE[buf_len]) == NULL) {
        buf_len = index = 0;
    }
}

CT_Log::~CT_Log()
15 {
    if (buffer) DosFreeMem( buffer );
}

BOOL CT_Log::fPostChar( char ch )
{
    // First check that the log buffer hasn't overflowed.
    if (!fIsFull()) {
20     // Store the char, update head.
        buffer[index++] = ch;
        return TRUE;
    }
    else return FALSE;
}

25  BOOL CT_Log::fDumpLog( const char *fname )
{
    fstream dump;

    dump.open( fname, ios::out );
    if (!dump) return FALSE;
    dump.write( buffer, index );
30  dump.close();

    return TRUE;
}

#define INCL_DOSNMPIPES
35  #include <os2.h>

#include <MessagePipe.HPP>

```

```

//*****
//*****
// SvrMsgPipeFactory Implementation.
//*****
//*****

5  SvrMsgPipeFactory::SvrMsgPipeFactory( PCSZ name, UINT
    msg_len, UINT pipe_len )
    {
        MsgPipeFactory( msg_len ),
        pipe_name( name ),
        pipe_len( pipe_len )
    }

10  FLAG SvrMsgPipeFactory::fCreatePipe( MessagePipe *ppipe
    )
    {
        ppipe = new MessagePipe( this );

        return TRUE;
    }

15  FLAG SvrMsgPipeFactory::fDestroyPipe( MessagePipe *ppipe
    )
    {
        delete ppipe;

        return TRUE;
    }

20  FLAG SvrMsgPipeFactory::fOpenPipe( MessagePipe *pipe )
    {
        HPIPE hPipe;

        // Create and connect the named pipe.
        pipe->rc = DosCreateNPipe( (PSZ)pipe_name, &hPipe,
                                NP_NOWRITEBEHIND |           //
25  Data sent to remote pipes immediatly.
                                NP_ACCESS_DUPLEX,             //
                                Two-way client/server communications.
                                NP_WAIT |                      //
                                I/O to pipe blocked until data avaliable.
                                NP_TYPE_MESSAGE |              //
                                Message pipe type.
                                NP_READMODE_MESSAGE |         //
30  Messafe read mode type.
                                0x00FF,                     //
                                Infinite number of allowed instances of this pipe.
                                (uMaxMsgLen() + 2) * pipe_len, //
                                Size of output buffer.
35  Size of input buffer.
                                (uMaxMsgLen() + 2) * pipe_len, //
                                0                             //
                                Client open timeout (see DosWaitNPipe).
    }

```

```

        };
        if (pipe->rc) return FALSE;

        pipe->rc = DosConnectNPipe( hPipe );
        if (pipe->rc) return FALSE;

5       pipe->SetHandle( hPipe );
        return TRUE;
    }

    FLAG SvrMsgPipeFactory::fClosePipe( MessagePipe *pipe )
    {
        HPIPE hPipe = pipe->GetHandle();

10       // Wait till the pipe is empty.
        pipe->rc = DosResetBuffer( hPipe );
        if (pipe->rc) return FALSE;
        // Disconnect the pipe handle.
        pipe->rc = DosDisconnectNPipe( hPipe );
        if (pipe->rc) return FALSE;

15       return TRUE;
    }

    //*****
    //*****
    // CltMsgPipeFactory Implementation.
    //*****
    //*****
20       CltMsgPipeFactory::ClMsgPipeFactory( PCSZ name, UINT
        msg_len )
        : MsgPipeFactory( msg_len ),
          pipe_name( name )
        {}

25       FLAG CltMsgPipeFactory::fCreatePipe( MessagePipe *&ppipe )
    {
        ppipe = new MessagePipe( this );

        return TRUE;
    }

30       FLAG CltMsgPipeFactory::fDestroyPipe( MessagePipe *ppipe )
    {
        delete ppipe;

35       return TRUE;
    }

    FLAG CltMsgPipeFactory::fOpenPipe( MessagePipe *pipe )

```

```

    {
        HPIPE hPipe;
        ULONG ulAction;

        pipe->rc = DosOpen( pipe_name, &hPipe, &ulAction, 0,
                           FILE_NORMAL, FILE_OPEN,
5       OPEN_ACCESS_READWRITE |
       OPEN_SHARE_DENYNONE,
        (PEAOP2) NULL );
        if (pipe->rc) return FALSE;

        pipe->SetHandle( hPipe );
        return TRUE;
10    }

    FLAG CltMsgPipeFactory::fClosePipe( MessagePipe *pipe )
    {
        HPIPE hPipe = pipe->GetHandle();

        // Wait till the pipe is empty.
        pipe->rc = DosResetBuffer( hPipe );
15        if (pipe->rc) return FALSE;
        // Close the pipe handle.
        rc = DosClose( hPipe );
        if (pipe->rc) return FALSE;

        return TRUE;
    }

20    //*****
    //*****
    // MessagePipe Implementation
    //*****
    //*****

    MessagePipe::MessagePipe( MsgPipeFactory *mom )
25    {
        : factory( mom )

        factory->InitPipe( this );
    }

    MessagePipe::~MessagePipe()
    {
30        factory->DeinitPipe( this );
    }

    FLAG MessagePipe::fOpenPipe()
    {
        return factory->fOpenPipe( this );
35    }

    FLAG MessagePipe::fClosePipe()
    {

```

```

        return factory->fClosePipe( this );
    }

    FLAG MessagePipe::fSendMessage( PCVOID msg, ULONG msg_len
    )
    {
5        ULONG cbWritten;

        rc = DosWrite( hPipe, (PVOID)msg, msg_len, &cbWritten
    );

        return (rc == 0 && msg_len == cbWritten) ? TRUE :
    FALSE;
10    }

    FLAG MessagePipe::fGetMessage( PVOID msg, PULONG msg_len
    )
    {
        // PRECONDITION( msg_len != 0 && *msg_len <=
    uMaxMsgLen() );

15        rc = DosRead( hPipe, msg, *msg_len, msg_len );

        return (rc == 0) ? TRUE : FALSE;
    }

    FLAG MessagePipe::fTransact( PCVOID out_msg, ULONG
    out_msg_len, PVOID in_msg, PULONG in_msg_len )
    {
20        // PRECONDITION( in_msg_len != 0 && *in_msg_len <=
    uMaxMsgLen() );

        rc = DosTransactNPipe( hPipe, (PVOID)out_msg,
    out_msg_len, in_msg, *in_msg_len, in_msg_len );

        return (rc == 0) ? TRUE : FALSE;
25    }

    MessagePipe::PIPE_STATE MessagePipe::eState()
    {
        ULONG cbRead;
        AVAILDATA avail;
        ULONG state;

30        // Use DosPeekNPipe to find the state of the pipe.
        rc = DosPeekNPipe( hPipe, NULL, 0, &cbRead, &avail,
    &state );

        return (PIPE_STATE)state;
35    }

#ifdef __OS2__
    #define INCL_DOSDATETIME

```

```

        #include <os2.h>
    #endif

    #include <ctype.h>

    #include <Objects.HPP>
5
    //*****
    //
    // TFlag members.
    //

10    TFlag::TFlag()
        : TNull( TRUE )
    {}

    TFlag::TFlag( FLAG flag )
        : value( (flag != FALSE) ),
          TNull( FALSE )
    {}

15    TFlag::~~TFlag()
    {
        #ifdef DEBUG
            fSetNull();
            value = UNINIT_DATA;
        #endif
    }

20    //*****
    //
    // TTimestamp members.
    //

25    const UINT TTimestamp::TSStringLen = 27;

    TTimestamp::TTimestamp()
        : TNull( TRUE )
    {
        #ifdef DEBUG
            Year = Month = Day = Hour = Minute = Second =
30    Millisec = UNINIT_DATA;
        #endif
    }

    TTimestamp::TTimestamp( USHORT yr, UCHAR mo, UCHAR dy,
        UCHAR hr, UCHAR mn, UCHAR sc,
35    USHORT ms )
        : Year( yr ),
          Month( mo ),
          Day( dy ),

```

```

        Hour( hr ),
        Minute( mn ),
        Second( sc ),
        Millisec( ms ),
        TNull( FALSE )
    {}
5   TTimestamp::~TTimestamp()
    {
        #ifdef DEBUG
            fSetNull();
            Year = Month = Day = Hour = Minute = Second =
10      Millisec = UNINIT_DATA;
        #endif

        FLAG TTimestamp::fValidate() const
        {
            if (fIsNull()) return FALSE;

            // Check year.
15      if (!Year || Year > 9999) return FALSE;
            // Check month and day.
            if (!Day) return FALSE;
            switch (Month) {
                case 1:
                    if (Day > 31) return FALSE;
                    break;
                case 2:
20      if (Year % 4 == 0 && Year % 100 != 0)          //
                    Check for a leapyear.
                    if (Day > 29) return FALSE;
                    else
                        if (Day > 28) return FALSE;
                    break;
                case 3:
25      if (Day > 31) return FALSE;
                    break;
                case 4:
                    if (Day > 30) return FALSE;
                    break;
                case 5:
                    if (Day > 31) return FALSE;
                    break;
30      case 6:
                    if (Day > 30) return FALSE;
                    break;
                case 7:
                    if (Day > 31) return FALSE;
                    break;
35      case 8:
                    if (Day > 31) return FALSE;
                    break;
            }
        }
    }

```

```

        case 9:
            if (Day > 30) return FALSE;
            break;
        case 10:
            if (Day > 31) return FALSE;
            break;
5       case 11:
            if (Day > 30) return FALSE;
            break;
        case 12:
            if (Day > 31) return FALSE;
            break;
        default:
10         return FALSE;
    }
    // Check hours.
    if (Hour > 23) {
        if (Hour > 24 || Minute || Second || Millisec)
            return FALSE;
    }
    // Check minutes, seconds and milliseconds.
15    if (Minute > 59 || Second > 59 || Millisec > 999)
        return FALSE;

    return TRUE;

void TTimestamp::ForceValidate()
20 {
    setNotNull();
    Year = Month = Day = 1;
    Hour = Minute = Second = Millisec = 0;
}

FLAG TTimestamp::IsValidTSString( STRING ts )
25 {
    if (    isdigit( ts[0] )           // Check Year.
        && isdigit( ts[1] )
        && isdigit( ts[2] )
        && isdigit( ts[3] )
        && ts[4] == '-'
        && isdigit( ts[5] )           // Check Month.
        && isdigit( ts[6] )
        && ts[7] == '-'
30     && isdigit( ts[8] )           // Check Day.
        && isdigit( ts[9] )
        && ts[10] == '-'
        && isdigit( ts[11] )          // Check Hour.
        && isdigit( ts[12] )
        && ts[13] == ':'
35     && isdigit( ts[14] )          // Check Minute.
        && isdigit( ts[15] )
        && ts[16] == ' ' )

```

```

        && isdigit( ts[17] )           // Check Second.
        && isdigit( ts[18] )
        && ts[19] == ','
        && isdigit( ts[20] )           // Check Millisec.
        && isdigit( ts[21] )
        && isdigit( ts[22] )
5      && isdigit( ts[23] )
        && isdigit( ts[24] )
        && isdigit( ts[25] )
        && ts[26] == '\x0'
      return TRUE;
    } else return FALSE;
  }

10  TTimestamp& TTimestamp::Assign( const TTimestamp &ts )
    {
      if (!ts) {
        fSetNull();
      }
      else {
15      setNotNull();
        Year = ts.Year;
        Month = ts.Month;
        Day = ts.Day;
        Hour = ts.Hour;
        Minute = ts.Minute;
        Second = ts.Second;
        Millisec = ts.Millisec;
20    }
      return (*this);
    }

    TTimestamp& TTimestamp::Assign( USHORT yr, UCHAR mo,
    UCHAR dy,
    UCHAR hr, UCHAR mn, UCHAR
    sc, USHORT ms )
25  {
    setNotNull();

    Year = yr;
    Month = mo;
    Day = dy;
    Hour = hr;
30  Minute = mn;
    Second = sc;
    Millisec = ms;

    return (*this);
  }

35  TTimestamp& TTimestamp::Assign( STRING ts, FLAG isnull )
    {
      unsigned num;

```

```

        if (isnull) {
            fSetNull();
            return *this;
        }

        setNotNull();

5       ASSERT( fIsValidTSString( ts ) );

        /* Convert year */
        num = (ts[0] - '0') * 1000;
        num += (ts[1] - '0') * 100;
        num += (ts[2] - '0') * 10;
10       num += (ts[3] - '0');
        Year = USHORT( num );

        /* Convert month */
        num = (ts[5] - '0') * 10;
        num += (ts[6] - '0');
        Month = UCHAR( num );

        /* Convert day */
        num = (ts[8] - '0') * 10;
15       num += (ts[9] - '0');
        Day = UCHAR( num );

        /* Convert hour */
        num = (ts[11] - '0') * 10;
        num += (ts[12] - '0');
        Hour = UCHAR( num );

        /* Convert minute */
        num = (ts[14] - '0') * 10;
20       num += (ts[15] - '0');
        Minute = UCHAR( num );

        /* Convert second */
        num = (ts[17] - '0') * 10;
        num += (ts[18] - '0');
        Second = UCHAR( num );

        /* Convert millisec */
25       num = (ts[20] - '0') * 100;
        num += (ts[21] - '0') * 10;
        num += (ts[22] - '0');
        Millisec = USHORT( num );

        return *this;
    }

30     #ifdef __OS2__
    TTimestamp& TTimestamp::Assign( const DATETIME &Date )
    {
        setNotNull();

35       Year = Date.year;
        Month = Date.month;
        Day = Date.day;
        Hour = Date.hours;
    }

```

```

        Minute = Date.minutes;
        Second = Date.seconds;
        Millisec = Date.hundredths * 10;

        return (*this);
    }
5  #endif // __OS2__

    STRING TTimestamp::ToSTRING( char *ts ) const
    {
        unsigned num;

        /* Convert year */
        num = Year;
10      ts[0] = (num%10000) / 1000 + '0';
        ts[1] = (num%1000) / 100 + '0';
        ts[2] = (num%100) / 10 + '0';
        ts[3] = (num % 10) + '0';
        ts[4] = '-';
        /* Convert month */
        num = Month;
15      ts[5] = (num%100) / 10 + '0';
        ts[6] = (num % 10) + '0';
        ts[7] = '-';
        /* Convert day */
        num = Day;
        ts[8] = (num%100) / 10 + '0';
        ts[9] = (num % 10) + '0';
20      ts[10] = '-';
        /* Convert hour */
        num = Hour;
        ts[11] = (num%100) / 10 + '0';
        ts[12] = (num % 10) + '0';
        ts[13] = '.';
        /* Convert minute */
        num = Minute;
25      ts[14] = (num%100) / 10 + '0';
        ts[15] = (num % 10) + '0';
        ts[16] = '.';
        /* Convert second */
        num = Second;
        ts[17] = (num%100) / 10 + '0';
        ts[18] = (num % 10) + '0';
30      ts[19] = '.';
        /* Convert millisec */
        num = Millisec;
        ts[20] = (num%1000) / 100 + '0';
        ts[21] = (num%100) / 10 + '0';
        ts[22] = (num % 10) + '0';
35      ts[23] = '0';
        ts[24] = '0';
        ts[25] = '0';
    }

```

```

    ts[26] = '\x0';
    return ts;
}

FLAG TTimestamp::operator > ( const TTimestamp &ts )
5  const
{
    useAsValue();

    if (Year > ts.Year) return TRUE;
    else if (Year == ts.Year) {
        if (Month > ts.Month) return TRUE;
        else if (Month == ts.Month) {
10         if (Day > ts.Day) return TRUE;
            else if (Day == ts.Day) {
                if (Hour > ts.Hour) return TRUE;
                else if (Hour == ts.Hour) {
                    if (Minute > ts.Minute) return TRUE;
                    else if (Minute == ts.Minute) {
15                     if (Second > ts.Second) return TRUE;
                        else if (Second == ts.Second) {
                            if (Millisec > ts.Millisec) return
TRUE;
                                else return FALSE;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    return FALSE;
}

FLAG TTimestamp::operator >= ( const TTimestamp &ts )
25  const
{
    return (*this > ts || *this == ts);
}

FLAG TTimestamp::operator == ( const TTimestamp &ts )
const
{
30  useAsValue();

    if (Year == ts.Year &&
        Month == ts.Month &&
        Day == ts.Day &&
        Hour == ts.Hour &&
35  Minute == ts.Minute &&
        Second == ts.Second &&
        Millisec == ts.Millisec) {
        return TRUE;
    }
}

```

```

    }
    else {
        return FALSE;
    }
}

5 // Date and time add function.
  TTimestamp& TTimestamp::AddToDate( UINT yr, UINT mon,
    UINT day,
                                UINT hr, UINT min,
    UINT sec, UINT ms )
  {
    if (!IsNull()) {
10      ms += Millisec;
      sec += Second;
      min += Minute;
      hr += Hour;
      day += Day;
      mon += Month;
      yr += Year;
    }

15 // Adjust and carry ms.
      while (ms > usMaxMillisec()) {
        ms -= usMaxMillisec() + 1;
        sec++;
      }
    // Adjust and carry sec.
20    while (sec > usMaxSecond()) {
      sec -= usMaxSecond() + 1;
      min++;
    }
    // Adjust and carry min.
      while (min > usMaxMinute()) {
        min -= usMaxMinute() + 1;
        hr++;
25    }
    // Adjust and carry hr.
      while (hr > usMaxHour()) {
        hr -= usMaxHour() + 1;
        day++;
      }
    // Adjust and carry mon (day adjust is dependent on mon
    and yr).
30    while (mon > usMaxMonth()) {
      mon -= usMaxMonth();
      yr++;
    }
    // Now adjust and carry day now that yr and mon is known.
35    while (day > usMaxDay( yr, mon )) {
      day -= usMaxDay( yr, mon );
      mon++;
      if (mon > usMaxMonth()) {

```

```

        mon -= usMaxMonth();
        yr++;

// Copy new values to members.
5   Assign( yr, mon, day, hr, min, sec, ms );

    CHECK( fValidate() );
    return *this;

10  // static member.
    USHORT TTimestamp::usMaxDay( USHORT year, USHORT month )
    {
        switch (month) {
            case 1:          // Jan.
                return 31;

            case 2:          // Feb.
15         return fIsLeapYear( year ) ? 29 : 28;

            case 3:          // Mar.
                return 31;

            case 4:          // Apr.
                return 30;

20         case 5:          // May.
                return 31;

            case 6:          // Jun.
                return 30;

            case 7:          // Jul.
25         return 31;

            case 8:          // Aug.
                return 31;

            case 9:          // Sep.
                return 30;

30         case 10:         // Oct.
                return 31;

            case 11:         // Nov.
                return 30;

35         case 12:         // Dec.
                return 31;
    }

```

```

//      default:
//      BOILERPLATE;
}

//*****
5 //*****
//
// TStream stream operators.
//
TStream& operator << ( TStream &buf, const TFlag &flag )
{
    if (!flag) return buf << FLAG( TRUE );
10     else return buf << FLAG( FALSE ) << flag.value;

TStream& operator >> ( TStream &buf, TFlag &flag )
{
    buf >> *(TNull*)&flag;
    if (flag.fIsNull() == FALSE)
15         buf >> flag.value;
    return buf;

TStream& operator << ( TStream &buf, const TTimestamp &ts
)
{
    if (!ts) return buf << FLAG( TRUE );
20     else {
        return buf << FLAG( FALSE )
            << ts.Year
            << ts.Month
            << ts.Day
            << ts.Hour
            << ts.Minute
25         << ts.Second
            << ts.Millisecond;

TStream& operator >> ( TStream &buf, TTimestamp &ts )
{
    buf >> *(TNull*)&ts;
30     if (!ts) {
        return buf;
    }
    else {
        return buf >> ts.Year
35         >> ts.Month
        >> ts.Day
        >> ts.Hour
        >> ts.Minute
        >> ts.Second

```

```

>> ts.Millisecond;

//*****
//*****
5 //
// ostream friend function members.
//

ostream& operator << ( ostream &os, const TFlag &flag )
{
    if (!flag) return os << NULL_TOK;
10   else return os << (STRING)flag;

//*****
istream& operator << ( istream &is, TFlag &flag )
{
    char ch, buffer[12];

15   is >> ws; // Extract leading
    whitespace.

    for (int i = 0; i < sizeof( buffer ); i++) {
        is >> buffer[i];
        if (!isalpha( buffer[i] )) break;
    }
20   if (i == sizeof( buffer ) ASSERT( FALSE );

    buffer[i] = '\x0';

    if (strcmp( buffer, NULL_TOK) == 0) {
        fSetNull();
    }
    else if (strcmp( buffer, TRUE_TOK) == 0) {
25     Assign( TRUE );
    }
    else if (strcmp( buffer, FALSE_TOK) == 0) {
        Assign( FALSE );
    }
    else ASSERT( FALSE );

30   return is;
}
//*****/

ostream& operator << ( ostream &os, const TTimestamp &ts
35 {
    {
        char tsstring[TTimestamp::TSStringLen];
        if (!ts) return os << "NULL";
        else return os << ts.ToSTRING( tsstring );
    }
}

```

```

#define INCL_NOPMAPI           // no PM in this program
#define INCL_DOS
// #define INCL_BSE
5 // #define INCL_DOSSEMAPHORES
#include <os2.h>

#include <usertype.h>
#include <TModem.HPP>

TModem::TModem( TPort &_port )
10 : port( _port )
{

TModem::RC TModem::rcSendCommand( STRING, ULONG timeout )
{
    NOTIMPLEMENTED;
}

15 STRING TModem::strSendCommand( STRING str, ULONG timeout )
{
    port.fWritePort( str );
    port.fPutChar( '\r' );
    STRING result = strGetString( timeout );
    if (strcmp( str, result ) == 0) {
20         return strGetString( timeout );
    }
    else {
        return result;
    }
}

25 STRING TModem::strGetString( ULONG timeout )
{
    UINT i = 0;
    last_result[0] = '\x0';

    // Eat Leading CR/NL.
    while (!port.fGetChar( last_result[i] )
           || last_result[i] == '\r'
           || last_result[i] == '\n') {}
30     i++;
    // (already got 1 char ok)
    // Grab text until a CR/NL.
    while (port.fGetChar( last_result[i] )
           && last_result[i] != '\n'
           && last_result[i] != '\r'
35     && i <= sizeof( last_result )) {
        i++;
    }
}

```

```

        last_result[i] = '\x0';          // Null terminate
    buffer.
        return last_result;
    }

#include <TObject.HPP>

5  //*****
   //*****
   //
   // TObject members.
   //

10 TObject::~TObject()
    {}

   //*****
   //*****
   //
   // TNull members.
   //

15 TNull::TNull( FLAG is_null )
    : isnull( is_null )
    {}

   FLAG TNull::fSetNull()
    {
20     isnull = TRUE;
        return TRUE;
    }

#define INCL_NOPMAPI          // no PM in this program
#define INCL_DOS
25 #define INCL_BSE
#define INCL_DOSSEMAPHORES
#define INCL_DOSNMPPIPES
#include <os2.h>

#include <usertype.h>
#include "TPacket.HPP"

30 TPacket::TPacket( TPort& p )
    : Port( p ),
      text_length( 0 ),
      state( TRANS_NULL )
    {}

35 TPacket::TRANS_STATE TPacket::rGetPacket()
    {
        enq_count = 0;
    }

```

```

    nak_count = 0;
    text_length = 0;

    if (state != TRANS_NULL) return TRANS_NULL;

// Enquiry Loop.
5   while (fSendENQ())
        if ((state = rReceivePacket()) == TRANS_NAK)
            while (fSendNAK())
                if ((state = rReceivePacket()) == TRANS_ACK)
10                fSendACK();
                return state;

            else if (state == TRANS_ACK)
15                fSendACK();
                return state;

        fSendEOT();
        return state;

20   TPacket::TRANS_STATE TPacket::rReceivePacket()
    {
        char ch;
        int i=0,j;

// Get STX.
25        if (!Port.fGetChar( ch ))
            return TRANS_ETD;
        packet_text[i++] = ch;
        if (ch != STX)
            return TRANS_NAK;

// Get Length.
30        if (!Port.fGetChar( ch ))
            return TRANS_NAK;
        packet_text[i++] = ch;

        text_length = (USHORT)ch;

35        if (!Port.fGetChar( ch ))
            return TRANS_NAK;
        packet_text[i++] = ch;

```

```

        text_length = (USHORT)(ch << 8) + text_length;
        if (text_length > MAX_TEXT_LEN)
            return TRANS_NAK;

// Get Text.
5   for (j=0 ; j < text_length; j++ )
        {
            if ( Port.fGetChar( ch ))
                packet_text[ j ] = ch;

            else
10         return ( TRANS_NAK );
        }

// Get ETX.
        if ( Port.fGetChar( ch ))
        {
            if ( ch == ETX )
15         packet_text[ i++ ] = ch;

            else
                return ( TRANS_NAK );
        }
        else
        {
20         return ( TRANS_NAK );
        }

// Get LRC.
        if (!Port.fGetChar( ch ))
            return TRANS_NAK;
// packet_text[i++] = ch;
        return TRANS_ACK;
25 }

UINT TPacket::cbCopyText( PVOID ptr, UINT len )
{
    len = len < text_length ? len : text_length;
    memcpy( ptr, packet_text, len );
    return len;
30 }

FLAG TPacket::fSendENQ()
{
    char enq = ENQ;

35     enq_count++;
    if (enq_count > MAX_ENQ) return FALSE;

    Port.FlushInputBuffer();

```

```

    return Port.fWritePort( &enq, 1 );
}

FLAG TPacket::fSendACK()
{
    char ack = ACK;
    Port.FlushInputBuffer();
5    return Port.fWritePort( &ack, 1 );
}

FLAG TPacket::fSendNAK()
{
    char nak = NAK;
10    nak_count++;
    if (nak_count > MAX_NAK) return FALSE;

    Port.FlushInputBuffer();
    return Port.fWritePort( &nak, 1 );
}

15 FLAG TPacket::fSendEOT()
{
    char eot = EOT;
    return Port.fWritePort( &eot, 1 );
}

20 #define INCL_NOPMAPI                // no PM in this program
#define INCL_DOS
#define INCL_BSE
#define INCL_DOSSEMAPHORES
#define INCL_DOSNMPPIPES
#define INCL_DOSDEVIOCTL
#include <os2.h>

25 #define _THREADS                    // This implemetation is
multi-threaded.

#include <process.h>
#include <string.h>
#include <stdlib.h>

30 #include "TPort.HPP"

TPort::TPort()
{
    manage_thread( -1 ),
    log_flag( FALSE )
}

35 TPort::~TPort()
{
    while (manage_thread != -1)

```

```

        KillManageThread();
        DosSleep( 1000 ); // Wait 1 second.
    }

    FLAG TPort::fOpenPort( const ComSettings &settings )
5    {
        LINECONTROL lctl;
        DCBINFO dcb;
        ULONG ulAction;
        ULONG ulPio, ulDio;
        ULONG cbTrans;

10    // Open the port.
        rc = DosOpen( settings.port_name, &hPort, &ulAction,
        0, 0, OPEN_ACTION_OPEN_IF_EXISTS,
        OPEN_FLAGS_WRITE_THROUGH |
        OPEN_ACCESS_READWRITE | OPEN_SHARE_DENYREADWRITE, NULL );
        if (rc) return FALSE;

        // Set the line speed.
15    ulPio = sizeof( settings.bps );
        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
        ASYNC_SETBAUDRATE, (PVOID)&settings.bps,
        ulPio, &ulPio, NULL, 0, NULL );

        if (rc) {
            DosClose( hPort );
            return FALSE;
20    }

        // Set the line characteristics.
        lctl.bDataBits = settings.data_bits;
        lctl.bParity = (BYTE)settings.parity;
        lctl.bStopBits = (BYTE)settings.stop_bits;
        ulPio = sizeof lctl;
        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
25    ASYNC_SETLINECTRL, &lctl, ulPio, &ulPio, NULL, 0, NULL );
        if (rc) {
            DosClose( hPort );
            return FALSE;
        }

        // Set the flow control.
30    ulDio = sizeof dcb;
        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
        ASYNC_GETDCBINFO, NULL, 0, NULL, &dcb, ulDio, &ulDio );
        if (rc) {
            DosClose( hPort );
            return FALSE;
35    }
    }

    /*****
    *****/
    dcb.usReadTimeout = 100;

```

```

        dcb.fbCtlHndShake = MODE_CTS_HANDSHAKE; // flags1 =
00001000

        dcb.fbFlowReplace &= 0x30; // flags2 =
00??0000
        dcb.fbFlowReplace |= MODE_RTS_HANDSHAKE; // flags2 =
5 10??0000

        dcb.fbTimeout &= 0xF8; // flags3 =
?????000
        dcb.fbTimeout |= MODE_WAIT_READ_TIMEOUT; // flags3 =
?????100
        *****/
10 dcb.usReadTimeout = 300;
        dcb.fbCtlHndShake = MODE_CTS_HANDSHAKE;
        dcb.fbFlowReplace = MODE_RTS_HANDSHAKE;
        dcb.fbTimeout = MODE_NO_WRITE_TIMEOUT |
MODE_WAIT_READ_TIMEOUT;

        rc = DosDevIOctl( hPort, IOCTL_ASYNC,
15 ASYNC_SETDCBINFO, &dcb, ulPio, &ulPio, NULL, 0, NULL );
        if (rc) {
            DosClose( hPort );
            return FALSE;
        }

        fRaisedDTR();
20 return TRUE;
    }

    FLAG TPort::fClosePort()
    {
        rc = DosClose( hPort );
        if (rc) return FALSE;
25 else return TRUE;
    }

    void TPort::FlushInputBuffer()
    {
        BYTE cmd; // Scratch, Needed
        by API.
        ULONG len; // Scratch, Needed
30 by API.

        rc = DosDevIOctl( hPort, IOCTL_GENERAL,
DEV_FLUSHINPUT, &cmd, sizeof( cmd ), &len,
&cmd, sizeof( cmd ), &len );
35 DosSleep(10); // Timing Kludge - Give the
Device Driver

```