

SAGE^{MAX} TM
USER MANUAL
VERSION 4.00



SAGE^{MAX} User Manual

Part Number 1E-04-00-0097

© 2002 American Auto-Matrix™

This document is protected by copyright and is the property of American Auto-Matrix. It may not be used or copied in whole or in part for any purpose other than that for which it is supplied without authorization. This document does not constitute any warranty, express or implied.

Every effort has been made to ensure that all information was correct at the time of publication. American Auto-Matrix reserves the right to alter specifications, performance, capabilities and presentation of this product at any time.

American Auto-Matrix and Auto-Matrix are trademarks of American Auto-Matrix and are not to be used for publication without the written consent of American Auto-Matrix.

All other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

WORLD HEADQUARTERS

American Auto-Matrix
One Technology Lane
Export, Pennsylvania 15632-8903 USA
Tel (1) 724-733-2000
Fax (1) 724-327-6124
Email aam@amatrix.com
www.amatrix.com

CHAPTER 1 - OVERVIEW OF THE SAGE^{MAX}

1.1 What is a SAGE^{MAX}?

The SAGE^{MAX} (see **Figure 1-1**) is a communications-intensive field panel that offers powerful programming and networking capabilities. SAGE^{MAX} is a fully distributed system that rivals the power and flexibility of centralized minicomputers at a fraction of the cost. SAGE^{MAX} is not an acronym, but rather "one who is wise and knowledgeable" as its name implies. In other words, the SAGE^{MAX} is the wisest choice for building automation.

The SAGE^{MAX} is the primary field panel of the Auto-Matrix system and offers fully distributed, programmable automation control. Like a traditional field panel, SAGE^{MAX} can be used in stand-alone control and monitoring applications, in a small network, in several small networks or in a large, high-performance local area network (LAN), such as Ethernet.

The SAGE^{MAX} field panel communicates to unit controllers using Public Unitary Protocol (PUP), a non-proprietary protocol that was developed by American Auto-Matrix and is in the public domain. Host systems communicate to the SAGE^{MAX} using Public Host Protocol (PHP), another non-proprietary protocol that was developed by American Auto-Matrix and is in the public domain. SAGE^{MAX} also provides communications drivers to support various other public domain and proprietary protocols. SAGE^{MAX} is backward-compatible with all existing Auto-Matrix systems.

SAGE^{MAX} brings together a powerful multitasking, real-time operating system and high performance microcomputer technology to provide a platform for satisfying today's complex monitoring and control demands.

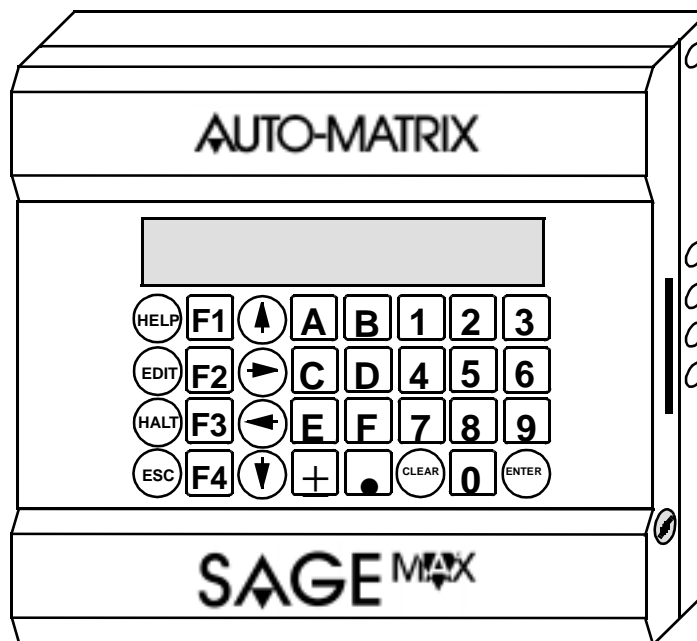


Figure 1-1 The SAGE^{MAX}

1.2 *Applications*

SAGE^{MAX} is suitable for a variety of applications, including the following:

- Add-ons to existing Auto-Matrix systems
- New Auto-Matrix systems
- Integrated information management systems
- Replacements for microcomputer-based automation systems.

1.2.1 *Add-ons to Existing Auto-Matrix Systems*

SAGE^{MAX} is designed to simply "plug in" to an existing Auto-Matrix system. Its multiple network and multi-protocol capabilities make SAGE^{MAX} an ideal addition to, or replacement for, SAC global controllers or STAR, MCU and RCU2 field panels.

1.2.2 *New Auto-Matrix Systems*

The same multiple network and multi-protocol capabilities that make SAGE^{MAX} ideal for add-ons to existing Auto-Matrix Systems make SAGE^{MAX} perfect for new Auto-Matrix systems. Combined with SPECTRA color graphic host systems and Auto-Matrix unit controllers, SAGE^{MAX} creates a building automation system ideal for any size installation. With SAGE^{MAX}, the number of Auto-Matrix products required to cover the spectrum of automation applications has been significantly reduced.

1.2.3 *Integrated Information Management Systems*

One of the reasons SAGE^{MAX} was developed was to integrate office and building automation systems. SAGE^{MAX} can address this marketplace due to its network and protocol flexibility. Compatibility with existing LAN standards for office automation allows SAGE^{MAX} to be integrated with a range of sophisticated information management tools such as spreadsheets, databases and graphic presentation programs.

1.2.4 *Replacements for Microcomputer-Based Automation Systems*

The multi-user and Ethernet capabilities of the SAGE^{MAX} make it possible to completely replace microcomputer-based automation systems. By distributing CRT terminals and PC-based workstations between SAGE^{MAX} field panels, a true emulation of the multi-user benefits of the old microcomputer-based systems can be achieved. This makes the SAGE^{MAX} an ideal, cost-effective alternative.

1.3 *Key Features of the SAGE^{MAX}*

The key features that give SAGE^{MAX} its power and flexibility are listed below and described in detail on the following pages.

- Interconnection of multiple networks
- Multi-protocol capabilities

- Supervisory and closed-loop control
- Rich control programming language
- Report generation
- Multi-user, language-independent, menu-driven operator interface
- Real-time multitasking operating system
- Industry-standard file system (MS-DOS)
- Historical data trending
- On-line/off-line database editing

1.3.1 *Interconnection of Multiple Networks and Multiprotocol Capabilities*

Two key features of SAGE^{MAX} are (1) its ability to connect multiple, possibly dissimilar, networks and (2) its ability to support multiple protocols over those networks.

As part of a network structure, SAGE^{MAX} can act in the following capacities:

- As a peer with STAR field panels on an EIA485 peernet
- As a host for MCU/RCU2/STAR field panels using eXtended Automation Network Protocol (XANP)
- As a host for PUP devices
- As a slave of a PC-based host such as SPECTRA using PHP
- As a host for PHP devices
- As part of a PHP dial-up configuration
- As a peer in a multi-SAGE^{MAX} system on an Ethernet

These diverse network structures and multiprotocol capabilities make SAGE^{MAX} suitable for a variety of applications.

1.3.2 *Supervisory and Closed-loop Control*

Another key feature of the SAGE^{MAX} is that it offers supervisory and closed-loop control.

In a supervisory capacity, SAGE^{MAX} can monitor system operations and can be programmed to perform actions based on the results.

Closed-loop control offers greater accuracy and response efficiency in direct digital control applications such as Proportional+Integral+Derivative (PID) control loops. By combining the high-speed networking and powerful programming capabilities of the SAGE^{MAX} with Auto-Matrix unit controllers for inputs and outputs, complex closed-loop control schemes such as alarm supervision, lighting and maximum demand can be created.

1.3.3 *Rich Control Programming Language with Report Generation*

Application programs that run in the SAGE^{MAX} are written in SAGE^{MAX} Programming Language (SPL). SPL is a BASIC-like programming language adapted for real-time control, with extra capabilities for custom report generation, data logging and manipulation of objects in the SAGE^{MAX} database. SPL allows the user to create customized control programs using a rich set of features that support all types of real-time control and monitoring applications.

Some features of an SPL program include:

- Unlimited number of lines
- 64K bytes of available program code
- 255 named two-character attributes
- 256 indirect references
- Floating point math support
- Access to lookup tables for scaling, conversion and general purpose storage in programs
- Job execution including report generation
- Formatted printing
- Ability to print to a log disk file for batch report generation and printing
- Standard function blocks (such as PID) that are usable by multiple programs
- Re-entrant subroutine calling ability for multiple programs
- A six-level expression stack for resolving nested expressions
- Trigonometric functions
- Mixed-mode arithmetic
- Asynchronous read/write of named object attributes
- 16 program registers (A-P)

For more information on SAGE^{MAX} programs or SPL, refer to **Chapter 11: Programming**.

1.3.4 *Multi-user, Language-independent Menu-driven Operator Interface*

SAGE^{MAX} builds on the success of previous Auto-Matrix systems by allowing multiple human operators to simultaneously interact with each field panel using easy-to-understand menu selections. While SAGE^{MAX} can support various types of sophisticated color graphic host systems, there are also times when a simple video terminal is all that is required. In either case, SAGE^{MAX} provides easy-to-understand menus.

SAGE^{MAX} menus are available in different languages, making operator interfaces very versatile. This multiple language capability of SAGE^{MAX} is easily adapted to new languages, too. In fact, every human operator using SAGE^{MAX} can use a different language at the same time.

SAGE^{MAX} provides password protection and a list of authorized user names to prevent unauthorized access. Specific operator privileges are assigned to each user so that operators with different levels of expertise can access appropriate SAGE^{MAX} functions.

1.3.5 *Real-time Multitasking Operating System*

Another key feature of SAGE^{MAX} is its real-time multitasking operating system. The principle behind a real-time multitasking operating system is that several functions or "tasks" (e.g., an operator interface task, the program executor task, the Ethernet server task, the alarm logger task, etc.) are performed simultaneously. One of the jobs of the operating system is to coordinate the multitasking of all SAGE^{MAX} tasks. The cyclic process of multitasking occurs at such a high speed that the tasks appear to run simultaneously.

1.3.6 *Industry-standard File System (MS-DOS)*

SAGE^{MAX} uses MS-DOS 6.22, an industry-standard file system, for organizing and managing its disk database and user-created files. File system functions are integrated into the multitasking system, extending the functionality of MS-DOS 6.22.

1.3.7 *Historical Data Trending*

SAGE^{MAX} offers sophisticated functions for trending and monitoring, making it easy to observe system operation and analyze its performance.

Trends can be infinite, cyclic or time-anchored on a daily, weekly, monthly or yearly basis. Samples can either be snapshot or averaged over the sample interval. For time-anchored trends, intervals of 1, 15, 30, 60 and 1440 minutes are available, as well as weekly and monthly sample intervals.

1.3.8 *On-line/Off-line Database Editing*

Having a file-based database system, the SAGE^{MAX} can offer both on-line and off-line database editing. This allows you to create database items, including programs, on a personal computer at your leisure. These items can later be copied to the SAGE^{MAX} via a 3.5" floppy disk or via Ethernet.

CHAPTER 2 - SAGE^{MAX} HARDWARE

The SAGE^{MAX} field panel consists of hardware (physical components such as the hard drive, network connectors and the front panel keypad as shown in Figure 2-1) and software (computer programs such as the operating system, the multitasking system and the menuing system). This section discusses the locations, descriptions and specifications of the hardware components of the SAGE^{MAX}. For information about SAGE^{MAX} software, refer to **Chapter 4: Software Overview**.

2.1 External Hardware Overview

The exterior of the SAGE^{MAX} field panel consists of a two-part hinged enclosure, network connectors, an optional 32-key keypad with a backlit LCD display and a contrast control, an enclosure latch, teardrop mounting cutouts and wiring knockouts (see **Figure 2-1**).

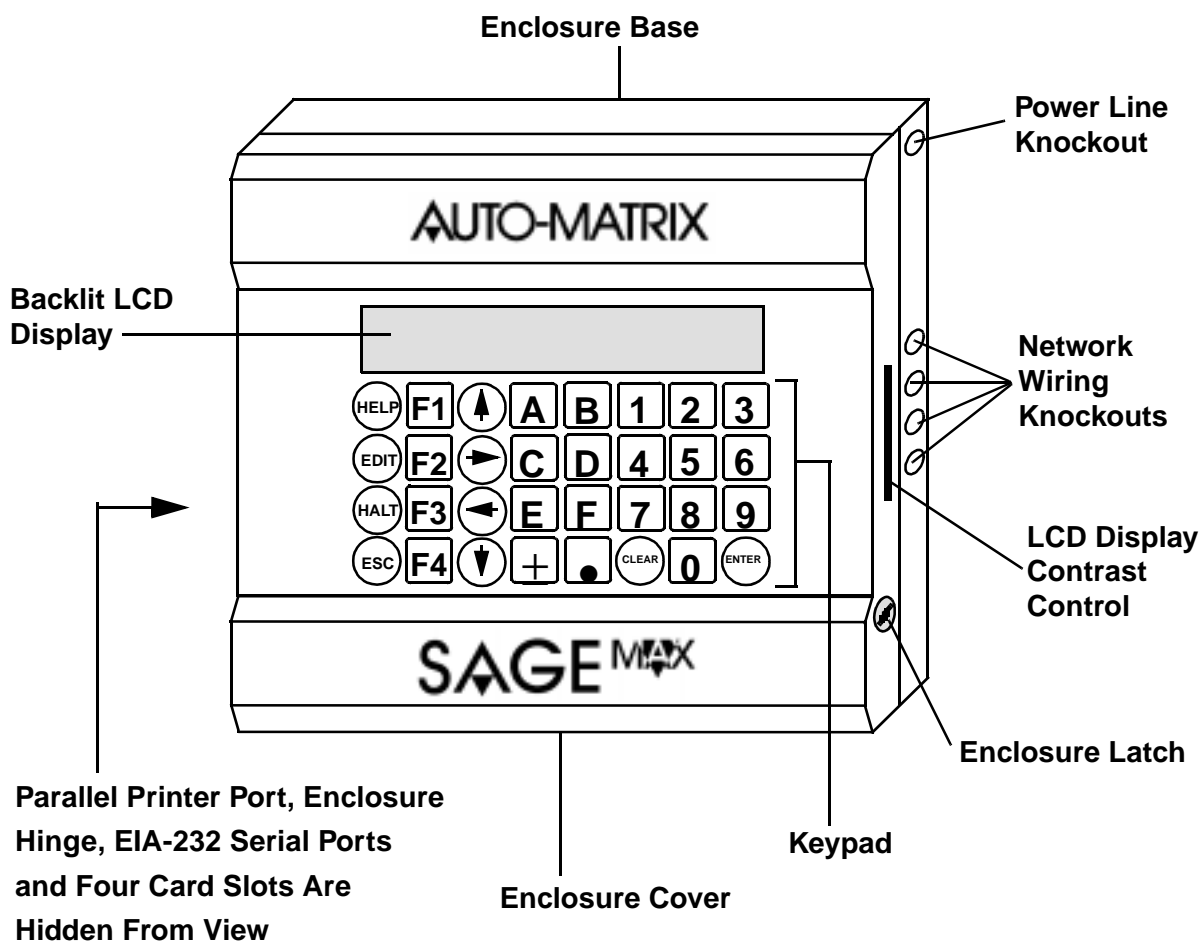


Figure 2-1 Front View of the SAGE^{MAX} Enclosure

The 32-key keypad and backlit LCD display are located on the front panel of the SAGE^{MAX} enclosure. This combination provides a functional operator interface. In addition, an LCD display contrast control is located on the right side of the SAGE^{MAX} enclosure. Sliding the control up or down increases or decreases contrast to the backlit LCD display. Refer to Figure 2-1 for the location of the contrast control.

The rugged two-part enclosure is hinged on the left side and provides protection for the internal components of the SAGE^{MAX} field panel. A latch on the right side of the enclosure provides access to the inside of the enclosure. Teardrop cutouts are located on the back of the enclosure to facilitate mounting (see **Appendix N**). Also, knockouts are provided for network wiring and power supply wiring. Refer to **Figure 2-2**.

Network and peripheral connectors are located on the left side of the SAGE^{MAX} enclosure (see **Figure 2-3**). Included are two 9-pin EIA-232D serial port connectors and a 2-pin connector for a parallel printer.

SAGE^{MAX} comes equipped with a 28.8K baud modem for dial applications. This internal modem occupies one of the four card slots (see **Figure 2-2** and **Figure 2-3**) and provides both a telephone and line connection which are accessible externally from the left side of the SAGE^{MAX} enclosure.

An optional EIA-232D card may be installed for special modem applications such as leased lines. Refer to part number 3A-01-00-0016.

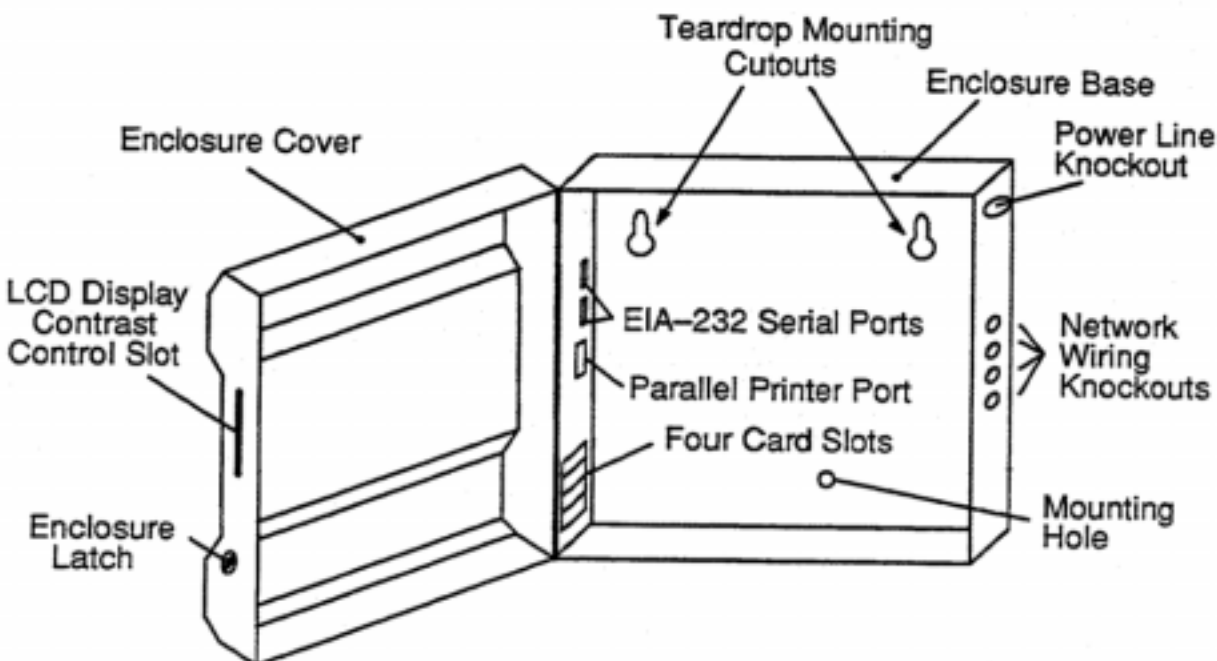


Figure 2-2 Inside the SAGE^{MAX} Enclosure

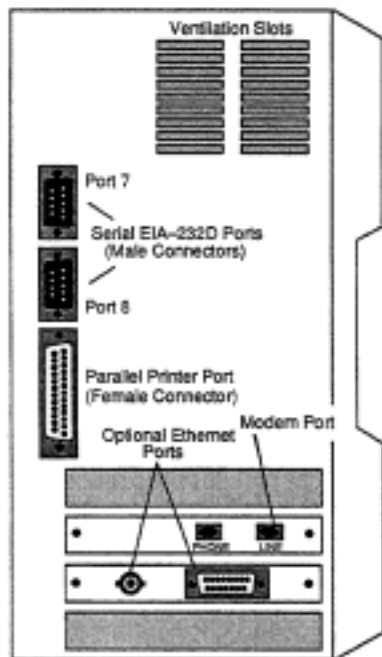


Figure 2-3 Left View of the SAGE^{MAX} Enclosure

An optical shroud is also available to cover the connectors and associated wiring of the ports on the left side of the SAGE^{MAX}. Refer to part number 4D-02-00-0129 when ordering.

Optionally, the SAGE^{MAX} offers an Ethernet card which provides your choice of a BNC connector for standard coaxial cable (i.e., *thinnet*) configurations or a DB15 connector for twisted pair (i.e., *thicknet*) Ethernet applications. Refer to **Figure 2-3**. Ethernet cards must be purchased from American Auto-Matrix only. Refer to part number 3A-01-00-0014 when ordering.

2.2 Internal Hardware Overview

The interior of the SAGE^{MAX} field panel consists of the following parts:

- 80486 (16-bit, 33Mhz) microprocessor
- watchdog timer reset
- 8MB of RAM
- 28.8 K baud Hayes-compatible modem
- 2 EIA-232D serial ports
- parallel printer port
- 4 EIA-485 dual-trunk serial ports
- 3.5" (1.44 MB) floppy disk drive and controller
- 3.5" hard disk drive (at least 400 MB) and controller
- optional Ethernet or EIA-232D card

Figure 2-4 is a dissection of major internal components of the SAGE^{MAX}.

The ventilation fan, power supply, power supply terminal block and power switch/circuit breaker are shielded from the motherboard by a metal shroud. These components are all located at the top of the SAGE^{MAX} enclosure.

Below and to the left of the power supply shroud is the disk storage module that contains the floppy and hard disk drives.

To the right of the disk storage module are network configuration switch blocks and EIA-485 terminal blocks. These components are located on the motherboard of the SAGE^{MAX}.

Below the disk storage module are four full-size card slots. These accept an internal modem, the processor board and a hard disk controller. This leaves one empty card slot which can accept an optional Ethernet or EIA-232D card.

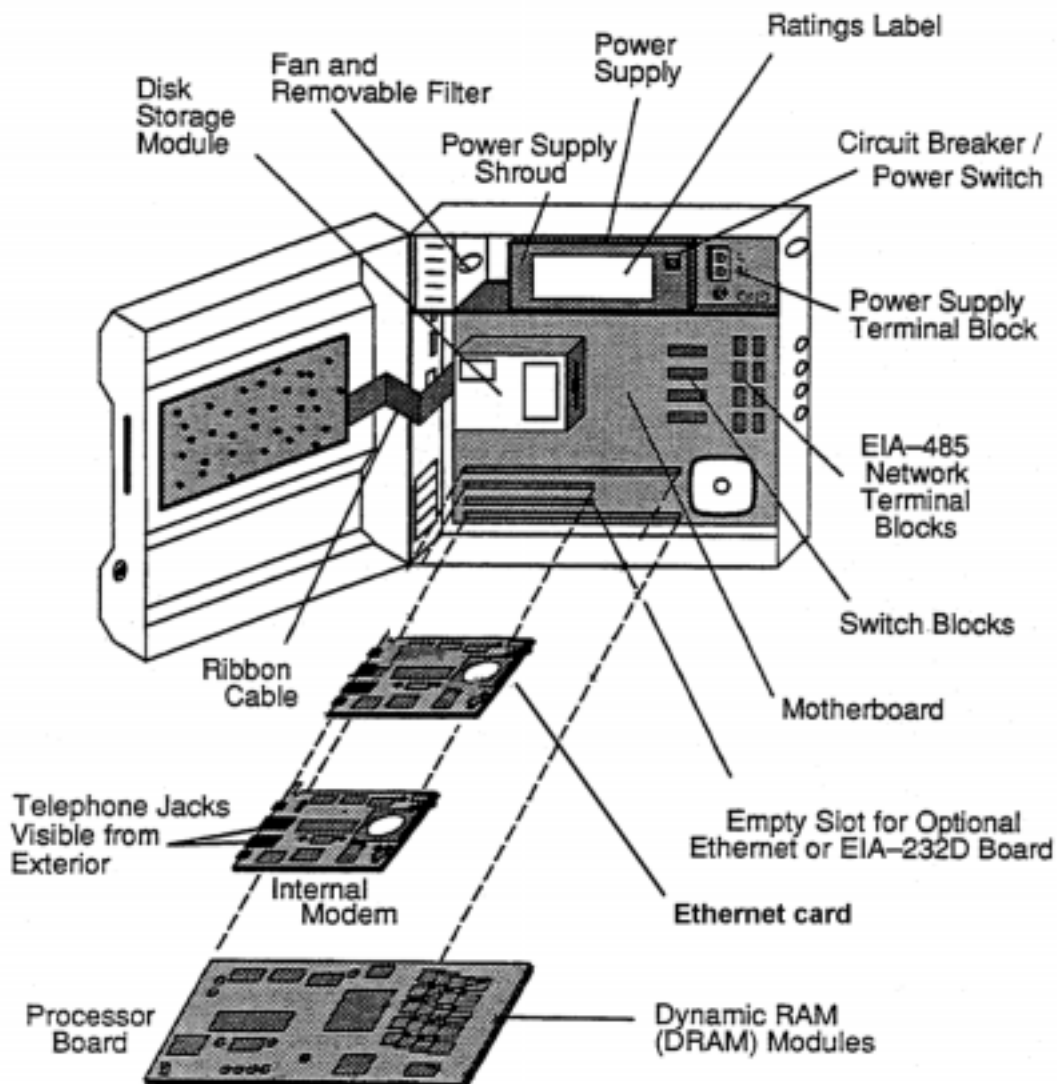


Figure 2-4 Major Internal Components of the SAGE^{MAX}

2.3 Specifications

Communication Ports

EIA-232D (2) ports:	CRTs, printers or host computers
Control Signals:	TXD, RXD, RTS, CTS
Communication Speed:	110 to 38,400 baud, programmable
Terminations:	DB9P connector
Network Protection:	Dual tranzorbs plus optical and magnetic isolation
EIA 485(4)ports:	
Network Configuration:	Daisy chain up to 5000 feet per trunk Dual trunk 2-wire/single trunk 4-wire
Communication Speed:	110 to 38,400 baud, programmable
Terminations:	Pluggable 0.2" (5.08 mm), 3-position terminal blocks (2 for each port)
Network Protection:	Dual tranzorbs plus PTCs; optical and magnetic isolation
Biasing:	Switch selectable 240 ohm termination resistors Switch selectable 510 ohm biasing resistors
Modem port:	Hayes-compatible
Communication Speed:	28.8K baud, adaptive to slower speeds
Protocol:	Public Host Protocol (PHP) or XON/XOFF Protocol
Termination:	Integral RJ-11 jack
Compatibility:	V.22bis, V.22 and V.21 international standards compatible Bell 103/212A U.S. standards compatible
Parallel port:	Centronics interface
Termination:	DB25S cornektor
Ethernet port:	Optional
Communication Speed:	10 Mbps
Termination:	BNC connector, DB15S AUI connector

Mounting

Flat surface mount, 1/2" teardrop holes, 1/4-20 recommended

Input Supply

100/120/240 VAC, auto-select, universal input, 50/60 Hz, 130 VA protected at 2.5A
0.375" (9.52 mm) fixed terminal block

Operating Environment

Temperature:	41° - 113° F (5° - 45°C)
Altitude:	0 -10,000 ft (0 - 3 km)
Relative Hulmidity:	20% - 80% Rh non-condensing

Shipping Weight

25 lb (11.3 kg)

Overall Dimensions

13.0" x 14.2" x 6.2" (33.0 cm x 36.1 cm x 15.7 cm)

CHAPTER 3 - FUNDAMENTAL CONCEPTS

This chapter describes several concepts that are fundamental to understanding the operation of the SAGE^{MAX} field panel. You may wish to skip this section if you already have a thorough understanding of object-oriented databases, ports, networks, protocols, multitasking, operating systems and MS-DOS.

3.1 *The Object-oriented Database*

In a building automation system, there are many different kinds of information available to you. For example, there are physical inputs and output control points. There are constants and known variables. There are control programs that calculate different kinds of values and there are programs used to control and implement operational strategies. In addition, there is historical information that must be gathered from the facility. Object-oriented refers to a method of organization which allows all of these different kinds of information to be examined and modified in the same basic way.

The term “object-oriented” comes from the concept that each piece of information is an independent *object* that has a name. This name describes what the object represents. For example, a temperature sensor that measures outside air temperature might be called “outside air temperature”.

In practice, most information objects have more than one principle characteristic associated with them. In the temperature sensor example, it is obvious that the temperature reading is an important characteristic of that object. However, the object in question may have other characteristics as well.

For example, various temperature limits may be of interest for alarming purposes. Information regarding the *scaling* of the input (i.e., 4-20 mA versus full scale) may be important. These types of parameters are called *attributes* and are properties of the object itself. These attributes act as the interface between you and the object (see **Figure 3-1**).

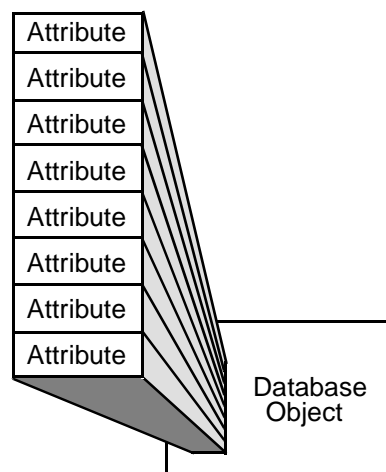


Figure 3-1 *Attributes: The Object/Operator Interface*

Object-oriented design is not limited to the physical points of monitoring and control. Control programs and control logic also share in this object-oriented design approach. In the same sense that an analog input point may have a name and attributes, you can write complex control programs that look at dozens or even hundreds of points, and then make complex sequences of logical decisions and calculations. The entire control program can then have a name and attributes that can control its behavior and influence what it does.

To simplify matters even more, with an object-oriented approach, it is not necessary to know how a built-in program was written or how it works in order to use it. In fact, the concept is to insulate the operator from the complex, internal functions of the program by presenting program information in attribute form. The internal implementation of a program can then be changed without impacting the object attributes which are visible to you. This approach also minimizes the interdependencies between interrelated program objects

Table 3-1 shows examples of object-oriented database items with sample object names..

OUTPUTS	Hot Water Valve	Floor Lights	Main AHU Fan
INPUTS	Zone Temperature	Switches	Duct Flow
CONTROL PROGRAMS	Discharge PID Loop		Schedules
VARIABLES AND CONSTANTS	Start Time	Temperature Setpoint	Holidays

Table 3-1 Examples of Named Objects

Taking the object-oriented approach one step further, Auto-Matrix has implemented all of its networking and communications systems around the notion of object-oriented devices. In other words, network communications can refer to objects and their attributes in much the same way that control programs interact with each other using named objects and attributes. This has turned out to be an extremely valuable approach, and is at the core of our ability to provide non-obsolescence and backward compatibility for all AI2100 products ever made. This is due to the fact that object-oriented networking, by its very nature, is arbitrarily extensible in the future. Since a control program on a SAGE^{MAX} field panel does not know the details of how the control is being done in the control object it is talking to (a SOLO unitary controller for example), we can completely change the control technology of the object we are talking to, and not have to change anything about the control network or communications to that object.

One advantage of using an object-oriented design is that it provides a uniform appearance among control programs, physical input and output points, calculated variables and all other forms of data and information.

Another advantage of using an object-oriented design is that it offers simplified control programming, making it easier, faster and less expensive to commission and develop a robust automation and control system.

Non-obsolescence and built-in backward compatibility allow for system expansion without penalty and lend themselves to an incremental expansion, replacement, refinement and development methodology.

3.2 What Are Ports?

A port is a communication channel on a field panel or host system that is used to connect peripheral equipment or other devices. A device's port is like a door through which all communication (both *to* and *from* the device) must pass.

The parallel printer port of the SAGE^{MAX} is a communication channel through which the SAGE^{MAX} sends information to be printed. *Parallel* refers to the way that the data is sent and received through the port--eight parallel wires have either a high or low signal at any given instant. This group of eight instantaneous high/low signals translates into a single byte (eight bits) which the printer interprets as an ASCII character and prints. A parallel port can, therefore, transmit or receive one character at a time.

The two EIA-232D serial ports of the SAGE^{MAX} are communication channels for local CRT terminals, serial printers or host computers. Serial ports differ from parallel ports in that data is sent and received over only two wires in a one-bit-at-a-time fashion. A device with a serial port must wait until at least eight bits (eight high/low signal time periods) are sent or received before the data byte can be interpreted or used.

Like the EIA-232D ports, the EIA-485 network ports of the SAGE^{MAX} also transmit and receive data serially. These ports are used as communication channels that link one or more unit controllers so that the SAGE^{MAX} can act as a supervisor and use their distributed resources.

SAGE^{MAX} also has a modem port - a communication channel through which dial-up applications may occur.

Ethernet and EIA-232D ports are optional communication channels that are available for use with the SAGE^{MAX}. These optional ports require the purchase of a separate card that must be installed in a free slot of the SAGE^{MAX} hardware platform.

Figure 3-2 shows the locations of the standard ports on the SAGE^{MAX} field panel. The optional Ethernet port is also shown. Notice that you have the option of selecting either the BNC connector for coaxial (*thinnet*) Ethernet applications or the DB15 connector for twisted pair (*thicknet*) Ethernet applications.

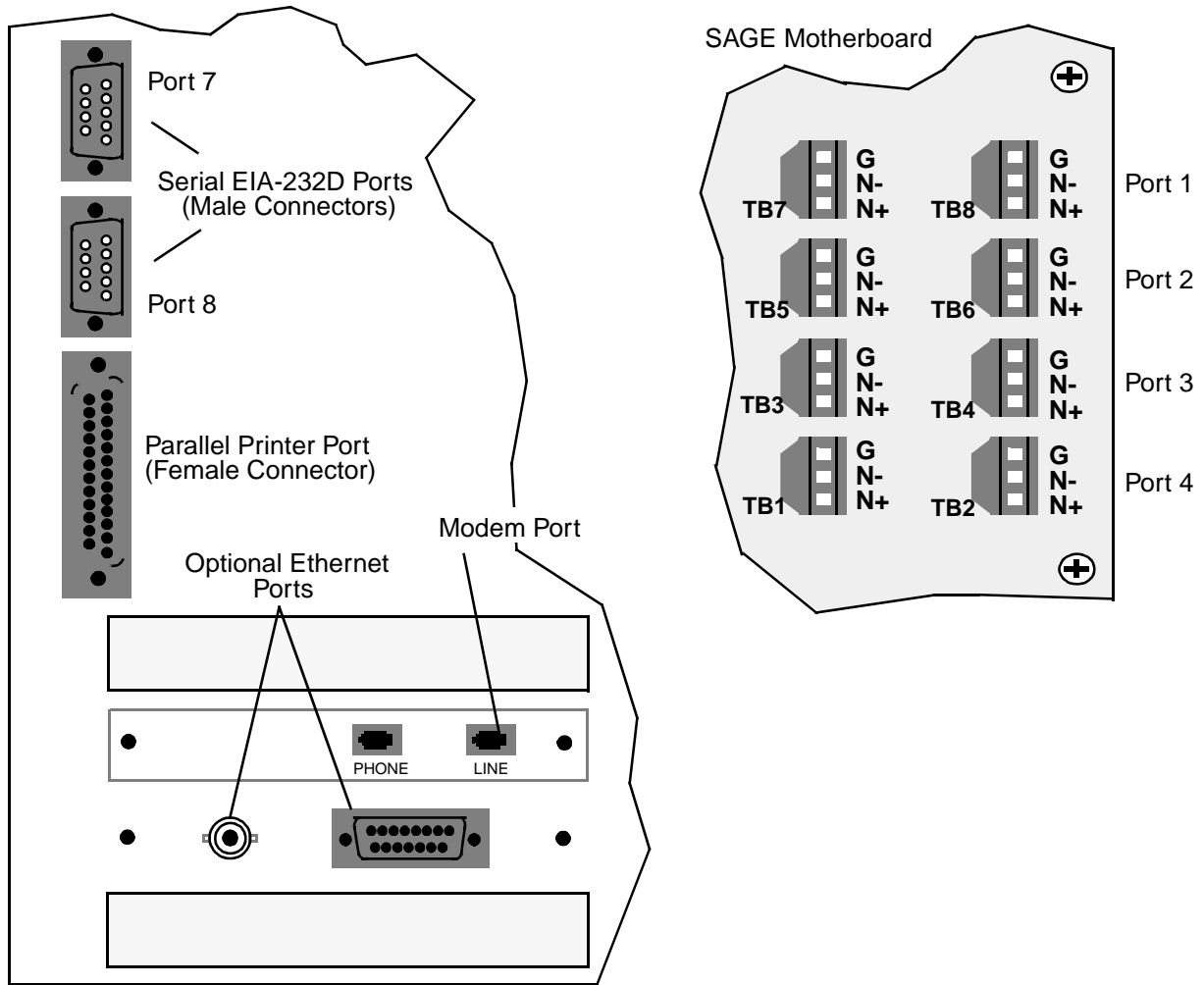


Figure 3-2 SAGEMAX Ports Enclosure (left) Internal EIA-485 Network Ports (right)

3.3 What Are Networks?

A network is a physical communication line that connects two or more similar devices through their ports. Networks allow similar devices to communicate with one another, either sharing information or acting in a host/slave capacity. Through hardware and software control, networks are able to carry information from the port of one device to the port of another device on the same network.

The information that is “carried” from one device to another is actually transmitted from one device and sent out over the entire network. All devices listening on this network must determine if the message that was transmitted should be processed or ignored.

There are many types of networks. It seems obvious that an arbitrary selection of devices cannot be randomly “connected” in hopes of creating a network. The key word in the definition of a network is “*similar* devices.” The term *similar*, in this case, refers to the rules that govern how the networked devices communicate -- more simply, the *language* or *protocol* that is spoken over the communication line. If the protocols of two or more connected ports are the same, you have a network. .

Other examples of networks are shown in **Figure 3-3** below.

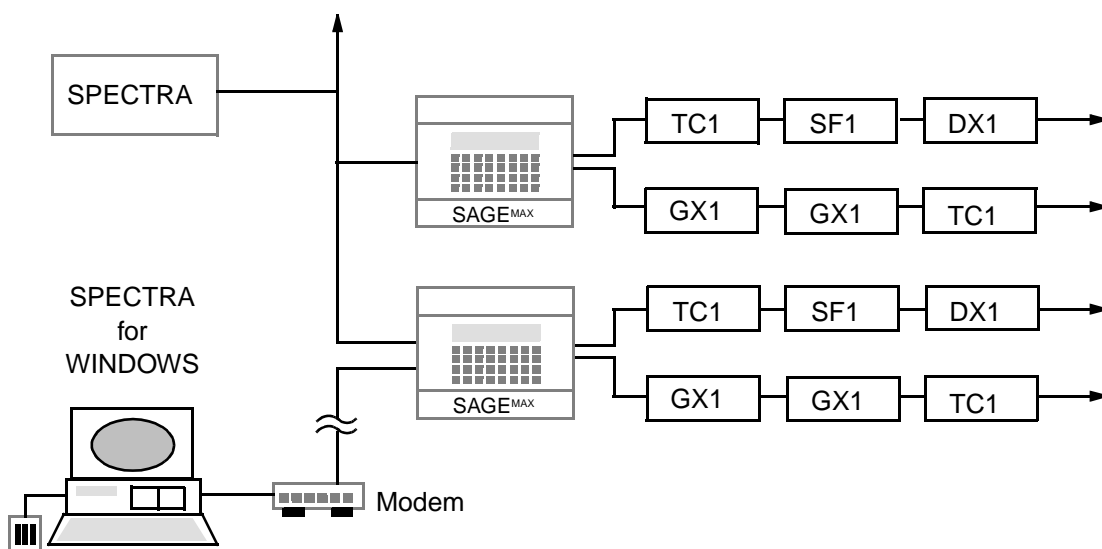


Figure 3-3 Examples of Auto-Matrix Networks

3.4 What Are Protocols?

A protocol is a set of rules that governs how a device communicates with other devices. The nature of the information that is sent or received through any given port over a network depends on the protocol. The protocol not only determines the format of what is to be sent, but also how it is to be sent, when to send it and if a response or acknowledgment are expected.

For example, two devices, "A" and "B", are connected over a simple network that uses a very simple protocol. The action requested by device “A” is converted into its corresponding protocol message and then transmitted over the network. Since devices “A” and “B” are

similar (i.e., they understand the same protocol), device “B” can interpret the protocol command and execute the requested action.

Typical network protocols are much more sophisticated and include much more information in their message formats. Many protocols require that destination devices issue acknowledgments or responses after a command is sent to them. Also, most protocols have a built-in error-checking procedure, like a checksum calculation for example, that is a function of the contents of the protocol message itself. A checksum byte might be placed at the end of the protocol message.

Just as there are many types of networks, there are many types of network protocols. The communications-intensive SAGE^{MAX}, available with several protocol-configurable ports, can link several dissimilar networks, creating a multiprotocol hub from which resources can be shared and networks can be integrated.

Protocols supported by SAGE^{MAX} include:

- PHP
- PUP
- XANP
- STAR Peernet Protocol
- Ethernet Datagram Protocol (EDP).

3.5 *What Is Multitasking?*

Multitasking is a software technique that allows the microprocessor to perform several tasks at the same time. The cyclic process of multitasking occurs at such a high speed that the tasks appear to run simultaneously. In reality, only one task is using the microprocessor of the SAGE^{MAX} at any given instant in time. Any unassigned tasks are ignored.

3.6 *What Is an Operating System?*

An operating system is a set of programs and routines that guides a computer in the performance of its tasks, routes requests, assists the programs (and the programmers) with supporting functions and increases the usefulness of the computer’s hardware. One of the jobs of the operating system is to coordinate the multitasking of the SAGE^{MAX}. For more information, refer to **Chapter 4.1: The OB Operating System**.

3.7 *How SAGE^{MAX} Uses MS-DOS*

MS-DOS is an operating system that provides a set of services for operating a PC-type device, an operator interface, file services and single task execution. SAGE^{MAX}, however, does not use MS-DOS in these capacities. By the nature of its design, MS-DOS is unable to handle the multitasking requirements of the SAGE^{MAX} by itself.

The role of MS-DOS within the SAGE^{MAX} is to provide file services such as reading and writing files to and from the SAGE^{MAX} hard disk. By making MS-DOS file services available

under a larger, more robust *multitasking* operating system, SAGE^{MAX} is able to take advantage of the file services provided by MS-DOS without its limitations.

Although MS-DOS diskettes are supplied with the SAGE^{MAX}, the parts of MS-DOS that are used by the SAGE^{MAX} are installed in the SAGE^{MAX} at the factory.

3.8 Files and Directories

When you store information on any media, whether on paper or on magnetic media such as a hard disk, as in the SAGE^{MAX}, it is convenient to organize the information into discrete, meaningful packets. When these packets of information are stored on magnetic media, we refer to them as *files*. It is convenient to think of files as folders of information.

If we visualize *files* as folders of information, then *directories* are simply drawers that hold groups of files in some relevant order. See **Figure 3-4**

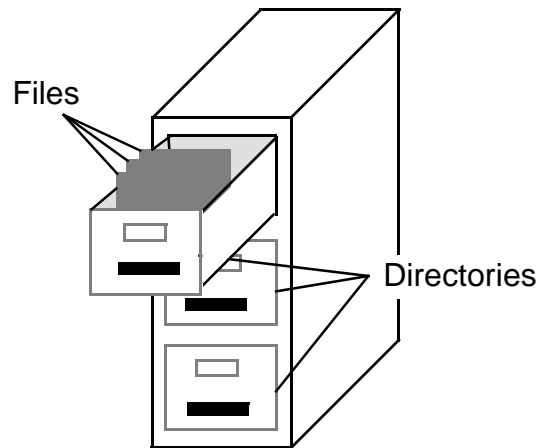


Figure 3-4 Files and Directories

There can be any number of files and/or other subdirectories in any given directory. In fact, files can be placed in directories that are elements of other directories. This *nesting* of directories allows many files to be stored on a hard disk and organized in a logical way.

It is often convenient to visualize the nesting of directories by using a *tree* diagram. In a tree diagram, the main or *root* directory is usually shown at the top. Files and subdirectories within this root directory are shown as branches from the root directory. Subdirectories that have further subdirectories are shown as branches further down the tree diagram. A sample tree diagram is illustrated in **Figure 3-5**.

In this example, the root directory **C:** is the first level in a multilevel directory system. **C:** represents the volume letter that corresponds to the disk volume on which the files are located. The physical hard disk drive of the SAGE^{MAX} is split into two volumes called **C:** and **D:**.

You can maneuver through the directory structure by changing your *working directory*. This enables you to navigate through the multilevel directory system. From your working directory you can view, create and delete files and other subdirectories.

The full path name of a file includes the root directory, subdirectories and the file name separated by a back slash “\.” For example, in **Figure 3-5**, the file **oss2.spl** is located in subdirectory **OSS**. Subdirectory **OSS** is located on directory **SPL** of the root directory **C:**. Therefore, the full path name of the file is **C:\SPL\OSS\loss2.spl**.

Root Directory C:\

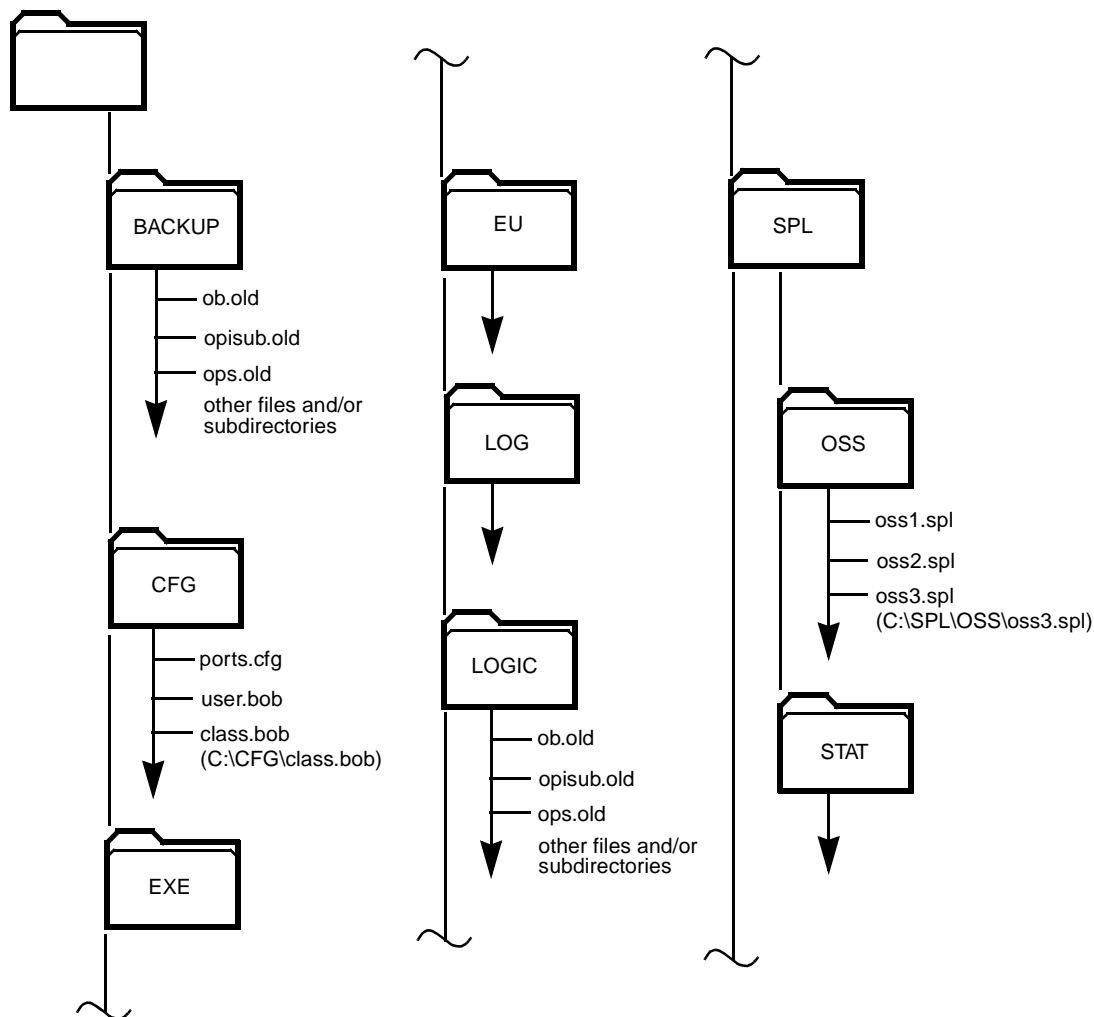


Figure 3-5 Sample Diagram Showing Directories and Files

Directories and subdirectories have names up to eight characters in length. Valid characters for directories and subdirectories are:

- a-z (lowercase letters)
- A-Z (uppercase letters)
- 0-9 (numbers)
- \$ (dollar sign)
- ~ (tilde)
- ! (exclamation point)

- # (number sign)
- % (percent sign)
- ' (apostrophe)
- ` (grave accent)
- - (hyphen)
- . (period)
- @ (at sign)
- _ (underline)
- { (left brace)
- } (right brace)
- , (comma)

File names consist of a name and an optional extension. The name portion can be up to eight characters in length. The extension portion is optional and can be up to three characters in length. Valid characters for file names are the same as valid characters for directories and subdirectories. The syntax for file names is **filename.ext**, where **filename** represents the name and **ext** represents the optional extension.

NOTE

The filenames aux, con, prn and nul (and their uppercase counterparts) are reserved by MS-DOS. Never use these names as filenames.

There are two special characters that can be used when you are using filenames in MS-DOS commands. These characters are called *wildcards*. The two wildcard characters are the question mark (?) and the asterisk (*).

The question mark is a position-sensitive wildcard. Used in a filename, it means that any valid character can occupy the position(s) it occupies. For example, **AUG??.TXT** represents all five-character **.TXT** files that have **AUG** as their first three characters.

The asterisk used in a filename means that any valid character(s) can occupy its position or any remaining positions in the name. For example, **SPL*.BOB** represents all **.BOB** files that begin with **SPL** such as **SPLOSS.BOB**, **SPLMX.BOB**, **SPLLITES.BOB** and **SPL.BOB**. Similarly, **SPL*.*** represents all files, regardless of their extensions, that begin with **SPL**. The special wildcard ***.*** represents all files of the current directory.

CHAPTER 4 - SOFTWARE OVERVIEW

4.1 *The OB Operating System*

The operating system requirements of SAGE^{MAX} are realized by the combination of MS-DOS file services and a multitasking executive program called OB.

The object-oriented multitasking kernel (OB) provides a pre-emptive multitasking environment for up to 32 tasks. A *pre-emptive* multitasking environment means that each task is given a maximum amount of processor time to use. When this *time slice* expires, OB “pre-empts” or “swaps” the task and passes processor control to the next task.

The responsibilities of OB are not limited to handling the multitasking system. OB is also responsible for allocating system resources and overseeing, routing and servicing requests.

4.2 *Messages*

Usually SAGE^{MAX} tasks rely on other SAGE^{MAX} tasks for information. For example, task 7 (a local CRT interface) may request information from a PUP network on the NET2 port (Task 2).

The SAGE^{MAX} provides a general mechanism for asynchronous intertask communications called the Intertask Message (ITM) System. This system sends messages back and forth between tasks. These messages are used internally to convey the necessary information from one task to another.

4.3 *Drivers*

A driver is a software program that is used to customize the functions of a particular port. By assigning “specific personalities” to standard ports, you can create very flexible, communications-intensive SAGE^{MAX} networks. Drivers manage/administrate networks (e.g., handle token passing, alarm polling, etc.) and provide access to network information for other parts of the SAGE^{MAX} (e.g., programs, trends, etc.).

It is the responsibility of the driver to provide a uniform set of functions that is used by the SAGE^{MAX}. Driver functions include:

- Management of read/write requests
- Alarm message management
- Virtual terminal handling
- File transfer capabilities

Not all drivers provide every function listed above. For example, the PUP and XANP drivers do not provide file transfer capabilities like the PHP driver.

Each driver has an associated setup program that is used to define driver-specific variables such as network unit numbers, units to be polled, units that are peers, etc.

The combination of a driver and its associated setup program is called a driver type. Driver types are divided into two categories: operator interface (OPI) and communications.

The OPI is a special driver type which is used to customize a port for use with a VT100 or dumb terminal.

Communications driver types customize ports for different network protocols/configurations including XANP, PUP, PHP slave, PHP host, STAR Peernet and Ethernet. The following list shows some of the driver types that are supported by the SAGE^{MAX}.

- VT 100 OPI
- Dumb terminal OPI
- XANP
- Pup
- PHP Slave
- PHP Host
- Star Peernet
- Ethernet

4.4 *The Operator Interface (OPI)*

The OPI is a special driver type which acts as the primary interface mechanism between you (through a terminal) and the SAGE^{MAX}. Both VT100 and dumb terminal modes of operation are supported through the OPI.

As the primary SAGE^{MAX}-to-human interface, the OPI, provides access to all SAGE^{MAX} functions through a menu-penetration mechanism that is part of the OPI. From the Main Menu you can access lower-level submenus.

Customizable user names, passwords and 32 definable privileges protect the SAGE^{MAX} Main Menu and other features from unauthorized users.

You can also customize SAGE^{MAX} menus and make them language-independent. These menus can then be associated with particular users, providing operator-assigned languages.

4.5 *The Database Subsystem*

As discussed in **Chapter 3.1**, pieces of information (e.g., inputs, outputs, control programs, constants, variables, trends, etc.) are stored in various forms as *named objects* in the SAGE^{MAX}.

The SAGE^{MAX} operator interface provides menus that allow you to create, modify, list and delete objects in the database.

In addition to the create/modify/delete mechanisms, SAGE^{MAX} has additional mechanisms that manage the migration of objects between disk and memory.

The following three elements comprise the Database Subsystem of the SAGE^{MAX}:

- A collection of named objects
- Create/modify/list/delete mechanisms for named objects
- Mechanisms for managing object migration between disk and memory

4.6 *The Alarm Logging Subsystem*

The Alarm Logging Subsystem is used to classify, record (log) and route alarms and their associated messages.

When an alarm occurs, SAGE^{MAX} determines the appropriate alarm class for proper alarm handling. Each alarm is then appropriately recorded in the proper alarm history files located on the **C:\LOG** subdirectory.

Each alarm class can be customized to send alarm messages to one or more “logging files.” The most common of these is the general alarm file **C:\LOG\GENERAL.LOG**. The **GENERAL.LOG** file, like all alarm history files, is not limited in size.

In addition to the **GENERAL.LOG** file, each alarm can be sent to an appropriate class alarm file. Class alarm files can have either numeric file names (e.g., **C:\LOG\001.LOG**, where the number **001** represents the class of the alarm) or three-character file names (e.g., **C:\LOG\FIR.LOG**, where the name **FIR** represents a user-defined class name).

Alarm classes may also be configured to send alarms to a special acknowledgment file. The file name **C:\LOG\ALARMS.LOG** contains unacknowledged alarms that must be acknowledged by privileged users before they are removed from the **ALARMS.LOG** file.

Through classes, alarms may be routed to multiple destinations based on time-of-day, day-of-week and holiday information. Alarms can be sent to a single port, multiple ports, a single unit on a port or multiple units on a port.

Dialing destination sets are used to route alarms to multiple remote destinations over telephone lines through the built-in modem of the SAGE^{MAX}.

4.7 *The Application Programming Subsystem*

The SAGE^{MAX} has an extremely versatile Application Programming Subsystem that can be used to make database objects which represent an underlying application program.

The Application Programming Subsystem uses SAGE^{MAX} Programming Language (SPL). SPL offers a “seamless” interface between program objects and other objects of the database. In addition, the SAGE^{MAX} operator interface includes integrated edit/compile/debug/execute mechanisms to simplify the development of SPL programs.

An SPL program is a collection of the following parts:

- A Program Logic Block
- An optional Program Reference Block
- An optional attribute Initial Value File
- Program control information

A program is a database object. Program objects must contain a Program Logic Block (PLB) and may contain an optional Program Reference Block (PRB) and/or an attribute Initial Value (INI) File.

The PLB is a binary data file that contains compiled program code which is ready to be executed. This file is created by the SPL Compiler after you create, edit and compile an ASCII source file. See **Figure 4-1**.

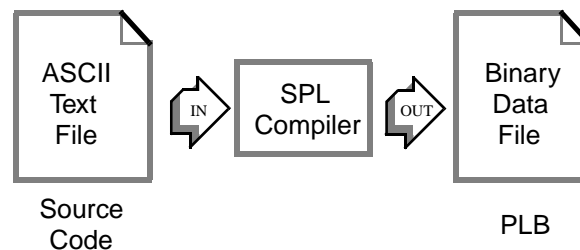


Figure 4-1 Creation of the PLB

SPL offers a referencing function that allows programs to reference groups of named objects indirectly. If a PLB uses these references as part of its logic, the associated program object must reference a PRB. The PRB is an ASCII text file that qualifies the references used in the PLB. The PRB can define up to 256 references (0-255) for use by the PLB.

A third component of an SPL program is an attribute Initial Value File or INI file. This ASCII text file is used to set program attributes to an initial value prior to program execution. If a program attribute is not listed in the INI File, the attribute is initialized to zero. SPL provides mechanisms that can be used to save new initial values to this file.

For more information on the Application Programming Subsystem, refer to **Chapter 11: Programming**.

4.8 *The LAN Subsystem*

The Local Area Network Subsystem (LAN) is a group of drivers that are used to customize a SAGE^{MAX} port for different LAN media.

It is the job of each LAN driver to handle peer-to-peer interaction across the LAN using the same set of functions that are used by other drivers. These functions include:

- management of read/write requests
- alarm message management
- virtual terminal handling
- file transfer capabilities

Additionally, the LAN Subsystem extends the Database Subsystem, allowing database objects to be shared by peers on the LAN.

Ethernet Datagram Protocol (EDP) is an example of a driver type that is used in LANs.

4.9 *The Scheduling Subsystem*

The Scheduling Subsystem provides a mechanism for executing certain application programs as background functions. These applications can be scheduled to execute at future times/dates or periodically.

Scheduling is available for several applications. They include the following:

- Broadcasting Messages To Ports/units
- Database Translations
- Data Capture/Data Stuff
- File Upload/download
- Report Generation
- Spooling

For more information on scheduling, refer to **Chapter 8: Menu Operations**.

4.10 *The Trending Subsystem*

The Trending Subsystem manages the collection of information at fixed intervals and the storage of this information in file-based trend objects. Each object specifies up to eight variables to be trended, the type of trend (infinite, circular or time-anchored), the anchor interval (hourly, daily, monthly, weekly and yearly) and the sample time (every minute, every 15 minutes, every 30 minutes, hourly, daily, weekly and monthly).

The OPI provides menus for displaying and manipulating trend objects. Trend uploading and downloading capabilities are also available through the OPI.

For more information on the Trending Subsystem, refer to **Chapter 19: Historical Data Trending**.

CHAPTER 5 - INSTALLATION & HARDWARE CONFIGURATION

5.1 *Choosing a Location*

Before you begin installing or configuring the SAGE^{MAX} field panel, you must first decide on a location that provides the following:

- a stable 100/120/240 VAC electrical outlet or electrical service conduit
- an environment that is 41°-113° F (5°-45° C), between 20-80% non-condensing relative humidity and at an altitude under 10,000 feet
- a mounting area free from moisture or leakage
- a mounting area unobstructed by equipment or machinery that could make accessing the system difficult
- adequate clearance for the enclosure door to swing completely open

5.2 *Equipment Needed*

To install your system, you will need the following:

- a flat-blade screwdriver
- mounting hardware (three 20x1/4" pan-head machine bolts and associated hardware as needed)

5.3 *Mounting the SAGE^{MAX}*

The SAGE^{MAX} is mounted using two teardrop cutouts on the back of the enclosure. These allow the SAGE^{MAX} enclosure to be slipped onto two 20x1/4" panhead machine screws and then slid into place. A third mounting hole is located 4.05" from the right teardrop cutout and secures the SAGE^{MAX}. **Figure 5-1** illustrates the SAGE^{MAX} being mounted on two parallel rails. The SAGE^{MAX} may also be flush mounted. Refer to **Appendix N** for additional SAGE^{MAX} mounting instructions.

5.4 *Power to the SAGE^{MAX}*

Connecting AC power to the SAGE^{MAX} is done by feeding the 3-wire power line through the power knockout on the right side of the SAGE^{MAX} into the enclosure.

The AC service to the SAGE^{MAX} power supply can range from 100-240VAC at 50/60hz. The power supply automatically adjusts according to the power being supplied.

NOTE

The power line must be enclosed in conduit to meet FCC Class B guidelines.

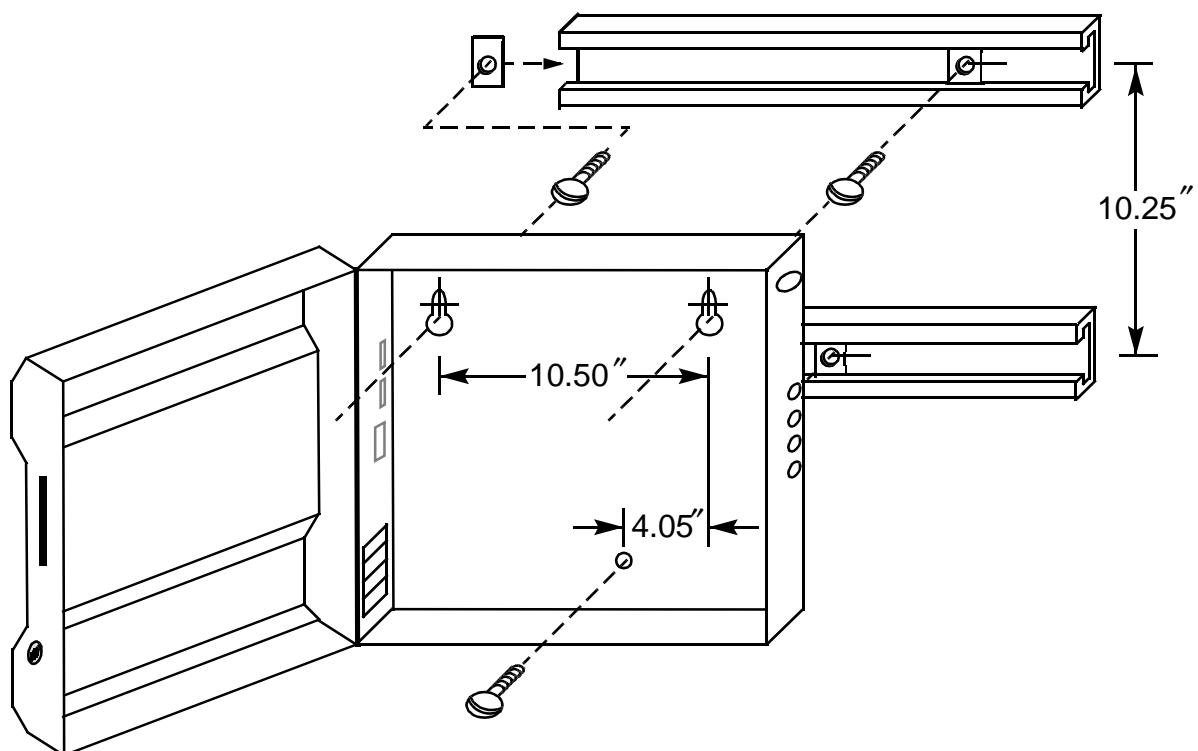


Figure 5-1 Typical Mounting for the SAGE^{MAX}

Before wiring the AC power line to the SAGE^{MAX}, be sure that the corresponding circuit breaker is **OFF** to prevent electrical shock or injury.

The use of a Romex connector or grommeted cable relief sleeve is advised to provide strain relief for the input power line.

Attach the neutral (cold) line to the **N** terminal, the line voltage (hot) to the **L** terminal and the ground line to the **GND** lug as shown in (REF TO FIGURE 5-2). Standard line cord color codes are shown in (REF TO TABLE 5-1).

	AMERICAN	EUROPEAN
Neutral (cold)	White	Blue
Line Voltage (hot)	Black	Brown
Ground	Green	Green with yellow stripe

Table 5-1 Standard Line Color Codes

Power may now be restored by resetting the appropriate circuit breaker and setting the power switch to the **ON** position.

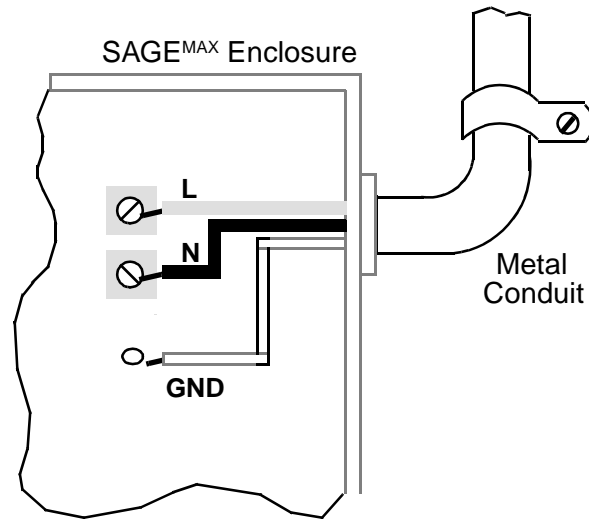


Figure 5-2 Supplying Power to the SAGE^{MAX}

NOTE

Although the SAGE^{MAX} power supply has built-in line filtering hardware, it is recommended that the AC power for the SAGE come from a dedicated source to further protect the SAGE^{MAX} against radio frequency interference (RFI), electrical spikes and power surges.

5.5 EIA-485 Network Connections

The SAGE^{MAX} can have up to four dual-trunk serial networks. The eight network terminal blocks (see **Figure 5-3**) are located inside the SAGE^{MAX} enclosure and are accessed through a series of four knockouts located on the right side of the SAGE^{MAX} enclosure.

Each pluggable terminal block is “keyed” (i.e., it can be inserted in only one direction). The terminal blocks accept 14-22 AWG wire (18-22 AWG is recommended) and are top loading with screw terminals on the left side.

The four EIA-485 networks can be used in half-duplex or full-duplex configurations. All associated network wiring extends through the SAGE^{MAX} enclosure knockouts into metal conduit. Refer to **Figure 5-4**.

Half-duplex communication is the most common EIA-485 configuration because only two-wire shielded cable is required. This is a low-cost alternative to four-wire cable when full-duplex operation is not required.

For half-duplex configurations, use terminal blocks TB1-TB8. Network wiring is connected to the positive and negative terminals of these terminal blocks. In half-duplex mode, all network communications (both transmit and receive signals) pass through the single terminal block (TB1-TB8).

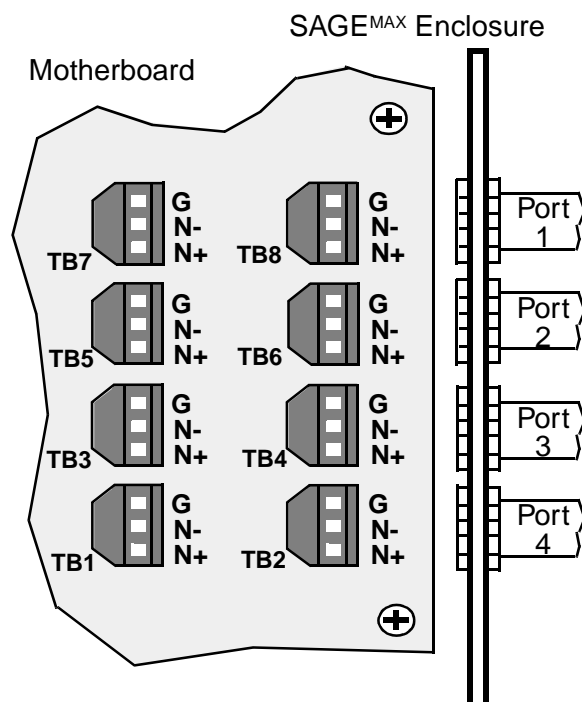


Figure 5-3 Terminal Blocks for the EIA-485 Networks

In the half-duplex (2-wire) mode, each port provides two communication trunks (A and B). Many of the SAGE^{MAX} driver types (not all) support operation on both trunks.

Unlike half-duplex mode, where transmit and receive signals must share time on the two-wire network, full-duplex mode has dedicated lines for both transmitting and receiving data. This enables full-duplex networks to communicate in two directions at the same time.

For full-duplex configurations, use the terminal block pairs TB1/TB2, TB3/TB4, TB5/TB6 and TB7/TB8. Network wiring for the transmit data lines is connected to the positive and negative terminals of terminal blocks TB1, TB3, TB5 and TB7. Network wiring for the receive data lines is connected to the positive and negative terminals of terminal blocks TB2, TB4, TB6 and TB8.

NOTE

When a SAGE^{MAX} network is configured for full-duplex operation, you must use two twisted pairs with a single shielded jacket.

To meet Class B FCC guidelines, wiring network devices that are in the same or different buildings requires that each network shield must be wired through a ceramic 470-650 pF, 3000 V capacitor and secured to the device's enclosure which must be grounded to earth. Intermediate shield wires must also be connected in addition to using the capacitor. The location of the capacitor must be within one inch of entering the device's enclosure to meet Class B FCC guidelines. Refer to **Figure 5-4**.

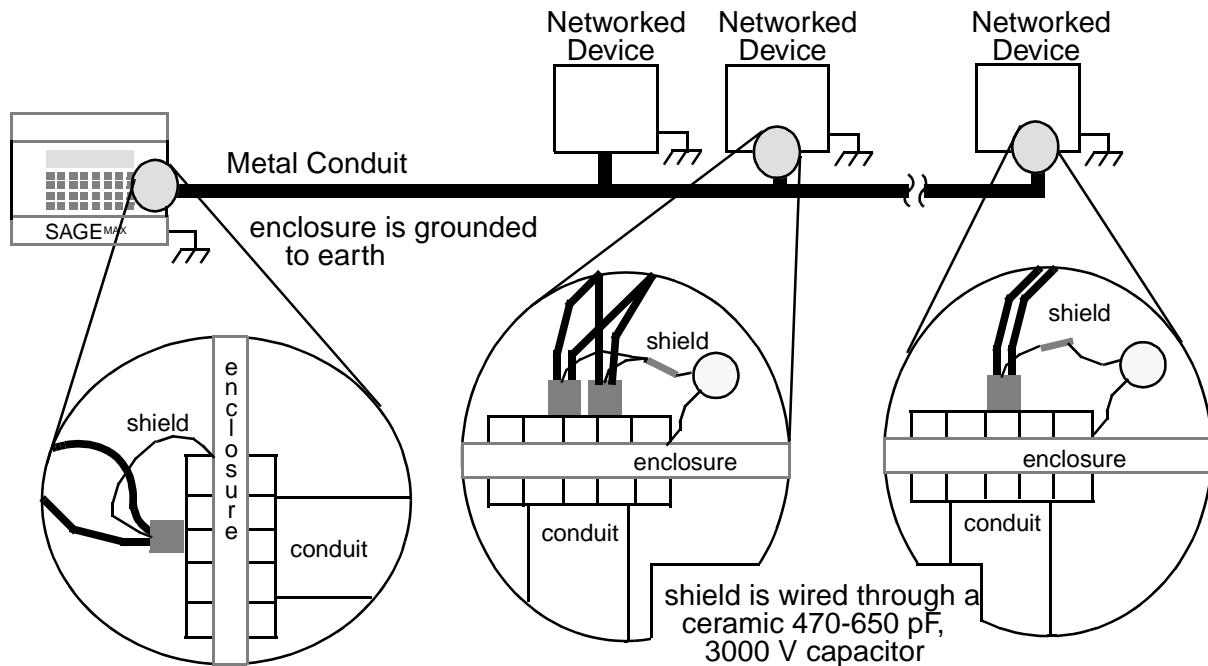


Figure 5-4 Proper Shield Wiring of Networked Devices

5.6 Switch Settings

A 7-position internal switch block is used to configure full- or half-duplex mode, a termination resistance and a biasing resistance for each of the four EIA-485 networks.

The four 7-position switch blocks are located to the left of their associated terminal block pairs on the motherboard of the SAGE^{MAX}. The meanings of the switches are shown in **Figure 5-5**.

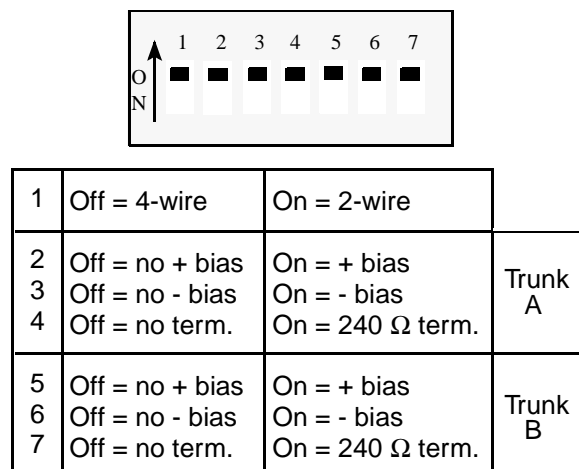


Figure 5-5 EIA-485 Network Switch Settings

NOTE

The pairs of biasing resistor switches (2/3 and 5/6) must be either both on or both off for each trunk.

Biasing resistors are used to maintain proper line voltage levels on an EIA-485 network when no driver is enabled. EIA-485 driver devices such as the SAGE^{MAX} allow these biasing resistors to be switched into and out of the network so proper line voltages are maintained. Other EIA-485 driver devices from American Auto-Matrix include:

- CC/PC
- Communication Converter
- FiberDrop
- GPUP/C1
- CC/485

For networks configured with a single EIA-485 driver device, the biasing resistors should be switched into the circuit (the **ON** position).

When configuring a network with multiple EIA-485 driver devices, only two of the devices should have biasing resistors enabled for network distances up to 5000 feet. Failure to limit this number to two causes the total bias of the network to be compounded to an extreme level which can cause problems with communications or a complete failure to communicate at all.

In their **ON** positions, switches 2/5 and 3/6 add biasing resistors to the +5 V (positive) and ground (negative) EIA-485 lines, respectively. If pull-up and pull-down biasing resistances are not required, these switches should be in the **OFF** position.

Termination resistors are used at the ends of EIA-485 multidrop networks. The SAGE^{MAX} allows a 240-ohm termination resistance to be switched across the network at the ends of a multidrop configuration.

Switches 4 and 7 activate termination resistances for EIA-485 trunks A and B respectively.

Using the 240-ohm or 120-ohm termination resistance can increase signal levels, which may decrease line reflections.

Termination resistances are selected for the ends of multidrop networks. All other termination resistors should be off.

5.7 Local Terminal Connections

The SAGE^{MAX} has two 9-pin EIA-232D serial ports for connecting one or two local CRT terminals. The DB9 male connectors are located on the left side of the SAGE^{MAX} enclosure. **Figure 5-6** shows the connectors, as well as identifies the connector pins.

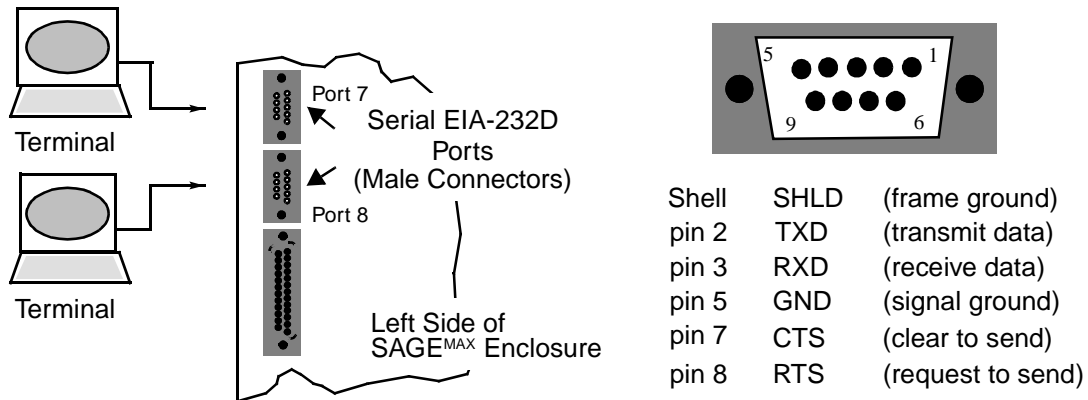


Figure 5-6 Local Terminal Ports (left), SAGE^{MAX} DB9 Pinouts for Local CRT Terminals (right)

5.8 Parallel Printer Connection

The SAGE^{MAX} has a standard DB25F D-subminiature parallel port for connecting a local printer. The DB25 female connector is located on the left side of the SAGE^{MAX} enclosure, below the DB9 local terminal ports, and is shown in **Figure 5-7**.

A standard parallel printer cable can be used to connect the SAGE^{MAX} to a parallel printer. The standard printer cable has a DB25 connector which attaches to the SAGE^{MAX} printer port and a Centronics connector which attaches to the parallel printer.

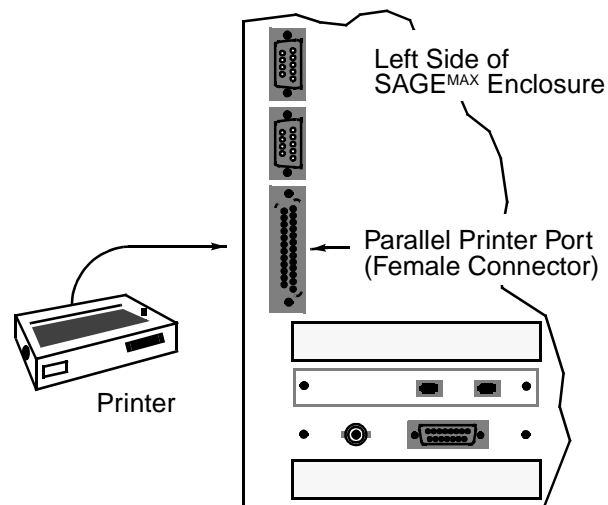


Figure 5-7 Local Parallel Printer Port on the SAGE^{MAX}

5.9 The Importance of a Telephone Connection

The SAGE^{MAX} has a 28.8K baud internal modem which is used for dial-in and dial-out applications. The modem has two RJ-11 jacks which can be accessed from the left side of the SAGE^{MAX} enclosure.

The two RJ-11 jacks are labeled **LINE** and **PHONE** and have two distinct functions. The **LINE** port is where the telephone line is to be connected for dialing operations. The **PHONE** port may be connected to a telephone if the operator needs to call out while on-site. Refer to **Figure 5-8**.

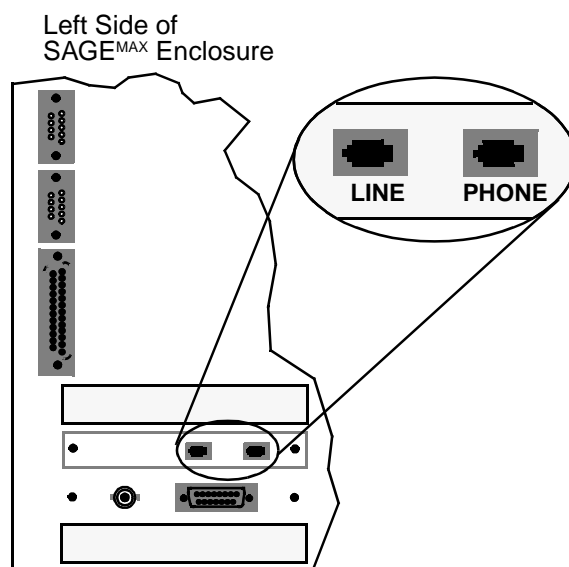


Figure 5-8 RJ-11 Modem Ports on the SAGE^{MAX}

IMPORTANT

The functions of the modem are not limited to alarm dial-out and terminal mode dial-in from a remote host. Troubleshooting your SAGE from the factory may be the only way to diagnose a problem that you are having. For this reason, a dedicated telephone line is vital. The importance of a telephone connection cannot be overemphasized!

NOTE

It is recommended that the telephone line be dedicated and not be part of a switchboard system.

In order to comply with FCC emissions regulations, it is necessary to install a ferrite core sleeve over two passes of the telephone line(s). This is shown in **Figure 5-9**.

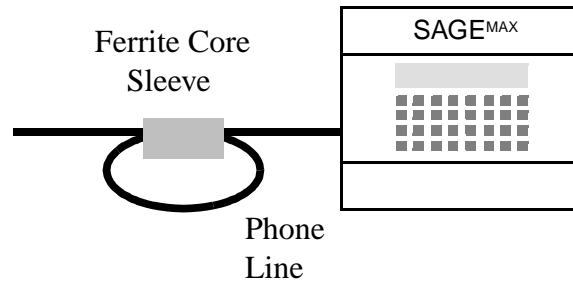


Figure 5-9 Ferrite Sleeve over the Telephone Line(s) of the SAGE^{MAX}

5.10 Optional Ethernet Connection

The SAGE^{MAX} can have an optional Ethernet card for connecting it to Ethernet networks. The Ethernet card is installed in one of the empty slots in the SAGE^{MAX}. Ethernet network connection is done through one of two ports on the Ethernet card.

A BNC connector is provided for coaxial cable (thin wire) Ethernet applications. A female DB15 connector is provided for thick wire Ethernet applications. These connectors are shown in **Figure 5-10**.

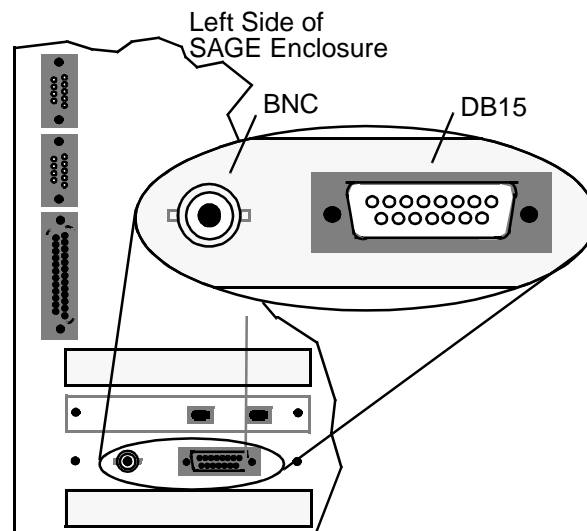


Figure 5-10 Optional Ethernet Ports on the SAGE^{MAX}

5.11 Optional EIA-232D Connection

The SAGE^{MAX} can have an optional EIA-232D card for EIA-232D applications. The card is installed in one of the empty slots in the SAGE^{MAX}. The EIA-232D network connection is done through a 25-pin male connector on the card. Refer to **Figure 5-11**.

The optional EIA-232D connection supports all the standard EIA-232D handshaking signals (unlike the local CRT ports which only use six signal lines) and can be used for leased line applications, an additional CRT port, or any other EIA-232D application.

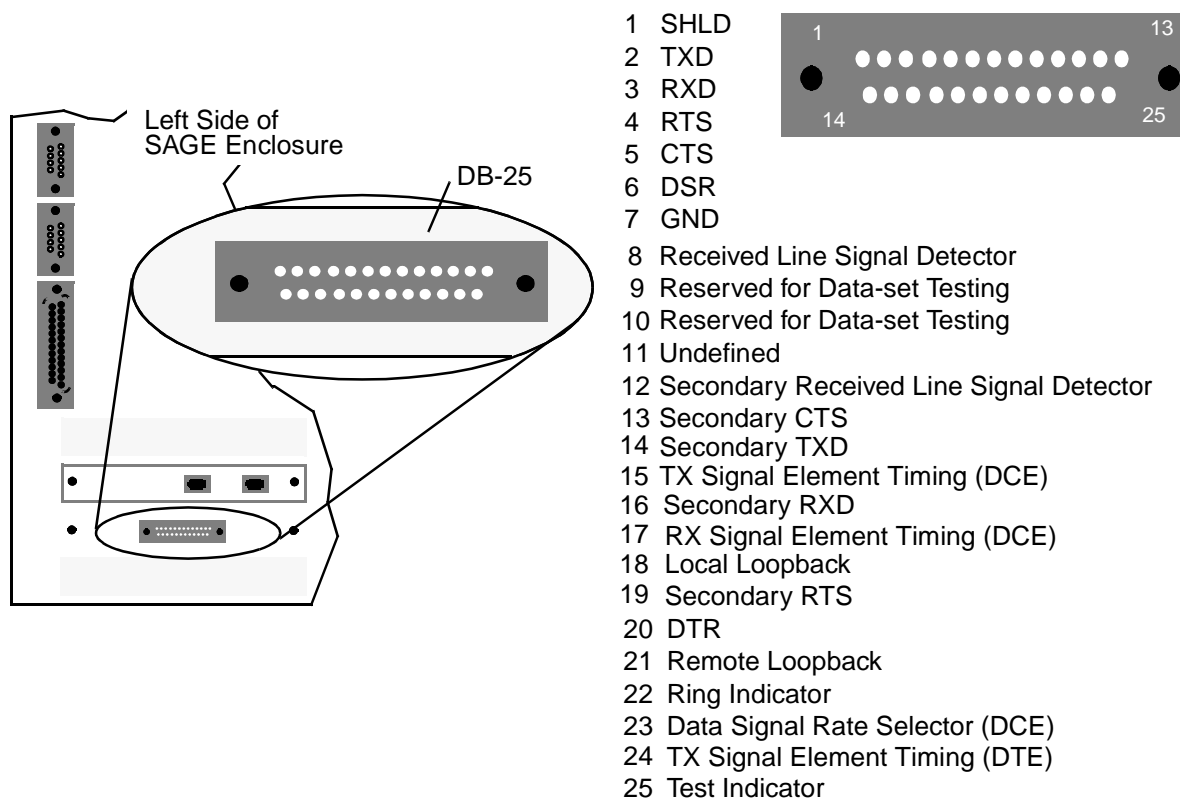


Figure 5-11 Optional EIA-232D Port (right), DB25 Pinouts (left)

5.12 Minimal Setup for Software Configuration

You can configure the SAGE^{MAX} software (i.e., setting the time and date, defining network protocols, setting system variables and creating points, trends, programs, schedules, dialing actions, etc.) before connecting all of your peripheral devices.

This minimal hardware configuration allows you to begin software setup using one of the local terminal ports before connecting networks, a local parallel printer, an optional Ethernet network or the second local terminal port.

The minimal setup for software configuration involves only two of the installation and hardware configuration steps listed in this section. You must supply power to the SAGE^{MAX} (see **Chapter 5.4: Power to the SAGE^{MAX}**) and connect a single local CRT terminal to port 7 (see **Chapter 5.7: Local Terminal Connections**). The local CRT terminal must be configured for 9600 baud, XON/XOFF Protocol, 8 data bits, 1 stop bit and no parity. Port 8 is disabled from the factory.

CHAPTER 6 - THE SAGE^{MAX} MENU SYSTEM

6.1 What Are Menus?

The SAGE^{MAX} offers a variety of operations that are available through its operator interface (OPI). These operations are listed in menus.

Menus consist of lists of options and unique, associated “keys” which are typed at the keyboard to perform each function in that menu. Typically, the keys of menu options are mnemonic codes that are easily associated. For example, **Q** for Quit (i.e., sign off the SAGE^{MAX}), **B** for Broadcast Message, etc. **Figure 6-1** shows the Main Menu of the SAGE^{MAX}.

The SAGE^{MAX} operator interface is based on a hierarchical menu structure. This scheme is designed to be simple to learn and use. It is also self-explanatory, once you understand certain basic principles.

The first menu that is presented after you sign on to the SAGE^{MAX} from an OPI is the Main Menu. Consider the Main Menu as the starting point of all menu activities.

From the Main Menu you have access to certain options as well as submenus. Some of these submenus have submenus of their own. This “nesting” effect of menus is the basic architecture of menu penetration interfaces.

The menu selections of the SAGE^{MAX} may vary, based on the privilege levels assigned to individual operators. An operator assigned many privileges has more menu options than an operator assigned very few privileges. For simplicity, further discussions of menus will assume that they are being accessed by operators with the maximum number of privileges except where noted.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                             Port: 07
Main:
key      to do
A       Alarm List
B       Broadcast Message
C       Calendar Edit
D       Database Functions
F       Find Objects
J       Job Scheduler
L       Table
M       Monitor/Change Point Attributes
P       Ports Status and Setup
Q       Quit
S       System Variables
T       Trend
U       Utility Functions
V       Virtual Terminal
PF1     Return to previous menu
Press key for desired action:
```

Figure 6-1 SAGE^{MAX} Main Menu

In some cases, the SAGE^{MAX} protects you from accidentally pressing the wrong key by prompting you to press several keys to select a given option. In general, if you press an incorrect key, the system will ignore it, and the terminal will beep briefly to signal that a mistake was made.

6.2 *Dumb versus VT100 Terminals*

There are two fundamental kinds of devices you can use to converse with the SAGE^{MAX} OPI. They are dumb terminals and VT100 terminals.

Dumb devices consist of interactive hard copy terminals such as Teletypes, TI700-type terminals, DEC LA34/LA120s, etc., or virtually any user terminal. In general, any device which can use the ASCII character set can be used as a dumb terminal.

SAGE^{MAX} provides extended support for user terminals that can emulate the functions of the Digital Equipment Corporation (DEC) VT100 family. This is a subset of the ANSI x3.64 specification.

There are many vendors of user terminals which conform to the VT100 standard. In the remainder of this document such terminals will be referred to generically as “VT-type” terminals. These include the terminals shown in **Table 6-1**. Table 6-1

<u>Manufacturer</u>	<u>User Terminal</u>
Digital Equipment Corp	VT100, VT102, VT220, VT131
Visual Computer Corp	Visual 100, 102
Wyse Technology	WY75
Liberty Electronic	Freedom 220, 222
Micro-Term	Ergo 301
Qume	QVT103

Table 6-1 VT-type Terminals Supported by the SAGE^{MAX}

6.3 *The No Scroll or Hold Screen Key and XON/XOFF Support*

An important key on VT terminals is the No Scroll or Hold Screen key. When pressed, this key sends a **CTRL-S** (stop or XOFF) character to the SAGE^{MAX} which suspends all further printout to the terminal screen. Pressing this key a second time causes a **CTRL-Q** (start or XON) character to be sent. This tells the SAGE^{MAX} to resume any suspended printout.

This feature can be used on dumb terminals as well by typing the **CTRL-S** and **CTRL-Q** characters directly. This feature is referred to as XON/XOFF support, and applies to printers as well as terminals.

6.4 *ESC and PF1 Keys*

The escape key on dumb terminal keyboards, sometimes marked **ESC**, performs the same function as the **PF1** key on VT-type terminal keyboards. Throughout this document, the **ESC** key is referred to as **PF1**.

PF1 is used as an escape signal. In general, at any point in the use of the OPI, the **PF1** key can be used to abort the operation that is in progress and return to the previous menu. In most cases, the use of the **PF1** key prevents any data from being altered by the operation that was in progress.

6.5 *PF4 and CTRL-Z Keys*

The **PF4** key on VT-type terminal keyboards has special uses in several instances. Its primary use is to signal the end of a “virtual terminal session.” For more information on virtual terminal mode, refer to **Chapter 8-17 : The Virtual Terminal Screen**.

If **PF4** is used in the Main Menu, it has the same effect as **Q** (Quit), i.e., sign off the SAGE^{MAX}. In any other menu, **PF4** is the same as **PF1**.

On dumb terminal keyboards, this function can be accomplished using the **CTRL-Z** character.

6.6 *The ENTER or RETURN Key*

When a menu option requires that you enter a line of text such as messages or point names, the **ENTER** or **RETURN** key is used to signal the SAGE^{MAX} that you are finished typing.

Once you type the **ENTER** or **RETURN** key, the SAGE^{MAX} accepts the information that you typed.

6.7 *The RUBOUT, DELETE and BACKSPACE Keys*

When a menu option requires that you enter a line of text such as messages or point names, the **RUBOUT**, **DELETE** or **BACKSPACE** keys may be used to erase erroneous characters.

Correcting mistakes can be done by using the **RUBOUT**, **DELETE** or **BACKSPACE** key *before* you type the **RETURN** or **ENTER** key.

6.8 *The Cursor Keys (→ ← ↑ ↓)*

The cursor keys (right, left, up and down arrow keys) on the VT-type keyboard perform intuitive functions.

The cursor keys are used in the SAGE^{MAX} text editor to move the cursor to different positions on the screen.

Cursor keys may also be used in various menus to move to next (▼) and previous (▲) entries in a list.

In some cases, multiple parameters within the SAGE^{MAX} may be entered all at once on the same line. In these cases, a *prototype* of the parameters is printed on one or more lines and the cursor is positioned on a line below the prototype. The left arrow (←) and right arrow (→) keys can be used to navigate around the prototype line. The **ENTER** or **RETURN** key is used to “accept” the contents of the prototype line. In most cases, the prototype line is initialized with some default characters which may be “typed over” or left alone. On dumb terminals, you may use the **TAB** key as right arrow (→) and the **RUBOUT** or **DELETE** key as left arrow (←).

Figure 6-2 shows an example of a SAGE^{MAX} prototype line. Note the position of the cursor, which is shown in reverse video, in position one of the prototype line. In this example (taken from the Alarm List Submenu), the prototype line is initialized to question marks (?). These default characters may be “typed over” or left alone.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
*text* = Find "text"  -- OR --
Trans=Transaction #, Unit=Device Unit, Cls=Class Name, Message=Alarm Msg Text
A Trans Unit- Cls Message-----
  ????? ????? ?? ?????????????????????????????????????????????????????????????

```

Figure 6-2 Example Prototype Line

On dumb terminals, you can still get the same effects of special keys like **PF1** and cursor keys (→ ← ↑ ↓) by using the associations shown in Table 6-2:

<u>VT Key</u>	<u>Dumb Terminal Equivalent</u>
PF1	ESC
PF2	CTRL-\
PF3	CTRL-]
PF4	CTRL-Z
(up arrow)	CTRL-K
(down arrow)	CTRL-J (linefeed)
(right arrow)	CTRL- <u>_</u>
(left arrow)	CTRL-H (Backspace)

Table 6-2 Dumb Terminal Equivalents for VT-type Keys

6.9 Special VT220 Keys

VT220 terminal keyboards have keys that can be used to perform additional special functions in the SAGE^{MAX}.

The **FIND**, **INSERT**, **REMOVE**, **SELECT**, **PREVIOUS SCREEN**, **NEXT SCREEN** and **HELP** keys are used in the SAGE^{MAX} text editor. These function keys are helpful if you are editing a text file and you are familiar with the VT220 keyboard.

6.10 Help Screens: The “?” Key

Help screens are available in some of the menus and submenus of the SAGE^{MAX}. To access these help screens, you must use the ? (question mark) key.

When the ? key is pressed, a help screen is displayed and explains available options and how to proceed.

Figure 6-3 shows the help screen of the Monitor/Change Point Attributes menu selection.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-9612:00:00
Opr: SYSTEM                                           Port: 07
...Press ? for Help
...Press PF1 to return to previous menu

TY\Name to Monitor:?

          - - - - - Help for Monitor - - - - -
TY\Name          for Type and Object Name (GR\Name or \Name for groups)
                  Valid Types: PT, VR, PG, GR, GL
N or +          Next Attribute
P or -          Previous Attribute
(space)         Default Attribute
;              Any Attribute
A or *          All Attributes
$              $$ Attributes for Programs
%              %A Register for Programs
(return)       Update Display
C or &         Continuous on/off
...Press ? for Help
...Press PF1 to return to previous menu

TY\Name to Monitor:

```

Figure 6-3 Help Screen from the Monitor/Change Point Attributes Prompt

6.11 Expert Mode: The “!” Key

The SAGE^{MAX} Menu System has a feature called expert mode. This feature is enabled and disabled using the ! (exclamation point) key.

When a SAGE^{MAX} OPI is in expert mode, the menu name is displayed but menu options are not. For example, the Main Menu in expert mode simply shows the word **Main:** on the terminal. The options of the Main Menu are not displayed. Full menus can be temporarily displayed using the ? (question mark) key.

NOTE

Expert mode should only be used by experienced SAGE^{MAX} users.

6.12 *Switching Languages: Using CTRL-L*

The SAGE^{MAX} OPI is designed to be usable in multiple languages. If you have multiple languages installed, you can switch among them by entering **CTRL-L** at any *menu* prompt. After entering **CTRL-L**, the SAGE^{MAX} prompts you with **New Language (xxx):**. From this prompt you must enter one, two or three decimal digits corresponding to the international language code for the language you want, followed by **ENTER** or **RETURN**.

Your SAGE^{MAX} may not have all of the languages listed below, but **Table 6-3** lists some of the international codes and their associated countries.

<u>Code</u>	<u>Country</u>
000	United States
001	United States
031	Netherlands
032	Belgium
033	France
034	Spain
039	Italy
041	Switzerland
042	Czechoslovakia
044	United Kingdom
045	Denmark
046	Sweden
047	Norway
049	Germany
061	Australia
081	Japan
086	China
102	Finland
204	Israel

Table 6-3 International Language Codes

6.13 *Displaying Groups*

The SAGE^{MAX} Menu System provides for the creation and dynamic display of *Groups*. Groups are collections of database objects that appear together in a menu and are discussed in detail in the next chapter.

When you monitor a group, you can use the - (minus) key to toggle how the objects in the group are displayed. The - (minus) key toggles the display between the object names of the group elements and the associated description text of the group elements.

6.14 *Paging*

There are certain submenus of the SAGE^{MAX} Menu System that display information in a list (e.g., Alarm List, Trend Display, List Job Requests, etc.). When there is more information requested than can fit on the screen, the SAGE^{MAX} Menu System will prompt you to press **N** (or **+**) to display the next page. Similarly, The SAGE^{MAX} will prompt you to press **P** (or **-**) to display the previous page.

Figure 6-4 shows a sample screen from the Unacknowledged Alarm List. Notice the prompt at the bottom of the screen which explains how to view the next page, view the previous page or escape to the previous menu.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
*text* = Find "text"  -- OR --
Trans=Transaction #, Unit=Device Unit, Cls=Class Name, Message=Alarm Msg Text
A  Trans Unit- Cls Message-----
  ????? ????? ??? ?????????????????????????????????????????????????????????????
0  02130/00000 003 Mon 23-Sep-96 11:21:02
1  -          003 PROGRAM7 halted at 0123:0021 due to #InvalidPcode
2  02121/00016 STR Mon 23-Sep-96 11:22:15
3  -          STR RM11_AI7          High Limit Alarm
4  -          STR Building 17 Room 11 AI#7 - Room Humidity Sensor
5  -          STR *** Humidity Beyond Upper Limit ***
6  02113/00000 SCN Mon 23-Sep-96 11:23:13
7  -          SCN Port 3, Unit 4111   Online Scan Alarm
8  02093/00000 SCN Mon 23-Sep-96 11:23:14
9  -          SCN Port 3, Unit 591    Online Scan Alarm

Press Key: N(+) next page, P(-) previous page, PF1 (Esc) escape
Press Key: 0-9 acknowledge this alarm, A acknowledge all alarms

```

Figure 6-4 Page One of a Sample Alarm List Screen

6.15 The Type Ahead Feature

The SAGE^{MAX} Menu System offers a type ahead feature that allows any OPI to accept valid keystrokes the moment they are typed, rather than printing the remainder of a menu.

This feature is very handy for users who are familiar with the Menu System of the SAGE^{MAX} and are interfacing through a telephone modem OPI at slower speeds.

6.16 The Video Terminal Display Screen

The display screen of a SAGE^{MAX} OPI appears differently for dumb terminals than for VT-type terminals because of the differences in features. Whenever a new menu is displayed on a VT-type terminal, the screen is erased and a new menu appears at the top of the screen. On dumb terminals, the same screen is printed, but it simply scrolls upward rather than beginning at the top of the screen.

VT-type screens are special in that the top two lines of the screen are dynamic, and display changing information about the time and date. Dumb terminals only update this information once every time the Main Menu is displayed. These two lines are displayed in highlighted (reverse video) characters on the screen of VT-type terminals. For both dumb and VT-type terminals, these two lines also contain the software version number of the SAGE^{MAX}, banner line text, the port number of the OPI, the user name of the operator signed on to that port and an **ALARMS** field that appears when there are unacknowledged alarms. Refer to **Figure 6-5**. The remaining lines on the screen (22 lines on the VT terminal) are used for menu display and general interaction with operators. On VT-type screens, these bottom 22 lines are set up as a scrolling region.

These 22 lines of the VT screen are dynamically divided into a static upper portion and a scrolling lower portion. The scrolling portion “rolls upward” underneath the static upper part under certain circumstances, such as when alarm messages are displayed on the VT screen.

Typically, when a menu is displayed, the portion of the screen below the menu is a scrolling region.

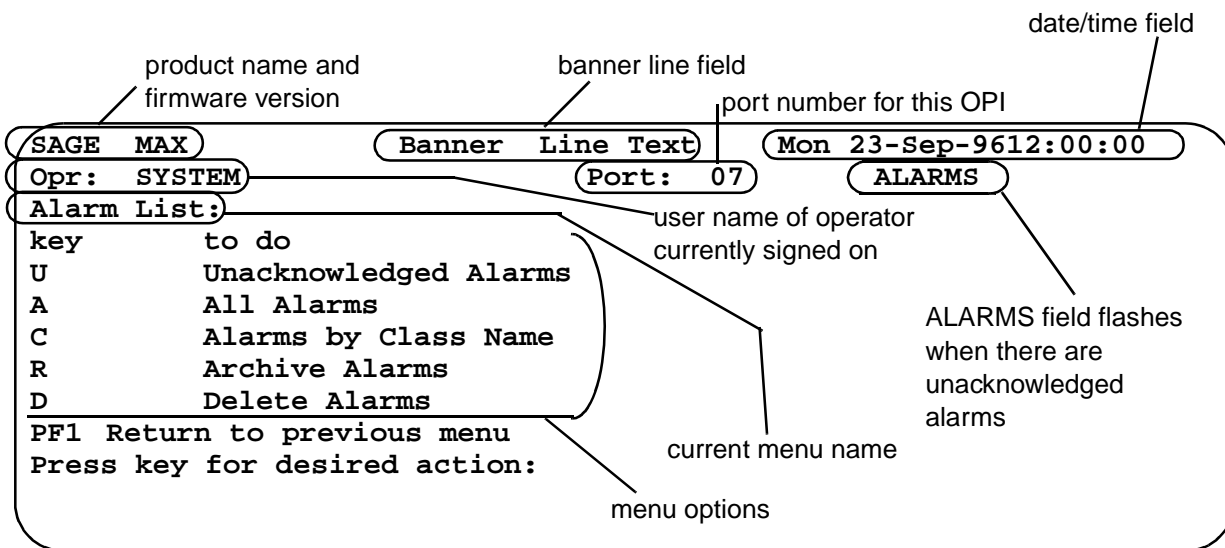


Figure 6-5 Components of the Video Terminal Display Screen

CHAPTER 7 - INITIAL CONFIGURATION

This chapter explains how to initially configure the software portion of the SAGE^{MAX}, including port setup, configuration of drivers for each port and how to set up a database.

To configure SAGE^{MAX} software, you must access the SAGE^{MAX} menus from a local OPI. In order to access SAGE^{MAX} menus, you must at least have a minimal hardware setup. This minimal configuration is explained in **Chapter 5-12: Minimal Setup for Software Configuration**.

7.1 Signing On to the SAGE^{MAX}

Before you can begin to configure the SAGE^{MAX} software, you must sign on to the SAGE^{MAX}. You sign on to the SAGE^{MAX} using a username and a corresponding codeword from the Sign-on Screen of a local OPI. The SAGE^{MAX} Sign-on Screen is shown in **Figure 7-1**.

Although usernames and codewords can be added and changed later, you must use the username **SYSTEM** and the codeword **AAM** when you initially sign on to the SAGE^{MAX}. This default username has the maximum number of privileges and allows access to all SAGE^{MAX} menus and features necessary to initially configure the software of the SAGE^{MAX}. This name should be deleted after system configuration.

From the **Enter User Identification:** prompt you enter the default username **SYSTEM**. The username is echoed (displayed on the screen as you type it). After you enter the username, you are prompted to **Enter Codeword:**. Enter the default codeword **AAM**. Unlike the username, the codeword is not shown on the screen for security reasons.

If you enter the username and codeword correctly, you are prompted with the Main Menu shown in **Figure 7-2**. You can begin to configure the ports, drivers, and database of your SAGE^{MAX} from this point.

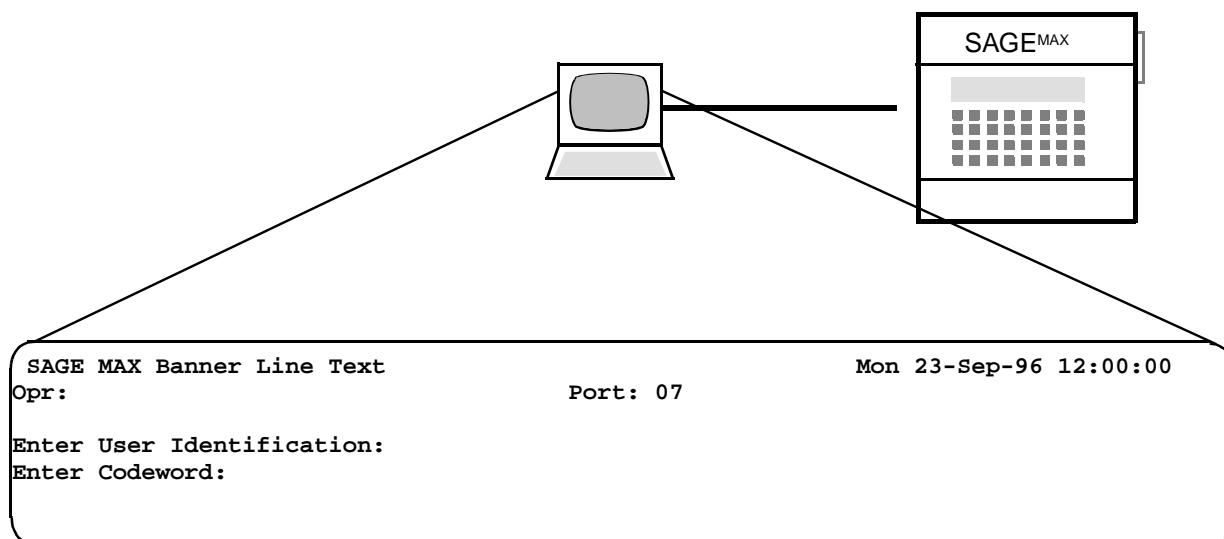


Figure 7-1 The SAGE^{MAX} Sign-On Screen

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Main:
key           to do
A             Alarm List
B             Broadcast Message
C             Calendar Edit
D             Database Functions
F             Find Objects
J             Job Scheduler
L             Table
M             Monitor/Change Point Attributes
P             Ports Status and Setup
Q             Quit
S             System Variables
T             Trend
U             Utility Functions
V             Virtual Terminal
PF1          Return to previous menu
Press key for desired action:
```

Figure 7-2 Main Menu

7.2 Ports of the SAGE^{MAX}

The SAGE^{MAX} can support up to 12 different software-configurable ports. These ports are used as network ports, OPIs and virtual terminal ports. Any of these ports may be configured to perform with any of the SAGE^{MAX} port driver personalities, although some of these combinations are not really useful.

- Ports 1-4 are dedicated dual-trunk EIA-485 network ports.
- Port 5 is dedicated for use with the 28.8K baud internal modem for dial-up applications. Port 6 is dedicated for use with an optional EIA-232D card for leased-line applications. Alternately, port 6 can be configured with a second internal modem (COM2). This configuration could be used if you wanted to make port 5 a dedicated alarm dial-out port, and make port 6 a dedicated host dial-up port.
- Ports 7 and 8 are local CRT or serial printer ports. These ports are typically the local OPI ports of the SAGE^{MAX}.
- Ports 9-12 are used as virtual terminal ports. These ports do not represent real physical connections to hardware devices. Instead, *virtual* ports are used when a dial-up or direct connect host enters terminal mode with the SAGE^{MAX} through another network.

You configure and monitor the status of these 12 ports of the SAGE^{MAX} from the Ports Status and Setup Menu. This menu is reached from the Ports Status and Setup (**P**) option of the Main Menu and is shown in **Figure 7-3**.

Port configuration information is edited through the menu selections provided by the Ports Status and Setup Submenu. This submenu also allows you to edit the Ethernet configuration information, watch the dynamic port status and save configuration changes in the configuration file **C:\CFG\PORTS.CFG**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07

Port Setup:
key          to do
T           Change Port Type
C           Change Port Communications Parameters
D           Change Port Driver Configuration
W           Watch Dynamic Port Status
S           Save Changed in Configuration File
E           Change Ethernet Configuration
PF1        Return to previous menu
Press key for desired action:

```

Figure 7-3 The Ports Status and Setup Menu

In addition to the 12 available configurable ports, there are three additional ports that have dedicated functions. Port 0 is the optional front panel display. Port 13 is a parallel printer port (LPT1). Port 14 is the optional Ethernet High Speed LAN interface. Refer to **Figure 7-4**.

PORT	CONFIGURATION	PORT	CONFIGURATION
0	Front Panel Display/Keypad	8	CRT Terminal, Serial Printer, or PHP Host
1	EIA-485 Network	9	Virtual Terminal
2	EIA-485 Network	10	Virtual Terminal
3	EIA-485 Network	11	Virtual Terminal
4	EIA-485 Network	12	Virtual Terminal
5	COM1 Modem to Phone Lines	13	Parallel Printer
6	COM2 Modem/EIA-232D Network (optional)	14	Ethernet
7	CRT Terminal, Serial Printer, or PHP Host		

Figure 7-4 Typical Hardware Configurations of SAGE^{MAX} Ports

7.3 Setting Up Ports

To set up the ports of the SAGE^{MAX}, you must first determine how you want the ports to be configured. The available driver types and their respective code numbers for the SAGE^{MAX} are displayed when you select the Change Port Type (T) option of the Ports Status and Setup Submenu and choose a port to modify. This screen is shown in **Figure 7-5**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07

Port --Type--      Lan  Baud- P   D   S
01= None           001   9600 N   8   1
02= None           001   9600 N   8   1
03= None           001   9600 N   8   1
04= None           001   9600 N   8   1
05= None           001  38400 N   8   1
06= None           001   9600 N   8   1
07= Dumb           000   9600 N   5   1
08= VT100          000   9600 N   5   1
09= VT100          000   9600 N   5   1
10= VT100          000   9600 N   5   1
11= VT100          000   9600 N   5   1
12= VT100          000   9600 N   5   1
Change Port (1-12): 7

Code Port Type
00      None
01      PUPhost
02      Peernet
03      XANP Host
04      Dumb
05      VT100
06      Printer
07      PHPdcon
08      PHPdial
10      PHPHdcon
11      PHPHdial
New Port Type: 5
Are you sure ? (Y/N): Y

```

Figure 7-5 Example of the Change Port Type Screen

When you select a port to change its type, SAGE^{MAX} presents you with a list of available driver types which have been loaded with the system. It is possible to add or delete drivers from this list, so the list you see may be different from the example.

The example shown in **Figure 7-5** shows how to change the driver type of port 7 from Dumb (code 04) to VT100 (code 05).

- The **(00) None** option implies that no driver type has been assigned to that particular port.
- The **(01) PUPhost** driver type is used by ports that have PUP network configurations.
- The **(02) Peernet** driver type is used by ports that are connected to a STAR Peernet.
- The **(03) XANP** driver type is used when the SAGE^{MAX} serves as a host for networks of XANP devices (STAR, RCU, RCU2, MCU).

- The **(04) Dumb** and the **(05) VT100** driver types are used to configure OPI ports for dumb and VT100 terminals respectively.
- The **(06) Printer** driver type is used to configure OPI ports for serial printer support.
- The **(07) PHPdcon** driver type is used to configure network ports for direct PHP connections. In these cases, the SAGE^{MAX} acts as a slave on the PHP network.
- The **(08) PHPdial** driver type is used to configure the modem port for dial-up PHP applications. In this case, the SAGE^{MAX} acts as a PHP slave.
- The **(09) PHPHdcon** driver type is used to configure network ports for direct PHP connections. In these cases, the SAGE^{MAX} acts as the host on the PHP network.
- The **(10) PHPHdial** driver type is used to configure the modem port for dial-up PHP applications. In this case, the SAGE^{MAX} acts as the PHP host.

Figure 7-5 shows the ports of the SAGE^{MAX} along with typical hardware configurations for each port.

7.4 *Setting Up Communications Parameters*

The baud rate, parity, number of data bits, number of stop bits and the language used by each port is set up by using the Change Port Communications Parameters (**C**) option of the Ports Status and Setup Submenu. This screen is shown in **Figure 7-6**.

Baud rate refers to the speed (measured in bits per second or bps) at which serial data is transmitted or received. The serial ports on the SAGE can be configured to 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19,200 and 38,400 bps. The modem port (port 5) defaults to 38,400 bps, while all other ports default to 9600 bps.

Although serial port baud rates range from 110-38,400 bps on the SAGE^{MAX}, the allowable baud rates for different port types vary, based on the kinds of devices that are connected to the ports.

For information about the available communication speeds of a serial device, consult the user manual of that particular device.

Parity refers to an error detection technique that counts the number of bits in each serial data byte and appends a ninth parity bit to the data byte. For *even parity* systems, the parity bit is set to 1 when there is an even number of “1” bits in the data byte, and 0 when there is an odd number of “1” bits in the data byte. For *odd parity* systems, the parity bit is set to 1 when there is an odd number of “1” bits in the data byte, and 0 when there is an even number of “1” bits in the data byte. A *no parity* system does not calculate a parity bit and does not use an extra ninth bit. None of the standard SAGE^{MAX} drivers uses the parity bit. The parity option is made available in the event that it is used by a terminal, printer or other peripheral SAGE^{MAX} device.

Although the number of *data bits* is typically 8 for most port types, the data bits option is made available in the event that a different number of data bits is used by a terminal, printer or other peripheral SAGE^{MAX} device. The SAGE^{MAX} can be configured to support serial communications using 5,6,7 and 8 data bits.

In typical serial communications, a *stop bit* is appended to each byte that is transmitted. Some devices use 2 stop bits after each transmitted character. SAGE^{MAX} supports both 1 *stop bit* and 2 *stop bit* configurations.

The default language for each port is selectable from 0-255 as defined in your SAGE^{MAX}.

In the example screen shown in **Figure 7-6**, the communications parameters of port 8 are changed to language 001, 4800 bps, no parity, 8 data bits and 1 stop bit. After you enter the new communications parameters, you are asked to confirm your selections with the **Are You Sure ? (Y/N)**: prompt.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07

Port --Type--      Lan  Baud- P   D   S
01= None           001   9600 N   8   1
02= None           001   9600 N   8   1
03= None           001   9600 N   8   1
04= None           001   9600 N   8   1
05= None           001  38400 N   8   1
06= None           001   9600 N   8   1
07= Dumb           000   9600 N   5   1
08= VT100          000   9600 N   5   1
09= VT100          000   9600 N   5   1
10= VT100          000   9600 N   5   1
11= VT100          000   9600 N   5   1
12= VT100          000   9600 N   5   1
Change Port (1-12): 8
Baud=110,150,300,600,1200,2400,4800,9600,19200,38400
Language=0 to 255 Parity=N, O, E DadaBits=5,6,7 or 8 Stop Bits=1 or2
Language,Baud,Parity,DataBits,StopBits: 001,4800,N,8,1

```

Figure 7-6 Example of the Change Port Communications Parameters Screen

7.5 Configuring Drivers

After you define drivers and communications parameters for serial ports on the SAGE^{MAX}, you may need to configure driver information. All of the driver types have driver variables that may need to be changed.

Driver configuration variables are accessed through the Change Port Driver Configuration (**D**) option of the Ports Status and Setup Submenu, and vary based on the associated driver type.

The PUPhost, Peernet, XANP, PHPdcon, PHPdial (for PHP slave applications), PHPHdcon and PHPHdial (for PHP host applications) driver types have variables as shown in **Tables 7-1** through **7-4**.

When editing driver variables, **RETURN** must be pressed for the line to take effect. You must then press **PF1** to return to the configuration menu. Be sure to save your changes (**S**) after you make changes to the driver variables. Refer to **Chapters 14-19** for detailed explanations of features and variables of driver types.

VARIABLE	DESCRIPTION (Default)
PUP Unit Number for this SAGE^{MAX}	The PUP network address for this SAGE ^{MAX} from 0-32767 (default=100)
Max Transaction before Token Pass	The maximum number of PUP network transactions that can occur before the SAGE ^{MAX} passes the token to the next PUP device on the network. SAGE ^{MAX} may voluntarily relinquish the token before this number of transactions is reached. (default=10)
Request Interleave	The maximum number of PUP network requests that can occur when the SAGE ^{MAX} gets the token (default=9)
Alarm Poll Interleave	The maximum number of times the SAGE ^{MAX} polls for alarms on the PUP network when the SAGE ^{MAX} gets the token (default=1)
Max Tries per Transaction	The maximum number of times a transaction is transmitted across the PUP network before it is considered a failed transaction (default=3)
Broadcast Time Synch?	If YES, enables SAGE ^{MAX} to broadcast current time at power up and every five minutes thereafter (default=NO)
Unit # / Peer / Alarm Poll 0-127	Unit P A (Peer, Alarmpoll) The unit number of another PUP device on the network. This variable also specifies if this unit is a peer (i.e., gets passed to token) and if this unit should be polled for alarms by entering P and/or A in the appropriate fields. (default is blank, i.e., no units, no peers, no alarm polling) Use the up and down arrow keys to scroll through all 128 units. Use the left and right arrow keys to maneuver through the field while editing. Press RETURN for new values to take effect. Press PF1 to return to the configuration menu.

Table 7-1 PUPhost Driver Type Variables

VARIABLE	DESCRIPTION (Default)
Peer Unit Number for this SAGEMAX	The Peer unit number for this SAGEMAX from 0-31 (default=0)
Max Transaction before Token Pass	The maximum number of Peernet transactions that can occur before the SAGEMAX passes the token to the next device on the Peernet. SAGEMAX may voluntarily relinquish the token before this number of transactions is reached. (default=5)
Request Interleave	The maximum number of Peernet requests that can occur when the SAGEMAX has the token (default=9)
File Session Interleave	The maximum number of file transfer session transactions which are interleaved with requests (default=1)
Max Tries per Transaction	The maximum number of times a transaction is transmitted across the Peernet before it is considered a failed transaction (default=3)
Broadcast Time Synch?	If YES, enables SAGEMAX to broadcast current time every hour, when first powered up and if requested by a Peer unit (default=NO)
Unit # / Peer / Alarm Poll 0-127	<pre> 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 - - - - -</pre> <p>Identifies which units (0-31) are on the Peernet (default is blank, i.e. no Peer units)</p> <p>Use the left and right arrow keys to position the cursor below the appropriate unit number and type * (an asterisk) to identify units on the Peernet. Press RETURN for new values to take effect. Press PF1 to return to the configuration menu.</p>

Table 7-2 PUPhost Driver Type Variables

VARIABLE	DESCRIPTION (Default)
Leased-line Modem Control	If YES, configures port for use with leased-line modems (default=NO)
Seek Factor	The number of transactions before retrying unresponsive units (default=100)
Request Interleave	The maximum number of XANP requests that can occur when the SAGE ^{MAX} has the token (default=9)
Alarm Poll Interleave	The maximum number of times the SAGE ^{MAX} polls for alarms on the XANP network when the SAGE ^{MAX} gets the token (default=1)
File Session Interleave	The maximum number of file transfer session transactions which are interleaved with requests (default=1)
Max Tries per Transaction	The maximum number of times a transaction is transmitted across the Peernet before it is considered a failed transaction (default=3)
Block Tries to Unresponsive Units	If YES, access to unresponsive units will be blocked. If NO, you may be able to access a previously unresponsive unit if it had come back on line, but has not yet been polled (default=YES).
Normal Response Timeout	The amount of time (in 25 ms increments) the SAGE ^{MAX} waits for a response from another unit on the XANP network before assuming the transaction failed. The SAGE ^{MAX} multiplies the number you enter by 25 for the actual response timeout in milliseconds (default=12).
Peer Units	<pre> 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 - - - - -</pre> <p>Identifies which units (0-31) are on the XANP network (default is blank, i.e. no units)</p> <p>Use the left and right arrow keys to position the cursor below the appropriate unit number and type * (an asterisk) to identify units on the Peernet. An "A" can be entered instead of an asterisk for automatic download. Press RETURN for new values to take effect. Press PF1 to return to the configuration menu.</p>

Table 7-3 XANP Driver Type Variables

VARIABLE	DESCRIPTION (Default)
PHP Unit Number	PHP unit number for this SAGE ^{MAX} from 48-57 or 62-255 (default=48)
Turnaround Delay	The delay imposed by the SAGE ^{MAX} before it responds to a PHP command. The SAGE ^{MAX} multiplies the number you enter by 25 ms (default=0).
Terminal Mode Delay	The delay imposed by the SAGE ^{MAX} before it responds to a PHP terminal mode request. The SAGE ^{MAX} multiplies the number you enter by 25 ms (default=2).
Auto Hangup Delay	For dial-up connections, this is the amount of time the SAGE ^{MAX} will wait between keystrokes before hanging up and breaking the dial-up connection (default=70).
Extended Site ID	An extended ID number (0-65535) for PHP dial-up systems (default=0).
Redial Interval	For dial-up connections, the amount of time the SAGE ^{MAX} waits between dial-out attempts (default=60 seconds)
Dial Out Tries Before Quiet Time	For dial-up connections, the number of times the SAGE ^{MAX} will dial out alarms before initiating a "quiet time", during which no dial-outs occur, so that remote sites have an opportunity to dial in to the SAGE ^{MAX} (default=5).
Object Modify Privileges	An object modification bitmap used by the PHP slave drivers (default=0000000B)
PHP Site Banner Line Text	For dial-up connections, 50-character ASCII text string displayed as banner line text when host connects via virtual terminal mode to this unit (default is blank).
PHP Upload/Download Path	SAGE ^{MAX} pathname where uploaded/downloaded PHP files are stored (default is blank).
PHP Reboot Password	The password which is required for the PHP reboot command (default is SYSTEM)
Default Operator	24-character case-insensitive operator name used in alarm messages when log modifications is enabled (default is a blank)

Table 7-4 PHPdcon and PHPdial Driver Type Variables

VARIABLE	DESCRIPTION (Default)
PHP Unit Number	PHP unit number for this SAGE ^{MAX} from 48-57 or 62-255 (default=48)
Extended Site ID	An extended ID number (0-65535) for PHP dial-up systems (default=0)
Terminal Mode Delay	The delay imposed by the SAGE ^{MAX} before it responds to a PHP terminal mode request. The SAGE ^{MAX} multiplies the number you enter by 25 ms (default=2).
Auto Hangup Delay	For dial-up connections, this is the amount of time the SAGE ^{MAX} will wait between keystrokes before hanging up and breaking the dial-up connection (default=70).
Response Timeout	An intercharacter timeout that is reset every time a valid PHP character is received (default=5 seconds).
Object Modify Privileges	An object modification bitmap used by the PHP slave drivers (default=00000000B)
Default Alarm Class	If nonzero, this PHP alarm class override is used by the SAGE ^{MAX} to process incoming alarms and override their class. If zero, the incoming alarm uses its own alarm class (default=0).
PHP Upload/Download Path	SAGE ^{MAX} pathname where uploaded/downloaded PHP files are stored (default is blank).
PHP Reboot Password	The password which is required for the PHP reboot command (default is SYSTEM)
Default Operator	24-character case-insensitive operator name used in alarm messages when log modifications is enabled (default is blank)
Unit # / Peer / Alarm Poll 0-63	Unit P A (Peer, Alarmpoll) The unit numbers of other PUP devices on the network. If this unit should be polled for alarms, enter A in the appropriate fields. Use the left and right arrow keys to maneuver through the field while editing. Press RETURN for new values to take effect. Press PF1 to return to the configuration menu (default is blank, i.e. no units polled).

Table 7-5 PHPHdcon and PHPHdial Driver Type Variables

7.6 *Planning a Database*

Now that the ports and drivers of the SAGE^{MAX} are configured, you can begin to create database objects so you can communicate over the networks you have configured.

The SAGE^{MAX} database consists of seven types of named objects, as shown below. The creation of database objects requires some careful consideration and planning.

- *Users* are database objects that define sign-on identification names, codewords, timeouts and privileges for security purposes.
- *Points* are database objects that reference hardware or software objects such as sensors, setpoints and output values.
- *Globals* are named objects which represent names in another SAGE^{MAX} database.
- *Programs* are software database objects that consist of user-definable logic statements that are used to develop specialized applications.
- *Variables* are database objects that are used in the same way as points, but have a definable data type and a single, definable value.
- *Groups* are collections of up to 16 references to database objects that appear together as a menu.
- *Classes* are database objects that are used in defining alarm information such as where to log alarms, acknowledgment information, alarm priority and dialing information.

The following sections explain the procedures for planning the individual elements of a database.

7.6.1 *Planning Users*

Users are database objects that define sign-on identification names, codewords, timeouts and privileges for security purposes. Planning users for your system requires that you understand the following:

- who will be using the system
- what access privileges should be made available to those users

SAGE^{MAX} user privileges are divided into nine functional groups. These groups (and their respective privilege codes) are:

Everyone 0000 0000	File Access 0000 0100	Object Modifier 0010 0000
Operator 0000 0001	File Expert 0000 1000	Scheduler 0100 0000
Administrator 0000 0010	User Administrator 0001 0000	Installer 1000 0000

The first three groups (Everyone, Operator and Administrator) are *base set* Menu Privileges. Each of these groups defines a basic collection of menu options and features that are available to that group. The remaining six groups (Object Modifier, Scheduler, Installer, File Access, File Expert and User Administrator) are *optional features* that provide specialized privileges which can be added to any of the base set privileges.

Using a menu access scheme of user privileges (rather than simply a *privilege level* scheme) permits access to menu functions based on the user's job. Typical users include, but are not limited to the following:

- an installer
- a casual operator
- a file expert operator
- a file expert, user administrator
- an operator/user administrator with scheduling and modify capabilities
- a basic user (everyone)

Before you can plan users, you must have an understanding of what is offered through the Menu Privilege groups.

The *Everyone* base set privilege allows very limited access to the SAGE^{MAX} Menu System. The only menu options available are:

- Alarm List (All Alarms, Unacknowledged Alarms and Alarms by Class Name only)
- Broadcast Message (except on the Ethernet port)
- Quit

The *Operator* base set privilege allows access to menu options that are considered typical *operator* functions. Operators have access to the following menus:

- Alarm List (All Alarms, Unacknowledged Alarms, Alarms by Class Name and Alarm Acknowledgment only)
- Broadcast Message (on all ports)
- Database Functions (List Objects only)
- Find Objects (Find Objects only)
- Monitor/Change Point Attributes (Monitor only)
- Table Menu (List Objects only)
- Ports Status and Setup (Watch Dynamic Port Status and Watch Driver Status only)
- Trend (List Trend Objects, Numerical Display and Stripchart Display only)
- Virtual Terminal
- Quit

The *Administrator* base set privilege allows access to menu options that are considered typical *administrator* functions. Administrators have access to the following menus:

- Alarm List (All Alarms, Unacknowledged Alarms, Alarms by Class Name, Archive Alarms and Delete Alarms only)
- Calendar Edit
- Database Functions (all functions except Create User, Erase User and List Users)
- Find Objects (all functions except Find and Stuff)
- Monitor/Change Point Attributes (Monitor only)
- Job Scheduler (Import/Export Database Files and STAR/SAC to SAGE^{MAX} Translations)
- System Variables
- Trend (all functions)
- Utility Functions (Directory, Make Directory, Delete Directory, Display File as Hex/ASCII, Format Diskette, Prepare System for Maintenance, Type File, Search for File, Math Calculator, Edit File, Copy File, Delete File and Remote Copy of certain files)
- Table Menu (List, List to File, and Edit)
- Quit

NOTE

Some of the available functions of the three base set Menu Privileges overlap.

Every user of the SAGE^{MAX} is assigned a base set privilege. In addition, it is possible to assign both Operator and Administrator base set privileges to the same user. This gives the user access to both operator and administrator functions.

There are six additional *special privileges* that can be assigned to any base privilege user. These privileges can be added in any combination to supply additional privileges to the user.

The *Scheduler* privilege adds *all* Job Scheduler functions to the user's group of privileges.

The *Installer* privilege adds *all* Ports Status and Setup functions to the user's group of privileges.

The *File Access* privilege adds the following list of Utility Functions to the user's group of privileges:

- Directory
- Edit File
- Copy File
- Delete File
- Remote Copy

For users with the File Access privilege, the features listed above only apply to the following directories on the SAGE:

- **D:** (all files)
- **C:\CFG** (all **.BOB** files except **USER.BOB**)
- **C:\EXE** (all files)
- **C:\GROUPS** (all files)
- **C:\INIT** (all files)
- **C:\LOGIC** (all files)
- **C:\LOG** (all files)
- **C:\REF** (all files)
- **C:\SPL** (all files)
- **C:\TABLES** (all files)
- **C:\TREND** (all files)

The *File Expert* privilege is essentially the same as the File Access privilege. The only difference is that functions are not limited to specific directories. The File Expert privilege adds the same five features but their capabilities extend to the entire **D:** and **C:** drives of the SAGE^{MAX}, including *all* directories and subdirectories.

The *User Administrator* privilege adds the following Database Functions and Utility Functions that relate to User objects:

- Database Functions (Create Users, List Users and Erase Users)
- Utility Functions (Directory, Edit, Copy, Delete and Remote Copy of the file **C:\CFG\USER.BOB**)

The *Object Modifier* privilege allows certain objects to be changed under certain conditions. Users with Object Modifier privileges have the following additional capabilities:

- Monitor/Change Point Attributes (with some restrictions)
- Job Scheduler (Data Capture/Stuff with some restrictions)
- Find Objects (Find and Stuff with some restrictions)

In your database, it may be the case that you define several users with the Object Modifier privilege. This gives each of these users modification capabilities for *every* point, program and variable in the database. Although this may be the desired effect, SAGE^{MAX} also gives you the ability to further restrict the modification scope of each of those users.

For example, assume that USER1-USER9 are Object Modifiers, each of whom is responsible for maintaining a single floor of a building. SAGE^{MAX} has the ability to restrict the modification privilege of each user to points/programs/variables that are specific to his/her respective area.

These Object Modifier restrictions are based on *object privileges*. User Object Privilege Patterns are 8-bit patterns that are definable when the user is created/modified. Similarly, an *Object Privilege Pattern* must also be defined when you create points, programs and variables.

The User Object Privilege Pattern allows users with modify privileges to have different levels of modification abilities.

The ability to modify an object is determined through a binary operation of the Privilege Pattern of the object (i.e., the point, program or variable) and the Privilege Pattern of the user attempting to modify the object. If the result of logically ANDing the User and Object Privilege Patterns is the same as the Object Privilege Pattern in question, the user is permitted to change the value of the object. **Figure 7-1** and **Figure 7-2** show examples of users (Object Modifiers) with and without proper modification privileges.

NOTE

If a database object (i.e., point, program or variable) has an Object Privilege Pattern of all zeros, then any user with the Object Modifier privilege is permitted to modify the object's value.

The User Privilege Pattern has an additional function. It is also used to determine whether or not a particular user is permitted to acknowledge alarms that originate from a given point or program. The determination of alarm acknowledgment ability is similar to the process used to determine modification ability. The determination of acknowledgment is made using the Object Privilege Pattern of the *alarm class*. This pattern is logically ANDed with the User Privilege Pattern. The result of this operation determines if the user *is* or *is not* permitted to acknowledge the alarm.

With an understanding of the privileges available to users, who will be using the system and what privileges should be made available to each user, you can plan your users by using an *information table* that contains all necessary information until you are ready to enter the information into your database (refer to **Table 7-6**). These user information tables will also be useful when you plan the Object Privilege Patterns of points, programs, variables and classes.

For each user you plan to create, you must have a username, an associated codeword, the required *base set* privilege, any optional privileges, a user timeout and a Privilege Pattern (if required) for object modification and alarm acknowledgment. This information should be accumulated in an information table as shown in **Table 7-6**.

The username consists of up to 24 characters and identifies the user. You enter the username from the Sign-on Screen of the SAGE^{MAX}.

The codeword consists of up to 12 characters and is a password that is unique to each username. The codeword is entered after you enter the username to gain access to the system.

The base set privileges and the optional *special* privileges are stored as a 16-bit Menu Privilege Pattern which can be modified when you create/modify the user. Currently, only 8 bits in the 16-bit Menu Privilege Pattern are used.

User Name (24)	Password (12)	Menu Privileges								User Object Privilege Pattern	Timeout	
		I	S	M	U	X	F	A	O			
SYSTEM	AAM	1	1	1	1	1	1	1	1	11111111 B	0	SECS
SCON374	JERRY	1	1	1	0	1	1	1	1	00000001 B	0	SECS
ALSE378	MIKE	0	0	1	0	0	1	0	1	00000011 B	60	SECS
BENE369	DENNY	1	0	0	0	0	1	0	1	00000000 B	255	SECS
KAGH338	EARL	0	1	0	0	0	1	0	1	00000000 B	255	SECS
MCIE999SUPERUSER	STUSH	1	1	1	1	1	1	1	1	11111111 B	0	SECS
BIRG888	MILO	0	1	0	0	0	1	0	1	00000000 B	255	SECS
BAER777	VENUNZIO	0	1	1	0	1	0	1	1	00110010 B	0	SECS
BRER362	PATTY	1	1	1	0	1	1	1	1	01110101 B	0	SECS
POAK1STAR	GEORGE	1	0	0	0	0	0	1	0	00000000 B	0	SECS

Table 7-6 Sample User Information Table

For users with multiple privileges, you simply set the appropriate menu privilege bit numbers (0-16) corresponding to the desired privileges. The Menu Privilege bits default to zeros. **Figure 7-7** shows the User's Menu Privilege Bitmap.

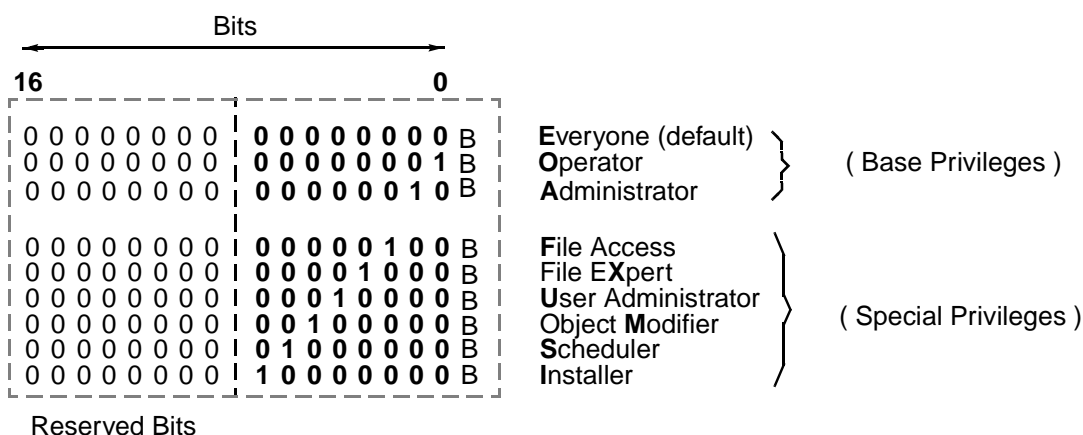


Figure 7-7 Dissection of the User's Menu Privilege Pattern

The User Object Privilege is stored as an 8-bit pattern which can also be changed when you create/modify the user.

Each user must be assigned a User Object Privilege Pattern which ranges from 00000000B to 11111111B (0-255 decimal). As previously explained, this pattern is used in conjunction with the Privilege Pattern of database objects to test whether or not a user (who is an Object Modifier) is permitted to modify a particular database object (i.e., a point, program or variable).

NOTE

If a user is an Object Modifier and has an Object Privilege Pattern of 11111111B, then the user is able to make modifications to all points/programs/variables, regardless of their Object Privilege Patterns.

The User Object Privilege Pattern is also used to test whether or not a user is permitted to acknowledge alarms of certain classes (refer to **Chapter 7.6.7: Planning Classes**). Therefore, you should plan User Object Privilege Patterns very carefully.

When planning a User Object Privilege Pattern, it is important to:

- decide that you (1) want to selectively limit a user's modification capabilities to certain points, programs and/or variables, and/or (2) want to allow alarm acknowledgment capabilities for this user on an object-by-object basis
- group the database objects based on areas where you desire selective modification (e.g., between buildings, floors, zones, wings, etc.)
- assign users to the appropriate areas
- specify Object Privileges for each user, based on modify and acknowledge patterns that you define
- define appropriate Object Privilege Patterns for each point, program, variable and class in each group you defined

NOTE

Due to their interrelationship, you may choose to plan User Privilege Patterns and the Object Privilege Patterns of points, programs, variables and classes simultaneously, after you have gathered the other pieces of information about these database objects. In general, this may simplify the process of planning Privilege Patterns.

The user timeout is a number of seconds that must pass with no keyboard activity before the SAGE^{MAX} automatically signs the operator off the SAGE^{MAX}. User timeouts range from 0-255 seconds. A timeout of 0 represents no timeout for the user.

After compiling all the necessary user information, you can move on to planning other objects in the SAGE^{MAX} database.

7.6.2 Planning Points

The creation of points requires some initial consideration and planning. The easiest approach is to start with your first network port and systematically create all points specific to that port. When all necessary points are created for port 1, move on to port 2, port 3, etc.

To effectively plan database points, there are several things you need (refer to **Figure 7-3**):

- User manuals of the network devices
- ID numbers of network devices
- Locations or uses of each network device
- Naming conventions for database points

- Information about users (e.g., Object Privilege Patterns for modification and alarm acknowledgment, and *modification areas* assigned to each Object Modifier)
- Tables for recording and later referencing information about each object

User manuals are used to gather information about attributes, ID numbers, channel and subchannel numbers, fundamental types, alarming, engineering units, etc. This information is vital when you create a database.

Each device on a network must have a unique *ID number* for proper network communication. Depending on the type of network you use, these device ID numbers may be selectable through software or based on hardware switches.

The ID numbers of Auto-Matrix unit controllers are programmable, but are set at the factory to the hardware serial number of the device. You can also find these ID numbers on the packing slip of your Auto-Matrix shipments or as attribute ;ID of the **FF00** channel of the unit controller. ID numbers of Auto-Matrix products are PUP unit numbers and range from 0-32767.

The SOLOTool, which is a hand-held operator interface for PUP networks, has a PUP ID number that ranges from 0-32767.

For PHP networks, the ID numbers of field panels such as the SAC3, STAR, RCU, RCU2 and MCU are programmable through software variables. Valid ID numbers are PHP unit numbers and range from 48-57 and from 65-255 for each of these field panels.

For XANP networks, the ID numbers of field panels such as the STAR, RCU, RCU2 and MCU are defined through switch settings on the hardware platform. The valid ID range is 0-31 for XANP networks.

For Peernet networks, the ID numbers of STAR field panels are defined through switch settings on the hardware platform. The valid ID range is 0-31 for Peernet networks. For SAGE^{MAX} field panels on Peernet networks, the ID number is software configured through driver configuration variables.

The SF1, which can be used in PUP and PHP networks, has separate PHP and PUP unit numbers. The PUP unit number ranges from 0-32767 and the PHP unit number ranges from 48-57 and from 65-255.

As can be seen from the examples above, how an “ID number” of a network device is *used* and how it is *derived* depends on the type of network being used. Be sure to use the proper ID number when you create database points. If you are unsure of how to find the unit number of a device, consult the appropriate user manuals for assistance.

When you are planning database points, it is helpful to know *where* each hardware object will be located or, at the very least, *what function* the object will perform. A floor plan of the job site might be helpful in this task. This information is used to plan description text to be associated with each point that you create. This text field can be up to 64 characters in length and is displayed when you monitor or modify the corresponding database point.

Developing a standard *naming convention* for database objects is important when planning a database. A well-planned naming convention adds information and consistency to your database object names. Good naming conventions can aid in searching for database points using match patterns in the name field. Database object names can contain up to 24 characters.

For example, if the SAGEMAX will be monitoring several air handler units (AHUs), you may have many points in your database that relate to these AHUs (e.g., control points, temperature points, flow points, etc.). Once you create the database, it may be desirable to have a list of all points that relate to the AHUs. If you use the convention that “*all AHU-related points must have the letters AHU as the first three characters of their name,*” you can easily get a list of all AHU points in the SAGE database by positionally matching those three characters in the name.

With a naming convention in place, you can now begin assigning names to the points you want to create. Since you will be accumulating a large amount of information about each point, it is helpful to keep an *information table* to record this data. Each table that you create should contain the name of the point, its network port number and the description text.

Using the user manuals of the devices, supply fundamental type information (AI, BO, TT, etc.), channel numbers (0-16, FE0H, etc.), and subchannel numbers (0-15) where appropriate to the information tables that you create. Depending on the type of network that you are using (e.g., PUP, PHP, Peernet, XANP, etc.), certain pieces of information will not be required. For example, points on PUP networks do not require fundamental type information, while certain database points do not use subchannels. The specific user manuals of the network devices explain how to address database objects.

If you want engineering units (EUs) to be attached to the attributes of the points that you create, you must define EU text files and associate them with each point. EU text files have the extension **EU** and are found on the **C:\EU** directory of the SAGEMAX. Enter the appropriate EU filenames on the information sheets that you create for each database point. The use of EUs is optional, so you may choose not to use them or you may choose to create them sometime later. The configuration of EU files is explained in **Chapter 7.9: Creating Engineering Units Files**.

When planning points, you may want to *log changes*. When enabled, this feature generates an alarm when any attribute of the point is modified by an operator. Attribute changes made from SPL programs are not logged.

Each point in the database has an 8-bit Object Privilege Pattern that is used in conjunction with a User Object Privilege Pattern to determine if a user is permitted to modify attributes in the point or acknowledge alarms that originate from the point. The ability to modify an object is determined through a binary operation of the Privilege Pattern of the object and the Privilege Pattern of the user attempting to modify or acknowledge the object. If the result of logically ANDing the User and Object Privilege Patterns is the same as the original Object Privilege Pattern, the user in question can change the value of the object or acknowledge alarms associated with the object.

NOTE

If a point has an Object Privilege Pattern of all zeros, then any user with the Object Modifier privilege can modify the object's value.

Some points may have alarm capabilities associated with them. For these objects, you may choose to create *alarm* and *return message text*. If you do, supply the appropriate text to the information tables that you create. Alarm information can be found in the user manuals of the network devices.

An example of an information table is shown in **Table 7-7**. In this example, port #1 is a PUP network, port #2 is a Peernet network and port #3 is a direct connect PHP host network.

Name (24) Description (64)	Net # ID #	Chan SChan	FType Card	EU File Famous	Log Changes Privileges	Alarm Message Return Message
FLR1_AHU1	1	FF00		RXFF00.EU	YES	
AHU1 for Floor 1 - Control Attributes	2234	0		NO	0 1 1 1 1 1 1 1	
FLR1_AHU1_DAT	1	FE00		RXTEMPS.EU	NO	Discharge Air Temp Alarm- AHU1 - Floor 1 ***
AHU1 for Floor 1 - Discharge Air Temp	2234	2		NO	0 0 0 0 1 1 1 1	Discharge Air Temp Normal - AHU1 - Floor 1
FLR1_LIGHTS	1	FB00		MXFB00.EU	NO	
Light Control for Floor 1 - SAGE # 1	4456	4		YES	0 0 0 0 1 1 1 1	
STAR1_SY	2	0	SY		NO	
System Point of STAR # 1	1	0	0	NO	0 0 0 0 0 0 1 1	
STAR3_MDDC_AI5	2	5	AI	IOMODAI.EU	YES	Relative Humidity Alarm - Floor 7 ***
Humidity Control - Floor 7	3	0	6	NO	0 0 0 0 1 1 1 1	Relative Humidity Normal - Floor 7
STAR37_ALC8_SY	3	0	SY		NO	
System Point of ALC8 on STAR 37	51	0	3	NO	0 0 0 0 1 1 1 1	

Table 7-7 Sample of an Information Table

7.6.3 Planning Programs

Programs are software database objects that are used to develop specialized applications. Programs on the SAGE^{MAX} are developed using SAGE^{MAX} Programming Language (SPL).

To effectively plan programs, there are several things you need:

- a list of system requirements
- user manuals of the network devices
- a naming convention for programs
- a list of information to be supplied to the program (if any)
- a list of information to be output from the program
- information about users (e.g., Object Privilege Patterns for modification and alarm acknowledgment, and *modification areas* assigned to each Object Modifier)
- a logic model

The process of planning programs is closely tied to the requirements of the application. You may not need programs in your database if the requirements of your application are met by the SAGE^{MAX}, a host system, a networked field panel(s) and/or unitary controllers.

The first important step to planning programs is to determine if any programs are actually needed. This is done by comparing system requirements to the features available to you. *System requirements* are part of the job specification, while available features can be found in the *user manuals* of the network devices.

If, for example, your application specifies a feature that is unavailable through the network devices being used, you may need to create SAGE^{MAX} programs to satisfy the system requirements. In some instances, however, it may prove beneficial to add to your network a device(s) that has the desired features rather than investing the time to plan, enter, debug and test specialized applications programs. In cases where adding devices to the network is cost-prohibitive, or the feature is so application-specific that it is not available in any unit controller or I/O Module, application programs may be the only alternative.

After you determine the need for an application program, you must select a name for the program object. Just as you developed a naming convention for database points, developing a standard *naming convention* for program objects is equally important. Program names can contain up to 24 characters, should supply information and should be consistent with other database object names.

After a program name is selected, you must decide what information, if any, is going to be supplied to the program (*input*) and what information is *output* from the program as a result. After you make the decision about inputs and outputs of the program, you must then associate them to named objects in the database.

Depending on your application, your program inputs may include the current time of day, information from another SAGE^{MAX} or an attribute of the program itself. Likewise, program outputs can include hardware points on another SAGE^{MAX} or attributes of the program itself.

With the program's inputs and outputs defined, you can begin to link them by creating logic for the application. The *logic* lists the sequenced, logical steps that must occur to achieve the desired outputs to satisfy your application. Due to a person's stylistic preference and experience, program logic may be as simple as several lines of *pseudocode* or as complex as a graphic representation using standard flowcharting symbols. **Figure 7-8** shows a segment of program logic in both pseudocode and flowchart forms.

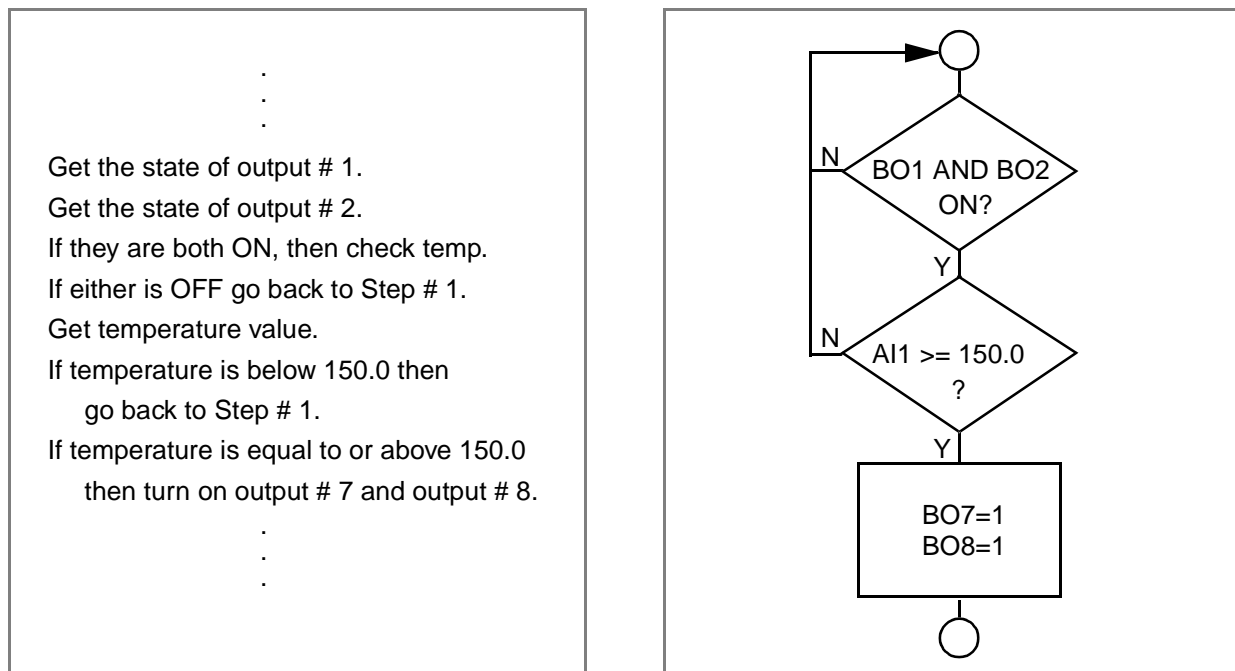


Figure 7-8 Sample Logic of a Program Segment in Pseudocode and Flowchart Forms

Once you have defined the inputs and outputs of your program and the program logic, you are ready to edit and compile your program source code. This involves converting your logic into SPL code that is recognizable by the SAGE^{MAX}. Due to the scope of programming in SPL, an entire chapter is devoted solely to this topic. Refer to **Chapter 11: Programming** for detailed information on creating and using SPL programs.

Each program in the database has an 8-bit Object Privilege Pattern that is used in conjunction with a User Object Privilege Pattern to determine if a user is permitted to modify program attributes or acknowledge program alarms. The ability to modify program attributes is determined through a binary operation of the Program Privilege Pattern and the Privilege Pattern of the user attempting to acknowledge alarms or modify attributes. If the result of logically ANDing the User and Program Privilege Patterns is the same as the original Program Privilege Pattern, the user in question can change the value of the program attributes or acknowledge program alarms.

Like points, programs can have an associated EU file. If you want engineering units (EUs) to be attached to the attributes of the programs that you create, you must define EU text files and associate them with each program. EU text files have the extension **EU** and are found on the **C:\EU** directory of the SAGE^{MAX}. The use of EUs is optional, so you may choose not to use them or you may choose to create them sometime later. The configuration of EU files is explained in **Chapter 7.9: Creating Engineering Units Files**.

7.6.4 Planning Variables

Variables are database objects that are used in the same way as points, but have a definable data type and a single, definable value.

The use of variables is best explained through an example. In the logic example of **Figure 7-8**, the flow of the logic depends on the value of a temperature sensor (i.e., MX_AI1;CV) and how that temperature compares with the number 150.0.

Assume that we have several programs that use this same number in similar comparisons. Also assume that someone may choose to change the comparison value at some point in the future. With a SAGE^{MAX} database containing 10 similar programs, for example, you would have to make changes in 10 different places. If, however, you create a *variable* with the value 150.0, you can use the variable name in the programs with similar comparisons. Then, when the comparison value needs to be adjusted, you can effectively change all 10 programs by simply changing the value of a single variable.

To effectively create variables, there are several things you need in your initial planning:

- a value to be assigned to each variable
- a data type for each variable
- a naming convention for variables
- information about users (e.g., Object Privilege Patterns used for variable modification by Object Modifiers)

For each variable that you plan to create, you should define an *initial value*. This value is directly related to the application for which the variable is being used.

In addition to an initial value, each variable must have a *data type*. The data type of the variable determines how the value is to be displayed (e.g., a hexadecimal word, a bitmap, a floating point number, an unsigned value with 2 decimal places, etc.).

Just as you developed a naming convention for database points and programs, developing a standard *naming convention* for variables is equally as important. Variable names can contain up to 24 characters, should supply information and should be consistent with other database object names.

Each variable in the database has an 8-bit Object Privilege Pattern that is used in conjunction with a User Object Privilege Pattern to determine if a user is permitted to modify the value of a variable. The ability to modify the value of a variable is determined through a binary operation of the Variable Privilege Pattern and the Privilege Pattern of the user attempting to modify the variable value. If the result of logically ANDing the User and Variable Privilege Patterns is the same as the original Variable Privilege Pattern, the user in question can change the value of the variable.

7.6.5 Planning Globals

In SAGE^{MAX} applications that use an Ethernet network, database objects (i.e., points, variables and programs) can be *famous*. Database objects that are famous automatically create *globals* on all other SAGE^{MAX} field panels on the Ethernet network. Globals can be created automatically in this fashion or manually by supplying a peername (up to 24 characters in length) and an optional EU filename. This gives you the ability to monitor/modify objects on a SAGE^{MAX} from any SAGE^{MAX} on the Ethernet by creating the object only once.

Planning globals uses the same process as planning points, variables or programs. Reserve a space on each information table to note whether or not you want an object to be famous. Creation of globals is then a simple process of creating famous points variables or programs on other SAGE^{MAX} field panels on the Ethernet. See **Figure 7-9**.

NOTE

A database object (point, variable or program) that is created on a SAGE^{MAX} can be famous. The result of creating a famous object on a SAGE^{MAX} is the automatic creation of a global object on every Ethernet SAGE^{MAX}.

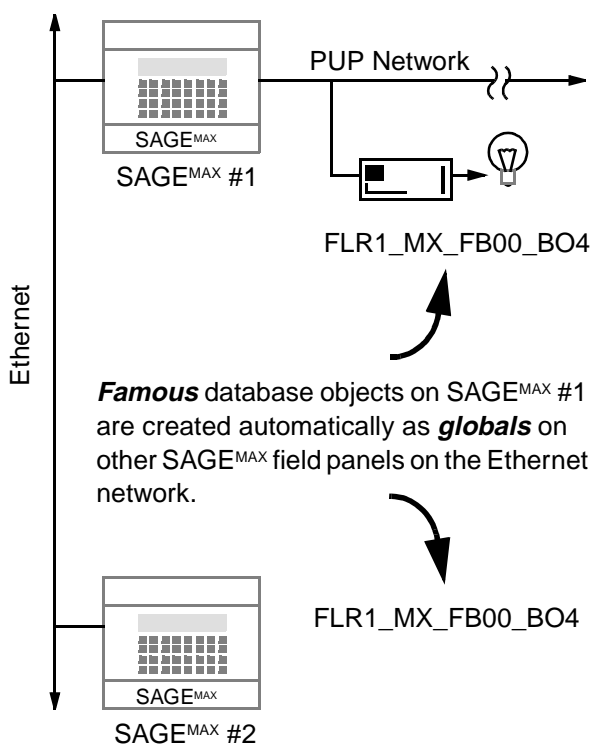


Figure 7-9 Automatic Creation of Globals from Famous Database Objects

7.6.6 Planning Groups

Groups are collections of up to 16 database objects that appear together as a menu. Groups can be nested to provide convenient access to other database objects.

Like the creation of other database objects, the creation of groups requires some initial consideration and planning.

To effectively create groups, there are several things you need in your initial planning:

- a list of point, program, variable and global names
- a naming convention for groups
- a grouping strategy

Having already planned points, programs, variables and globals, you should have lists of these *object names*.

Just as you developed a naming convention for database points, programs and variables, development of a standard *naming convention* for groups is equally as important. Group names can contain up to 24 characters, should supply information and should be consistent with other database object names.

For each group that you create, you must plan a *grouping strategy*. The particular grouping strategy that you choose will depend on the projected size of the database, the types of operators, and the system requirements.

Typically you will want to create a main group whose elements are subgroups that represent certain areas of interest. For example, the group **MAIN** may have six subgroups as shown in **Figure 7-10**. Each of the nested groups may have several nested groups representing floors within each building. This nesting of groups can continue to the component level (e.g., fans, sensors, setpoints, etc.) or can continue based on logical groups of devices (e.g., air handlers, unit ventilators, lighting, security, etc.). The specific nature of the application, the types of operators and the size of the database will affect the grouping strategy that you choose.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr:                                                    Port: 07

Group Name: MAIN
key          to select
A           BUILDING1
B           BUILDING2
C           BUILDING3
D           BUILDING4
E           BUILDING5
F           BUILDING6
PF1        Return to previous menu
```

Figure 7-10 Sample Group with Six Nested Groups

Once a grouping strategy is developed, you can save your ideas so they may be entered at a later time. You may, however, choose to enter *group files* directly into the SAGE^{MAX} using the SAGE^{MAX} text editor. A third alternative is to create the desired group files offline using any standard text editor.

Group files are text files that contain an unlimited number of lines. These files have three types of lines in them.

Comment lines begin with a semicolon in the first position and are used to document the group file.

Subgroup path lines begin with a backslash character (\) in the first position and specify a path fragment on the **\GROUP** subdirectory. The path fragment may contain up to 26 characters and may specify up to two subdirectories. If an extension is specified, it is removed and assumed to be **.GRP**. This line may also contain an optional description text field.

Object reference lines contain an object name up to 24 characters long, one of its attributes (optional), and an optional description of the object.

Refer to Appendix L for an overview of the structure of SAGE^{MAX} group files.

7.6.7 Planning Classes

Classes are database objects that are used in defining alarm information such as where to log alarms, acknowledgment information, alarm priority and dialing information. The creation of classes takes some initial consideration and planning. Effective planning requires that you have the following information at hand:

- user manuals of the network devices
- a list of driver types for the SAGE^{MAX} ports
- alarm destination information
- an optional naming convention for class names

User manuals are used to gather information about alarm capabilities of database objects. This information, in conjunction with a list of the *port driver types*, is used to determine a SAGE^{MAX} class number for alarms on your network configuration. SAGE^{MAX} class numbers are shown in **Table 7-8**.

Once you determine the class numbers of alarms that can be produced by your particular configuration, you can optionally change the class number (000-255) to an associated *class name* (e.g., FIR, PUP, STR, TMP, etc.). Since alarm text includes the class of the alarm, a class name such as FIR may be more meaningful than an alarm of class 027.

Alarm Class	Description	Examples
000 General 001 Action Required 002 Operator 003 Program Exec 004 Warning 005 Trends 006 007 008 009 Unit Scan	SAGE Alarm Class 000 SAGE Alarm Class 001 SAGE Alarm Class 002 SAGE Alarm Class 003 SAGE Alarm Class 004 SAGE Alarm Class 005 SAGE Alarm Class 006 SAGE Alarm Class 007 SAGE Alarm Class 008 SAGE Alarm Class 009	Broadcasts Printer Out of Paper Sign-on, Sign-off, Log Alarm Acks SAGE Program Alarms Bad "Time" Field in Schedule Request SAGE Trend Alarms PUP Unit Scan Alarms and Returns
010 STAR Action 000 011 STAR Action 001 012 STAR Action 002 013 STAR Action 003 014 STAR Action 004 015 STAR Action 005 016 STAR Action 006 017 STAR Action 007 018 STAR Action 008 019 STAR Action 009 020 STAR Action 010 021 STAR Action 011 022 STAR Action 012 023 STAR Action 013 024 STAR Action 014 025 STAR Action 015	STAR Alarms Using Action 000 STAR Alarms Using Action 001 STAR Alarms Using Action 002 STAR Alarms Using Action 003 STAR Alarms Using Action 004 STAR Alarms Using Action 005 STAR Alarms Using Action 006 STAR Alarms Using Action 007 STAR Alarms Using Action 008 STAR Alarms Using Action 009 STAR Alarms Using Action 010 STAR Alarms Using Action 011 STAR Alarms Using Action 012 STAR Alarms Using Action 013 STAR Alarms Using Action 014 STAR Alarms Using Action 015	Any STAR Alarms Using Action 000 Any STAR Alarms Using Action 001 Any STAR Alarms Using Action 002 Any STAR Alarms Using Action 003 Any STAR Alarms Using Action 004 Any STAR Alarms Using Action 005 Any STAR Alarms Using Action 006 Any STAR Alarms Using Action 007 Any STAR Alarms Using Action 008 Any STAR Alarms Using Action 009 Any STAR Alarms Using Action 010 Any STAR Alarms Using Action 011 Any STAR Alarms Using Action 012 Any STAR Alarms Using Action 013 Any STAR Alarms Using Action 014 Any STAR Alarms Using Action 015
026 PUP Class 000 027 PUP Class 001 028 PUP Class 002 029 PUP Class 003 030 PUP Class 004 031 PUP Class 005 032 PUP Class 006 033 PUP Class 007 034 PUP Class 008 035 PUP Class 009 . . . 253 PUP Class 227 254 PUP Class 228 255 PUP Class 229	System Applications Alarms Life Safety Alarms Mechanical Alarms General Alarms PUP Alarms of Class 004 PUP Alarms of Class 005 PUP Alarms of Class 006 PUP Alarms of Class 007 PUP Alarms of Class 008 PUP Alarms of Class 009 . . . PUP Alarms of Class 227 PUP Alarms of Class 228 PUP Alarms of Class 229	Fire Alarms Fan Run Limit Alarms Temperature/Flow Limit Alarms

Table 7-8 SAGEMAX Alarm Classes

Every class has information regarding the *destination* of its alarms. Proper planning of classes involves understanding the types of alarms that may be generated by your system, where these alarms are to be reported based on the time of day and day of the week, dialing options such as baud rates and phone numbers, and, in manual acknowledge cases, the types of users and their associated privilege levels. **Figure 7-11** shows a tree diagram detailing the information needed to plan classes on the SAGEMAX. This information should be planned and then saved in information tables similar to those used in planning database points.

Each alarm class has the option of logging its alarms to the general alarm log file (**C:\ALARMS\GENERAL.LOG**) and/or its associated class alarm log file (**C:\ALARMS\027.LOG** or **C:\ALARMS\FIR.LOG** if you assigned the class name **FIR**). For each alarm class you must decide if you want the alarms to be logged to either (or both) of these text files.

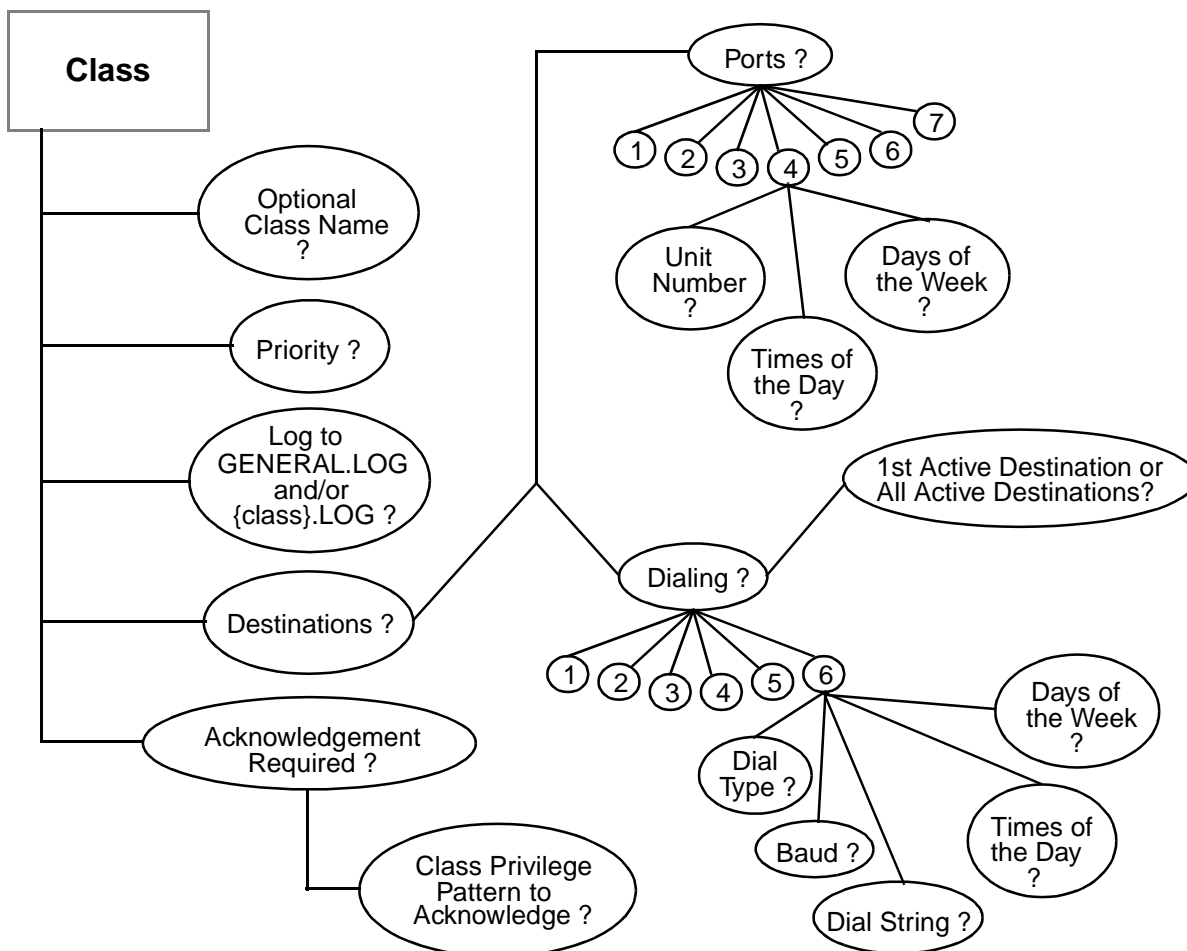


Figure 7-11 Tree Diagram for Planning Classes

You can configure each class so that alarms of that class are not acknowledged automatically. If acknowledgment is required for alarms of certain classes, you must decide which operators will have the proper privilege levels to do the acknowledgment. The privilege code that is used for acknowledgment of alarms is an 8-bit Acknowledgment Privilege Pattern associated with each class.

The use of the Acknowledgment Privilege Pattern in classes allows users with like privilege levels to have different levels of alarm acknowledgment. The ability to perform alarm acknowledgment is determined through a binary operation of the Acknowledgment Privilege Pattern and the User Object Privilege Pattern of the user attempting the acknowledgment. If the result of logically ANDing the User and Acknowledgment Privilege Patterns is the same as the Acknowledgment Privilege Pattern of the class, the user in question can acknowledge the alarm.

Alarm priorities are assignable by class. Each alarm class has an associated priority from 0-255. Large numbers dictate a higher priority in servicing the alarm. The highest priority is level 255 and the lowest is level 0.

Alarms of each class can be reported to as many as 7 SAGE^{MAX} ports. Each port destination includes the port number where you want alarms to be sent, the desired unit number scheduled to receive the alarms, and the days of the week and time interval when alarms should be sent to the port/unit.

Classes also have dialing capabilities. Like port destinations, classes can specify 6 dial destinations. Each dial destination includes the dial type (e.g., ring only, printer and automatic), the dial-out baud rate, the dial string (up to 40 characters) for the destination, and the days of the week and time interval when alarms should be dialed out. Additionally, each class specifies whether the first active dial destination should be used or if all active dial destinations should be used.

Remember to save all class information in tables similar to those used when you plan database points. This will simplify the creation process of SAGE^{MAX} classes.

An entire chapter is devoted to Alarm and Event Management. Refer to **Chapter 10: Alarm and Event Management**. This chapter gives detailed information on creating, managing, logging, acknowledging and dialing alarms.

7.7 *Creating a Database*

After you compile all the necessary information in the database planning phase, the actual *creation* of the database can begin. If the information you gather in the planning phase is complete and accurate, the creation phase is nothing more than entering data.

You can begin creating a database by signing on to the SAGE^{MAX} from a local terminal using the default username **SYSTEM** and the default codeword **AAM**.

Chapter 8: SAGE^{MAX} Menu Operations gives a detailed explanation of every function of the SAGE^{MAX} and how to maneuver through every menu.

7.8 *Off-line Database Creation*

To create a SAGE^{MAX} database offline, follow the steps below:

1. Use any standard ASCII text editor on a personal computer that has a 3.5" 1.44MB (high density) disk drive to write the SAGE^{MAX} database.
2. Save this *Name Binding File* (NBF) to a 3.5" 1.44MB (high density) diskette.
3. Place the diskette into the floppy disk drive on the SAGE^{MAX}.
4. Translation programs convert the text files into the appropriate *Binary Object* (BOB) database files that are used by the SAGE^{MAX}.

Name Binding Files are ASCII text files that always have the extension **NBF**. Each file may contain an unlimited number of lines. There are three basic types of lines in a Named Binding File: *comment lines*, *binding definition lines* and *continuation lines*.

Comment lines begin with a semicolon (;) as the first character of the line. The remainder of the line is ignored when you convert the file.

Name Binding definition lines share a similar format. Each of these lines begins with a unique two-character mnemonic which identifies the type of definition. **VR**, **PT**, **PG** and **GL** are the mnemonics used to identify variable, point, program and global objects, respectively.

The mnemonic code is followed by a space and then a symbol name that is up to 24 characters in length. The symbol name may contain any combination of the following characters:

- A through Z
- a through z
- 0 through 9
- \$ (dollar sign)
- . (period)
- _ (under bar)
- (space)

Symbol names are not case sensitive, so “a” and “A” are considered to be equivalent in names. Names may contain embedded spaces, but trailing spaces are removed when the file is converted. The symbol name is followed by an equal sign (=) which separates the name from its definition parameters. The definition parameters are different for each type of object definition. See **Appendix J: Name Binding Files** for a description of the binding definition parameters of each SAGE^{MAX} object type.

Continuation lines begin with a **TAB** character as the first character of the line. The format of the continuation line is dependent on the type of binding definition being used. Continuation lines are used when it is not practical to fit the entire Name Binding definition on one line.

7.9 Engineering Units Files

Engineering units are text descriptions that are appended to points, programs and globals when their values are monitored. These descriptions are used to enhance the meaning of a database object’s value. For a point that represents a temperature sensor, a typical engineering unit might be “Degrees F.” A point that represents a binary output might have two text descriptions (“Off” and “On”) which are indexed on the two possible values of the output (0 and 1).

Some points/programs/globals have several attributes that have non-unique engineering units. For example, a point might contain a current value attribute (**;CV**) that represents a temperature (“Degrees F”) and an alarming enable attribute (**;AE**) that represents a binary state (“Enabled” or “Disabled”). To allow different engineering unit text to follow different attributes within the same point (or program or global), engineering units can be assigned to each of the attributes *individually*, through a text file that is referenced by the point/program/global.

Engineering units (EU) table files are text files that contain a series of attributes, each followed by either a definable text string (16 characters maximum) or a series of indexed, delimited text strings (16 characters maximum for each index). The text files must be located on the **C:EU** directory of the SAGE^{MAX} and must have the extension **.EU**. If any file errors occur when the

SAGE^{MAX} tries to access an EU file, the EU field will be left blank.

When you create a point/program/global, you can assign an EU file. If you do not assign an EU file, engineering units will not be displayed for that particular point/program/global. It is possible to assign many points/programs/globals to the same EU file. You are also free to create a separate EU file for each point/program/global object type in your database.

The EU file can contain comment lines which begin with a semicolon in column 1. Comment lines are for documenting the EU file and are not used by the SAGE^{MAX}.

The first two characters of each non-commented line must be an attribute name. Remember that attribute names are case-sensitive, so **cv** is not the same as **CV**.

Direct EU assignments must have an equal sign (=) following the attribute name. Any other character (except for the pound sign #) in position three is considered a delimiter for an indexed-style engineering unit. Direct EU assignments are used when the EU text must remain the same, regardless of the value of the attribute.

Indexed EU assignments display different text based on the attribute value of the database object. The text fields must be delimited using a character that will not appear in any of the indexed options for that attribute.

No more than four indices may appear on a single line of an EU file. For attributes that have more than four states, up to three index continuation lines may be added to the EU file, but they must be contiguous.

The indices of indexed attributes are zero-based starting with the first occurrence in the EU file, therefore there are no negative indices. It is not required that the last occurrence of an indexed attribute include four indices.

If the value of an indexed attribute exceeds the number of indices provided in the EU file, no engineering unit is appended to the attribute value.

NOTE

There is a maximum of 32 possible indexed states for an indexed attribute in an EU file.

Since an attribute's value acts as its index, indexed engineering units are valid only for integer-style data types such as:

- 00H hex byte
- 01H hex word
- 02H hex double word
- 05H day of the week
- 07H no/yes

- FEH unsigned 10 digit
- FFH signed 10 digit (positive only)

NOTE

Both direct and indexed engineering units are limited to a maximum of 16 characters.

Bitmap EU assignments display different text fields based on bits that are selected in an 8-bit bitmap. Bitmap EU assignments are similar to indexed assignments, but they use the pound sign (#) as the delimiter. The active days bitmap is a good example of a bitmap EU assignment. The format for an active days bitmap example is:

AD#MON#TUE#WED#THU#FRI#SAT#SUN#.

Bitmap EU assignments may have up to 8 labels, each of which may contain up to 7 characters. Each bitmap EU assignment must be on a single line in the text file.

Figure 7-12 shows the default EU file for all SPL programs that are created on the SAGE^{MAX}. This file shows both direct and indexed EU assignments and acts as an example when you create other EU files on the SAGE^{MAX}.

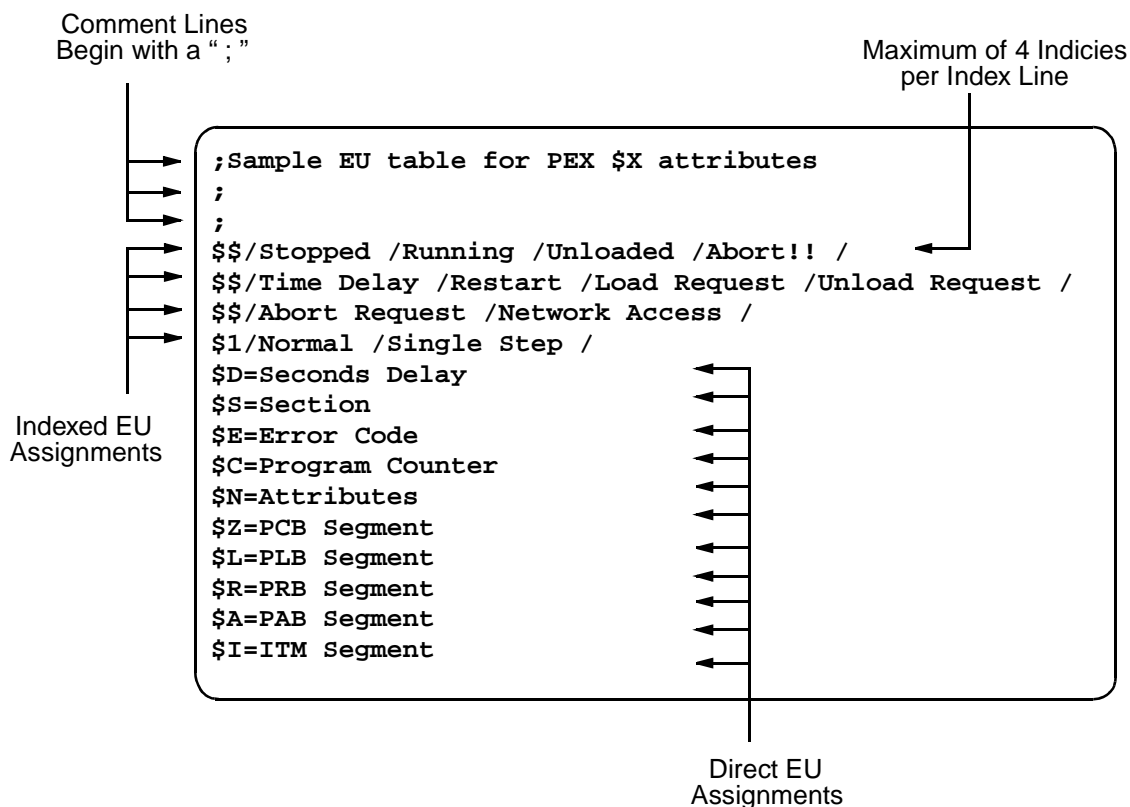


Figure 7-12 PEX.EU - The Default EU File for SPL Programs

7.10 Table Files

Tables are collections of up to 1,073,741,824 data values which are stored as linear, one-dimensional arrays in disk-resident files on the SAGE^{MAX}. All table files are stored in the reserved directory **C:\TABLES**. Table files may be stored in this directory or in any of its subdirectories. Table files must have the extension **.TBL**. A typical table file might have the full path name **C:\TABLES\XXX\YYY.TBL** which would be represented by the path fragment **XXX\YYY**.

Table files are useful for creating, among other things, linearization tables. Tables are created and edited using a special table editor which is built-in to the SAGE. This editor is accessed through the **Table (L)** option of the Main Menu. Tables can also be created and updated using an SPL program.

NOTE

Although table files can contain 1,073,741,824 data values, the SAGE^{MAX} table editor is only capable of handling 65,536 entries. Tables with more entries than this must be created/edited using an SPL program.

After recognizing the need for a table within your database, you must decide what type of table you require. The SAGE^{MAX} offers two types of tables. *Full tables* contain a separate value and a separate data type for every element of the table. *Sparse tables* contain a separate value for every element of the table, but have a single data type which applies to every element of the table. **Figure 7-13** illustrates the difference between a full table and a sparse table.

Table: FULL		Table: SPARSE Data Type : [FCh]	
0	[FEh] 100	0	100.5
1	[FCh] 100.5	1	100.5
2	[FCh] 200.7	2	200.7
3	[FAh] 90.75	3	907.5
4	[FAh] 45.01	4	45.1
5	[FEh] 200	5	200.9

Figure 7-13 Full Tables Versus Sparse Tables

After deciding what type of table is needed for your application, you must create a name for your table. Table names can consist of up to 8 characters and may consist of the characters A-Z, a-z, 0-9 and \$. If a file fragment is used, up to 2 subdirectories (up to 8 characters in each) may be included in the table name (e.g., **\SENSORS\STAEFA\XYZ**).

Once you have decided on a name, you can begin to plan the data that will be stored in the table. After the table is created, you will be able to access the elements of the table (using a zero-based index) from an SPL program. This gives you the flexibility to create linearization tables for non-standard sensors.

In planning a table, you should know the number of elements that will be in the table. If, however, you determine that you are going to need more elements in a table that you have already defined, you can automatically extend the size of your table when you edit the table.

If your SAGE^{MAX} requires the use of tables for specialized applications, you should follow the same rules as mentioned earlier in this section for planning your database. All of the information pertaining to SAGE^{MAX} tables should be saved in information tables (i.e., on sheets of paper) to facilitate data entry when the actual table creation occurs.

7.11 *Group Files*

Group files are ASCII text files that contain the references of group objects in the SAGE^{MAX} database. Group files are unlike other database objects in how they are used by the SAGE^{MAX}. While point, program, global and variable objects take the form of binary object (**.BOB**) files in the SAGE^{MAX} database, group files are simply ASCII text files.

In the case of off-line database editing, point, program, global and variable objects are entered as ASCII text files (**.NBFs**) and must be converted to binary object (**.BOB**) files. For group files, off-line editing is done by creating/editing the file and then copying the file to the **C:\GROUPS** subdirectory of the SAGE^{MAX}.

The format of group files is shown in **Figure 7-14**.

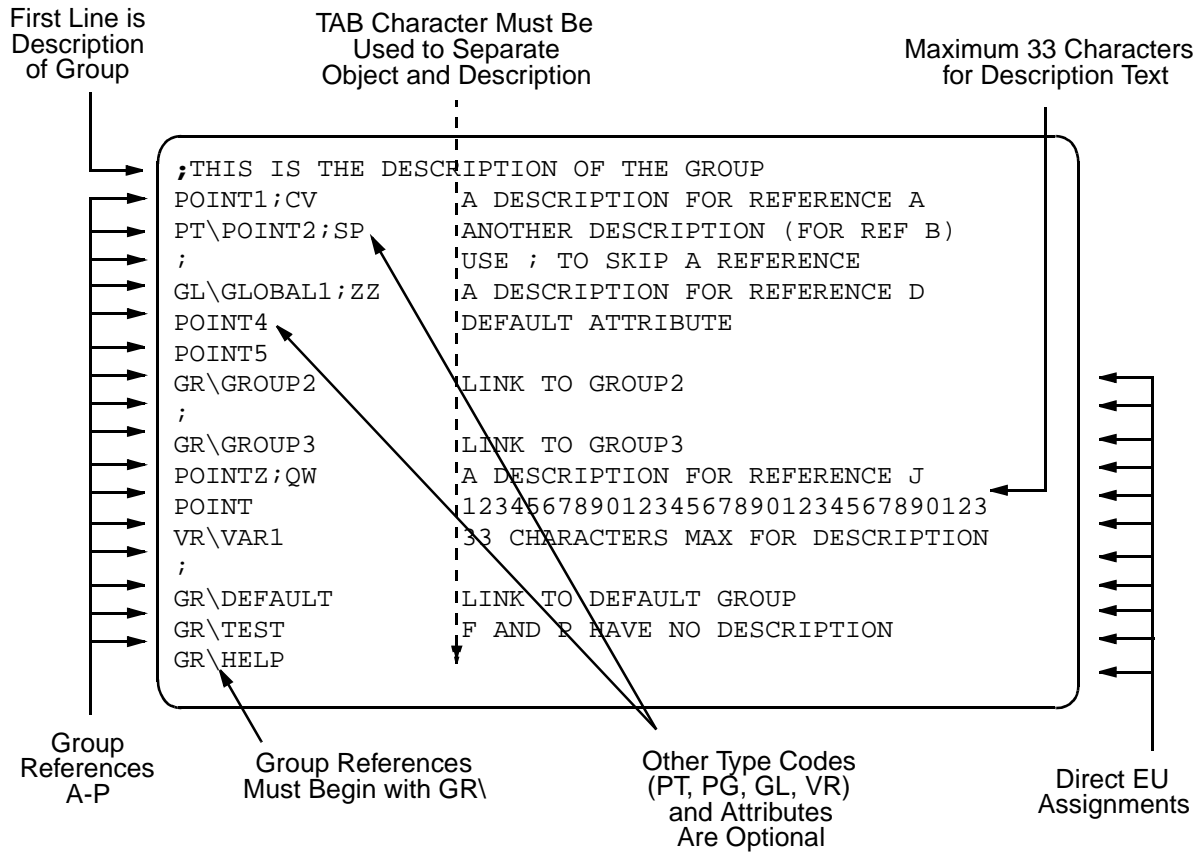


Figure 7-14 Sample SAGEMAX Group File

CHAPTER 8 - SAGE^{MAX} MENU OPERATIONS

This section explains in detail all of the SAGE^{MAX} menus, submenus and options that are available from an OPI in a depth-first fashion. Sample screens are shown for every menu/option from the perspective of an operator with the maximum number of privileges on a VT100 terminal.

To access the SAGE^{MAX} menus from an OPI, you must at least have a minimal hardware setup. This minimal configuration is explained and illustrated in **Chapter 5.12: Minimal Setup for Software Configuration**.

8.1 *The Sign-on Screen*

You sign on to the SAGE^{MAX} using a username and a corresponding codeword from the **Sign-on** Screen of the OPI. The **Sign-on** Screen of the SAGE^{MAX} displays the software version number, the banner line text, the date and time, the operator field (which is blank before you sign on), the SAGE^{MAX} port to which the terminal is connected, and an **ALARMS** field that flashes when unacknowledged alarms are present. The SAGE^{MAX} **Sign-on** Screen is shown in **Figure 8-1**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr:                                                     Port: 07
Enter User Identification: SYSTEM
Enter Codeword:
```

Figure 8-1 The SAGE^{MAX} Sign-on Screen

Although usernames and codewords can be added and changed, you must use the username **SYSTEM** and the codeword **AAM** when you initially sign on to the SAGE^{MAX}. This username has the maximum number of privileges and allows access to all SAGE^{MAX} menus and features. Subsequent usernames and codewords are assigned by the system manager of each site, or may be created during the commissioning of the system by the installer. Appropriate privileges are assigned to the usernames at that time.

From the **Enter User Identification:** prompt, you enter the default username **SYSTEM**. The username is echoed (displayed on the screen as you type it). After you enter the username, you are prompted to **Enter Codeword:**. Enter the default codeword **AAM**. Unlike the username, the codeword is not shown on the screen for security reasons.

If the username and codeword are valid, you are prompted with the **Main Menu**. From then on, the **Opr:** field of the banner line shows the username. Refer to **Figure 8-2**.

If you enter an invalid username and codeword, the error message **Invalid ID/Code** is displayed. Press any key and the OPI will again prompt you for a username.

8.2 The Main Menu

The **Main Menu**, shown in **Figure 8-2**, is the first menu you see after signing on to the SAGE^{MAX}.

The **Main Menu** is considered the starting point of all menu activities. From any level of the Menuing System, you can usually return to the Main Menu by simply typing a series of **ESC**s (if your OPI is a dumb terminal) or **PF1**s (if your OPI is a VT-type terminal). Typing **ESC** or **PF1** from the **Main Menu** simply refreshes the menu.

You choose **Main Menu** options by pressing the letter key for the function you want to perform.

Typing **!** from the **Main Menu** toggles the OPI between normal and Expert Mode. For more information on the **!** key, refer to **Chapter 6.11: Expert Mode: The “!” Key**.

Each **Main Menu** option is described on the pages that follow.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Main:
key          to do
A           Alarm List
B           Broadcast Message
C           Calendar Edit
D           Database Functions
F           Find Objects
J           Job Scheduler
L           Table
M           Monitor/Change Point Attributes
P           Ports Status and Setup
Q           Quit
S           System Variables
T           Trend
U           Utility Functions
V           Virtual Terminal
PF1         Return to previous menu
Press key for desired action:
```

Figure 8-2 The Main Menu of the SAGE^{MAX}

8.3 The Alarm List Submenu (A)

Selecting the **Alarm List (A)** option from the **Main Menu** displays the **Alarm List** Submenu shown in **Figure 8-3**. From this submenu you select options that allow you to selectively view, archive and delete alarms.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Alarm List:
Key to do
U Unacknowledged Alarms
A All Alarms
C Alarms by Class Name
R Archive Alarms
D Delete Alarms
PF1 Return to Previous Menu
Press key for desired action:
    
```

Figure 8-3 The Alarm List Submenu

8.3.1 Unacknowledged Alarms

The **Unacknowledged Alarms (U)** option of the **Alarm List** Submenu allows you to view all or a select group of unacknowledged alarms from **C:\LOG\ALARMS.ACK** that have not yet been acknowledged. After choosing this option, you are presented with the Alarm Selection Template Screen shown in **Figure 8-4**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
*text* = Find "text" -- OR --
Trans=Transaction #, Unit=Device Unit, Cls=Class Name, Message=Alarm Msg Text
A Trans Unit- Cls Message-----
  ?????  ?????  ???  ?????????????????????????????????????????????????????????????
    
```

Figure 8-4 The Alarm Selection Wild Card Template Screen for Listing Unacknowledged Alarms

From this screen, you can selectively choose transaction, unit, class and text information as a match pattern for the unacknowledged alarms that you want to view. You use the left and right arrow keys to maneuver through the match pattern. When the cursor is moved to the desired match position(s), type the text match pattern you want. After you enter the match pattern (or simply type **RETURN**), the matching unacknowledged alarm text will be displayed in groups of up to ten lines at a time. See **Figure 8-5**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
*text* = Find "text"  -- OR --
Trans=Transaction #, Unit=Device Unit, Cls=Class Name, Message=Alarm Msg Text
A  Trans Unit- Cls Message-----
0  02130/00000 003 Mon 23-Sep-96 11:21:02
1  -          003 PROGRAM7 halted at 0123:0021 due to #InvalidPcode
2  02121/00016 STR Mon 23-Sep-96 11:22:15
3  -          STR RM11_AI7          High Limit Alarm
4  -          STR Building 17 Room 11 AI#7 - Room Humidity Sensor
5  -          STR *** Humidity Beyond Upper Limit ***
6  02113/00000 SCN Mon 23-Sep-96 11:23:13
7  -          SCN Port 3, Unit 4111   Online Scan Alarm
8  02093/00000 SCN Mon 23-Sep-96 11:23:14
9  -          SCN Port 3, Unit 591    Online Scan Alarm

Press Key: N(+) next page, P(-) previous page, PF1 (Esc) escape
Press Key: 0-9 acknowledge this alarm, A acknowledge all alarms

```

Figure 8-5 Example Display Screen of Unacknowledged Alarms

The question marks in the match pattern are wild card characters which, if left in the match pattern, represent positional matches regardless of the alarm text in those positions. A match pattern of all question marks, for example (as shown in **Figure 8-4**), displays *all* unacknowledged alarms.

The selective use of question marks in the match pattern can be used to retrieve groups of alarms such as:

- all unack'd alarms from port 3
- all unack'd alarms of class STR from unit 00016
- all unack'd TMP alarms from port 2 that occurred last Saturday

The use of question marks in the alarm template allows you to be as selective as you like when listing unacknowledged alarms.

Rather than specify text match patterns *positionally*, you can enter a desired match pattern delimited with asterisks (e.g., ***Limit***). Using a match pattern in this way causes the SAGE^{MAX} to find all alarm lines that contain the match pattern in *any* position.

The **A** column of the alarm list is present only from the **Unacknowledged Alarms (U)** option of the **Alarm List** Submenu. When you list a group of unacknowledged alarms, the **A** column contains alarm line numbers from 0-9 (the alarms that have occurred most recently appear first in the list). You can acknowledge any alarm in this list by typing the number that corresponds to the first line of the alarm you want to acknowledge.

For example, to acknowledge the High Limit Alarm from STAR unit number 16, type 2 at the prompt shown in **Figure 8-5**. This causes an asterisk (*) to appear before the transaction code column of the display for that alarm. In this example, typing **0, 2, 6** or **8** acknowledges an alarm and causes an asterisk to appear before the transaction code of the associated alarm. Typing **1, 3, 4, 5, 7** or **9** does not acknowledge an alarm since these numbers do not correspond

to the first lines of the alarms. **Figure 8-6** shows an example display screen with every alarm acknowledged.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
*text* = Find "text"  -- OR --
Trans=Transaction #, Unit=Device Unit, Cls=Class Name, Message=Alarm Msg Text
A  Trans Unit- Cls Message-----
  ????? ???? ???? ?????????????????????????????????????????????????????????
0 *02130/00000 003 Mon 23-Sep-96 11:21:02
1 -           003 PROGRAM7 halted at 0123:0021 due to #InvalidPcode
2 *02121/00016 STR Mon 23-Sep-96 11:22:15
3 -           STR RM11_AI7           High Limit Alarm
4 -           STR Building 17 Room 11 AI#7 - Room Humidity Sensor
5 -           STR *** Humidity Beyond Upper Limit ***
6 *02113/04111 SCN Mon 23-Sep-96 11:23:13
7 -           SCN Port 3, Unit 4111   Online Scan Alarm
8 *02093/00591 SCN Mon 23-Sep-96 11:23:14
9 -           SCN Port 3, Unit 591     Online Scan Alarm

Press Key: N(+) next page, P(-) previous page, PF1 (Esc) escape
Press Key: 0-9 acknowledge this alarm, A acknowledge all alarms
    
```

Figure 8-6 Example Display Screen of Acknowledged Alarms

Once you type **PF1** to leave this screen and then return to it, the asterisks and all of the associated lines of alarm text that were acknowledged are no longer listed in the unacknowledged alarm list.

The **Trans** column of the alarm list contains a 5-digit number that represents a transaction number and the port number of the alarm. In the example shown in **Figure 8-7**, transaction code **02113** represents transaction number **0211** and port number **3**.

In this example, the alarm shown is the 211th transaction to occur. (The transaction code is initially set to 0 when the SAGE^{MAX} leaves the factory, and wraps to 9999 before repeating.) Also, the scan alarm is from a unit on port number 3. By selecting the first position of the transaction code column in the alarm selection template, you can display all unacknowledged alarms from a specific port.

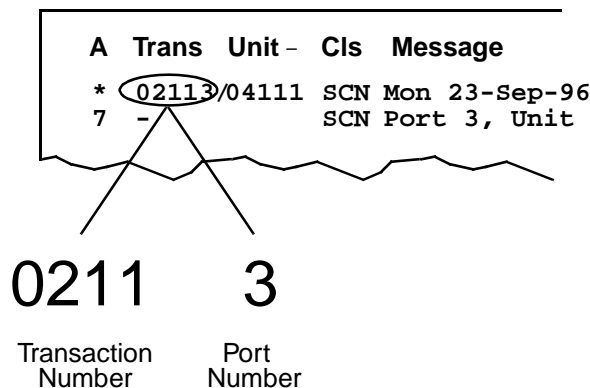


Figure 8-7 Components of the Transaction Code

The **Unit** column of the alarm list shows the unit number of the device associated with the alarm. The example in **Figure 8-7** shows that the alarm refers to unit number 4111. You can use the **Unit** column in the alarm selection template to display all unacknowledged alarms from a specific unit number.

The **Cls** column of the alarm list shows the alarm class of the transaction. This column may contain the class number or the class name that you may optionally assign. Use the **Cls** column in the alarm selection template to display all unacknowledged alarms of a particular class.

The **Message** column of the alarm list contains text that varies based on the source of the transaction.

If you create a match pattern in the message field of the alarm selection wild card template, only those lines of text that match exactly will be displayed. If the matching text is not from the first line of the alarm (which usually contains the date, day of the week and time of day when the alarm occurred), the displayed alarm list may not be very useful.

NOTE

*The alarm file **GENERAL.LOG** is an ASCII text file. Alarms in this file (or any .LOG file) may consist of multiple lines of text. When using match patterns, the SAGE^{MAX} treats this file as a series of single lines of text -- not a series of groups of alarms.*

8.3.2 All Alarms

The **All Alarms (A)** option of the **Alarm List** Submenu allows you to view all or a select group of alarms from the alarm list. Alarms will be displayed in groups of ten at a time. After choosing this option, you are presented with the Alarm Selection Wild Card Template Screen shown in **Figure 8-4**. If there are no alarms in the alarm list, the **Alarm List** Submenu is simply reprinted.

From this screen, you can selectively choose transaction, unit, class and message text information as a match pattern for the alarms that you want to view. You use the left and right arrow keys to maneuver through the match pattern. When the cursor is moved to the desired match position(s), type the text match pattern you want. After you enter the match pattern, all matching alarms will be displayed in groups of ten at a time. Refer to **Figure 8-5**.

Question marks in the match pattern are used as wild card characters as described in **Chapter 8.3.1: Unacknowledged Alarms (U)**.

8.3.3 Alarms by Class Name

The **Alarms by Class Name (C)** option of the **Alarm List** Submenu allows you selectively view all alarms of a particular class from the alarm list. After choosing this option, you are presented with the Alarm Selection Template Screen shown in **Figure 8-4**.

From this screen, you can selectively choose transaction, unit, class and message text information as a match pattern for the alarms that you want to view. You use the left and right arrow keys to maneuver through the match pattern. When the cursor is moved to the desired match position(s), type the text match pattern you want. After you enter the match pattern, all matching alarms will be displayed in groups of ten at a time as shown in **Figure 8-5**.

Question marks in the match pattern are used as wild card characters as described in **Chapter 8.3.1: Unacknowledged Alarms**.

8.3.4 Archive Alarms

The **Archive Alarms (R)** option of the **Alarm List** Submenu allows you save all alarms or alarms of a particular class to the hard disk or a floppy disk by specifying a full pathname.

After choosing this option, you are presented with a prompt to enter a class name. If you only want to archive a single class of alarms, enter the class name. If you want to archive the general alarm file, type **GENERAL**. If you want to archive all alarms (the default), just type the enter key.

After entering the alarm class(es) of the alarms you want to archive, you are prompted to enter a drive and directory for the archive file. You can save archive files to the hard disk (the **C:** drive) or a floppy disk (the **A:** drive). **Figure 8-8** shows an example of archiving **FIR** alarms to the filename **FIR_ALMS.TXT** on the **ARCHIVES** directory of a diskette in drive **A:**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Alarm List:
key          to do
U           Unacknowledged Alarms
A           All Alarms
C           Alarms by Class Name
R           Archive Alarms
D           Delete Alarms
PF1         Return to Previous Menu
Press key for desired action:R

Type class name or 'GENERAL' (default=all):FIR
Enter destination [drive:]\pathname:A:\ARCHIVES\FIR_ALMS.TXT

Copying File \LOG\FIR.LOG
to A:\ARCHIVES\FIR_ALMS.TXT

```

Figure 8-8 Sample Archive Alarms Screen

If you specify an alarm file that does not exist, the error message:

#[21]Errorreadingsourcefile

is displayed when you enter the destination pathname. Pressing any key at this error message causes the **Alarm List** Submenu to be displayed.

IMPORTANT

After you archive a file, the source file is deleted.

8.3.5 Delete Alarms

The **Delete Alarms (D)** option of the **Alarm List** Submenu allows you delete the general alarm file or class alarm files from the **LOG** subdirectory of the hard disk.

After choosing this option, you are presented with a prompt to enter a class name. If you only want to delete a single class of alarms, enter the class name. If you want to delete the general alarm file, type **GENERAL**. If you want to delete all alarms (the default), just type the enter key.

After you choose the alarm files that you want to delete, you are prompted to confirm the deletion by typing **Y** for yes and **N** for no. Typing **N** causes the **Alarm List** Submenu to be displayed. Typing **Y** causes the specified alarm file to be deleted from the hard disk. If you try to delete an alarm file that does not exist, the error message

#[21]FileNotFound

is displayed. Pressing any key at this error message causes the **Alarm List** Submenu to be displayed.

8.4 The Broadcast Message Screen

The **Broadcast Message (B)** option of the **Main Menu** allows you to send messages to other units that are connected to the SAGE^{MAX}. This is particularly useful when operators are decentralized and need to communicate from remote terminals such as dial-up terminals, for example.

Selecting **Broadcast Message (B)** from the **Main Menu** prompts you with the message **Broadcast [Port/Unit#/Text]:**. From this prompt you enter the following three pieces of information:

- Port number
- Unit number (optional)
- Message text

You must use “/” key to separate the port and unit numbers, and the unit number and message text string. If an optional unit number is not used (e.g., broadcasting messages to a local CRT port), type two “/” characters between the port number and message text strings.

Figure 8-9 shows an example of a conversation between the two CRT ports using the Broadcast Message feature of the SAGE^{MAX}.

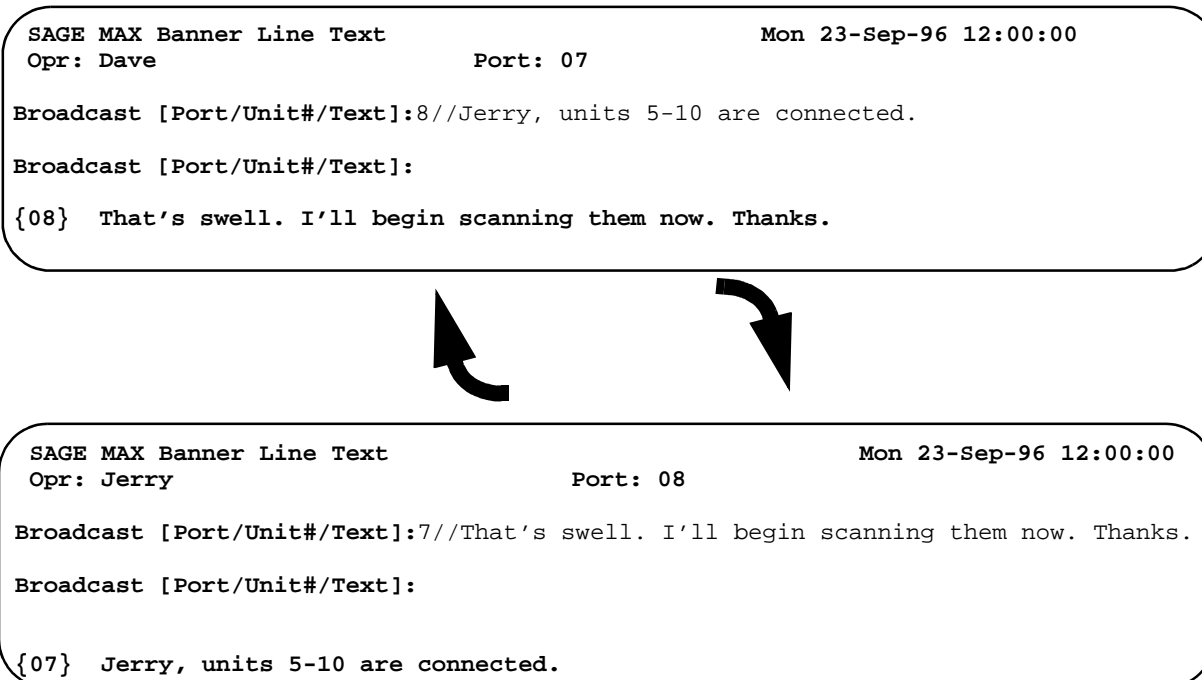


Figure 8-9 Sample Broadcast Message Screens

The port number that you enter must be a valid port number in the range of 1-12.

Received messages begin with the number of the sender's port enclosed in braces, i.e.,
{07} Message Text.

8.5 The Calendar Edit Submenu

The internal SAGE^{MAX} calendar (**Figure 8-10**) allows you to specify days of the year that are holidays and to specify a programmable *mode name* for every 5-minute increment of a day. This information is stored in a *daytype file* which can be assigned to any day of any year. On that day, the **\$HOLIDAY** and **\$MODE** variables of the SAGE^{MAX} will reflect your settings. For that day, the **\$MODE** variable may change 288 times (there are 288 5-minute increments in a day). The **\$HOLIDAY** and **\$MODE** variables are accessible through SPL programs and standard operator interfaces.

The two calendar functions of **Edit Calendar for Some Year (Y)** and **Edit Daytype Mode Schedule (D)** are the options available in the **Calendar Edit** Submenu shown in **Figure 8-11**.

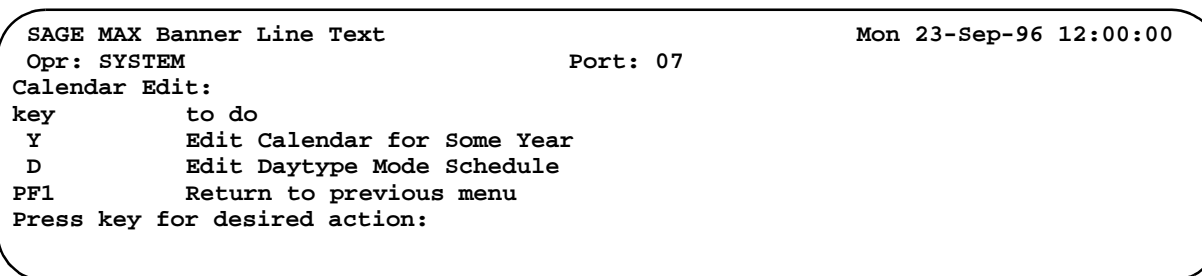


Figure 8-10 Calendar Edit Submenu

8.5.1 Edit Calendar for Some Year

The **Edit Calendar for Some Year (Y)** option allows you to specify a 3-character month (e.g., Jan, Feb, etc.), a day of the month (0-31) and an 8-character day type for every day of a given year (i.e., there are 366 entries possible). Type represents the day type filename that specifies up to 288 modes for every 5-minute increments of the day. The day file also specifies if the day is a holiday.

Figure 8-11 shows a sample screen using the **Edit Calendar for Some Year** option. You use the left, right, up and down arrow keys to maneuver through the fields and days of the year. You must press **ENTER** if you make changes to the day type.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Calendar Edit:
key          to do
Y           Edit Calendar for Some Year
D           Edit Daytype Mode Schedule
PF1        Return to previous menu
Press key for desired action: Y

Year to Edit (1991):1991
mon day type
Jan 01  HOLIDAY
Jan 02
Jan 03
Jan 04  WEEKEND
Jan 05  WEEKEND
Jan 06
Jan 07
Jan 08
Jan 09

```

Figure 8-11 Example Display of the Edit Calendar for Some Year Option

8.5.2 Edit Daytype Mode Schedule

The **Edit Daytype Mode Schedule (D)** option allows you to define a daytype file. Each daytype file has an associated 8-character (maximum) name (default is **WEEKDAY**) and specifies up to 288 modes for every 5-minute increments of the day. The dayfile also specifies if the day is a holiday (**Yes/No**).

Dayfiles have the extension **.DAY** and are located on the **C:\CFG** subdirectory of the SAGE^{MAX}.

Figure 8-12 shows a sample screen using the **Edit Daytype Mode Schedule (D)** option. You use the left, right, up and down arrow keys to maneuver through the fields and time increments. You must press **ENTER** if you make changes to the mode field.

When you edit the daytype mode schedule, every 5-minute increment defaults to **MODE000**. This is the default mode name for mode 0. Default mode names range from **MODE000** to **MODE255**. When you edit the daytype mode schedule, you can assign these mode names, or you can assign mode names that have been previously customized.

Individual mode names can be customized by creating and/or editing the ASCII text file

C:\CFG\MODENAME.TXT by using the SAGE^{MAX} text editor. In this file, you can assign customized, 24-character (maximum) mode names (e.g., **WARMUP**, **OCCUPIED**, **UNOCC**, **COOLDOWN**, **OPTSTART**, **OPTSTOP**, etc.) to any of the 256 modes, rather than using the default mode names **MODE000** to **MODE255**.

The text format of the **MODENAME.TXT** file is the numeric mode number (0-255) followed by a comma and the 24-character (maximum) mode name. If a customized mode name has not been assigned to a particular mode number in the **MODENAME.TXT** file, you can still use the default mode name **MODEnnn** (where *nnn* represents a number from **000** to **255**) when editing the daytype mode schedule. Refer to **Figure 8-12**.

Figure 8-12 also shows an example of the **MODENAME.TXT** file being edited from the SAGE^{MAX} text editor.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Calendar Edit:
key          to do
Y            Edit Calendar for Some Year
D            Edit Daytype Mode Schedule
PF1         Return to previous menu
Press key for desired action: D

Daytype to Edit (WEEKDAY):
Is this a Holiday? (No ):
time      mode
00:00    MODE000
00:05    MODE000
00:10    COOLDOWN
00:15    MODE000
00:20    MODE000
00:25    MODE000
00:30    MODE002

```

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Edit v3 Insert   \CFG\MODENAME                       .TXT
1,COOLDOWN
15,OCCUPIED
16,UNOCC
17,SETUP

Press PF2 or HELP for editor Help

```

Figure 8-12 Sample Display of the Edit Daytime Mode Schedule Option (C:\CFG\MODENAME.TXT)

8.6 The Database Functions Submenu

The **Database Functions (D)** Submenu allows you to create, modify, erase and list database objects (e.g., points, groups, programs, classes, variables and globals). In addition, this submenu allows you to move database points between ports and/or units on SAGE^{MAX} networks. The **Database Functions** Submenu is shown in **Figure 8-13**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Database:
key           to do
C             Create/Modify Objects
E             Erase Objects
L             List Objects
F             List to File
M             Move Database Points
PF1          Return to previous menu
Press key for desired action:
```

Figure 8-13 The Database Functions Submenu

8.6.1 Create/Modify Objects

Selecting **C** from the **Database Functions** Submenu displays the **Create/Modify Objects** Submenu shown in **Figure 8-14**. From this submenu, you can create or modify points, groups, programs, classes, variables and globals.

Points are database items that are used to control and/or monitor equipment and activities of the system.

A group is a collection of up to 16 database objects (points, programs, globals, variables or other groups) that appear together as a menu.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Create/Modify:
key           to do
P             Points
G             Groups
R             Programs
C             Classes
V             Variables
X             Globals
U             Users
PF1          Return to previous menu
Press key for desired action:
```

Figure 8-14 The Create/Modify Objects Submenu

A program is a database object that represents a collection of program objects: a PLB, PRB, INI file, attributes and control registers. The PLB is a binary data file that contains the set of statements or *logic*. The PRB is an optional text file that contains up to 256 program references. The INI file is an optional file that is used to initialize program attributes to non-zero values. These files along with the attributes and control registers make up a program database object.

A class is a three-character name that is used in routing, classifying and sorting alarms. Classes specify where alarms are to be logged, if alarm acknowledgment is required, the privileges required to acknowledge the alarm, ports and/or units to report the alarm to, times and days-of-the-week to report the alarm, dialing options and phone numbers. You can define up to 256 alarm classes on the SAGE^{MAX}.

A variable is a named object that has an associated value that can be changed through operator control or through a program.

A global is a database object that references a database name that resides on another SAGE^{MAX}. The term *peername* refers to the name of the other SAGE^{MAX}. The *euname* is the name of an engineering units override file.

8.6.1.1 Create Modify Points

Selecting **P** from the **Create/Modify Objects** Submenu displays the **Point Name:** prompt. You then enter the name of the point that you wish to create or edit.

If you enter a point name that does not exist in the database, the SAGE^{MAX} prompts you with **Create New Database Object? (Y/N):**. Typing **Y** creates a new database item which you are asked to define through a series of prompts shown in **Figure 8-15**. Typing **N** causes the **Point Name:** prompt to be redisplayed.

Some information shown in **Figure 8-15** is followed by a default or current value. On the next line, the same piece of information is duplicated. From this prompt you may edit the information or type **RETURN** to move on to the next prompt.

NOTE

The Fundamental Type, Subchannel and Card information is not shown if the specified Port is configured as a PUP network. This information is not used by the PUP network in this form.

Port (1-8) refers to the eight SAGE^{MAX} ports that have network capabilities. Ports 1-4 are the four EIA-485 network ports. Ports 5 and 6 are the two serial ports, COM1 (modem) and COM2 (for optional leased line applications), which can potentially be required to perform network functions. Ports 7 and 8 are the local CRT ports which have limited network capabilities. At this prompt, you define the network port on which the point is located.

For example, if you want to define a SOLO/MX point that is located on EIA-485 network 3, a PUP network port, set this variable equal to 3.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07

Port          (1-8)= 0
Port          (1-8)=
Unit   (0-65535)= 0
Unit   (0-65535)=
Channel (0-FFFFH)= 0000H
Channel (0-FFFFH)=
Fundamental Type= SY
Fundamental Type=
Subchannel (0-15)= 0
Subchannel (0-15)=
Card          (0-15)= 0
Card          (0-15)=
EU Override File=
EU Override File=
Famous Name ?= No
Famous Name ?=
Override Class = 0
Override Class =
Description =
Description =
Alarm Message=
Alarm Message=
Return Message=
Return Message=

```

First Line of Each Pair Shows the Default or Current Value (May Be Blank)

After The Default or Current Value, a Second Line Is Displayed for Editing

Figure 8-15 The Create/Modify Point Screen

Unit (0-65535) refers to the unit number (on the specified port) of the device on which the point is located. For example, if you want to define a point on a STAR that is located on network 1, a STAR Peernet network, set this variable equal to the STAR's unit identification number (limited to 0-31 in the case of the STAR).

Similarly, if you want to define a SOLO/MX point on a PUP network, set this variable equal to the unit number of the SOLO/MX (0-65,535).

Channel (0-FFFFH) refers to the channel number of the point you want to create. For example, if you want to define a point on a STAR, set this variable equal to the channel number of the point (0-128 depending on the type of I/O Module used).

For some networks, as in the example above, it may be easier to use the decimal form of the channel number, although binary format (e.g., 10101111b) and hexadecimal format (e.g., 7Fh) are supported.

If, for example, you want to define a SOLO/MX point on a PUP network, set this variable equal to the PUP channel number of the point (0-FFFFH).

In cases involving PUP networks, it is usually easier to enter the channel number in hexadecimal form, although several forms (e.g., decimal, binary, hexadecimal) are supported.

Fundamental Type refers to the uppercase, two-letter code that identifies a point's type (e.g., AI for analog input, BO for binary output and TT for terminal temperature).

If you want to define a system point on a STAR that is on a Peernet network, for example, set this variable equal to the two-letter fundamental type SY.

On PUP networks, this “type” information is defined through the PUP channel number, making fundamental type information unnecessary for these networks. This information is not shown on Create/Modify Point screens when dealing with PUP networks.

Subchannel (0-15) refers to the subchannel of a point you want to create. If you want to define system point SY0\$1 of a GPUP/C1 Module on a STAR that is on a Peernet network, for example, set this variable equal to 1. This defines the point on subchannel 1. SY0\$2 would equal 2, etc.

On PUP networks, subchannel information can be directly incorporated into the PUP channel number, making subchannel information unnecessary for these networks. This information is not shown on Create/Modify Point screens when dealing with PUP networks.

Card (0-255) refers to the card number associated with the point you want to create.

If you want to define a binary output point of an MDDC/A2 I/O Module on a STAR that is on a Peernet network, set this variable equal to the card slot number of the MDDC/A2 I/O Module (slot 1-7).

If you want to define a program point from the same STAR on the SAGE^{MAX}, you must first determine the *card number* associated with the program point. Since STAR program points are *pseudopoints*, there is no physical card association. The card number is the *bank number* in memory where the STAR program resides. For this reason, the card number variable of the SAGE^{MAX} has a range well above seven.

On PUP networks, card information is unnecessary and is not shown on **Create/Modify Points Screens**.

EU Override File refers to the optional engineering units file that you may assign to this point object.

The engineering units file is a text file that contains engineering units text which is appended to point, program, and global attributes when they are monitored.

For information on creating engineering units files, refer to **Chapter 7: Initial Configuration**.

Famous Name allows you to specify if you want the point to be famous.

A famous point is one which is created automatically as a global on all SAGES^{MAX} on the Ethernet network. If you do not want this point to be *famous* on all SAGES^{MAX} on the Ethernet, or if the point exists on a SAGE^{MAX} that does not (and will not) have an Ethernet network, then **Famous Name** should be set to **No**.

For information on famous points, refer to **Chapter 7: Initial Configuration**.

Override Class allows you to specify a SAGE^{MAX} alarm class for this point. When alarms

occur from this point (e.g., alarms from a PUP device), the override class is used rather than the class determined by the driver.

The determination of alarm class varies based on the driver type of the network. For more information on how a driver determines alarm classes, refer to the individual driver section.

Description is an optional text field that is used to describe the point. This description is displayed when the point is monitored.

Alarm Message is an optional text field that is displayed as part of the alarm text when an alarm from this point occurs.

Return Message is an optional text field that is displayed as part of the alarm text when a return to normal occurs for this point. This message is only used by points that can generate alarms *and* returns (i.e., high limit alarm and high limit return).

8.6.1.2 Create/Modify Groups

Selecting **G** from the **Create/Modify Objects** Submenu displays the **Group Name:** prompt. You then enter the name of the group that you wish to create or edit. **Figure 8-16** shows a blank **Create/Edit Group** screen.

If you enter a group name that does not exist in the database, the SAGE^{MAX} prompts you with **Create New Database Object? (Y/N):**. Typing **Y** creates a new group in the database. Typing **N** causes the **Group Name:** prompt to be redisplayed.

```
SAGE MAX  Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Group Name: SAMPLE
key          to modify
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
X           Erase a Member
Y           Change Group Name
Z           Change Group Description
PF1        Return to Previous Menu
```

Figure 8-16 Sample Create/Edit Group Screen 1

Once you have entered an existing group name (or confirmed the creation of a new group in the database), the **Create/Edit Group** Screen is displayed. Refer to **Figure 8-16**.

The letters **A-P** represent elements in the group. To create/edit elements in the group, type the letter corresponding to the element that you wish to create/edit.

The elements of a group can be points, programs, variables and globals. In addition, the elements of groups can be other groups. When you create/edit an element (**A-P**), you must tell the SAGE^{MAX} if the element is to be a reference to another group.

If the group element is another group, you enter the name of the group. You are prompted to enter a new name for the group, if you like, and a new description for the group.

If the group element is a point, program, variable or global, you enter the corresponding name. You are then prompted to enter the desired attribute and a new description for the group element.

Type **X** if you want to erase a member of a group without replacing that element with another member. Type **Y** if you want to change the name of the group. Type **Z** if you want to change the description of the group.

Type **PF1** if you want to return to the previous menu. Before the Create/Modify Menu is displayed, you are asked if you want to save the group. Type **Y** to save the changes you made. Type **N** to cancel any modifications that you made to the group.

Figure 8-17 shows a sample group being created from the **Create/Edit Group** Screen of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Group Name: SAMPLE                                   Group Description Text Appears Here...
key          to modify
A            AHU1 Ctl                               Statistics of AHU#1 Flows/Setpoints
B            Temps                                 Zone/Duct Temperatures
C            BO Group                              Binary Outputs
D
E            GPUP2;U1                               Unit Map of GPUP/C1 #2 (0-31)
F            GPUP2;U2                               Unit Map of GPUP/C1 #2 (32-63)
G
H            OSS1;$$                                Status of OSS Program Number 1
I            OSS1;$D                               OSS Seconds Remaining in Time Delay
J
K            OSS_MAX_PREHEAT_TIME                 Variables for OSS program
L            OSS_MAX_PRECOOL_TIME
M
N            OAT;CV                                Outside Air Temp from SAGE#2 (Global)
O            OA_RH;CV                              Outdoor Air Relative Humidity (Global)
P
X            Erase a Member
Y            Change Group Name
Z            Change Group Description
PF1         Return to Previous Menu
Save this group? (YN): Y

```

Figure 8-17 Sample Create/Edit Group Screen 2

When you are finished creating the group, SAGE^{MAX} prompts you with **Save this group? (Y/N)**:. After your response, SAGE^{MAX} returns you to the **Create/Modify Objects** Submenu shown in **Figure 8-18**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                             Port: 07
Create/Modify:
key             to do
P               Points
G               Groups
R               Programs
C               Classes
V               Variables
X               Globals
U               Users
PF1            Return to previous menu
Press key for desired action:
```

Figure 8-18 The Create/Modify Objects Submenu

8.6.1.3 Create/Modify Programs

Figure 8-19 shows the **Create/Modify Program** Submenu. This submenu contains eight options. The **Edit/Compile (E)** option offers a submenu that permits you to edit the program logic source code, program references, initial values, and the program list file. In addition, this menu contains a selection to compile the program logic source code. Refer to **Figure 8-20**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                             Port: 07
Program SAMPLE
Program:
key             to do
E               Edit/Compile
N               Change Program Name
L               Program Logic Block
R               Program Reference Block
I               Program Initial Values
P               Program Object Privileges
U               Engineering Units File
O               Program Options
PF1            Return to previous menu
Press key for desired action:
```

Figure 8-19 The Create/Modify Programs Submenu

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program SAMPLE
Program Edit/Compile:
key          to do
S            Edit Program Logic Source
R            Edit Program References
I            Edit Program Initial Values
L            Edit Program List File
C            Compile Program Logic Source
PF1         Return to previous menu
Press key for desired action:

```

Figure 8-20 The Program Edit/Compile Submenu

From the **Edit/Compile** Submenu (shown in **Figure 8-20**), you can choose the **Edit Program Logic Source (S)** option. When selected, you are prompted to enter the program source filename. The program object name that you created is the default name used for the logic source code. This name is displayed at the prompt. You can use the default name by typing the **RETURN** key, or edit the source name using the arrow, backspace or delete keys. After you type the **RETURN** key, the SAGE^{MAX} editor is displayed with the first page of any text that is contained in the file you specified. **Figure 8-21** shows the SAGE^{MAX} text editor as it appears when you first create a new logic source file. Pressing **PF1** (a prefix key which precedes most editor commands) causes a command line to be displayed. Each option in this command line contains a capitalized letter which you must type for that function to occur. The SAGE^{MAX} text editor with command line is shown in **Figure 8-22**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Edit v3 Insert          \SPL\SAMPLE          .SPL

Press PF2 or HELP for editor Help

```

Figure 8-21 The SAGE^{MAX} Text Editor

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Bot Copy Dn End Find Home Ins Kill deL Move Next Quit maRk Save Top Up eXit Zap

```

Figure 8-22 The SAGE^{MAX} Text Editor with Command Line

Figure 8-23 shows the SAGE^{MAX} Text Editor Help Screen which is displayed by typing the **PF2** key or the **HELP** key. Press any key to return to the Text Editor Screen.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Edit v3 Insert   \SPL\SAMPLE                           .SPL
Most EDIT commands are preceded by PF1:
  B bottom of file                                     M move marked block to cursor
  C copy marked block to cursor                       N find/change next
  D page down                                         Q quit without saving
  E end of line                                       R set marker at cursor
  F find/change first                                 S save file
  G goto line number                                 T top of file
  H home (beginning of line)                         U page up
  I toggle insert mode                               V invert case of marked block
  K delete (kill) current line                       X save and exit
  L delete to end of line                             Z delete marked block

Use cursor keys to move up, down, left, right
RUBOUT or DELETE key deletes left, Ctrl-G or REMOVE deletes right

Find/Change search string is ended with RETURN or ENTER for case-sensitive
or end with Up-Arrow cursor key or Ctrl-K for case-insensitive searches
When Changing, separate find string from change string with PF1:
      PF1 F findstring PF1 changestring ENTER

```

Figure 8-23 The SAGE^{MAX} Text Editor Help Screen

From the **Edit/Compile** Submenu (shown in **Figure 8-20**), you can choose the **Edit Program References (R)** option. When selected, you are prompted to enter the program reference filename. Note that a default name is not specified for the program reference file. You enter the name of the reference file without the extension (it is assumed to be **.PRB**) and without a path (it is assumed to be on the **C:\REF** subdirectory). After you type the **RETURN** key, the SAGE^{MAX} editor is displayed with the first page of any text that is contained in the file you specified.

From the **Edit/Compile** Submenu (shown in **Figure 8-20**), you can choose the **Edit Program Initial Values (I)** option. When selected, you are prompted to enter the program initial value filename. This prompt is displayed only once when the initial values file is first assigned to this program object. Note that a default name is not specified for the program initial values file. You enter the name of the initial values file without the extension (it is assumed to be **.INI**) and without a path (it is assumed to be on the **C:\INIT** subdirectory). After an initial values file has been assigned, SAGE^{MAX} prompts you for a series of attributes and associated initial values. This information is either saved to newly created files or is appended to existing files. **Figure 8-24** shows the SAGE^{MAX} prompting for attribute names and initial values.

From the **Edit/Compile** Submenu (shown in **Figure 8-20**), you can choose the **Edit Program List File (L)** option. When selected, you are prompted to enter the program list filename. The program object name that you created is the default name used for the logic source code. This name is displayed at the prompt. You can use the default name by typing the **RETURN** key, or edit the list filename using the arrow, backspace or delete keys. The list filename you enter must not contain an extension (it is assumed to be **.LST**) and must not contain a path (it is assumed to be on the **C:\SPL** subdirectory).


```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program SAMPLE
Program Edit/Compile:
key          to do
S           Edit Program Logic Source
R           Edit Program References
I           Edit Program Initial Values
L           Edit Program List File
C           Compile Program Logic Source
PF1         Return to previous menu
Press key for desired action: I
Enter program initial values filename: INIFILE
attr: CV
CV=123
```

Figure 8-24 The SAGE^{MAX} Prompting for Attributes & Initial Values from the Edit Program Initial Values Option

After you type the **RETURN** key, the SAGE^{MAX} editor is displayed with the first page of any text that is contained in the file you specified. Pressing **PF1** (a prefix key which precedes most editor commands) causes a command line to be displayed. Each option in this command line contain a capitalized letter which you must type for that function to occur. The SAGE^{MAX} text editor with command line is shown in **Figure 8-22**.

From the **Edit/Compile** Submenu (shown in **Figure 8-20**), you can choose the **Compile Program Logic Source (C)** option. When selected, you are prompted to enter the program list filename. The program object name that you created is the default name used for the logic source code. This name is displayed at the prompt. You can use the default name by typing the **RETURN** key, or edit the list filename using the arrow, backspace or delete keys. The list filename you enter must not contain an extension (it is assumed to be **.SPL**) and must not contain a path (it is assumed to be on the **C:\SPL** subdirectory).

After you type the **RETURN** key, the SAGE^{MAX} compiles the program source code you specified and creates a program logic block (**.PLB**) file on the **C:\LOGIC** subdirectory. Compile information is displayed on the screen as shown in **Figure 8-4**. Any errors that may occur during the compile process are reflected in this information. In the event that your program contains errors, it is necessary to edit the program logic source code, correct the errors, and recompile the source code.

The **Compile Program Logic Source (C)** is the last option of the **Program Edit/Compile** Submenu.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program SAMPLE
Program Edit/Compile:
key          to do
S            Edit Program Logic Source
R            Edit Program References
I            Edit Program Initial Values
L            Edit Program List File
C            Compile Program Logic Source
PF1         Return to previous menu
Press key for desired action: C

**** SPL Compiler v1.00 (c) American Auto-Matrix 1991 ****

          Source: \SPL\SAMPLE.SPL
          Logic:  \LOGIC\SAMPLE.PLB

**** SPL Compile Complete ****

[00015] Lines   [00274] Bytes   [00000] Symbols   [00:00:00.43] Compile Time
[00000] Syntax Errors [00000] Undefined Symbols [00000] Multiple Symbols

...Press any key to continue:

```

Figure 8-25 Sample Display after Compiling a Program's Logic Source Code

The next option in the **Create/Modify Program** Submenu is the **Change Program Name (N)** option. This option is used to rename a program object. This procedure is illustrated in **Figure 8-26**.

After you enter the new program name, the **Create/Modify Program** Submenu is displayed with the new program name in the **Program** field.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program OLDNAME
Program:
key          to do
E            Edit/Compile
N            Change Program Name
L            Program Logic Block
R            Program Reference Block
I            Program Initial Values
P            Program Object Privileges
U            Engineering Units File
O            Program Options
PF1         Return to previous menu
Press key for desired action: N
Enter program name: NEWNAME

```

Figure 8-26 Example of Changing a Program Name from the Create/Edit Program Submenu

The next option in the **Create/Modify Program** Submenu is the **Program Logic Block (L)** option. This option is used to change the name of the program logic block that is assigned to the program object.

The next option in the **Create/Modify Program** Submenu is the **Program Reference Block (R)** option. This option is used to change the name of the program reference block that is assigned to the program object.

The next option in the **Create/Modify Program** Submenu is the **Program Initial Values (I)** option. This option is used to change the name of the program initial values file that is assigned to the program object.

The next option in the **Create/Modify Program** Submenu is the **Program Object Privileges (P)** option. This option is used to change the object privilege bitmap associated with the program object.

The next option in the **Create/Modify Program** Submenu is the **Engineering Units File (U)** option. This option is used to change the engineering units file that is associated with the program object.

The next option in the **Create/Modify Program** Submenu is the **Program Options (O)** Submenu. You use this submenu to make the program famous, toggle the Log Modifications flag, define the preload option, set the program logic option, and set the program reference option. The **Program Options (O)** Submenu is shown in **Figure 8-27**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program NEWNAME
Program Options:
key          to do
F           Famous Option
L           Log Object Changes
P           Preload Option
G           Set Program Logic Option
R           Set Program Reference Option
PF1        Return to previous menu
Press key for desired action:

```

Figure 8-27 The Program Options Submenu

The **Famous Option (F)** can be set to either **Y** (Yes) or **N** (No). If **Y**, the program is *famous*. This means that global objects are created automatically on all SAGES^{MAX} on the Ethernet. The global objects have the same name as the program object and give users on other SAGES^{MAX} (on the Ethernet) access to the programs attributes. If this option is set to **N**, the program is not famous and global objects are not created. Refer to **Figure 8-28**.

The **Log Object Changes (L)** option can be set to either **Y** (Yes) or **N** (No). If **Y**, any changes made to the program's attributes through an operator interface cause the event to be logged according to SAGE^{MAX} class 002 (the OPR class). If this option is set to **N**, changes to program attributes are not logged. Refer to **Figure 8-29**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program NEWNAME
Program Options:
key          to do
F           Famous Option
L           Log Object Changes
P           Preload Option
G           Set Program Logic Option
R           Set Program Reference Option
PF1        Return to previous menu
Press key for desired action: F
Program is private.
Famous? (Y/N):

```

Figure 8-28 Program Options Submenu - Famous Option

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program NEWNAME
Program Options:
key          to do
F           Famous Option
L           Log Object Changes
P           Preload Option
G           Set Program Logic Option
R           Set Program Reference Option
PF1        Return to previous menu
Press key for desired action: L
Program Object Changes are NOT logged.
Log Program Object Changes? (Y/N):

```

Figure 8-29 Program Options Submenu - Log Object Changes Option

The **Preload Option (P)** is used to define how the program's logic block (PLB) is loaded into memory. If you set this option to **0**, the PLB is loaded on demand (e.g., from the **Monitor/Change Point Attributes** Menu, from an SPL program, etc.). If you set this option to **1**, the PLB is loaded into memory at bootstrap (i.e., when the SAGE^{MAX} is turned on or reset). Refer to **Figure 8-30**.

The **Set Program Logic Option (G)** is used to define how the PLB is treated after it is loaded into memory and after the program requests unloading. If the Logic Option is *not sticky* (**0**), the program does not "stick in RAM" and is removed from memory when the program requests unloading. If the Logic Option is *sticky* (**1**), the program remains in memory once it is loaded. Refer to **Figure 8-31**.

The **Set Program Reference Option (R)** allows you to define the RAM-resident nature of a Program Reference Block (PRB). PRBs can be always file resident (**0**), RAM-resident, but able to be unloaded (**1**), or not unloadable (sticky) once loaded into RAM (**2**). Refer to **Figure 8-32**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program NEWNAME
Program Options:
key          to do
F           Famous Option
L           Log Object Changes
P           Preload Option
G           Set Program Logic Option
R           Set Program Reference Option
PF1        Return to previous menu
Press key for desired action: P
Program is loaded at bootstrap.
Preload option (0=load on demand,1=loaded at bootstrap):

```

Figure 8-30 Program Options Submenu - Preload Option

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program NEWNAME
Program Options:
key          to do
F           Famous Option
L           Log Object Changes
P           Preload Option
G           Set Program Logic Option
R           Set Program Reference Option
PF1        Return to previous menu
Press key for desired action: G
Logic is not sticky.
Logic Option (0=not sticky,1=sticky):

```

Figure 8-31 Program Options Submenu - Set Program Logic Option

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Program NEWNAME
Program Options:
key          to do
F           Famous Option
L           Log Object Changes
P           Preload Option
G           Set Program Logic Option
R           Set Program Reference Option
PF1        Return to previous menu
Press key for desired action: R
References are file resident.
Reference Option (0=file res,1=not sticky,2=sticky):

```

Figure 8-32 Program Options Submenu - Set Program Reference Option

Each alarm class has the option of logging its alarms to the general alarm log file (**C:\ALARMS\GENERAL.LOG**) and/or its associated class alarm log file (**C:\ALARMS\027.LOG** or **C:\ALARMS\FIR.LOG** if you assigned the class name **FIR**). For each alarm class you must decide if you want the alarms to be logged to either (or both) of these text files.

You can configure each class so that alarms of that class are not acknowledged automatically. If acknowledgment is required for alarms of certain classes, you must decide which operators will have the proper privilege levels to do the acknowledgment. The privilege code that is used for acknowledgment of alarms is the 8-bit Acknowledgment Privilege Pattern associated with each class.

Alarm priorities are assignable by class. Each alarm class has an associated priority from 0-255. Large numbers dictate a higher priority in servicing the alarm. The highest priority is level 255 and the lowest is level 0.

Alarms of each class can be reported to as many as 7 SAGE^{MAX} ports. Each port destination includes the port number where you want alarms to be sent (0-14), the desired unit number scheduled to receive the alarms, and the days of the week and time interval when alarms should be sent to the port/unit.

Classes also have dialing capabilities. Like port destinations, classes can specify 6 dial destinations. Each dial destination includes the dial type (e.g., ring only, printer and automatic), the dial out baud rate, the dial string (up to 40 characters) for the destination, and the days of the week and time interval when alarms should be dialed out. Additionally, each class specifies whether the first active dial destination should be used or if all active dial destinations should be used.

For more information about classes, refer to **Chapter 10: Alarm and Event Management**.

8.6.1.5 *Create/Modify Variables*

Selecting **V** from the **Create/Modify Objects** Submenu (**Figure 8-14**) displays the **Variable Name:** prompt. From this prompt, you enter the name of a variable you want to create or edit. If the name does not already exist in the SAGE^{MAX} database, you are asked if you want to create a new database object. If you respond with **Y** (Yes), the variable name is created.

After you create a variable name, SAGE^{MAX} prompts you for a **Data Type (0-FFh)**. This is the data type that you assign to the variable. SAGE^{MAX} displays a list of the available data types to aid you in selecting the hexadecimal code. Refer to **Figure 8-34**.

You are then prompted to enter the **Variable Value**. This value follows the data type that you specified.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Variable Name: HUMAN
Create New Database Object? (Y/N) Y
Data Type (0-FFh)=00H
value      means      value      means      value      means
00        hex byte    E7        Time HH:MM:SS F4        xxxxx.xxxxx
01        hex word    E8        Bitmap      F5        +xxxxx.xxxxx
02        hex dword   EA        .xxxxxxxxxxx F6        xxxxxx.xxxx
03        segment:offset EB      +.xxxxxxxxxxx F7        +xxxxxx.xxxx
04        DOSTime     EC        x.xxxxxxxxxxx F8        xxxxxxxx.xxx
05        day of week  ED      +x.xxxxxxxxxxx F9        +xxxxxxxx.xxxx
07        no/yes      EE      xx.xxxxxxxxxxx FA        xxxxxxxx.xx
7F        bogus value  EF      +xxx.xxxxxxxxx FB        +xxxxxxxx.xx
E0        float       F0      xxx.xxxxxxxxx FC        xxxxxxxxx.x
E3/E4    Hex/BCD Date  F1      +xxx.xxxxxxxxx FD        +xxxxxxxx.x
E5        BCD         F2      xxx.xxxxxxxxx FE        xxxxxxxxx.
E6        Time HH:MM  F3      +xxxx.xxxxxxx FF        +xxxxxxxxxxx.

Data Type (0-FFh)= FCh
Variable Value = .0
Variable Value = 98.6
Object Privs = 00000000B
Object Privs = 11111111B
Log Changes = No
Log Changes =
Famous Name = No
Famous Name =

```

Figure 8-34 The Create/Modify Variables Screen

Next, you are prompted to enter an **Object Privilege Pattern**. This pattern is used in conjunction with a User Object Privilege Pattern to determine if a user is permitted to modify the value of a variable. The ability to modify the value of a variable is determined through a binary operation of the Variable Privilege Pattern and the Privilege Pattern of the user attempting to modify the variable value. If the result of logically ANDing the User and Variable Privilege Patterns is the same as the original Variable Privilege Pattern, the user in question can change the value of the variable.

The **Log Changes** flag can be set to either **Yes** or **No**. If **Yes**, any changes made to the variable through an operator interface cause the event to be logged according to SAGE^{MAX} class 002. If this option is set to **No**, changes to the variable are not logged. Refer to **Figure 8-34**.

The **Famous Name** flag can be set to either **Yes** or **No**. If **Yes**, the variable is *famous*. This means that global objects are created automatically on all SAGES^{MAX} on the Ethernet. The global objects have the same name as the variable object and give users on other SAGES^{MAX} (on the Ethernet) access to the variable. If this option is set to **No**, the variable is not famous and global objects are not created. Refer to **Figure 8-34**.

8.6.1.6 Create/Modify Globals

Selecting **X** from the **Create/Modify Objects** Submenu (**Figure 8-14**) displays the **Global Name:** prompt. From this prompt, you enter the name of a global that you want to create or edit. If the name does not already exist in the SAGE^{MAX} database, you are asked if you want to create a new database object. If you respond with **Y** (Yes), the global name is created. Refer to **Figure 8-35**.

The SAGE^{MAX} prompts you to enter a **Peer Name** for the global object. This is the name of the SAGE^{MAX} where the associated object lives. The Peer Name is found in the Ethernet Configuration Driver Variables.

Next, the SAGE^{MAX} prompts you to enter an **EU Override File**. This is a text file used to supply engineering units to the global object. Refer to **Figure 8-35**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Global Name: OATEMP
Create New Database Object? (Y/N) Y
Peer Name      =
Peer Name      = SAGE A5
EU Override File =
EU Override File = TEMPS

```

Figure 8-35 The Create/Modify Globals Screen

8.6.1.7 Create/Modify Users

Selecting **U** from the **Create/Modify Objects** Submenu (**Figure 8-14**) displays the **User Name:** prompt. From this prompt, you enter the name of a user that you want to create or edit. If the name does not already exist in the SAGE^{MAX} database, you are asked if you want to create a new database object. If you respond with **Y** (Yes), the user name is created. Refer to **Figure 8-36**.

The SAGE^{MAX} prompts you to enter a new **User Name** if you are editing an old user. You are then prompted to enter the **Password** for this user name. These two entries are the user name and password that are used at the **Sign-on** Screen.

Next, the SAGE^{MAX} prompts you to enter a **Menu Privilege Pattern**. This pattern defines what features are available to this user. Different privileges have access to different menus throughout the SAGE^{MAX}. Use the left and right arrow keys to maneuver through the pattern. Type a **1** under user privileges that you want to assign. Type a **0** under user privileges that you do not want to assign.

Next, you are prompted to enter a **User Privilege Pattern**. This pattern is used in conjunction with Object Privilege Patterns to determine if a user is permitted to modify the value of an object. The ability to modify the value of an object is determined through a binary operation of the Object Privilege Pattern and the Privilege Pattern of the user attempting to modify the object value. If the result of logically ANDing the User the and Object Privilege Patterns is the same as the original Object Privilege Pattern, the user in question can change the value of the object.

The SAGE^{MAX} then prompts you to enter a **Timeout** value for this user. This timeout value is a number of seconds between 0 and 65,535 and represents the time that must pass with no keystrokes before this user is signed off. A timeout of 0 means no timeout.

Next, the SAGE^{MAX} prompts you for a **Language Code**. This represents the language that is to be used when this user signs on to the SAGE^{MAX}. Language text is stored in text files on the **C:\EXE** subdirectory. These files have the format **TEXTnnn.TXT**, where **nnn** represents the language code. Language code 1 is English and uses text file **C:\EXE\TEXT001.TXT**. Language code 0 also uses **C:\EXE\TEXT001.TXT**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
User Name: FASTEDDIE
Create New Database Object? (Y/N) Y
User Name: FASTEDDIE
User Name:
Password : FINGERS
Password :

                Menu Privilege Help:
                -----1 = Operator Privs-----1- = Administrator Privs
                -----1-- = File Access (limited)----1--- = File Expert (all)
                ---1---- = User Administrator--1----- = Object Modifier
                -1----- = Scheduler                1----- = Installer
Menu Privs      = -----ISMUXFAO
Menu Privs      = 1111111111111111111B
Object Privs    = 11111111B
Object Privs    =
Timeout (secs) =          0
Timeout (secs) =
Language Code   =          0
Language Code   =

```

Figure 8-36 The Create/Modify Users Screen

8.6.2 Erase Objects

Selecting **E** from the **Database Function** Submenu (Figure 8-13) displays the **Erase Objects** Submenu shown in Figure 8-37. From this submenu, you can remove objects from the SAGE^{MAX} database.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Erase:
key            to do
F              First Name Matched (Point, Program, Variable, Global)
P              Points
G              Groups
R              Programs
V              Variables
X              Globals
U              Users
PF1           Return to previous menu
Press key for desired action:

```

Figure 8-37 The Erase Object Submenu

8.6.2.1 Erase First Name Matched

The **First Name Matched (F)** option allows you to erase a named object from the SAGE^{MAX} database. Selecting this option causes the SAGE^{MAX} to erase the named database object that you specify. To perform this function, the SAGE^{MAX} begins searching its database for the specified object name in the list of point objects. If no matching name is found, the search continues to the list of program names, variable names, and global names. The first object to match is erased from the database. This is of significance if your database uses the same name for multiple objects in the database (e.g., a program called **ZONETEMP**, and a point called **ZONETEMP**). If, however, the object name is unique, it can be removed from the database using this option.

After you specify the name of the database object to be erased, the SAGE^{MAX} searches the database for the object name. If the name is found, the SAGE^{MAX} prompts you to confirm the erasure of the database object with **Erase this Object from the Database? (Y/N)**. This is illustrated in **Figure 8-38**.

If an erase request is made for an object that does not exist in the SAGE^{MAX} database, the **#RequestInvalid** error message is displayed and the SAGE^{MAX} prompts you for another object name to be erased.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Name: SAMPLE
Database search is in progress...
Erase this Object from the Database? (Y/N)

```

Figure 8-38 Example Erase Screen - Erase First Name Matched

8.6.2.2 Erase Points

The **Points (P)** option allows you to erase a point object that you specify from the SAGE^{MAX} database. After you select this option, SAGE^{MAX} prompts you to enter the **Point Name:**.

After you specify the name of the point to be erased, the SAGE^{MAX} searches for the name. If the name is found, the SAGE^{MAX} prompts you to confirm the erasure of the point with **Erase this Object from the Database? (Y/N)**. If an erase request is made for a point that does not exist in the SAGE^{MAX} database, the **#[NC]NoSuchName** error message is displayed and the SAGE^{MAX} prompts you for another point name to be erased.

8.6.2.3 Erase Groups

The **Group (G)** option allows you to erase a group object that you specify from the SAGE^{MAX} database. After you select this option, SAGE^{MAX} prompts you to enter the **Group Name:**.

After you specify the name of the point to be erased, the SAGE^{MAX} searches for the name. If the name is found, the SAGE^{MAX} prompts you to confirm the erasure of the group with **Erase this Object from the Database? (Y/N)**. If an erase request is made for a group that does not exist in the SAGE^{MAX} database, the **#[21]FileNotFound** error message is displayed and the SAGE^{MAX} prompts you for another group name to be erased.

8.6.2.4 Erase Programs

The **Programs (R)** option allows you to erase a program object that you specify from the SAGE^{MAX} database. After you select this option, SAGE^{MAX} prompts you to enter the **Program Name:**.

After you specify the name of the program to be erased, the SAGE^{MAX} searches for the name. If the name is found, the SAGE^{MAX} prompts you to confirm the erasure of the program with **Erase this Object from the Database? (Y/N)**. If an erase request is made for a program that does not exist in the SAGE^{MAX} database, the **#[NC]NoSuchName** error message is displayed and the SAGE^{MAX} prompts you for another program name to be erased.

8.6.2.5 Erase Variables

The **Variables (V)** option allows you to erase a variable object that you specify from the SAGE^{MAX} database. After you select this option, SAGE^{MAX} prompts you to enter the **Variable Name:**.

After you specify the name of the variable to be erased, the SAGE^{MAX} searches for the name. If the name is found, the SAGE^{MAX} prompts you to confirm the erasure of the variable with **Erase this Object from the Database? (Y/N)**. If an erase request is made for a variable that does not exist in the SAGE^{MAX} database, the **#[NC]NoSuchName** error message is displayed and the SAGE^{MAX} prompts you for another variable name to be erased.

8.6.2.6 Erase Globals

The **Globals (X)** option allows you to erase a global object that you specify from the SAGE^{MAX} database. After you select this option, SAGE^{MAX} prompts you to enter the **Global Name:**.

After you specify the name of the global to be erased, the SAGE^{MAX} searches for the name. If the name is found, the SAGE^{MAX} prompts you to confirm the erasure of the global with **Erase this Object from the Database? (Y/N)**. If an erase request is made for a global that does not exist in the SAGE^{MAX} database, the **#[NC]NoSuchName** error message is displayed and the SAGE^{MAX} prompts you for another global name to be erased.

8.6.2.7 Erase Users

The **Users (U)** option allows you to erase a user object that you specify from the SAGE^{MAX} database. After you select this option, SAGE^{MAX} prompts you to enter the **User Name:**.

After you specify the name of the user to be erased, the SAGE^{MAX} searches for the name. If the name is found, the SAGE^{MAX} prompts you to confirm the erasure of the user with **Erase this Object from the Database? (Y/N)**. If an erase request is made for a user that does not exist in the SAGE^{MAX} database, the **#[NC]NoSuchName** error message is displayed and the SAGE^{MAX} prompts you for another user name to be erased.

8.6.3 List Objects

Selecting **L** from the **Database Function** Submenu (Figure 8-13) displays the **List Objects** Submenu shown in Figure 8-39. From this submenu, you can list (to your port) objects in the SAGE^{MAX} database.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
List:
key          to do
E            Every (Point, Program, Variable, Global)
N            by Name (Point, Program, Variable, Global)
P            Points
G            Groups
R            Programs
C            Classes
V            Variables
X            Globals
U            Users
PF1         Return to previous menu
Press key for desired action:

```

Figure 8-39 The List Object Submenu

8.6.3.1 List Every

The **List Every (E)** option generates a list of every database object that is defined in the SAGE^{MAX} database. After you select this option, the SAGE^{MAX} begins displaying all database objects (starting with points) one page at a time. You display each successive page by typing any key to continue, or **PF1** to quit the listing. An example of the **List Every** option is illustrated in Figure 8-40.

Along with the name of every database object is the definition information that you supply when you create the database objects. This information varies based on the type of database object that is being displayed. For point objects, this information includes the point name, the SAGE^{MAX} port on which it is defined, the unit number on which the point is defined, the channel and subchannel on which the unit is defined, etc.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Ty Name----- P Unit- Chan- Ft Sc Crd ObjPrivs- L EUFile-- F Ovr
PT FL1 AHU1          1  2234 FF00H SY  0  0 01111111B Y  RXFF00 N  0
PT FL1 AHU1 DAT      1  2234 FE00H SY  2  0 00001111B N  RXTEMPS N  0
PT FL1 LIGHTS        1  4456 FB00H SY  4  0 00001111B N  MXFB00 Y  0

Top line is [ 1 of 3] PF1=End, Any key to continue:

```

Figure 8-40 Example List Every Screen

8.6.3.2 List by Name

The **List by Name (N)** option uses an object type (i.e., PT, VR, GL, PG) and an object name to display definition information about database objects. When you select this option, SAGE^{MAX} displays a wild card template for object type and object name information. SAGE^{MAX} will perform a search on the database based on a positional match of what you enter in the type (**Ty**) and **Name** fields of the wild card template. This is convenient if you know a “piece” of the object name.

8.6.3.3 List Points

The **List Points (P)** option is used to display definition information about selected database points. When you select this option, SAGE^{MAX} displays a wild card template for point definition information. SAGE^{MAX} performs a search on all database points based on a positional match of what you enter in the fields of the wild card template. If you type **RETURN** without modifying the wild card template, the SAGE^{MAX} will display all point objects in the SAGE^{MAX} database.

8.6.3.4 List Groups

The **List Groups (G)** option is used to display selected groups in the SAGE^{MAX} database. When you select this option, SAGE^{MAX} displays a wild card template for group names. SAGE^{MAX} performs a search on all database groups based on a positional match of what you enter in the fields of the wild card template. If you type **RETURN** without modifying the wild card template, the SAGE^{MAX} will display all group objects in the SAGE^{MAX} database.

When group names are listed, they are displayed one page at a time on your screen. If there is more than one page of group names in your database, you can access the next consecutive page by pressing any key.

8.6.3.5 List Programs

The **List Programs (R)** option is used to display program definition fields or program pathnames of selected groups in the SAGE^{MAX} database. When you select this option, SAGE^{MAX} displays the **List Programs** Submenu shown in **Figure 8-41**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
List Programs:
key           to do
F             Program Fields
P             Program Pathnames
PF1          Return to previous menu
Press key for desired action:
```

Figure 8-41 List Programs Submenu

Selecting either the **Program Fields (F)** option or the **Program Pathnames (P)** option of the **List Programs** Submenu displays a wild card template.

When program fields and pathnames are listed, they are displayed one page at a time on your screen. If there is more than one page, you can access the next consecutive page by pressing any key.

SAGE^{MAX} performs a search on all database programs based on a positional match of what you enter in the program fields and program pathname templates. If you type **RETURN** without modifying the wild card template, the SAGE^{MAX} will display all requested program information from the SAGE^{MAX} database.

8.6.3.6 *List Classes*

The **List Classes (C)** option is used to display selected classes from the SAGE^{MAX} database. When you select this option, SAGE^{MAX} displays a wild card template for class number or name. SAGE^{MAX} performs a search on all database classes based on a positional match of what you enter in the fields of the wild card template. If you type **RETURN** without modifying the wild card template, the SAGE^{MAX} will display all class objects from the SAGE^{MAX} database.

When classes are listed, they are displayed one class per page on your screen. If there is more than one class to be displayed, you can access the next consecutive class by pressing any key.

8.6.3.7 *List Variables*

The **List Variables (V)** option is used to display selected variables from the SAGE^{MAX} database. When you select this option, SAGE^{MAX} displays a wild card template for selecting variables. SAGE^{MAX} performs a search on all database variables based on a positional match of what you enter in the fields of the wild card template. If you type **RETURN** without modifying the wild card template, the SAGE^{MAX} will display all variable objects and associated variable fields from the SAGE^{MAX} database.

When variables are listed, they are displayed one screen at a time. If there are additional variables in your database, you can access the next consecutive screen by pressing any key.

8.6.3.8 *List Globals*

The **List Globals (X)** option is used to display selected global objects from the SAGE^{MAX} database. When you select this option, SAGE^{MAX} displays a wild card template for selecting globals. SAGE^{MAX} performs a search on all database globals based on a positional match of what you enter in the fields of the wild card template. If you type **RETURN** without modifying the wild card template, the SAGE^{MAX} will display all global objects from the SAGE^{MAX} database.

When globals are listed, they are displayed one screen at a time. If there are additional globals in your database, you can access the next consecutive screen by pressing any key.

8.6.3.9 *List Users*

The **List Users (U)** option is used to display selected users from the SAGE^{MAX} database. When you select this option, SAGE^{MAX} displays a wild card template for selecting users. SAGE^{MAX} performs a search on all database users based on a positional match of what you enter in the fields of the wild card template. If you type **RETURN** without modifying the wild card template, the SAGE^{MAX} will display all user names and associated user fields from the SAGE^{MAX} database.

When user information is listed, it is displayed one screen at a time. If there are additional users in your database, you can access the next consecutive screen by pressing any key.

8.6.4 *List to File*

The **List to File (F)** option of the **Database Functions** Submenu does the same thing as the previous menu option (**List Objects**), except that it routes the information to a file and optionally to a SAGE^{MAX} printer port.

Selecting **F** from the **Database Functions** Submenu (**Figure 8-13**) displays the **Enter filename:** prompt. At this prompt, you enter the filename you want to contain the database object listing information.

Next, the SAGE^{MAX} prompts you with **Spool file to a printer port? (Y/N):**. If you select **Y**, the SAGE^{MAX} also prompts you for the printer port number by displaying **Spool on Port:**. This is the SAGE^{MAX} port where the spool job is sent. The SAGE^{MAX} then asks if you want to **Delete file after spooling? (Y/N):**.

The SAGE^{MAX} then displays the **List Objects** Submenu shown in **Figure 8-39**. At this point, the options are the same as in **Chapter 8.6.3: List Objects**. The only difference is that you do not see the results of your selections. The results are sent to the filename that you specified and are optionally spooled to a SAGE^{MAX} port for printing.

Refer to **Chapter 8.6.3: List Objects** for more information on listing objects.

8.6.5 Move Database Points

Selecting **M** from the **Database Functions** Submenu (Figure 8-13) displays the **Move Database Points** Submenu shown in Figure 8-42. From this submenu, you can change the definition of database points so that all points that are defined on one port can be redefined or *moved* to another port.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Move Database Points:
key          to do
M           Move All Points on Port
U           Undo Move on Port
P           Make Move Permanent
N           Move All Points on Unit
PF1        Return to previous menu
Press key for desired action:
```

Figure 8-42 The Move Database Points Submenu

8.6.5.1 Move All Points on Port

The **Move All Points on Port (M)** option of the **Move Database Points** Submenu is used to change the definition of all database points on one port to points on another port. You specify the source and destination ports from prompts that the SAGE^{MAX} displays. The SAGE^{MAX} also asks if you want to update the network configuration file (e.g., C:\CFG\PUP.1, C:\CFG\PUP.2, etc.). The network configuration file contains unit numbers used for polling. If you choose to update the network configuration file, the list of units to be polled from the source configuration file is added to the list of units to be polled in the destination configuration file.

If, for example, you are polling 14 PHP slave units on port 3, and you move all points on port 3 to port 4, you will cause scan alarms to occur for the 14 PHP slave units if you do not update the network configuration file.

NOTE

*Do not update the network configuration file if you intend to **Undo Move on Port (U)**. Update the network configuration file only if you are certain the change is permanent.*

A sample screen showing the **Move All Points on Port (M)** option is shown in Figure 8-43.

If a **Move All Points on Port (M)** request is made, and there are no points defined for the specified port, the **No Points found!** message is displayed.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Source port number: 1
Destination port number: 2
Update network configuration file? (Y/N): N

Database search is in progress...
[      2135] records processed
...Press any key to continue:
```

Figure 8-43 Sample Move All Points on Port Screen

8.6.5.2 Undo Move on Port

The **Undo Move on Port (U)** option of the **Move Database Points** Submenu is used to restore database points to their original definition (i.e., the point definition that existed before a **Move All Points on Port (M)** or a **Move All Points on Unit (N)** option was selected).

When either of the two “move all points” options is selected, each of the resulting *moved* points gets an associated internal flag set. This flag indicates that the associated point has been *moved*. By selecting the **Undo Move on Port (U)** option, all flagged points on the source port that you specify are moved to their original port, providing you have not made the move a permanent one (see **Make Move Permanent (P)** option in the next section).

IMPORTANT

The Undo Move on Port (U) option is not effective if you have selected the Make Move Permanent (P) option previously.

If an **Undo Move on Port (U)** request is made, and there are no points with their *moved* flags set, the **No Points found!** message is displayed.

8.6.5.3 Make Move Permanent

The **Make Move Permanent (P)** option of the **Move Database Points** Submenu is used to clear the *moved* flag associated with database points that have been moved. Once a move is made permanent, the **Undo Move on Port (U)** option is not effective, since the flag used to denote moved points has been cleared.

A sample screen showing the **Make Move Permanent (P)** option is shown in **Figure 8-44**.

If a **Make Move Permanent (P)** request is made, and there are no points with their *moved* flags set, the **No Points found!** message is displayed.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Warning!!! Changes made will be irreversible. Continue? (Y/N): Y
Make changes permanent on port: 1
Database search is in progress...
[          2135] records processed
...Press any key to continue:

```

Figure 8-44 Sample Make Move Permanent Screen

8.6.5.4 Move All Points on Unit

The **Move All Points on Unit (N)** option of the **Move Database Points** Submenu is used to change the definition of all database points on one unit to points on another unit on the same SAGE^{MAX} port. You specify the source and destination units from prompts that the SAGE^{MAX} displays. The SAGE^{MAX} also asks if you want to update the network configuration file (e.g., **C:\CFG\IPUP.1**, **C:\CFG\IPUP.2**, etc.). The network configuration file contains logical unit numbers used for polling. If you choose to update the network configuration file, the list of units to be polled are updated based on the new unit number you defined.

For example, you use this option if you want to move all points on port 1 that are associated with unit 17, to unit 27. If you specify that you want to update the network configuration file, the SAGE^{MAX} will poll unit 27 (providing you had previously specified unit 17 to be polled).

NOTE

*Do not update the network configuration file if you intend to **Undo Move on Port (U)**. Update the network configuration file only if you are certain the change is permanent.*

A sample screen showing the **Move All Points on Unit (N)** option is shown in **Figure 8-45**.

If a **Move All Points on Unit (N)** request is made, and there are no points defined for the specified unit, the **No Points found!** message is displayed.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Source port number: 1
Unit number: 17
Unit number to assign to points: 27
Update network configuration file? (Y/N): N

Database search is in progress...
[          2135] records processed
...Press any key to continue:

```

Figure 8-45 Sample Move All Points on Unit Screen

8.7 The Find Objects Submenu

Selecting **F** from the **Main Menu** displays the **Find Objects (F)** Submenu shown in **Figure 8-46**. From this submenu, you can search the SAGE^{MAX} database for points, programs, globals and variables using a wild card match pattern and then display the object names and current attribute values to the screen or to a file. In addition, you can *stuff* (assign) a new value to the objects that you find.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Find Objects:
key           to do
F             Find Objects
S             Find and Stuff
L             Find and List to File
PF1          Return to previous menu
Press key for desired action:

```

Figure 8-46 The Find Objects Submenu

The wild card match pattern is used to specify a desired subset of database objects and contains a *type field*, an *object name field*, an *attribute field*, and a special *display timeout (t/o) flag* that gives you two options for handling attribute fetches that timeout.

8.7.1 Find Objects

The **Find Objects (F)** option of the **Find Objects** Submenu finds database objects based on a wild card match pattern and displays the object names and current attribute values to the screen. Use the left and right arrow keys to maneuver through the fields of the wild card match pattern and then enter the desired match characters in the appropriate positions. The wild card match pattern is shown in **Figure 8-47**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Find Objects:
key           to do
F             Find Objects
S             Find and Stuff
L             Find and List to File
PF1          Return to previous menu
Press key for desired action: F

TY=(PT,PG,VR,GL), D=display t/o (Y/N), Name=obj name, AT=attribute
TY D Name----- AT
?? ? ??????????????????????????????????????????? ??

```

Figure 8-47 The Find Objects Wild Card Template Screen

You use the *type field* to specify the type of object you want to find by entering the appropriate 2-character object type code (**PT** for points, **PG** for programs, **VR** for variables, and **GL** for globals). Initially, this field contains **??** (wild card characters) which means that objects of all four object types are included in the database search.

You use the *display t/o field* to specify how you want to display attribute values that are not fetched when you first request them (i.e., a timeout occurs). If you specify **Y** in this field, requests that timeout simply display **#Timeout** as the attribute value. If you specify **N** in this field, and a timeout occurs, the SAGE^{MAX} leaves the line blank.

You use the *name field* to specify all or part of the object name(s) that you want to find. This field is 24 characters long and initially contains all wild card characters.

You use the *attribute field* to specify an object attribute. Initially, this field contains **??** (wild card characters) which means that all attributes of the object are used. In the case of variable objects, the attribute field is not used since variables do not have attributes.

After you enter the desired match pattern, SAGE^{MAX} displays object names and attribute values as shown in **Figure 8-48**. Objects are displayed 16 per page. If more than one page of objects results from the database search, you can press any key to view the next page.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
TY Name----- AT +Value-----
PT FLR1_AHU1_DAT      ;DT      84.0
PT FLR1_AHU1_DAT      ;HL      100.0
PT FLR1_AHU1_DAT      ;LL      70.0
PT FLR1_AHU1_DAT      ;DB       5.0
PT FLR1_AHU1_DAT      ;AE       1
PT FLR1_AHU1_DAT      ;ST       1
PT FLR1_AHU1_DAT      ;MN       0.0
PT FLR1_AHU1_DAT      ;MX       0.0
PT FLR1_AHU1         ; #Timeout
VR $ALARMS           ; No
VR $MODE             ;      0
VR $HOLIDAY          ; No
GL OATEMP            ;      52.6

...Press any key to continue, Press PF1 to return to previous menu

```

Figure 8-48 Sample Find Objects Screen

8.7.2 Find and Stuff

The **Find and Stuff (S)** option of the **Find Objects** Submenu finds database objects based on a wild card match pattern, stuffs the attributes with a value you define, and optionally displays the object names and newly stuffed attribute values to the screen. You use the left and right arrow keys to maneuver through the fields of the wild card match pattern and then enter the desired match characters and new value in the appropriate positions. The wild card match pattern for the **Find and Stuff (S)** option is shown in **Figure 8-49**.

You use the *type field* to specify the type of object you want to “find and stuff” by entering the appropriate 2-character object type code (**PT** for points, **PG** for programs, **VR** for variables, and **GL** for globals). Initially, this field contains **??** (wild card characters) which means that objects of all four object types are included in the database search.

8.7.3 Find and List to File

The **Find and List to File (L)** option of the **Find Objects** Submenu finds database objects based on a wild card match pattern and directs the object names and current attribute values to a filename that can be viewed or spooled to a printer port.

When you select this option, SAGE^{MAX} prompts you for a filename and then asks if you want to spool the file to a printer port. If you respond with **Y**, SAGE^{MAX} also asks for the port number to spool the file to, and if the file should be deleted after the file is spooled. Next the SAGE^{MAX} displays the wild card match pattern for selecting database objects. You use the left and right arrow keys to maneuver through the fields of the wild card match pattern and then enter the desired match characters in the appropriate positions. Refer to **Figure 8-50**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Enter filename: findfile
Spool file to a printer port? (Y/N) Y
Spool on Port: 13
Delete file after spooling? (Y/N): N

TY=(PT,PG,VR,GL), D=display t/o (Y/N), Name=obj name, AT=attribute
TY D Name----- AT
?? ? ??????????????????????????????????????????? ??

```

Figure 8-50 The Find and List to a File Wild Card Template Screen

You use the *type field* to specify the type of object you want to find by entering the appropriate 2-character object type code (**PT** for points, **PG** for programs, **VR** for variables, and **GL** for globals). Initially, this field contains **??** (wild card characters) which means that objects of all four object types are included in the database search.

You use the *display t/o field* to specify how you want to display attribute values that are not fetched when you first request them (i.e., a timeout occurs). If you specify **Y** in this field, requests that timeout simply display **#Timeout** as the attribute value. If you specify **N** in this field, and a timeout occurs, the SAGE^{MAX} leaves the line blank.

You use the *name field* to specify all or part of the object name(s) that you want to find. This field is 24 characters long and initially contains all wild card characters.

You use the *attribute field* to specify an object attribute. Initially, this field contains **??** (wild card characters) which means that all attributes of the object are used. In the case of variable objects, the attribute field is not used since variables do not have attributes.

8.8 The Job Scheduler Submenu

The **Job Scheduler (J)** Submenu is used to request jobs from an operator interface. SAGE^{MAX} jobs include the following:

- Edit/Create a File
- Archive Alarms
- Broadcast Message
- Data Capture/Stuff
- Export Database File
- Create Report File
- Translate STAR to SAGE
- Spool Files for Printing
- Upload/Download File

Jobs may be executed immediately, or they may be queued for later execution at a particular time and date, or repetitively at some interval. Job requests are loaded and executed by the Job Scheduling Task which is also responsible for maintaining the global SAGE^{MAX} calendar.

The **Job Scheduler (J)** Submenu offers five options. You can **Create Scheduled Request**, **List Scheduled Requests**, **Erase Scheduled Request**, **Make "As Soon As Possible" Request**, and **Make Job Request Directly**. Refer to **Figure 8-51**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Scheduler:
key           to do
C             Create Scheduled Request
L             List Scheduled Requests
E             Erase Scheduled Request
A             Make "As Soon As Possible" Request
J             Make Job Request Directly
PF1          Return to previous menu
Press key for desired action:
```

Figure 8-51 The Job Scheduler Menu

8.8.1 Create Scheduled Request

The **Create Scheduled Request (C)** option of the **Job Scheduler (J)** Submenu is used to create a SAGE^{MAX} job that is queued for later execution at a particular time and date, or repetitively at some interval.

When you select this option, SAGE^{MAX} prompts you with a series of questions regarding the nature of the schedule request that you are creating. Refer to **Figure 8-52**.


```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Scheduler:
key           to do
C             Create Scheduled Request
L             List Scheduled Requests
E             Erase Scheduled Request
A             Make "As Soon As Possible" Request
J             Make Job Request Directly
PF1          Return to previous menu
Press key for desired action: C
Day of Month(1-31 or                                   Return="any day"):
Month        (Jan Feb... or 1-12 or                    Return="any month"):
Year         (1996 or 96 or                             Return="any year"):
Day of Week  (Sun Mon... or 1-7 {1=Sun} or             Return="any day"):
Time         (hours:minutes):

```

Figure 8-52 The Create Scheduled Request Screen

The first prompt asks for the day of the month (1-31) when you want the job to occur. You can type the **RETURN** key for “any day.” This means that the job is scheduled to occur independently of the day of the month.

The second prompt that the SAGE^{MAX} displays is for the month when you want the job to occur. The month can be entered in either 3-character abbreviated form (e.g., Jan, Feb, etc.) or numerically (1-12). You can type the **RETURN** key for “any month.” This means that the job is scheduled to occur independently of the month.

The third prompt that the SAGE^{MAX} displays is for the year when you want the job to occur. The year can be entered in either 4-digit form (e.g., 1996) or in 2-digit form (e.g., 96). You can type the **RETURN** key for “any year.” This means that the job is scheduled to occur independently of the year.

The fourth prompt that the SAGE^{MAX} displays is for the day of the week when you want the job to occur. The day of the week can be entered in either 3-character abbreviated form (e.g., Sun, Mon, etc.) or in numerical format (e.g., 1, 2, etc.) where 1=Sunday, 2=Monday, etc. You can type the **RETURN** key for “any day.” This means that the job is scheduled to occur independently of the day of the week.

The fifth prompt that the SAGE^{MAX} displays is for the time when you want the job to occur. The time must be entered in short time format (e.g., 09:15, 17:00, etc.). You can type the **RETURN** key to indicate the time 00:00.

Next, the SAGE^{MAX} displays the **Job Request** Submenu. You must select the type of job that you want to occur. The choices are:

- Edit/Create a File (E)
- Archive Alarms (0)
- Broadcast Message (1)

- Data Capture/Stuff (2)
- Export Database Files (3)
- Create Report File (4)
- Translate STAR to SAGE^{MAX} (5)
- Spool Files for Printing (6)
- Upload/Download File (7)

The **Job Request** Submenu is shown in **Figure 8-53**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Request:
key          to do
E           Edit/Create a File
0           Archive Alarms
1           Broadcast Message
2           Data Capture/Stuff
3           Export Database File
4           Create Report File
5           Translate STAR to SAGE
6           Spool Files for Printing
7           Upload/Download a File
PF1         Return to previous menu
Press key for desired action:
```

Figure 8-53 The Job Request Submenu

8.8.1.1 Edit/Create a File

Selecting the **Edit/Create a File (E)** option from the **Job Request** Submenu allows you to create or edit a text file. The SAGE^{MAX} displays the prompt: **Enter [drive]:\pathname:.** After you respond to the prompt, the SAGE^{MAX} Text Editor (see **Figure 8-21**) is displayed, allowing you to create or edit the text file. When you are finished editing the text file, press **PF1** to display the text editor menu, **S** to save changes to the text file, and **X** to return to the Job Scheduler menu.

8.8.1.2 Archive Alarms

Selecting the **Archive Alarms (0)** option from the **Job Request** Submenu allows you to archive the alarms the SAGE^{MAX} registers into a file. The SAGE^{MAX} displays the prompt: **Enter alarm class code (*'=all):.** Enter the appropriate alarm code and press **RETURN**. The SAGE^{MAX} then displays the prompt: **Enter [drive]:\directory for archive (return):.** This prompt allows you to choose where the alarm archive file will be stored. After you press **RETURN**, the SAGE^{MAX} will archive all alarms matching the appropriate class code, and create an archive file in the directory that you chose.

8.8.1.3 Creating a "Broadcast Message" Job

At this point, you have specified *when* you want a particular job to occur. You must now specify *what kind* of job you want to occur. Selecting the **Broadcast Message (1)** option from the **Job Request** Submenu allows you to request a Broadcast Message job.

When you select this option, the SAGE^{MAX} prompts you with the message **Broadcast [Port/Unit#/Text]:**. From this prompt you enter the following three pieces of information:

- Port number
- Unit number (optional)
- Message text

You must use "/" keys to separate the port and unit numbers, and the unit number and message text string. If the optional unit number is not used (e.g., broadcasting messages to a local CRT port or to all units on the port), type two "/" characters between the port number and message text strings.

Figure 8-54 illustrates the Broadcast Message job request screen of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Request:
key          to do
E           Edit/Create a File
0           Archive Alarms
1           Broadcast Message
2           Data Capture/Stuff
3           Export Database File
4           Create Report File
5           Translate STAR to SAGE
6           Spool Files for Printing
7           Upload/Download a File
PF1         Return to previous menu
Press key for desired action: B
Broadcast [Port/Unit#/Text]:

```

Figure 8-54 The Broadcast Message Job Request Screen

8.8.1.4 Creating a "Data Capture/Stuff" Job

At this point, you have specified *when* you want a particular job to occur. You must now specify *what kind* of job you want to occur. Selecting the **Data Capture/Stuff (2)** option from the **Job Request** Submenu allows you to request a data capture or data stuff job.

When you select this option, the SAGE^{MAX} prompts you with the message **Pathname for Points Description File (PDF):**. From this prompt you enter the path and filename of the PDF file that you want to use for the Data Capture/Stuff job. For information on the structure of Points Description Files, refer to **Appendix H: DCS Files**.

Next, the SAGE^{MAX} prompts you with the message **Use "Stuff Format" for captured data? (Y/N):**. This allows you to specify the output format for data captures. If you type **Y**, the object, attribute, and captured data are stored in the format used by PDFs (e.g., **OATEMP;CV=45.0**). This allows you to use the captured data as data to be stuffed. If you type **N**, the object and attribute are stored as quoted strings, the captured data follows, and the fields are separated by commas (e.g., **"OATEMP","CV",45.0**).

Figure 8-55 illustrates the Data Capture/Stuff job request screen of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Request:
key          to do
E           Edit/Create a File
0           Archive Alarms
1           Broadcast Message
2           Data Capture/Stuff
3           Export Database File
4           Create Report File
5           Translate STAR to SAGE
6           Spool Files for Printing
7           Upload/Download a File
PF1         Return to previous menu
Press key for desired action: C
Pathname for Points Description File (PDF): SAMPLE
Use "Stuff Format" for captured data? (Y/N): Y

```

Figure 8-55 Sample Data Capture/Stuff Job Request Screen

8.8.1.5 Creating an "Export Database Files" Job

At this point, you have specified *when* you want a particular job to occur. You must now specify *what kind* of job you want to occur. Selecting the **Export Database Files (E)** option from the **Job Request** Submenu allows you to request a Name Bindings File (NBF) of all or selected objects of the SAGE^{MAX} database. This exported file can be transferred to floppy disk and can be edited using a standard text editor.

When you select this option, the SAGE^{MAX} prompts you with the message **Export to NBF pathname:**. From this prompt you enter the SAGE^{MAX} path and filename for the Name Bindings File (e.g., **A:\DATABASE.NBF**).

Next, the SAGE^{MAX} prompts you with the message **Include All Objects? (Y/N):**. From this prompt you enter **Y** if you want all SAGE^{MAX} database objects in the NBF. If you enter **N**, SAGE^{MAX} will prompt you with each individual database object. For each object type, you respond with **Y** or **N** depending on whether or not you want those types included in the NBF.

When you select the type(s) of objects that you want in the NBF, SAGE^{MAX} creates the NBF file that you specified.

Figure 8-56 illustrates the Export Database Files job request screen of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Request:
key           to do
E             Edit/Create a File
0             Archive Alarms
1             Broadcast Message
2             Data Capture/Stuff
3             Export Database File
4             Create Report File
5             Translate STAR to SAGE
6             Spool Files for Printing
7             Upload/Download a File
PF1          Return to previous menu
Press key for desired action: 3
Export to NBF pathname: A:\VARS.NBF
Include All Objects? (Y/N): N
Include Points? (Y/N): N
Include Variables? (Y/N): Y
Include Programs? (Y/N): N
Include Globals? (Y/N): N
Include Cards? (Y/N): N
Include Profiles? (Y/N): N
Include Rooms? (Y/N): N
Include Readers? (Y/N): N

```

Figure 8-56 Sample Export Database Files Job Request Screen

NOTE

Importing Name Bindings Files is done through the Import Name Bindings (I) option of the Backup Utilities Submenu found in the Maintenance Shell of the SAGE^{MAX}. To access the Backup Utilities Submenu, you must use the Prepare System for Maintenance (Z) option of the Utility Functions Submenu. Refer to Chapter 8.15.10: Prepare System for Maintenance. Access to the Backup Utilities Submenu is then possible from port Z of the SAGE^{MAX}.

8.8.1.6 Creating a Report File

At this point, you have specified *when* you want a particular job to occur. You must now specify *what kind* of job you want to occur. Selecting the **Create Report File (4)** option from the **Job Request** Submenu allows you to request a report job.

Reports are created by first merging a *text template file* with a file containing *data values* to produce a final *report file* which may be printed or saved for later reference.

The *text template file* contains the basic, underlying text of the report. This text file identifies locations to be filled in with data by including special *tokens* that are enclosed in parentheses. Within this file, every place that will contain a dynamic field is marked with a place holder. This place holder takes the format **%...%** and is called a format specifier. A list of format specifiers used in text template files is shown in **Table 8-1**.

FORMAT	DESCRIPTION
% %	the character % itself
%.?%	use the data type of the expression - default characters wide
%.?Rx%	use the data type of the expression - only show the x rightmost characters
%.?Lx%	use the data type of the expression - only shoe the x leftmost characters
%.Fx.y%	for floating point xxx.yyy format
%.Ix%	for integer with field that is x digits wide
%.Ux%	for unsigned integer with field that is x digits wide
%.Hx%	for hexadecimal mode with field that is x digits wide
%.Bx%	for binary mode with field that is x digits wide
%.xx%	for control characters using decimal numbers xx (e.g., %10% for <lf>, %13% for <cr> or %07% for <bel>)
%.R%	the 24-character name of the reference that matches the variable
%.Tx%	current time in HH:MM:SS format (x=8), HH:MM format (x=5) or HH format (x=2)
%.W%	current day of the week (e.g., MON, TUE, WED, etc.)
%.D%	current day of the month as two decimal digits
%.Mx%	current month as two digits (x=2) or as a 3 letter abbreviation (x=3)
%.Yx%	current year as last two digits (x=2) or as full year (x=4)

Table 8-1 Format Specifiers Used in Report Template Files

The *data file* contains values that are merged into the place holders of the text template file. The *data file* may be either a SAGE^{MAX} table file (.TBL), a trend file, or an ASCII text file.

The resulting output report file is a text file that has the extension .RPT. You can spool this file to a printer port or save it for later reference.

When you select this option, the SAGE^{MAX} prompts you with the message **Report Template pathname:**. From this prompt you enter the SAGE^{MAX} path and filename for the report template file (e.g., C:\MONTHLY.TXT).

Next, the SAGE^{MAX} prompts you with the message **Report Datafile pathname:**. From this prompt you enter the SAGE^{MAX} path and filename for the file that contains the data that will be merged with the template file to produce the report (e.g., C:\USAGE.TBL).

Finally, SAGE^{MAX} prompts you with the message **Report Output pathname:**. From this prompt you enter the SAGE^{MAX} path and filename for the output report that will be created (e.g., C:\MONTHLY.RPT). The filename is assumed to be .RPT.

Figure 8-57 illustrates the Export Database Files job request screen of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Request:
key          to do
E           Edit/Create a File
0           Archive Alarms
1           Broadcast Message
2           Data Capture/Stuff
3           Export Database File
4           Create Report File
5           Translate STAR to SAGE
6           Spool Files for Printing
7           Upload/Download a File
PF1        Return to previous menu
Press key for desired action: 4
Report Template pathname: C:\MONTHLY.TXT
Report Datafile pathname: C:\USAGE.TBL
Report Output pathname: C:\MONTHLY.RPT

```

Figure 8-57 Sample Create Report File Job Request Screen

A sample template file is shown in **Figure 8-58**. This sample template file was created using the SAGE^{MAX} text editor. This template file is merged with the ASCII data file shown in **Figure 8-59** to create the sample report file shown in **Figure 8-60**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Edit v3 Insert   C:\RPT\ELECTRIC.TXT

*****
          E L E C T R I C       D E M A N D       R E P O R T
*****

REPORT DATE : %W%, %M3%, %Y4%                TIME : %T8%

DEMAND LEVEL PLAN   : %?% KW
DAILY PEAK DEMAND   : %?% KW, PEAK TIME : %?%
EXCESS MAXIMUM DEMAND: %?% KW

CALCULATED COST OF EXCESS DEMAND BASED ON $ %F1.2% / KW IS $ %F1.2%

WEEKLY CONSUMPTION  : %?% KW
MONTHLY CONSUMPTION : %?% KW
YEARLY CONSUMPTION  : %?% KW

AVERAGE OUTSIDE AIR TEMPERATURE: %?R4% DEGREES FAHRENHEIT
LOW / HIGH OUTSIDE TEMPERATURES: %?R4%, %?R4% DEGREES F
AVERAGE OUTSIDE AIR HUMIDITY: %?R4% %% RELATIVE HUMIDITY

Press PF2 or HELP for editor Help

```

Figure 8-58 Sample Report Template File Using the SAGE^{MAX} Text Editor

For an actual monthly report of this type, it would be more practical to use a SAGE^{MAX} table file as the report data file rather than an ASCII text data file. A monthly table file could be created automatically using an SPL program that assigns the table values from trended data and from values calculated within the program. **Figure 8-61** shows such a sample SPL program. If no calculated values were needed in this example, you could simply use the trend file as the input data file for the report job.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Edit v3 Insert   C:\RPT\ELECTRIC.DAT
225.0
245.0
20.0
5.85
117.00
1453.5
5066
52355
52.5
38.0
57.0
45.0

Press PF2 or HELP for editor Help

```

Figure 8-59 Sample ASCII Data File Using the SAGE^{MAX} Text Editor

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Edit v3 Insert   C:\RPT\ELECTRIC.TXT
*****
          E L E C T R I C   D E M A N D   R E P O R T
*****
REPORT DATE : Wed, Mar, 1991           TIME : 12:00:00

DEMAND LEVEL PLAN      : 225.0 KW
DAILY PEAK DEMAND     : 245.0 KW, PEAK TIME : 11:58:05
EXCESS MAXIMUM DEMAND: 20.0 KW

CALCULATED COST OF EXCESS DEMAND BASED ON $ 5.85 / KW IS $ 117.00

WEEKLY CONSUMPTION    : 1453.5 KW
MONTHLY CONSUMPTION   : 5066 KW
YEARLY CONSUMPTION    : 52355 KW

AVERAGE OUTSIDE AIR TEMPERATURE: 52 DEGREES FAHRENHEIT
LOW / HIGH OUTSIDE TEMPERATURES: 38, 57 DEGREES F
AVERAGE OUTSIDE AIR HUMIDITY: 45 % RELATIVE HUMIDITY

Press PF2 or HELP for editor Help

```

Figure 8-60 Sample Report Template File Using the SAGE^{MAX} Text Editor**NOTE**

If you use a trend file as the data file for a report job, the extension of the trend filename is .TRN if the trend is inactive and is .TR\$ if the trend is active. When you specify the report data filename, be sure to include the proper trend extension.


```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Edit v3 Insert   C:\SPL\ELECTRIC.SPL
;-----
; This SPL program is used to stuff trended and calculated values into a Table
; File which is to be used as the input data file for the ELECTRIC report job.
;
;   C:\TABLES\ELEC.TBL- table file used as data file for report job ELECTRIC
;   C:\TREND\DEMAND.TRN- trend of collected data
;-----

        SWAIT 5
        &\TABLES\ELEC(0)=&\TREND\DEMAND.TRN(0)
        &\TABLES\ELEC(1)=&\TREND\DEMAND.TRN(1)
        &\TABLES\ELEC(2)=&\TREND\DEMAND.TRN(2)
        &\TABLES\ELEC(3)=&\TREND\DEMAND.TRN(3)
        &\TABLES\ELEC(4)=&\TREND\DEMAND.TRN(4)
        &\TABLES\ELEC(5)=&\TREND\DEMAND.TRN(5)

;-----
;   EXCESS is used to determine the calculated cost of the excess demand
;   and to stuff the calculated value into C:\TABLES\ELEC(6).
;-----

        CALL EXCESS

        &ELEC(7)=&\TREND\DEMAND.TRN(7)
        &ELEC(8)=&\TREND\DEMAND.TRN(8)
        &ELEC(9)=&\TREND\DEMAND.TRN(9)
        &ELEC(10)=&\TREND\DEMAND.TRN(10)
        &ELEC(11)=&\TREND\DEMAND.TRN(11)
        &ELEC(12)=&\TREND\DEMAND.TRN(12)
        &ELEC(13)=&\TREND\DEMAND.TRN(13)

        STOP

Press PF2 or HELP for editor Help

```

Figure 8-61 Creating a Table Using an SPL Program

8.8.1.7 Creating a "Translate STAR to SAGE" Job

At this point, you have specified *when* you want a particular job to occur. You must now specify *what kind* of job you want to occur. Selecting the **Translate STAR to SAGE (5)** option from the **Job Request** Submenu allows you to translate an uploaded STAR database (**SYSTEM.SYS**) to SAGE^{MAX} format.

When you select this option, the SAGE^{MAX} prompts you with the message **Pathname of STAR [SYSTEM.SYS] file:**. From this prompt you enter the filename of the **SYSTEM.SYS** file that you want to convert. Typically this file is uploaded to a disk using XANP (e.g., from a STAR) or PHP (e.g., from SPECTRA). The disk is then inserted into the **A:** drive of the SAGE^{MAX} and the appropriate filename is specified at the prompt (e.g., **A:\STR1SYS.SYS**).

Next, SAGE^{MAX} prompts you with the message **Pathname of file to put STAR Messages in:**. From this prompt you enter the SAGE^{MAX} pathname of an ASCII text file that will contain the messages that were part of the STAR database. This text file is created simply for reference.

The SAGE^{MAX} then prompts you with the message **SAGE port to use as USER port:**. From this prompt you enter the SAGE^{MAX} port number that will represent the old STAR's User port (e.g., SAGE^{MAX} ports **7** or **8**).

Next, the SAGE^{MAX} prompts you with the message **SAGE port to use as HOST port:**. From this prompt you enter the SAGE^{MAX} port number that will represent the old STAR's Host port (e.g., SAGE^{MAX} ports **1, 2, 3, 4, 5, 7** or **8**).

The SAGE^{MAX} then prompts you with the message **SAGE port to use as Peernet:**. From this prompt you enter the SAGE^{MAX} port number that will represent the old STAR's Peernet port (e.g., SAGE^{MAX} ports **1, 2, 3, 4, 5, 7** or **8**).

Next, the SAGE prompts you with the message **Peernet unit number of STAR that [SYSTEM.SYS] came from [0-31]:**. From this prompt you enter the Peernet unit number of the STAR whose database you want to translate (e.g., STAR unit number **0-31**).

The SAGE^{MAX} then prompts you with the message **Peernet unit number of this SAGE^{MAX} [0-31]:**. From this prompt you enter the Peernet unit number of this SAGE^{MAX} (e.g., SAGE^{MAX} unit number **0-31**). This is a Peernet driver variable of the SAGE^{MAX} port that you define as a Peernet driver type.

Next, the SAGE^{MAX} prompts you with the message **Point name prefix:**. From this prompt you enter a text string that will be added to the beginning of all STAR point object names. These new names (that can be up to 24 characters in length) represent the SAGE^{MAX} point names after the STAR translation. If you are replacing several STARS with a SAGE^{MAX}, for example, you may want to use prefixes such as **STAR1_**, **STAR2_**, etc. This might yield new SAGE^{MAX} points such as **STAR1_SYSTEM**, **STAR2_OATEMP**, etc.

The SAGE^{MAX} then prompts you with the message **Group name prefix:**. From this prompt you enter a text string that will be added to the beginning of all STAR group object names. These new names (that can be up to 24 characters in length) represent the SAGE^{MAX} group names after the STAR translation. If you are replacing several STARS with a SAGE^{MAX}, for example, you may want to use a prefix such as **S2G_**. This might yield new SAGE^{MAX} groups such as **S2G_MAIN**, **S2G_TEMPS**, etc.

Next, the SAGE^{MAX} prompts you with the message **Program name prefix:**. From this prompt you enter a text string that will be added to the beginning of all STAR program object names. These new names (that can be up to 24 characters in length) represent the SAGE^{MAX} program names after the STAR translation. If you are replacing several STARS with a SAGE^{MAX}, for example, you may want to use a prefix such as **S2P_**. This might yield new SAGE^{MAX} programs such as **S2P_OSS**, **S2P_DEMAND**, etc.

From the information that you enter in the prompts listed above, the SAGE^{MAX} translates the database of the STAR into database components that are compatible with the SAGE^{MAX}. Other STAR database items are also translated including trends, users, engineering units, etc.

Figure 8-62 illustrates the Translate STAR to SAGE^{MAX} job request screen of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Request:
key           to do
E             Edit/Create a File
0             Archive Alarms
1             Broadcast Message
2             Data Capture/Stuff
3             Export Database File
4             Create Report File
5             Translate STAR to SAGE
6             Spool Files for Printing
7             Upload/Download a File
PF1          Return to previous menu

Press key for desired action: 5
Pathname of STAR [SYSTEM.SYS] file: A:\STR2SYS.SYS
Pathname of file to put STAR Messages in: C:\STAR\STR2MSG.TXT
SAGE port to use as USER port: 7
SAGE port to use as HOST port: 5
SAGE port to use as Peernet: 4
Peernet unit number of STAR that [SYSTEM.SYS] came from [0-31]: 2
Peernet unit number of this SAGE [0-31]: 2
Point name prefix: STAR2_
Group name prefix: S2G_
Program name prefix: S2P_

```

Figure 8-62 Sample Translate STAR to SAGE^{MAX} Job Request Screen

8.8.1.8 Creating a “Spool Files for Printing” Job

At this point, you have specified *when* you want a particular job to occur. You must now specify *what kind* of job you want to occur. Selecting the **Spool Files for Printing (6)** option from the **Job Request** Submenu allows you to send a file to a SAGE^{MAX} port to be printed.

When you select this option, the SAGE^{MAX} prompts you with the message **Spool on Port:**. From this prompt you enter the SAGE^{MAX} port number where you want the file to be printed. This port must be configured as a printer or PHPdial driver type. You may also specify port 13 which is the dedicated parallel printer port on the SAGE^{MAX}.

Next, SAGE^{MAX} prompts you with the message **Spool Pathname:**. From this prompt you enter the SAGE^{MAX} pathname of the file that you want printed.

The SAGE^{MAX} then prompts you with the message **Spool Dialout Speed (300, 1200, 2400, 4800, 9600):**. From this prompt you enter the dialout baud rate that you want to use. This assumes that you are printing to a port configured as a PHPdial type. You enter the dialout baud rate from the list of choices specified. If you did not specify a dial port, simply type **RETURN**.

Next, SAGE^{MAX} prompts you with the message **Include Banner Line with spool output? (Y/N):**. From this prompt you enter **Y** if you want your printout to include a banner line, or **N** if not.

Next, SAGE^{MAX} prompts you with the message **Delete file after spooling? (Y/N)**:. From this prompt you enter **Y** if you want the file to be deleted after it is spooled, or **N** if you do not.

Figure 8-63 illustrates the Spool Files for Printing job request screen of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Request:
key          to do
E           Edit/Create a File
0           Archive Alarms
1           Broadcast Message
2           Data Capture/Stuff
3           Export Database File
4           Create Report File
5           Translate STAR to SAGE
6           Spool Files for Printing
7           Upload/Download a File
PF1        Return to previous menu
Press key for desired action: 6
Spool on Port: 13
Spool pathname: C:\RPT\ELECTRIC.RPT
Spool dialout speed (300,1200,2400,4800,9600):
Include banner line with spool output? (Y/N): N
Delete file after spooling? (Y/N): N

```

Figure 8-63 Sample Spool Files for Printing Job Request Screen

8.8.1.9 Creating a "Upload/Download File" Job

At this point, you have specified *when* you want a particular job to occur. You must now specify *what kind* of job you want to occur. Selecting the **Upload/Download a File (7)** option from the **Job Request** Submenu allows you to request a file download to a unit on a SAGE^{MAX} port.

When you select this option, the SAGE^{MAX} prompts you with the message **Type D to Download or U to Upload a file**:. From this prompt you enter your selection to either upload or download a file.

After you have made your selection, the SAGE^{MAX} prompts you with the message **Port# for peer**:. From this prompt you enter the SAGE^{MAX} port number of the destination unit.

Next, the SAGE^{MAX} prompts you with the message **Remote device unit#**:. From this prompt you enter the unit number of the destination unit.

The SAGE^{MAX} then prompts you with the message **Remote device drive:\pathname**:. This information specifies the location of the destination file within the destination unit (e.g., **2:SYSTEM.SYS**). The format of this information varies based on the type of protocol that is being used. For more information about the syntax of remote pathnames, refer to the individual driver chapters (refer to **Chapter 16: The XANP Driver** and **Chapter 17: The STAR Peernet Driver**).

Next, the SAGE^{MAX} prompts you with the message **Local drive:\pathname:**. This information specifies the location of the SAGE^{MAX} source file to be downloaded (e.g., **C:\SYSTEM02.SYS**).

Figure 8-64 illustrates the Download File job request screen of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Request:
key          to do
E            Edit/Create a File
0            Archive Alarms
1            Broadcast Message
2            Data Capture/Stuff
3            Export Database File
4            Create Report File
5            Translate STAR to SAGE
6            Spool Files for Printing
7            Upload/Download a File
PF1         Return to previous menu
Press key for desired action: 7
Type D to Download or U to Upload a File: D
Port# for peer: 1
Remote device unit#: 14
Remote device drive:\pathname: 2:SYSTEM.SYS
Local drive:\pathname: C:\SYSTEM02.SYS

```

Figure 8-64 Sample Download File Job Request Screen

Figure 8-65 illustrates the Upload File job request screen of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Request:
key          to do
E            Edit/Create a File
0            Archive Alarms
1            Broadcast Message
2            Data Capture/Stuff
3            Export Database File
4            Create Report File
5            Translate STAR to SAGE
6            Spool Files for Printing
7            Upload/Download a File
PF1         Return to previous menu
Type D to Download or U to Upload a File: U
Press key for desired action: 7
Port# for peer: 1
Remote device unit#: 14
Remote device drive:\pathname: 2:SYSTEM.SYS
Local drive:\pathname: C:\SYSTEM02.SYS

```

Figure 8-65 Sample Upload File Job Request Screen

8.8.2 List Scheduled Requests

The **List Scheduled Request (L)** option of the **Job Scheduler (J)** Submenu is used to list scheduled jobs that have been requested, but have not yet been performed. This list does not include “As Soon As Possible” job requests since they are not true *scheduled* requests.

When you select this option, SAGE^{MAX} prompts you with a wild card template to use as a match pattern if you want to list only certain scheduled jobs (e.g., DCS jobs for the current month, Broadcast jobs for next Tuesday, Report Generation jobs for the year, etc.). You use the left and right arrow keys to maneuver through the wild card template. Enter your match pattern in the appropriate field for a partial list of scheduled jobs, or just type **RETURN** for a list of all scheduled jobs. Scheduled job requests are listed one page at a time. If more than one page of scheduled jobs exist, you can press any key to view the next page. A sample list of scheduled jobs is shown in **Figure 8-66**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
sched [day,month,year,dow,time] (operator,port)
????????????????????????????????????????????????????????????????????????????????????
00001 [08,09,1996,?,09:30] (SYSTEM,08)
      BC 8//Boiler #5 shutdown in 30 minutes...
00002 [08,09,1996,?,09:30] (SYSTEM,08)
      DCS SAMPLE.PDF/S
00003 [08,09,1996,?,09:30] (SYSTEM,08)
      UDL D 1 14/2:SYSTEM.SYS C:\SYSTEM02.SYS
00004 [08,09,1996,?,09:30] (SYSTEM,08)
      EXPORT a:\VARS.NBF
00005 [08,09,1996,?,09:30] (SYSTEM,08)
      RPT C:\MONTHLY.TXT C:\USAGE.TBL C:\MONTHLY
00006 [08,09,1996,?,09:30] (SYSTEM,08)
      SPOOL 13 C:\RPT\ELECTRIC.RPT /B
00007 [08,09,1996,?,09:30] (SYSTEM,08)
      S2SAGE A:\STR2SYS.SYS C:\STR\STR2MSG.TXT/U
00008 [08,09,1996,?,09:30] (SYSTEM,08)
      UDL U 1 14/2:SYSTEM.SYS C:\SYSTEM02.SYS

(N)ext page, (P)revious page, (T)op, or PF1 to exit:

```

Figure 8-66 Sample List Scheduled Request Screen

8.8.3 Erase Scheduled Request

The **Erase Scheduled Request (E)** option of the **Job Scheduler (J)** Submenu is used to remove a scheduled job request from the list of scheduled job requests.

When you select this option, SAGE^{MAX} prompts you with **Schedule to Erase:**. Refer to **Figure 8-67**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Job Scheduler:
key          to do
C           Create Scheduled Request
L           List Scheduled Requests
E           Erase Scheduled Request
A           Make "As Soon As Possible" Request
J           Make Job Request Directly
PF1         Return to previous menu
Press key for desired action: E
Schedule to Erase:

```

Figure 8-67 The Erase Scheduled Request Screen

At this prompt you enter the number of the job that you want to remove from the list of scheduled job requests. This number is a 5-digit (maximum) number that precedes each job in the list of scheduled requests (see **Figure 8-66**).

After you enter the job number to be erased, the SAGE^{MAX} attempts to remove the scheduled job. If the job number you specify exists, SAGE^{MAX} displays the **Job Scheduler** Submenu. If the job number you specify does not exist, SAGE^{MAX} displays the message **#BadRecordNumber** and prompts you to **Press any key to continue:**.

8.8.4 Make "As Soon As Possible" Request Submenu

The **Make "As Soon As Possible" Request (A)** option of the **Job Scheduler (J)** Submenu is used to create an immediate job request. This option is identical to the **Create Scheduled Request (C)** option, only you do not have to specify a date and time. The request is made immediately and the job is performed as soon as possible. When you select this option, the **Job Request** Submenu (see **Figure 8-53**) is displayed, bypassing the prompts for scheduling information such as **Day of Month, Month, Year**, etc. From this point on, the process of creating an "as soon as possible" request is the same as creating a scheduled request.

For more information, refer to the individual jobs (Broadcast Message, Data Capture/Stuff, Upload/Download, Export Database Files, etc.) in **Chapter 8.8.1: Create Scheduled Requests (C)**.

8.8.5 Make Job Request Directly

The **Make Job Request Directly (J)** option of the **Job Scheduler (J)** Submenu is used to enter either a scheduled or “as soon as possible” job request in a format that can be directly sent to the Scheduler Task. This is in lieu of being prompted for the individual pieces of information. Samples of this Scheduler Task format can be seen when you **List Scheduled Requests**. Refer to **Figure 8-66**.

When you select this option, SAGE^{MAX} prompts you with **SCHED>**. From this prompt you enter the job request. The information that you enter must be in the proper Scheduler Task format or the job will not run properly.

Figure 8-68 illustrates a series of “as soon as possible” jobs being directly created, followed by a series of scheduled jobs being directly created.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
SCHED>BC 8//Boiler #5 shutdown in 30 minutes...
SCHED>DCS SAMPLE.PDF/S
SCHED>UDL D 1 14/2:SYSTEM.SYS C:\SYSTEM02.SYS
SCHED>EXPORT a:\VARS.NBF
SCHED>RPT C:\MONTHLY.TXT C:\USAGE.TBL C:\MONTHLY
SCHED>SPOOL 13 C:\RPT\ELECTRIC.RPT /B
SCHED>S2SAGE A:\STR2SYS.SYS C:\STR\STR2MSG.TXT/U
SCHED>UDL U 1 14/2:SYSTEM.SYS C:\SYSTEM02.SYS

SCHED>[08,09,1996,?,09:30] BC 8//Boiler #5 shutdown in 30 minutes...
SCHED>[08,09,1996,?,09:30] DCS SAMPLE.PDF/S
SCHED>[08,09,1996,?,09:30] UDL D 1 14/2:SYSTEM.SYS C:\SYSTEM02.SYS
SCHED>[08,09,1996,?,09:30] EXPORT a:\VARS.NBF
SCHED>[08,09,1996,?,09:30] RPT C:\MONTHLY.TXT C:\USAGE.TBL C:\MONTHLY
SCHED>[08,09,1996,?,09:30] SPOOL 13 C:\RPT\ELECTRIC.RPT /B
SCHED>[08,09,1996,?,09:30] S2SAGE A:\STR2SYS.SYS C:\STR\STR2MSG.TXT/U
SCHED>[08,09,1996,?,09:30] UDL U 1 14/2:SYSTEM.SYS C:\SYSTEM02.SYS

```

Figure 8-68 Sample List Scheduled Request Screen

For more information on the syntax of job requests, refer to **Chapter 11.31: Programming - The JOB Statement**.

8.9 The Table Submenu

The **Table (L)** Submenu of the **Main Menu** allows you to create, edit and display SAGE^{MAX} tables. You can also list SAGE^{MAX} tables to files or spool them to a printer. The **Table** Submenu is shown in **Figure 8-69**.

A table is a collection of up to 1,073,741,824 data values which are stored as linear, one-dimensional arrays in disk-resident files. These files cannot exceed 4,294,967,296 bytes in size. Table values can be referenced individually by SPL programs using each value's relative offset as an index in the list.

Internally, tables are stored in one of two ways based on their type. *Full* tables contain a value and a data type for each entry in the table. *Sparse* tables contain values for each entry in the table, but only have a single data type that reflects the data type of all elements of the table. It is more efficient to use a sparse table for table values having the same data type.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Table:
key          to do
E            Edit/Create Table
N            Numerical Display
L            List to File
PF1         Return to previous menu
Press key for desired action:

```

Figure 8-69 The Table Submenu

8.9.1 Edit/Create Table

The **Edit/Create Table (E)** option of the **Table (L)** Submenu is used to create a new SAGE^{MAX} table or edit the entries of a previously created table.

When you select this option, SAGE^{MAX} displays the prompt **Table Name:.** From this prompt you enter the name of the table that you want to create or edit.

If you enter a table name that does not exist, SAGE^{MAX} displays the prompt **Create new table? (Y/N):.** If you type **Y**, SAGE^{MAX} creates the SAGE^{MAX} table with the specified name.

When a new table name is created, you must also decide what type of table you want to create. SAGE^{MAX} prompts you with **Single datatype for entire table? (Y/N).** Type **Y** if you want to define a *sparse* table, or **N** if you want to define a *full* table.

NOTE

Although SAGE^{MAX} tables can contain up to 1,073,741,824 data values, the SAGE^{MAX} table editor can only create/edit up to 65,536 table elements. Tables with more than 65,536 elements can only be manipulated using SPL programs.

Next, the SAGE^{MAX} prompts you with **Initialize table? (Y/N).** Typing **Y** allows you to specify the number of elements in the table and initializes all values to zero. When you initialize a full

table, the data types default to **FEh**. You enter the number of elements in the table from the **Number of entries in table:** prompt. This allows the SAGE^{MAX} to create the proper size table. However, tables are automatically extended if you specify a table entry that is beyond the initial table size. When a table file is auto-extended, the contents of the *extended* elements are not initialized, and may contain bogus data.

If you specify **N** at the **Initialize table? (Y/N)** prompt, the table elements are not initialized, and may contain bogus data, whether or not the file is auto-extended.

Next, the SAGE^{MAX} prompts you with **Entry to edit:**. From this prompt you enter the number of the element that you want to edit. If you type **RETURN**, SAGE^{MAX} displays the table's zero element ready for data entry.

Table information is displayed in three columns. The *Index* column is zero-based and specifies the sequential element number within the table. The *DT* column specifies the hexadecimal data type of each element. In sparse tables, the values in the data type column are the same. The *Value* column specifies the value that is assigned to the particular table element.

NOTE

Although the DT (data type) column is displayed when you edit or display a sparse table, all the values are the same and they cannot be edited.

When entering data into the table, the up and down arrow keys take you to the next and previous table entries. Use the left and right arrow keys to maneuver within a particular entry. Type the **RETURN** key to enter the new data that you type.

To quickly reach a particular entry, use the arrow keys to position the cursor over the index field and type the number of the element that you want to edit followed by **RETURN**. **Figure 8-70** shows the process of creating a sample table on the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Table:
key          to do
E            Edit/Create Table
N            Numerical Display
L            List to File
PF1         Return to previous menu
Press key for desired action: E
Table name: SAMPLE
Create new table? (Y/N): Y
Single datatype for entire table? (Y/N): N
Initialize table? (Y/N): Y
Number of entries in table: 50
Entry to edit: 0
Index | DT | Value
-----
0: [FEh] _           0

```

Figure 8-70 Sample Edit/Create Table Screen

8.9.2 Numerical Display

The **Numerical Display (N)** option of the **Table (L)** Submenu is used to list the elements of tables to the operator interface.

When you select this option, SAGE^{MAX} prompts you for a **Table name:**. This is the name of the table that you want to list to your operator interface. Only specify the name of the table and not the **.TBL** extension.

When you enter a valid table name, the SAGE^{MAX} displays a heading line that contains the filename of the table, the size (number of elements) of the table and the size of each element in the table.

Next, the SAGE^{MAX} displays one page of table elements. The index, data type and value are displayed for each element of the table. You use the **N** (or **+**) and **P** (or **-**) to display next and previous pages if more than one page is needed to display all elements of the table. You type **PF1** to exit the table listing.

Figure 8-71 illustrates a sample numerical display of a SAGE^{MAX} table.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Table: C:\TABLES\SAMPLE.TBL contains 10 entries, 5 bytes per entry.
Index | DT | Value
-----|---|-----
0: [FEh]                10
1: [FEh]                20
2: [FEh]                30
3: [FEh]                40
4: [FEh]                50
5: [FEh]                60
6: [FEh]                70
7: [FEh]                80
8: [FEh]                90
9: [FEh]               100
Press N(+) for next page, P(-) for previous, PF! to escape

```

Figure 8-71 Sample Numerical Display Screen

8.9.3 List to File

The **List to File (L)** option of the **Table (L)** Submenu is used to list the elements of a table to a file and to optionally spool a table file to a SAGE^{MAX} printer port.

When you select this option, SAGE^{MAX} prompts you to **Enter filename:**. This is the name of the file that will contain the listing of table elements.

Next, SAGE^{MAX} asks if you want to **Spool file to a printer port? (Y/N):**. If you type **Y**, SAGE^{MAX} prompts you for a spool port. The port number that you specify must be the port number of a printer port. SAGE^{MAX} also asks if you want to **Delete file after spooling? (Y/N):**. If you type **Y**, SAGE^{MAX} will delete the file after the spool job request is submitted.

The SAGE^{MAX} then asks for the **Table name:**. This is the name of the table file that you want to list to a file. You only specify the name of the table, and not the **.TBL** extension. You may specify a full pathname if you want the file on a directory other than the root directory.

Figure 8-72 shows the process of listing a table to a file.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Table:
key          to do
  E          Edit/Create Table
  N          Numerical Display
  L          List to File
PF1          Return to previous menu
Press key for desired action: L
Enter filename: C:\TABLES\TABLE01.TXT
Spool file to a printer port? (Y/N): Y
Spool on port: 13
Delete file after spooling? (Y/N): N
Table name: SAMPLE
```

Figure 8-72 Sample List to File Screen

8.10 The Monitor/Change Point Attributes Screen

Selecting **M** from the **Main Menu** displays the **Monitor/Change Point Attributes** Screen shown in **Figure 8-73**. It is from this screen that you can monitor or change attributes of points, programs, globals, groups and variables.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
...Press ? for Help
...Press PF1 to return to previous menu

TY\Name to Monitor:

```

Figure 8-73 The Monitor/Change Point Attribute Screen

From the **Monitor/Change Point Attributes Screen** you can type **?** to display the help screen shown in **Figure 8-74**, **PF1** to return to the **Main Menu**, or the name of the object that you want to monitor. The help screen shows the proper format required for monitoring database objects.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
...Press ? for Help
...Press PF1 to return to previous menu

TY\Name to Monitor:?

      - - - - - Help for Monitor - - - - -
TY\Name      for Type and Object Name (GR\Name or \Name for groups)
              Valid Types:  PT, VR, PG, GR, GL
N or +      Next Attribute
P or -      Previous Attribute
(space)     Default Attribute
;           Any Attribute
A or *      All Attributes
$           $$ Attributes for Programs
%           %A Register for Programs
(return)    Update Display
C or &      Continuous on/off
...Press ? for Help
...Press PF1 to return to previous menu

TY\Name to Monitor:

```

Figure 8-74 Monitor/Change Point Attribute - Help Screen

From the **TY\Name to Monitor:** prompt, you enter the name of the database object that you want to monitor. Optionally, this object name may be preceded by the 2-character type code (**PT**, **VR**, **PG**, **GR**, **GL**) and the “\” character. Group object names, however, **must** be preceded by either a “\” character or **GR**. If the type code prefixes **PT**, **VR**, **PG**, and **GL** are omitted, the SAGE^{MAX} searches all object types (except groups) for a name match. The object name may also be optionally followed by a semicolon and an attribute name. If the attribute name is omitted, then the default attribute (“space space”) is assumed.

Figure 8-75 illustrates a sample point object being monitored from the **Monitor/Change Point Attributes** Screen using the object type code, point name and an attribute.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
...Press ? for Help
...Press PF1 to return to previous menu

TY/Name to monitor: PT/STAR4;AH

STAR4;AH= 4464

```

Figure 8-75 Monitoring a Point Object

The **Monitor/Change Point Attributes** Screen can also monitor variable objects, program objects, and program group objects.

When a group is first displayed, the description text associated with each element of the group is displayed in the **to select** field (see **Figure 8-76**). This is the description that you enter for each element when you create the group. If you do not specify a description for a group element, the element name is used in the **to select** field of the group.

If descriptions are assigned to group elements and you want to see the actual element names rather than the descriptions when monitoring a group, press the - (minus) key. This toggles the display of the **to select** field between message text and element name. Refer to **Figure 8-77**.

```

SAGE (1.00) Banner Line Text                           Wed 20-Mar-91 12:00:00
Opr: SYSTEM                                           Port: 07
Group Name: \MAIN GROUP
key           to select
A  STAR 4 SYSTEM POINT                               ;TI= 12:00:00
B  Are there any unack'd alarms?                     ; = No
C  PROGRAM SAMPLE                                   ;$$= 2
PF1 Return to previous menu

```

Figure 8-76 Monitoring a Group Object (Case 1)

```

SAGE (1.00) Banner Line Text                           Wed 20-Mar-91 12:00:00
Opr: SYSTEM                                           Port: 07
Group Name: \MAIN GROUP
key           to select
A  STAR                                             ;TI= 12:00:00
B  $ALARMS                                         ; = No
C  SAMPLE                                           ;$$= 2
PF1 Return to previous menu

```

Figure 8-77 Monitoring a Group Object (Case 2)

Figure 8-78 illustrates a sample global object being monitored from the **Monitor/Change Point Attributes** Screen.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
...Press ? for Help
...Press PF1 to return to previous menu

TY\Name to Monitor: GL\OATEMP

OATEMP      = 39.7
```

Figure 8-78 Monitoring a Global Object

If you request an object from the **TYName to Monitor:** prompt that requires longer than 2 seconds for a response, SAGE^{MAX} will display a spinning icon to denote that it is in the process of fetching your request. This can be seen when you try to monitor a global object from the Ethernet when a timeout occurs. This can also be seen when you try to monitor a remote point that must be fetched over the PHPHDial port of the SAGE^{MAX}.

8.10.1 Monitoring a Specific Attribute

After SAGE^{MAX} displays the default (or specified) attribute that you requested, you can view any other object attribute and its current value by typing a semicolon (;) followed by the two-character attribute name and **RETURN**.

To return to the object's default attribute, you can enter the default attribute name following a semicolon (e.g., ;**CV**), or you can simply type the space character. This is a quick method to return to an object's default attribute.

8.10.2 Monitoring the Next Attribute

When you monitor an object's attribute and current value, you can view the object's next sequential attribute and its current value by typing **N** for **next** (or +).

Most SAGE^{MAX} objects have at least two attributes (variables being the exception, with only a single value for each). You can think of a database object's attributes as being arranged sequentially in a list, with the default attribute at the top of the list. By typing a series of **Ns**, you can sequentially monitor each of the object's attributes and their respective current values. Entering **N** (or +) when you are at the bottom of the list of attributes causes you to wrap around to the first (default) attribute.

8.10.3 Monitoring the Previous Attribute

When you monitor an object's attribute and current value, you can view the object's previous sequential attribute and its current value by typing **P** for **previous** (or -). Entering **P** (or -) when you are at the top of the list of attributes causes you to wrap around to the last attribute.

8.10.4 Monitoring All Attributes

When you monitor an object's attribute and current value, you can view all of the object's attributes and their current values by typing **A** for **all**.

If the object you are monitoring has more attributes than will fit on a single screen, you may want to use the **NO SCROLL** key on VT-type operator interfaces to temporarily stop the scrolling. Typing the **NO SCROLL** key a second time resumes the list of all attributes of the object.

8.10.5 Monitoring Program Attributes

Monitoring program attributes is handled differently because program objects have at least two (and may have three) lists of *attributes* as opposed to a single list. Program objects have program control attributes (they begin with **\$**), user-defined attributes (optional two-character attributes that you create in the program), and program registers (they begin with **%**).

When you first monitor a program object, the default attribute that is displayed depends on the status of the program. If the program is unloaded, then the **\$\$** attribute is displayed. If the program is in any other state, the user-defined attribute that was defined first in the program is displayed. If no user-defined attributes have been defined in the program, the **\$\$** attribute is displayed.

Once a program attribute and its current value are displayed, you can access attributes from any of the other lists of attributes. However, user-defined attributes can only be accessed if they have been defined in the program. Monitoring a specific program attribute is done by typing a semicolon (;) followed by the two-character attribute name and **RETURN**. Examples include **;\$S**, **;**AA**** and **;%C**.

Typing a space character from any attribute (program control, user-defined or program register) returns you to the default attribute (either the first-defined, user-defined attribute or the **\$\$** attribute as explained earlier).

NOTE

Since the first attribute that is displayed when you monitor a loaded program is the first attribute that is defined in the program, you should make your first attribute the most important attribute of the program.

Typing a **\$** character from any attribute (program control, user-defined or program register) returns you to the first attribute of the list of program control attributes of the program (i.e., **\$\$**, **\$1**, **\$D**, etc.).

Typing a **%** character from any attribute (program control, user-defined or program register) returns you to the first attribute of the list of program register attributes of the program (i.e., **%A**, **%B**, **%C**, etc.).

8.10.6 Monitoring in Continuous Mode

When you monitor an object's attribute and current value, the value that is displayed is a snapshot of the attribute value when you made the request. To see the value of an attribute dynamically, type **C** to toggle into **continuous mode**. This causes the screen to continuously update the value of the current attribute. Typing **C** a second time turns off continuous mode.

8.10.7 Modifying Attribute Values

To change the value of an attribute, simply press = when the attribute is displayed. This *opens* the attribute and allows its value to be modified.

NOTE

To modify an attribute's value, the attribute must not be a read-only attribute (e.g., sensor values, etc.) and you must have the proper user privilege to allow such operations.

There are five types of attributes whose values may be modified. These types are:

1. simple numeric
2. indexed EU assignments
3. bitmap EU assignments
4. times
5. floating point

You can modify the value of a *simple numeric* attribute by entering the new value after the equal sign. The value that you enter may be in binary, decimal or hexadecimal format.

The default format is decimal. To modify the value of a simple numeric attribute using decimal format you simply enter the new value using the **0 - 9**, **+**, **-** and **.** keys. Examples include the following:

- 12345
- 123.45
- 0
- -5
- -.75
- -12345

To modify the value of a simple numeric attribute using hexadecimal format you simply enter the new value using the **0 - 9** and **A - F** keys. The lowercase keys (**a - f**) may also be used. The letter **H** or **h** must follow hexadecimal values to signify hexadecimal format.

- FF00h
- FC01H
- 0f7h
- fh
- AH
- fb00H

To modify the value of a simple numeric attribute using binary format you simply enter the new value using the **0** and **1** keys. The letter **B** or **b** must follow binary values to signify binary format.

Attributes may have *indexed engineering units (EU) assignments* associated with them. Indexed EU assignments display different text based on the value of the attribute. When you monitor an attribute that has an indexed EU assignment, the text associated with the current value of the attribute is displayed.

When you press = to modify the value of an attribute with an indexed EU assignment, SAGE^{MAX} displays a list of options. This list contains the valid new values (or *indices*, which are zero-based) and the text associated with each value. You modify the attribute value by selecting the new value or index from the list. **Figure 8-79 (left)** shows an example using a program attribute. Refer to file **C:\EU\PEX.EU** and **Chapter 7: Initial Configuration** for engineering units file format information.

Attributes may have *bitmap EU assignments* associated with them. Bitmap EU assignments may have up to 8 labels, each of which may contain up to 8 characters.

When you press = to modify the value of an attribute with a bitmap EU assignment, SAGE^{MAX} displays a list of bitmap patterns associated with each label. An example using active days is shown in **Figure 8-79 (right)**. The new value that you enter can be in binary format (the same format as the bitmap patterns), decimal or hexadecimal. Refer to **Chapter 7: Initial Configuration** for engineering units file format information.

<pre> SAGE MAX Banner Line Text Opr: SYSTEM SAMPLE;\$1 = 0 Normal = key to select 00 Normal 01 Single Step Enter New Value for ;\$1 (0..01): </pre>	<pre> SAGE MAX Banner Line Text Opr: SYSTEM SAMPLE;AD = MON = pattern to select 00000001b MON 00000010b TUE 00000100b WED 00001000b THU 00010000b FRI 00100000b SAT 01000000b SUN 10000000b HOL Enter New Value for ;AD: 10101b SAMPLE;AD = MON WED FRI </pre>
---	--

Figure 8-79 Modifying an Attribute Having an Indexed EU Assignment (left), or Bitmap EU Assignment (right)

You can modify the value of a *time* attribute by entering the new value after the equal sign. You enter the value using the **0 - 9** and **:** keys. *Long time* values consist of hours, minutes and seconds fields, while *short time* values consist of hours and minutes only. In both cases, the fields are separated by colons. Time examples include the following:

- 12:34:56
- 00:00:00
- 23:59:59
- 12:34
- 00:00
- 23:59

You can modify the value of a *floating point* attribute by entering the new value after the equal sign. For floating point, the value that you enter must be in decimal format.

To modify the value of a floating point attribute you simply enter the new value using the **0 - 9**, **+**, **-**, **.**, and **E** keys. The lowercase **e** key may be used in place of the **E** key.

The general format for floating point numbers is as follows:

- a leading **+** or **-** (optional)
- integer portion of value
- decimal portion of value
- **E** or **e** (optional)
- a **+** or **-** (optional)
- the exponent value (optional)

The **E** (or **e**) is used to separate the significant digits of the value from the exponent or power of ten. For example, the floating point representation of **2,000,000,000** is **2E9** (or **2*10⁹**), while the floating point representation of **0.000000002** is **2E-9** (or **2*10⁻⁹**). If the **E** is not included, the exponent defaults to zero. Since **10⁰=1**, this does not affect the significant digits. Other examples of floating point representations include the following:

- 1
- -1.06
- 1E9
- 2.6e-7
- -123.456789E+17
- 5.56789E-2

8.10.8 *Modifying Attribute Values of Group Elements*

When you monitor group objects on the SAGE^{MAX}, you can select an element of the group by typing the key associated with the element (i.e., **A**, **B**, **C**, ..., **P**). If the group element that you select is another group, SAGE^{MAX} displays the new group. If the group element is some other database object, SAGE^{MAX} displays the object, an attribute and its current value. From this point you can use the **N**ext, **P**revious, **A**ll and **C**ontinuous mode commands, as explained earlier, to view the values of selected attributes. You can also modify an attribute by typing = and using the previous instructions based on the attribute type.

When you monitor or modify any attribute of an object by selecting that object from a group, you can return to the group by typing **G**. This *return* feature is not available when the group element that you select is another group (i.e., a *nested group*) since **G** is a valid element key in groups.

8.10.9 *Logging Changes*

When you create database objects, you can enable a feature that causes a log transaction to occur when any attribute of the object is modified. Log transactions are events that use SAGE^{MAX} class 002. By modifying this class, you can have object modifications logged to **C:\LOG\GENERAL.LOG**, **C:\LOG\002.LOG**, local SAGE^{MAX} ports or remote dialed destinations.

In the case of program objects with the logging feature enabled, attribute changes are logged if the change is performed by an operator or another program. Attribute changes that occur from within the program itself may or may not be logged, depending on the syntax that is used by the statement that does the modification.

If a program attribute is modified using its short form (e.g., **[:SP] = 72.0**) rather than its long form (e.g., **[SAMPLE;SP] = 72.0**), the modification is not logged. Using the long form forces the program name to be looked up in the SAGE^{MAX} database. This look-up takes more time than its short method counterpart, but forces modifications of a program's own attributes to be logged.

8.11 *The Ports Status and Setup Submenu*

Selecting the **Ports Status and Setup (P)** option from the **Main Menu** displays the **Port Setup** Submenu shown in **Figure 8-80**.

From this submenu you can edit port types, communications parameters and driver configurations for SAGE^{MAX} ports including the optional Ethernet port. Included is an option that saves all your changes in the ports configuration file. In addition, there is an option that provides a dynamic display of the status of the SAGE^{MAX} ports.

8.11.1 *Change Port Type*

Selecting **T** from the **Port Setup** Submenu displays the **Change Port Type** Screen. This option is used to assign or change the driver type of SAGE^{MAX} ports 1-12.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Port Setup:
key          to do
T           Change Port Type
C           Change Port Communications Parameters
D           Change Port Driver Configuration
W           Watch Dynamic Port Status
S           Save Changes in Configuration File
E           Change Ethernet Configuration
PF1        Return to previous menu
Press key for desired action:

```

Figure 8-80 The Port Setup Submenu

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Port --Type-- Lan Baud- P D S
01= PUPhost  001  9600 N 8 1
02= Peernet  001  9600 N 8 1
03= None     001  9600 N 8 1
04= PUPhost  001  9600 N 8 1
05= PHPdial  001 38400 N 8 1
06= None     001  9600 N 8 1
07= VT100    000  9600 N 8 1
08= VT100    000  9600 N 8 1
09= VT100    000  9600 N 8 1
10= VT100    000  9600 N 8 1
11= VT100    000  9600 N 8 1
12= VT100    000  9600 N 8 1
Change Port (1-12): 3
Code Port Type
00      None
01      PUPhost
02      Peernet
03      XANPhost
04      Dumb
05      VT100
06      Printer
07      PHPdcon
08      PHPdial
10      PHPHDcon
11      PHPHDial
New Port Type: 3
Are you sure? (Y/N): Y

```

Figure 8-81 Sample Change Port Type Screen

When you select this option, SAGE^{MAX} displays a current list of driver type assignments for ports 1-12. SAGE^{MAX} then prompts you to enter the number of the port whose driver type you want to change. Refer to **Figure 8-81**.

Next, SAGE^{MAX} displays a list of available driver types that can be assigned to the port you selected. This list may be different from the list shown in **Figure 8-81**. SAGE^{MAX} then displays the prompt **New Port Type:**. From this prompt you type the number of the driver type that you want assigned to the port. Refer to **Figure 8-81**. After you select a valid driver type, SAGE^{MAX} prompts you with **Are you sure? (Y/N):** to verify your choice. After you type your choice, SAGE^{MAX} returns to the **Port Setup** Submenu shown in **Figure 8-80**.

IMPORTANT

After you change a port's driver type, you should *Save Changes in Configuration File (S)* (refer to Chapter 8.11.5). This updates the file C:\CFG\PORTS.CFG so that the port you modified will be initialized to the proper driver type if the SAGE^{MAX} is reset or loses power.

8.11.2 Change Port Communications Parameters

Selecting **C** from the **Port Setup** Submenu displays the **Change Port Communications Parameters** Screen. This option is used to assign or change the default language, baud rate, parity, number of data bits, and number of stop bits of ports 1-12.

When you select this option, SAGE^{MAX} displays a current list of communications parameter assignments for ports 1-12. SAGE^{MAX} then prompts you to enter the number of the port whose communications parameters you want to change. Refer to **Figure 8-82**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Port --Type-- Lan Baud- P D S
01= PUPhost   001  9600 N 8 1
02= Peernet   001  9600 N 8 1
03= None      001  9600 N 8 1
04= PUPhost   001  9600 N 8 1
05= PHPdial   001 38400 N 8 1
06= None      001  9600 N 8 1
07= VT100     000  9600 N 8 1
08= VT100     000  9600 N 8 1
09= VT100     000  9600 N 8 1
10= VT100     000  9600 N 8 1
11= VT100     000  9600 N 8 1
12= VT100     000  9600 N 8 1
Change Port (1-12): 3
Baud=110,150,300,600,1200,2400,4800,9600,19200,38400
Language=0 to 255 Parity=N,O,E DataBits=5,6,7 or 8 StopBits=1 or 2
Language,Baud,Parity,DataBits,StopBits: 001,9600,N,8,2
Are you sure? (Y/N): Y

```

Figure 8-82 Sample Change Communications Parameters Screen

Next, SAGE^{MAX} displays a list of available communications parameters that can be assigned to the port you selected. SAGE^{MAX} then displays the prompt **Language,Baud,Parity,DataBits,StopBits:**. From this prompt you type the appropriate communications parameters that you want assigned to the port. Refer to **Figure 8-82**. After you enter your list of parameters (separated by commas), SAGE^{MAX} prompts you with **Are you sure? (Y/N):** to verify your choice. After you type your choice, SAGE^{MAX} returns to the **Port Setup** Submenu shown in **Figure 8-80**.

IMPORTANT

After you change a port's communications parameters, you should *Save Changes in Configuration File (S)* (refer to Chapter 8.11.5). This updates the file C:\CFG\PORTS.CFG so that the port you modified will be initialized to the proper driver type if the SAGE^{MAX} is reset or loses power.

For more information about the communications parameters that are used by SAGE^{MAX} driver types, refer to the individual chapter for the particular driver in question.

8.11.3 Change Port Driver Configuration

This option is used to edit variables and watch performance statistics of particular drivers assigned to ports 1-12.

When you select this option, SAGE^{MAX} displays a current list of driver type assignments for ports 1-12. SAGE^{MAX} then prompts you to enter the number of the port whose driver configuration you want to change.

NOTE

Some SAGE^{MAX} driver types do not have variables that can be edited (e.g., None, Dumb, VT100 and Printer). For this reason, driver configuration submenus are not available for these SAGE^{MAX} driver types.

Depending on the driver type assigned to the port that you choose, SAGE^{MAX} displays a **Change Port Driver Configuration** Submenu. The options of this menu are different for different driver types that have driver configuration submenus. Above the **Port Driver Configuration** Submenu, SAGE^{MAX} displays two message lines that show you the port number of the driver you are configuring and the actual driver type of the port. **Figure 8-83** shows the **Port Driver Configuration** Submenu for a SAGE^{MAX} port (port 1 in this example) that is set up for a **PUPhost** driver configuration.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Change Driver Configuration for Port 1
PUP Configuration:
key          to do
E           Edit Current Configuration
S           Save Changes in Configuration File
M           Make these changes NOW...
W           Watch Performance Statistics
PF1         Return to previous menu
Press key for desired action:

```

Figure 8-83 Change Port Driver Communication Submenu (PUPhost Ports)

Table 8-2 shows the possible commands, and the driver configurations that offer those command on in their respective configuration submenus.

	PUPHost	Peernet	XANPhost	PHPdcon/ PHPdial	PHPHdcon/ PHPHdial
Edit Current Configuration	X	X	X	X	X
Save Changes in Configuration File	X	X	X	X	X
Make these changes NOW...	X	X	X	X	X
Watch Performance Statistics	X		X		X
Upload/Download Unit			X		

Table 8-2 Port Configuration Commands

8.11.3.1 Edit Current Configuration

The **Edit Current Configuration (E)** option of the **Port Driver Configuration** Submenu allows you to edit driver variables for the port that you selected. There are different variables for each particular driver type. The driver variable names and their meanings are discussed in detail in the individual chapter for the particular driver in question.

8.11.3.2 Save Changes in Configuration File

The **Save Changes in Configuration File (S)** option of the **Port Driver Configuration** Submenu causes your driver variable changes to take effect and updates the driver's configuration file for the port you specified. This ensures that the port's driver will be initialized using the proper driver variables if the SAGE^{MAX} is reset or loses power.

SAGE^{MAX} maintains a driver configuration file for each port on the SAGE^{MAX}, providing the port's driver type is not **None**, **Dumb**, **VT100** or **Printer**). These driver configuration files are located on the **C:\CFG** subdirectory. The filename includes the name of the driver type (e.g., **PUP**, **PEER**, **PHP**, **XANP**, **XANPMAP**, etc.), and the extension is the port number (e.g., **.1**, **.2**, **.3**, **.4**, etc.).

8.11.3.3 Make These Changes NOW...

The **Make these changes NOW... (M)** option of the **Port Driver Configuration** Submenu causes your driver variable changes to take effect. This changes the operation of the driver now and updates the port's driver configuration file.

8.11.3.4 Watch Performance Statistics

The **Watch Performance Statistics (W)** option of the **Port Driver Configuration** Submenu allows you to view dynamic driver statistics for host drivers. This option is available for **PHPHdcon**, **XANPhost**, and **PUPhost** driver types.

Figure 8-84 illustrates a sample **Watch Performance Statistics** screen for a SAGE^{MAX} port configured as an **XANPhost**. The screen shows the unit numbers of the XANP devices that are on the network as specified in the **Peer Units XANPhost** driver variable. Each unit shows either the type of device or its current status. The device type/status codes for the **XANPhost** screen have the following meanings:

- ? - unresponsive unit
- 1 - RCU1 or RCU2 pre-version 6.1
- 2 - RCU2 or MCU
- * - STAR
- D - unit is downloading unit data file
- U - unit is uploading unit data file

SAGE MAX Banner Line Text								Mon 23-Sep-96 12:00:00
Opr: SYSTEM				Port: 07				
0=*	1=*	2=*	3=*	4=*	5=*	6=*	7=*	
8=*	9=*	10=*	11=*	12=*	13=*	14=*	15=*	
16=2	17=2	18=2	19=2	20=2	21=2	22=2	23=2	
24=2	25=1	26=1	27=1	28=1	29=U	30=1	31=1	

Figure 8-84 Sample Dynamic Performance Statistics Screen from the Change Driver Configuration Submenu

When viewing the **Watch Performance Statistics** screen for a SAGE^{MAX} port configured as a **PUPhost**, the screen shows the unit numbers of the PUP devices that are on the network as specified in the **Unit#/Peer/Alarm Poll** driver variable. Each unit shows if it is responsive, polling, or polling and token passing. The device type/status codes for the **PUPhost** screen have the following meanings:

- ? - unresponsive unit
- * - SAGE^{MAX} is polling the unit

When viewing the **Watch Performance Statistics** screen for a SAGE^{MAX} port configured as a **PHPHdcon**, the screen shows the unit numbers of the PHP devices that are on the network as specified in the **Unit#/Peer/Alarm Poll** driver variable. Each unit shows if it is unresponsive or responsive and polling. The device type/status codes for the **PHPHdcon** screen have the following meanings:

- ? - unresponsive unit
- * - SAGE^{MAX} is polling the unit

8.11.3.5 Upload Unit

The **Upload Unit (U)** option of the **Port Driver Configuration** Submenu is only available for **XANPhost** drivers and gives you the ability to upload the unit data file of RCUs, RCU2s and MCUs.

When this option is selected, SAGE^{MAX} prompts you with **XANP Unit Number =**. This represents the XANP unit number whose unit data file you want to upload to the SAGE^{MAX}.

8.11.3.6 Download Unit

The **Download Unit (D)** option of the **Port Driver Configuration** Submenu is only available for **XANPhost** drivers and gives you the ability to download the unit data file of RCUs, RCU2s and MCUs.

When this option is selected, SAGE^{MAX} prompts you with **XANP Unit Number =**. This represents the XANP unit number whose unit data file you want to download to the SAGE^{MAX}.

8.11.4 Watch Dynamic Port Status

The **Watch Dynamic Port Status (W)** option of the **Ports Setup** Submenu displays a dynamically updated screen that contains driver types assigned to ports 1-12. With each driver type is a field that shows the operator using the port, the activity that is currently being performed, and statistics information for certain drivers. A sample **Dynamic Ports Status** Screen is shown in **Figure 8-85**.

SAGE MAX Banner Line Text		Port: 07		Mon 23-Sep-96 12:00:00	
Opr: SYSTEM					
Port	-----Operator-----	--Type--	Activity	-----Statistics-----	
01		PUPhost	Listen	t=00	c=00 f=00 o=00
02		Peernet	Listen		
03		XANPhost	Scanning	t=00	c=00 f=00 o=00
04		PHPdcon	PHPIdle		
05	PHPSLAVE	PHPdial	WaitCall		
06	PHPSLAVE	PHPdcon	PHPSlave		
07	SYSTEM	VT100	Ports SU		
08		VT100	Logon		
09		Dumb	Logon		
10		VT100	Logon		
11		VT100	Logon		
12		VT100	Logon		

Figure 8-85 Sample Dynamic Ports Status Screen

The **Operator** field displays the name of the operator using the port. This may be the username of a human operator on a VT100, for example, or the default name **PHPSLAVE** used by ports configured as **PHPdial** or **PHPdcon**.

The **Type** field displays the driver type that is assigned to SAGE^{MAX} ports 1-12.

The **Activity** field displays the current activity of the port. This activity may be **Scanning**, **WaitCall** or **PHPIdle**, or it may be the name of a SAGE^{MAX} menu that an operator is using (e.g., **Ports SU**, **Logon**, etc.).

The **Statistics** field displays communications statistics for certain driver types. The **t** field displays a cumulative timeout counter. The **c** field displays a CRC/check byte error accumulation. The **f** field displays a cumulative framing error count. The **o** field displays a cumulative counter of overrun errors that occur.

The values in these fields can be helpful in troubleshooting or diagnosing problems that result from improper wiring, excessive noise, improper hardware configurations and possible hardware failures.

8.11.5 Save Changes in Configuration File

The **Save Changes in Configuration File (S)** option of the **Ports Setup** Submenu (shown in **Figure 8-80**) is used to update the **C:\CFG\PORTS.CFG** file with any changes you make to driver types or communications parameters of ports 1-12. This provides proper initialization of port driver types and communications parameters if the SAGE^{MAX} is reset or loses power.

IMPORTANT

After you change a port's driver type or a port's communications parameters, you should Save Changes in Configuration File (S). Failure to do this will result in losing any of these changes and may result in improper port configurations if the SAGE^{MAX} is reset or loses power.

8.11.6 Change Ethernet Configuration Submenu

The **Change Ethernet Configuration (E)** option of the **Ports Setup** Submenu is used to configure an optional Ethernet network and watch its dynamic performance statistics.

When you select this option, SAGE^{MAX} displays the **Change Ethernet Configuration** Submenu shown in **Figure 8-86**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Change Driver Configuration for Port 6
Ethernet Configuration:
key           to do
E             Edit Current Configuration
S             Save Changes in Configuration File
M             Make these changes NOW...
W             Watch Performance Statistics
PF1          Return to previous menu
Press key for desired action:
```

Figure 8-86 Change Ethernet Configuration Submenu

8.11.6.1 Edit Current Configuration

The **Edit Current Configuration (E)** option of the **Change Ethernet Configuration** Submenu allows you to edit Ethernet driver variables. The Ethernet variable names and their meanings are discussed in detail in **Chapter 13: High Speed LANs**.

8.11.6.2 Save Changes in Configuration File

The **Save Changes in Configuration File (S)** option of the **Change Ethernet Configuration** Submenu causes your Ethernet driver variable changes to take effect. This option also updates the Ethernet driver's configuration file **C:\CFG\ETH.CFG** to ensure that the Ethernet driver will be initialized properly if the SAGE^{MAX} is reset or loses power.

8.11.6.3 Make These Changes NOW...

The **Make these changes NOW... (M)** option of the **Change Ethernet Configuration** Submenu causes your driver variable changes to take effect. This changes the operation of the Ethernet driver now. In addition, the changes are saved in the configuration file **C:\CFG\ETH.CFG**.

8.11.6.4 Watch Performance Statistics

The **Watch Performance Statistics (W)** option of the **Change Ethernet Configuration** Submenu allows you to view dynamic driver statistics for the Ethernet driver. The values displayed in the statistics fields are in hexadecimal format.

Figure 8-87 illustrates a sample **Watch Performance Statistics** screen for the Ethernet port.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07

RxInt -FCS- Drib -Ovf- Runt  Rx-OK >Max  NoBuf NotUs Coll Coll16 Tx-OK TxDrop
00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000

```

Figure 8-87 Sample Dynamic Performance Statistics Screen for the Ethernet

- The **RxInt** field shows the number of receive interrupts that have been received since the last reset of the SAGE^{MAX}.
- The **-FCS-** field shows the number of framing and checkbyte errors that have occurred since the last reset of the SAGE^{MAX}.
- The **Drib** field shows the number of Ethernet dribble errors that have occurred since the last reset of the SAGE^{MAX}.
- The **-Ovf-** field shows the number of Ethernet overflow errors that have occurred since the last reset of the SAGE^{MAX}.
- The **Runt** field shows the number of short frames (*runts*) that have been received since the last reset of the SAGE^{MAX}.
- The **Rx-OK** field shows the number of good frames that have been received since the last reset of the SAGE^{MAX}.
- The **>Max** field shows the number of received frames that were larger than the SAGE^{MAX} internal buffer. This field is zeroed when the SAGE^{MAX} is reset.
- The **NoBuf** field shows the number of packets that the SAGE^{MAX} received, but did not have room for in the free list queue. These packets were dropped. This field is zeroed when the SAGE^{MAX} is reset.
- The **NotUs** field shows the number of packets that the SAGE^{MAX} received that were not recognizable by the SAGE^{MAX}. This field is zeroed when the SAGE^{MAX} is reset.
- The **Coll** field shows the number of transmit collisions that have occurred since the SAGE^{MAX} was last reset. The **Coll16** field shows the number of times 16 collisions in a row caused a “back off” event.
- The **Tx-OK** field shows the number of successfully completed transmissions since the SAGE^{MAX} was last reset.
- The **Tx-Drop** field shows the number of unsuccessful transmit packets that were dropped due to collision errors after a maximum number of retries occurred. This field is zeroed when the SAGE^{MAX} is reset.

8.12 The Quit Screen

Selecting the **Quit (Q)** option from the **Main Menu** station locks your port and displays the **Sign-on** Screen. From the **Sign-on** Screen, you must enter a valid username and codeword to gain access to the **Main Menu** of the SAGE^{MAX}.

8.13 The System Variables Submenu

Selecting the **System Variables (S)** option from the **Main Menu** displays the **System Variables** Submenu shown in **Figure 8-88**.

From this submenu you can customize a variety of system parameters which are explained in the following sections.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
System Variables:
key           to do
D             Change System Date
T             Change System Time
B             Change Banner Line Text
L             System Language
A             Log Alarm Acknowledgments
P             Change Printer Flag
I             Change Audible Alarm Indicator
PF1          Return to previous menu
Press key for desired action:
```

Figure 8-88 System Variables Submenu

8.13.1 Change System Date

Selecting the **Change System Date (D)** option from the **System Variables** Submenu allows you to change the date that is used by the SAGE^{MAX}. The system date is displayed to the right of the banner line text on all SAGE^{MAX} menus.

When you select this option, the SAGE^{MAX} displays the prompt **New Date (day/month/year)=**. You enter the new system date from this prompt.

The **day** field must be a number from 1-31. The **month** field can be entered in either numeric form (1-12) or 3-character abbreviated form (Jan, Feb, Mar, Apr, etc.). The **year** field can be entered in either 2-digit (92) or 4-digit (1992) format. These fields must be separated by slash characters.

8.13.2 Change System Time

Selecting the **Change System Time (T)** option from the **System Variables** Submenu allows you to change the time that is used by the SAGE^{MAX}. The system time is displayed to the right of the banner line text and date on all SAGE^{MAX} menus.

When you select this option, the SAGE^{MAX} displays the prompt **New Time (hh:mm:ss)=**. You enter the new system time in military format from this prompt.

The **hh** (hour) field must be a number from 00-23. The **mm** (minute) field must be a number from 00-59. The **ss** (second) field must be a number from 00-59. These fields must be separated by colon (:) characters.

8.13.3 Change Banner Line Text

Selecting the **Change Banner Line Text (B)** option from the **System Variables** Submenu allows you to change the banner line text that is used by the SAGE^{MAX}. The banner line text is displayed to the right of the SAGE^{MAX} version number on all SAGE^{MAX} menus.

When you select this option, the SAGE^{MAX} displays the prompt **New Banner Line=**. You enter the new banner line text from this prompt. The banner line text can contain up to 40 characters.

Figure 8-89 shows how to change the banner line text of the SAGE^{MAX}. To see the new banner line text in the header of your screen, you must type **PF1** to refresh the screen.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
System Variables:
key          to do
D            Change System Date
T            Change System Time
B            Change Banner Line Text
E            Edit System Variables
PF1         Return to previous menu
Press key for desired action: B
New Banner Line: Fast Eddie's Commercial Widget Factory

```

Figure 8-89 Changing the Banner Line Text

8.13.4 Edit System Variables

Selecting the **Edit System Variables (E)** option from the **System Variables** Submenu scrolls through the list of system variables and allows you to make changes to them. **Figure 8-90** shows the complete list of system variables that you can edit.

To edit the system variable, type in the new value and press **RETURN**. To scroll past the system variable without editing it, press the down arrow key. Press **PF1** to return to the previous menu.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
System Variables:
key           to do
D             Change System Date
T             Change System Time
B             Change Banner Line Text
E             Edit System Variables
PF1          Return to previous menu
Press key for desired action: E
System Language Code:                                [ 0]:
ErrorWait Retry Interval:                            [ 5]:
Log Alarm Acknowledgements?:                        [No ]:
Log Object Changes made by PEX?:                    [No ]:
Toss LPT Alarms if printer is offline?:             [No ]:
Beep Terminals if Unacked Alarms?:                  [No ]:
Recognize & Filter JIS characters?:                  [No ]:
Timer for Auto-Scroll of Front Panel Display (secs): [ 0]:
Timeout for Front Panel Activity (secs):              [ 60]:
Enable Daylight Savings Time Function? (Y/N):        [No ]
Date on or after which Daylight Savings starts (mmm/ddd): [ 0 / 0 ]
Weekday on which Daylight Savings starts (0=Sun, 6=Sat): [ 0 ]
Date on or after which Daylight Savings ends (mmm/ddd): [ 0 / 0 ]
Weekday on which Daylight Savings ends (0=Sun, 6=Sat): [ 0 ]
Is it Daylight Savings Time now? (Y/N):              [No ]:

```

Figure 8-90 System Variables

8.14 The Trend Submenu

The **Trend (T)** Submenu of the **Main Menu** offers options that allow you to create/modify, initialize, activate, deactivate, list, display, and erase trend objects. There is also a translation option for converting PHP-style trend files to SAGE^{MAX} format. The **Trend** Submenu is shown in **Figure 8-91**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Trends:
key           to do
A             Activate Trend
C             Create/Modify Trend
D             Deactivate Trend
E             Erase Trend Object
F             List to File
L             List Trend Objects
I             Initialize Trend
N             Numerical Display
S             Stripchart Display
X             Translate PHP Trend Object
PF1          Return to previous menu
Press key for desired action:

```

Figure 8-91 The Trend Submenu

Before you continue with the options of the **Trend** Submenu, you may want to read **Chapter 19: Historical Data Trending**. Chapter 19 gives a detailed explanation of the types of trends that you can create on the SAGE^{MAX}, as well as some of the features of trending and some of the terminology that is used.

8.14.1 Activate Trend

The **Activate Trend (A)** option of the **Trend** Submenu is used to start a trend. Although a trend has been started, this does not necessarily mean that the trend has begun collecting data immediately.

Once a trend has been activated, it may take up to one minute before the first sample is collected. For activated *time-anchored trends*, the first sample may not be taken for one minute, 15 minutes, 30 minutes, 60 minutes, a day, a week, a month, or even a year, depending on the anchor interval of the trend.

When the **Activate Trend (A)** option is selected, SAGE^{MAX} prompts you with **Trend name:.** From this prompt you enter the name of the trend that you want to activate. The trend name can be up to 8 characters maximum.

After you enter a valid trend name, SAGE^{MAX} activates the trend and displays the **Trend** Submenu. If an invalid trend name is specified, SAGE^{MAX} displays the error message **#[21]FileNotFound** and prompts you to **...Press any key to continue:.**

8.14.2 Create/Modify Trend

The **Create/Modify Trend (C)** option of the **Trend** Submenu is used to create a new SAGE^{MAX} trend or modify a previously created SAGE^{MAX} trend.

When you select this option, SAGE^{MAX} prompts you for the **Trend name:.** This can be the name of a new trend or a previously created trend.

Next, the SAGE^{MAX} prompts you for a **Trend type (0-21)**. If you are creating a new trend, you enter the numeric code for the type of trend that you want to create or use the default of zero. If this is a previously created trend, you can enter the new type code for the trend or you can just type **RETURN** and keep the old trend type. SAGE^{MAX} displays a complete list of trend types.

The SAGE^{MAX} then prompts you to choose either an averaging or snapshot mode of data sampling with **Averaging (0=Snapshot, 1=Averaged)**. You enter **0** for snapshot mode or **1** for averaging mode.

Next, SAGE^{MAX} prompts you for a **Sample Interval (minutes)**. This is how frequently you want samples to be taken. For *time-anchored* trends (types 02-21), the sample interval is determined automatically, based on the type that you choose.

The **Sample failure alarm limit** prompt allows you to specify a number of unsuccessful fetches of references that must occur before a trend alarm is generated. Trend alarms use SAGE^{MAX} class 005 to determine how and where they are to be reported.

The **Max samples** prompt allows you to specify the total number of samples that will be allowed for the trend.

For *infinite* trends, this number acts as a ceiling for the number of samples that the trend can take. This is extremely important for infinite trends since their size is variable and disk space is dynamically allocated automatically (as needed). Without **Max samples**, an infinite trend could dynamically allocate the entire SAGE^{MAX} hard drive over time.

For *circular* trends, **Max samples** is the number of samples that occur before the trend wraps back to the first sample.

For *time-anchored* trends, **Max samples** is determined automatically, based on the trend type (02-21) that you choose.

Next, the SAGE^{MAX} prompts you for 8 trend references. These references are labeled **Reference A - Reference H**. References may contain point, variable, global and program objects. If no attribute is specified in the reference, the default attribute is assumed.

Figure 8-92 shows the process of creating/editing a trend object.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Trend name: TREND001
Trend type (0-21) = 0
00=Infinite,          01=Circular,          02=Daily_01_Min,    03=Daily_15_Min,
04=Daily_30_Min,     05=Daily_60_Min,    06=Weekly_01_Min,  07=Weekly_15_Min,
08=Weekly_30_Min,   09=Weekly_60_Min,  10=Weekly_1440_Min, 11=Monthly_01_Min,
12=Monthly_15_Min,  13=Monthly_30_Min,  14=Monthly_60_Min,  15=Monthly_1440_Min,
16=Yearly_15_Min,   17=Yearly_30_Min,  18=Yearly_60_Min,   19=Yearly_1440_Min,
20=Yearly_week,     21=Yearly_month.
Trend type (0-21) = 1
Averaging (0=Snapshot, 1=Averaged) = 0
Averaging (0=Snapshot, 1=Averaged) = 1
Sample Interval (minutes) = 0
Sample Interval (minutes) = 10
Sample failure alarm limit = 0
Sample failure alarm limit = 5
Maximum samples = 0
Maximum samples = 144
Reference A =
Reference A = GL\OATEMP;CV
Reference B =
Reference B = ZONE1;SP
Reference C =
Reference C = $MODE
Reference D =
Reference D = PG\RESET;SP
Reference E =
Reference E = RESET;ER
Reference F =
Reference F =
Reference G =
Reference G =
Reference H =
Reference H =

```

Figure 8-92 Creating/Editing a Trend

After the last reference, SAGE^{MAX} displays the message **Warning!!! All trend samples will be lost. Update trend? (Y/N):**. Typing **Y** causes the trend to be initialized. This means that the trend object is updated (or created if it is a new trend) and any previously collected samples will be zeroed. Initializing a trend also deactivates the trend. After your response, SAGE^{MAX} displays the **Trend** Submenu.

IMPORTANT

*After you create/edit a trend and the **Trend** Submenu is displayed, you must select the **A** option and specify the trend name to activate the trend you just created/edited.*

8.14.3 Deactivate Trend

The **Deactivate Trend (D)** option of the **Trend** Submenu is used to stop a trend from collecting any further samples.

When this option is selected, SAGE^{MAX} prompts you with **Trend name:**. From this prompt you enter the name of the trend that you want to deactivate. The trend name can be up to 8 characters maximum.

After you enter a valid trend name, SAGE^{MAX} deactivates the trend and displays the **Trend** Submenu. If an invalid trend name is specified, SAGE^{MAX} displays the error message **#[21]FileNotFound** and prompts you to **...Press any key to continue:**.

8.14.4 Erase Trend Object

The **Erase Trend Object (E)** option of the **Trend** Submenu is used to delete a trend object from the SAGE^{MAX} database.

When this option is selected, SAGE^{MAX} prompts you with **Trend name:**. From this prompt you enter the name of the trend object that you want to remove from the SAGE^{MAX} database. The trend name can be up to 8 characters maximum.

After you enter a valid trend name, SAGE^{MAX} deletes the trend and displays the **Trend** Submenu. If an invalid trend name is specified, SAGE^{MAX} displays the error message **#[21]FileNotFound** and prompts you to **...Press any key to continue:**.

8.14.5 List to File

The **List to File (F)** option of the **Trend** Submenu is used to send trend output to a file that you specify. In addition, you can spool the file to a SAGE^{MAX} printer port and optionally have the file deleted after it is spooled.

When this option is selected, SAGE^{MAX} prompts you with **Enter filename:**. From this prompt you enter the filename you want to contain the trend output.

Next, SAGE^{MAX} prompts you with **Spool file to a printer port? (Y/N)**. If you respond with **Y**, SAGE^{MAX} prompts you for a printer port number and asks if you want the file to be deleted after it is spooled.

The SAGE^{MAX} then displays a submenu from which you can choose either a **Numerical Display (N)** or a **Stripchart Display (S)**. After you make your selection, SAGE^{MAX} prompts you with **Trend name:**. From this prompt you enter the name of the trend whose output you want to list to a file. The trend name can be up to 8 characters maximum.

After you enter a valid trend name, SAGE^{MAX} formats the trend output, writes the output to the file that you specified, and displays the **Trend Submenu**. If an invalid trend name is specified, SAGE^{MAX} displays the error message **#[21]FileNotFound** and prompts you to **...Press any key to continue:**.

The process of listing trend output to a file is shown in **Figure 8-93**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Enter filename: C:\OUTPUT.TXT
Spool file to a printer port? (Y/N): Y
Spool on port: 13
Delete file after spooling? (Y/N): N

```



```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Trends:
key          to do
N            Numerical Display
S            Stripchart Display
PF1         Return to previous menu
Press key for desired action: N
Trend name: TREND001
Formatting trend output. Please wait...

```

Figure 8-93 Listing a Trend to a File

8.14.6 List Trend Objects

The **List Trend Objects (L)** option of the **Trend Submenu** is used to list all SAGE^{MAX} trend names and trend statistics associated with each trend.

Included with each trend name is the current status of the trend (Active or Inactive), the type of trend (Infinite, Circular, Daily, Weekly, etc.), whether or not averaging mode is being used, the sample interval, the sample count, the accumulated number of errors, and the current sample.

When this option is selected, SAGE^{MAX} prompts you with an 8-character wild card template for the trend name. From this prompt you enter the name of the trend whose statistics you want to list. You can simply type **RETURN** at the wild card prompt for a list of all SAGE^{MAX} trends, or enter selected characters in the name field for a partial list of SAGE^{MAX} trends.

Next, the trend name and statistics information are displayed one page at a time. If your SAGE^{MAX} database contains more than one page of trend objects, you can view consecutive pages by typing any key. A sample trend list is shown in **Figure 8-94**.

```

SAGE MAX Banner Line Text          Mon 23-Sep-96 12:00:00
Opr: SYSTEM                        Port: 07
Name---- Status-- Type---- Avg Intvl Samplecnt- Error Cursample-
TREND001 Inactive Infinite No      1      25      0      25
TREND002 Active   Circular Yes    2      10      0      5
TREND003 Active   Daily   No     1     1440    0      2
...Press any key to continue, Press PF1 to return to previous menu

```

Figure 8-94 Sample List of Trend Objects

8.14.7 Initialize Trend

The **Initialize Trend (I)** option of the **Trend** Submenu is used to initialize a trend that you specify.

After you select this option, SAGE^{MAX} displays the message **Warning!!! All trend samples will be lost. Update trend? (Y/N):**. Typing **Y** causes the trend to be initialized. This means that any previously collected samples are set to zero, current sample is set to zero, and the trend is deactivated. After your response, SAGE^{MAX} displays the **Trend** Submenu.

8.14.8 Numerical Display

The **Numerical Display (N)** option of the **Trend** Submenu is used to list trend output to your operator interface in a numerical format. The numerical display shows the time, date and value of the information that was referenced for each of the eight possible references. This option also shows the name of the trend, when it was activated, the type of trend, how many samples have been collected so far, the sample interval, how many failures have occurred and the low and high values of all samples that have been collected so far.

When you select this option, SAGE^{MAX} prompts you with **Trend name:**. From this prompt you enter the name of the trend whose output you want displayed. The trend name can be up to 8 characters maximum.

If you specify an invalid trend name, SAGE^{MAX} displays the message **#NotCollected.**, and instructs you to **...Press any key to continue:**.

If you specify a trend that has not yet collected any data, SAGE^{MAX} displays the error message **#NotCollected**, and instructs you to **...Press any key to continue:**.

If a valid trend name with sampled data is specified, SAGE^{MAX} displays the message **Formatting trend output. Please wait.**

Next, the SAGE^{MAX} displays the trend output in numerical format. A sample is shown in **Figure 8-95**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Trend SAMPLE beginning on 03-20-91 11:30 is Daily.
  1440 samples were collected every 1 minutes with 0 failures.
A = GL\OATEMP;CV                                     Low = 64.5 High = 79.5
B = ZONE1;SP                                         Low = 66.8 High = 74.6
C = $MODE                                           Low = 5 High = 9
D = PG\RESET;SP                                     Low = 67.8 High = 69.2
E = RESET;ER                                         Low = 0.4 High = 4.2
-----
03-20-91 11:30 A = 65.1
03-20-91 11:30 B = 72.0
03-20-91 11:30 C = 5
03-20-91 11:30 D = 68.0
03-20-91 11:30 E = 0.4
03-20-91 11:31 A = 64.8
03-20-91 11:31 B = 71.8
03-20-91 11:31 C = 5
03-20-91 11:31 D = 68.5
03-20-91 11:31 E = 0.8

Press N(+) for next page, P(-) for previous, PF1 to escape

```

Figure 8-95 Numerical Display of a Sample Trend

If there is more information than can fit on a single screen, SAGE^{MAX} prompts you with instructions on how to page between screens.

8.14.9 Stripchart Display

The **Stripchart Display (S)** option of the **Trend** Submenu is used to list trend output to your operator interface in a stripchart format. Stripchart display includes all the information shown in numerical displays, except that the referenced data values are replaced with a scaled grid and a marker that shows where each referenced value lies within the scale.

When you select this option, SAGE^{MAX} prompts you with **Trend name:.** From this prompt you enter the name of the trend whose output you want displayed. The trend name can be up to 8 characters maximum.

If you specify an invalid trend name, SAGE^{MAX} displays the message **#NotCollected.**, and instructs you to **...Press any key to continue:.**

If you specify a trend that has not yet collected any data, SAGE^{MAX} displays the error message **#NotCollected**, and instructs you to **...Press any key to continue:.**

If a valid trend name with sampled data is specified, SAGE^{MAX} displays the message **Formatting trend output. Please wait.**

Next, the SAGE^{MAX} displays the trend output in stripchart format. A sample is shown in **Figure 8-96.**

If there is more information than can fit on a single screen, SAGE^{MAX} prompts you with instructions on how to page between screens.

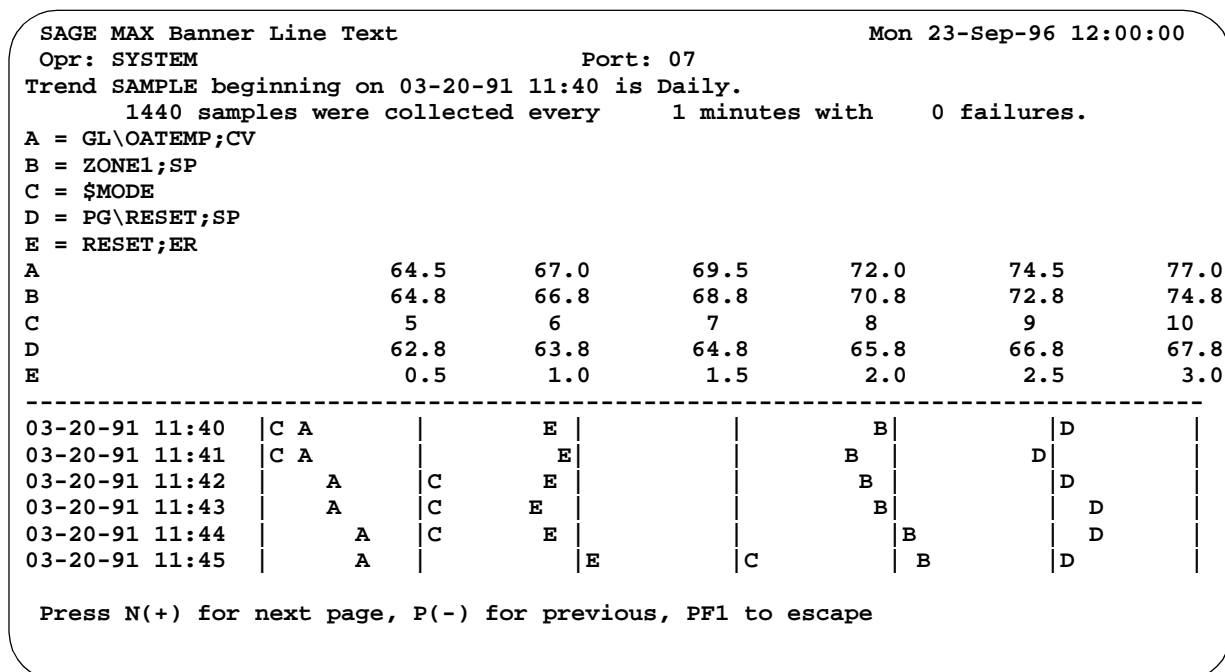


Figure 8-96 Stripchart Display of a Sample Trend

NOTE

Stripchart displays are only valid for references that have data types other than time or date data types.

The stripchart display prints a separate scale for each reference that is used in the trend. For example, if a trend uses five references, then five scales will be printed.

8.14.10 Translate PHP Trend Object

The **Translate PHP Trend Object (X)** option of the **Trend** Submenu is used to convert a PHP-style trend (e.g., a trend created on a STAR or RCU2) to a SAGE^{MAX} trend.

When you select this option, SAGE^{MAX} prompts you to **Enter PHP trend pathname:**. From this prompt you enter the full pathname of the PHP trend you want to convert. For example, this might be a trend that you uploaded to the SAGE^{MAX} from an RCU2 over an XANP network.

If you specify an invalid trend name, SAGE^{MAX} displays the message **#[21]FileNotFound**, and instructs you to **...Press any key to continue:**. If you specify an invalid pathname, SAGE^{MAX} displays the message **#[21]PathNotFound**, and instructs you to **...Press any key to continue:**.

Next, SAGE^{MAX} prompts you for the new **Trend name:**. From this prompt you enter the new SAGE^{MAX} trend name (8 characters maximum) that you want to assign to the converted trend.

After you enter a valid trend name, SAGE^{MAX} converts the trend. When the conversion is complete, SAGE^{MAX} displays the **Trend** Submenu.

Figure 8-97 illustrates the process of converting a sample PHP trend named **XYZ.TRN** to a SAGE^{MAX} trend named **NEW.TRN**. In this illustration, the PHP trend was uploaded to the SAGE^{MAX} and stored on the subdirectory **C:\UNIT31**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Trends:
key           to do
A            Activate Trend
C            Create/Modify Trend
D            Deactivate Trend
E            Erase Trend Object
F            List to File
L            List Trend Objects
I            Initialize Trend
N            Numerical Display
S            Stripchart Display
X            Translate PHP Trend Object
PF1         Return to previous menu
Press key for desired action: X
Enter PHP trend pathname: C:\UNIT31\XYZ
Trend name: NEW

```

Figure 8-97 Translating a PHP Trend Object

8.15 The Utility Functions Submenu

The **Utility Functions (U)** Submenu of the **Main Menu** allows you to perform various file maintenance operations.

From this submenu you can list a directory of the files on the SAGE^{MAX} **A:**, **C:**, and **D:** drives, change directories, type the contents of a file to the screen, edit a file, copy a file from one location to another (e.g., from the **C:** drive to the **A:** drive), delete a file, make a directory, delete a directory, prepare the SAGE^{MAX} for maintenance (shutdown), perform remote file copying to and from network devices, display a file in hexadecimal and ASCII format, use the SAGE^{MAX} math calculator, and format a diskette.

The **Utility Functions (U)** Submenu is shown in **Figure 8-98**.

Most of the options of this submenu are file oriented. When these options are selected, you are usually asked to enter a file pathname (e.g., the pathname of the file you want to type, to edit, to copy, to delete, etc.). From this prompt, you may enter the entire pathname of the file you want to act on.

SAGE^{MAX} also gives you a second option. When you select the **Utility Functions (U)** Submenu of the **Main Menu**, the SAGE^{MAX} selects **C:** as the *default* pathname. This default pathname can be changed (using the **Change Directory (G)** option) to any valid SAGE^{MAX} pathname while you remain in the **Utility Functions (U)** Submenu. This makes it unnecessary to enter the full pathname repeatedly when performing several file operations from the same subdirectory. It is only necessary that you change the default pathname once.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Directory = [C:\]
File/Directory Utilities:
key           to do
D             Directory
G             Change Directory
T             Type File
E             Edit File
C             Copy File
L             Delete File
S             Search for File
M             Make Directory
K             Delete Directory
Z             Prepare System for Maintenance
R             Remote Copy
H             Display File as Hex/ASCII
U             Math Calculator
F             Format diskette
PF1          Return to previous menu
Press key for desired action:

```

Figure 8-98 The Utility Functions Submenu

The current pathname is displayed above the **Utility Functions (U)** Submenu. It defaults to **C:** when you first enter this submenu, but can be changed using the **Change Directory (G)** option which is discussed later in this section.

8.15.1 Directory

The **Directory (D)** option of the **Utility Functions** Submenu shows a list of the files that reside on a SAGE^{MAX} pathname that you can specify. If you do not specify a particular pathname, the default pathname (displayed at the top of the **Utility Functions** Submenu) is used.

When you select this option, SAGE^{MAX} displays the prompt **Enter [drive:]pathname:**. From this prompt you may optionally enter the drive name and/or the pathname whose files you want to list.

If you simply type **RETURN** at this prompt, SAGE^{MAX} will list the files from the pathname specified by the default pathname (displayed at the top of the **Utility Functions** Submenu).

Figure 8-99 shows a sample directory of the initial **C:\EU** subdirectory of the SAGE^{MAX}.

The directory screen shows the names and extensions of files, parent and subdirectories, file sizes, the times and dates when the files were created, the amount of available disk space, and file type codes.

File type codes are used to define the nature of certain files. There are four types of codes that can be displayed within a SAGE^{MAX} directory. The **A** code means that the file has not been archived. The **R** code means that the file is a read-only file. The **S** code means that the file is a system code. The **H** code means that the file is a hidden file (although it is displayed anyway on the SAGE^{MAX}). SAGE^{MAX} files may contain none, any, or all of the codes mentioned above.


```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Directory of C:\EU.
.                <DIR>                                13-Feb-91  12:34:56
..               <DIR>                                13-Feb-91  12:34:56
PEX.EU          546  A13-Feb-91  12:34:56
                546 Byte(s) in  3 File(s)
[ 32180224] bytes available on disk
...Press any key to continue:

```

Figure 8-99 Sample Directory Screen

These codes, if present, are displayed to the right of the size field in the directory.

For more information about files and directories, refer to **Chapter 3: Fundamental Concepts**.

8.15.2 Change Directory

The **Change Directory (G)** option of the **Utility Functions** Submenu is used to change the default pathname (displayed at the top of the **Utility Functions** Submenu).

When you select this option, SAGE^{MAX} displays the prompt **Enter pathname:**. From this prompt you may enter the drive name and/or the pathname to which you want to change.

If you simply type **RETURN** at this prompt, the SAGE^{MAX} pathname will remain unchanged.

8.15.3 Type File

The **Type File (T)** option of the **Utility Functions** Submenu is used to type (list) a file to your screen.

When you select this option, SAGE^{MAX} displays the prompt **Enter pathname:**. From this prompt you may enter the drive name, the pathname, and filename of the file you want to list to the screen.

If you do not specify a drive or pathname, SAGE^{MAX} will assume the file is located on the directory specified by the default pathname (displayed at the top of the **Utility Functions** Submenu).

Figure 8-100 shows a sample type file screen. This example illustrates the standard engineering units file for SPL programs (**C:\EU\PEX.EU**).

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                             Port: 07
;Sample EU table for PEX $X attributes
$$/Stopped /Running /Unloaded /Abort!! /
$$/Time Delay /Restart /Load Request /Unload Request /
$$/Abort Request /Network Access /
$I/Normal /Single Step /
$D=Seconds Delay
$S=Section
$E=Error Code
$C=Program Counter
$N=Attributes
$Z=PCB Segment
$L=PLB Segment
$R=PRB Segment
$A=PAB Segment
$I=ITM Segment
$P=Stack Pointer

...Press any key to continue, Press PF1 to return to previous menu

```

Figure 8-100 Sample Type File Screen

8.15.4 Edit File

The **Edit File (E)** option of the **Utility Functions** Submenu is used to make changes to SAGE^{MAX} text files. This option provides access to the SAGE^{MAX} text editor.

When you select this option, SAGE^{MAX} displays the prompt **Enter pathname:**. From this prompt you may optionally enter the drive name, the pathname, and filename of the file you want to edit.

After you enter a filename, SAGE^{MAX} displays the SAGE^{MAX} text editor screen. This screen consists of the two standard SAGE^{MAX} header lines, a third status line, the first 20 lines of the file (these are blank if this is a new file you are editing), and a help line at the bottom of the screen. These components are shown in **Figure 8-21**.

The status line shows the version of the SAGE^{MAX} editor software, the current text entry mode (insert or replace), and the file name that you are editing.

From the text editor screen you can begin entering or editing text. Maneuvering through the file is done using the standard arrow keys. VT220 keyboards have special keys that can be used to perform special functions with the SAGE^{MAX} text editor. These keys include the **Find, Insert, Remove, Select, Prev Screen, Next Screen, and Help** keys. For a complete list of these keys and their functions within the editor, refer to **Chapter 6.0: The SAGE^{MAX} Menu System**.

It is not necessary to have a VT220 keyboard in order to take advantage of these functions. The SAGE^{MAX} text editor has a complete set of functions to facilitate the text entry and editing process. These functions are called *edit commands* and are usually performed by preceding the command letter with **PF1**. The edit commands are shown in **Figure 8-22**.

If you type **PF2** from the text editor screen, SAGE^{MAX} display a help screen that shows how to use the SAGE^{MAX} text editor and the meanings of the edit commands. The SAGE^{MAX} **Text Editor Help** screen is shown in **Figure 8-23**.

8.15.5 Copy File

The **Copy File (C)** option of the **Utility Functions** Submenu is used to copy SAGE^{MAX} files.

When you select this option, SAGE^{MAX} displays the prompt **Enter source [drive:]pathname:**. From this prompt you may optionally enter the drive name, the pathname, and filename of the file you want to copy. This is called the source file.

Next, SAGE^{MAX} prompts you with **Enter destination [drive:]pathname:**. From this prompt you may optionally enter the drive name, the pathname, and a filename for the copied file. This is called the destination file.

SAGE^{MAX} then prints the message **Copying File C:\sourcename to C:\destinationname**. SAGE^{MAX} then prompts you to **...Press any key to continue:**.

Figure 8-101 shows the process of copying a file from the **A:** drive to a subdirectory on the **C:** drive.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Directory = [C:\]
File/Directory Utilities:
key          to do
D            Directory
G            Change Directory
T            Type File
E            Edit File
C            Copy File
L            Delete File
S            Search for File
M            Make Directory
K            Delete Directory
Z            Prepare System for Maintenance
R            Remote Copy
H            Display File as Hex/ASCII
U            Math Calculator
F            Format diskette
PF1         Return to previous menu
Press key for desired action: C
Enter source [drive:]\pathname: A:\TREND1.TRN
Enter destination [drive:]\pathname: C:\TREND\TREND1.TRN
Copying File A:\TREND1.TRN
           to C:\TREND\TREND1.TRN.
...Press any key to continue:
```

Figure 8-101 Copying a File

8.15.6 Delete File

The **Delete File (L)** option of the **Utility Functions** Submenu is used to delete SAGE^{MAX} files.

When you select this option, SAGE^{MAX} displays the prompt **Enter [drive:]\pathname:**. From this prompt you may optionally enter the drive name, the pathname, and filename of the file you want to delete.

After you enter a valid filename, SAGE^{MAX} prompts you with **Delete filename ? (Y/N):**. From this prompt you confirm whether or not you want the specified filename to be deleted.

Figure 8-102 shows the process of deleting a SAGE^{MAX} file.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Directory = [C:\]
File/Directory Utilities:
key           to do
D             Directory
G             Change Directory
T             Type File
E             Edit File
C             Copy File
L             Delete File
S             Search for File
M             Make Directory
K             Delete Directory
Z             Prepare System for Maintenance
R             Remote Copy
H             Display File as Hex/ASCII
U             Math Calculator
F             Format diskette
PF1          Return to previous menu
Press key for desired action: L
Enter [drive:]\pathname: C:\TREND\TREND1.TRN
Delete C:\TREND\TREND1.TRN ? (Y?N): Y
File C:\TREND\TREND1.TRN Deleted
...Press any key to continue:

```

Figure 8-102 Deleting a File

8.15.7 Search for File

The **Search for File (S)** option of the **Utility Functions** Submenu is used to search the entire SAGE^{MAX} hard drive (i.e., all subdirectories) for filenames or file fragments that you specify.

When you select this option, SAGE^{MAX} displays the prompt **Enter file search string:**. From this prompt you enter the name or name fragment of the file(s) you want to locate on the SAGE^{MAX} hard drive. You may enter the wild card characters * and ? within this search string.

The question mark is a position-sensitive wildcard. Used in the search string, it means that any valid character can occupy the position(s) it occupies. For example, **AUG???.TXT** represents all five-character **.TXT** files that have **AUG** as their first three characters.

The asterisk used in the search string means that any valid character(s) can occupy its position or any remaining positions in the filename. For example, **SPL*.BOB** represents all **.BOB** files that begin with **SPL** such as **SPLOSS.BOB**, **SPLMX.BOB**, **SPLLITES.BOB** and **SPL.BOB**. Similarly, **SPL*.*** represents all files, regardless of their extensions, that begin with **SPL**. The special wildcard ***.*** represents all files of the current directory.

After you enter a file search string, SAGE^{MAX} searches the entire hard drive to find the file(s) that you specified. If SAGE^{MAX} finds files, they are displayed with their full pathnames, when they were created, and the size of each file. An example is shown in **Figure 8-103**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Directory = [C:\]
File/Directory Utilities:
key           to do
D             Directory
G             Change Directory
T             Type File
E             Edit File
C             Copy File
L             Delete File
S             Search for File
M             Make Directory
K             Delete Directory
Z             Prepare System for Maintenance
R             Remote Copy
H             Display File as Hex/ASCII
U             Math Calculator
F             Format diskette
PF1          Return to previous menu
Press key for desired action: S
Enter file search string: TREND*

           0 31-Jan-9112:49:50C:\TREND
          4071 30-Jan-9111:30:58C:\EXE\TREND.EXE
           962 11-Feb-9113:56:58C:\TREND\TREND002.TRN
          65312 12-Feb-9112:26:56C:\TREND\TREND003.TR$
          91140 05-Feb-9109:20:20C:\UPLOAD\TREND1.TRN
[           161485] Bytes Total Size
...Press any key to continue:

```

Figure 8-103 Searching for a File

8.15.8 Make Directory

The **Make Directory (M)** option of the **Utility Functions** Submenu is used to create a subdirectory on the SAGE^{MAX} hard drive or on the diskette in drive **A:**.

When you select this option, SAGE^{MAX} displays the prompt **Enter [drive:]pathname:**. From this prompt you enter the name of the directory or subdirectory that you want to create.

Figure 8-104 shows the process of making a subdirectory in the **C:\TREND** directory.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Directory = [C:\]
File/Directory Utilities:
key          to do
D            Directory
G            Change Directory
T            Type File
E            Edit File
C            Copy File
L            Delete File
S            Search for File
M            Make Directory
K            Delete Directory
Z            Prepare System for Maintenance
R            Remote Copy
H            Display File as Hex/ASCII
U            Math Calculator
F            Format diskette
PF1         Return to previous menu
Press key for desired action: M
Enter [drive:]\pathname: C:\TREND\UNIT007

Directory C:\TREND\UNIT007. created!
...Press any key to continue:
```

Figure 8-104 Making a Directory

8.15.9 Delete Directory

The **Delete Directory (K)** option of the **Utility Functions** Submenu is used to remove a directory or subdirectory from the SAGE^{MAX} hard drive or from the diskette in drive **A:**.

When you select this option, SAGE^{MAX} displays the prompt **Enter [drive:]pathname:**. From this prompt you enter the name of the directory or subdirectory that you want to delete.

After you specify a valid directory or subdirectory name, SAGE^{MAX} prompts you to confirm the deletion.

8.15.10 Prepare System for Maintenance

The **Prepare System for Maintenance (Z)** option of the **Utility Functions** Submenu is used to prepare the SAGE^{MAX} for maintenance operations such as backup and hard drive optimization procedures. Such operations require that the SAGE^{MAX} be shut down while they are being performed.

IMPORTANT

The Prepare System for Maintenance (Z) option can only be selected by users on SAGE^{MAX} port number 7. Selecting this option from any other SAGE^{MAX} port causes the following message to be displayed: Invalid port for maintenance operations!

When you select this option, SAGE^{MAX} confirms your selection by displaying the prompt **Shutdown system for maintenance? Are you sure? (Y/N):**. If you respond **Y**, the SAGE^{MAX} begins its shutdown procedure.

After you select this option, a brief moment passes and the **Backup Utilities** Submenu is displayed. The options available in this menu are discussed in depth in **Chapter 8.16: The Backup Utilities Submenu**.

After the desired maintenance functions have been performed, you exit the **Backup Utilities** Submenu. This causes SAGE^{MAX} to display the message **Restarting system...please wait**.

NOTES

Before you select this option, you must be sure that there is no diskette in disk drive A:/ of the SAGE^{MAX}.

If you Prepare System for Maintenance with a diskette in drive A:/, SAGE will prompt you to remove it before the Backup Utilities Submenu is displayed. This precaution ensures that the SAGE will reboot properly when you exit the Backup Utilities Submenu.

8.15.11 Remote Copy

The **Remote Copy (R)** option of the **Utility Functions** Submenu is used to upload and download files to and from devices on the SAGE^{MAX} networks.

When you select this option, SAGE^{MAX} displays the **Remote File Copy** Submenu as shown in **Figure 8-105**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Remote File Copy:
key           to do
F             Copy from Network Device
T             Copy to Network Device
PF1          Return to previous menu
Press key for desired action:

```

Figure 8-105 The Remote File Copy Submenu

Selecting either of these options causes the SAGE^{MAX} to display the prompt **Port# for peer:**. From this prompt you enter the SAGE^{MAX} port number of the unit whose file you want to upload (receive) or download (send).

Next, SAGE^{MAX} prompts you for the **Remote device unit#:**. From this prompt you enter the unit number of the device whose file you want to upload (receive) or download (send).

The SAGE^{MAX} then displays the prompt **Remote device drive:\pathname:**. From this prompt you enter the proper remote path of the file that you want to upload or download. The format of this remote path varies based on the driver type of the port in question. For more

information on the remote path syntax for different driver types, refer to the individual section of the driver in question.

Next, the SAGE^{MAX} displays the prompt **Local drive:\pathname:**. From this prompt you enter the SAGE^{MAX} pathname you want to assign to the file that you are uploading or the name of the SAGE^{MAX} file that you want to download.

The SAGE^{MAX} then begins transferring records, showing you a dynamic display of the status of the remote upload or download.

Figure 8-106 shows the process of uploading a trend file from an RCU2 on an XANP network to the SAGE^{MAX}. **Figure 8-107** shows the process of downloading a database file to an RCU2 on an XANP network from a SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Remote File Copy:
key           to do
  F           Copy from Network Device
  T           Copy to Network Device
  PF1        Return to previous menu
Press key for desired action: F
Port# for peer: 3
Remote device unit#: 31
Remote device drive:\pathname: 0:TREND1.TRN
Local drive:\pathname: C:\TREND\TREND1.TRN
File byte number: [    640] ... Press PF1 to cancel transfer
...Press any key to continue:

```

Figure 8-106 Remote Copy from a Network Device

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Remote File Copy:
key           to do
  F           Copy from Network Device
  T           Copy to Network Device
  PF1        Return to previous menu
Press key for desired action: T
Port# for peer: 3
Remote device unit#: 30
Remote device drive:\pathname: 3:SYSTEM.SYS
Local drive:\pathname: C:\UNIT003\SYS.SYS
File byte number: [   36042] ... Press PF1 to cancel transfer
...Press any key to continue:

```

Figure 8-107 Remote File Copy to a Network Device

8.15.12 Display File as Hex/ASCII

The **Display File as Hex/ASCII (H)** option of the **Utility Functions** Submenu is used to display the contents of a file one record at a time (1 record = 256 bytes) in its hexadecimal and ASCII formats.

When you select this option, SAGE^{MAX} prompts you to **Enter [drive:]pathname:**. From this prompt you enter the name of the file (with optional drive and subdirectory) that you want to view.

After you select a valid filename, SAGE^{MAX} displays the first 256 bytes of the file in hexadecimal and ASCII formats. If the file is larger than 256 bytes, SAGE^{MAX} displays a message explaining how to maneuver throughout the file.

Figure 8-108 shows the first 256 bytes (record 0) of file **C:\READ.ME** of the SAGE^{MAX} displayed as hexadecimal and ASCII.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Record [      0]-C:\READ.ME

-Rel Byte----- Hex Values ----- --ASCII Values--
 00H 53 41 47 45 20 28 31 2E 30 33 29 09 33 30 2D 4A  SAGE (1.03).30-J
 10H 61 6E 2D 39 32 0D 0A 0D 0A 54 68 69 73 20 6D 61  an-92....This ma
 20H 73 74 65 72 20 64 69 73 6B 20 63 6F 6E 74 61 69  ster disk contai
 30H 6E 73 20 53 41 47 45 20 50 72 6F 64 75 63 74 69  ns SAGE Producti
 40H 6F 6E 20 53 6F 66 74 77 61 72 65 2C 20 61 73 20  on Software, as
 50H 77 65 6C 6C 20 61 73 20 61 0D 0A 53 41 47 45 20  well as a..SAGE
 60H 52 65 62 75 69 6C 64 20 70 72 6F 63 65 64 75 72  Rebuild procedur
 70H 65 20 28 42 4C 44 58 29 2E 20 54 68 69 73 20 69  e (BLDX). This i
 80H 73 20 61 20 62 6F 6F 74 61 62 6C 65 20 4D 53 44  s a bootable MSD
 90H 4F 53 20 64 69 73 6B 20 77 68 69 63 68 0D 0A 73  OS disk which..s
 A0H 68 6F 75 6C 64 20 6F 6E 6C 79 20 62 65 20 75 73  hould only be us
 B0H 65 64 20 77 69 74 68 20 53 41 47 45 2E 20 0D 0A  ed with SAGE. ..
 C0H 0D 0A 09 44 4F 20 4E 4F 54 20 42 4F 4F 54 20 54  ...DO NOT BOOT T
 D0H 48 49 53 20 44 49 53 4B 20 4F 4E 20 41 20 52 45  HIS DISK ON A RE
 E0H 47 55 4C 41 52 20 50 43 21 0D 0A 09 54 48 45 20  GULAR PC!...THE
 F0H 50 52 4F 47 52 41 4D 53 20 4F 4E 20 54 48 49 53  PROGRAMS ON THIS

Press Key: N(+) next, P(-) previous, B begin, E end, PF1(Esc) new prompt

```

Figure 8-108 Sample Hex/ASCII Display of a File

8.15.13 Math Calculator

The **Math Calculator (U)** option of the **Utility Functions** Submenu is used to access the SAGE^{MAX} calculator.

The SAGE^{MAX} calculator allows you to perform a variety of mathematics functions using mathematical, unary, logical, shift, transcendental, compare, and convert operators.

When you select this option, SAGE^{MAX} prompts you for five pieces of information: the data type and value of the first term, the type of operator, and the data type and value of the second term.

When you are prompted for a data type, you must enter a valid SAGE^{MAX} data type code. If you enter an invalid data type code (or if you enter ? for help) SAGE^{MAX} displays a help screen containing the valid SAGE^{MAX} data types. This is illustrated in **Figure 8-109**. For more information about data types used by the SAGE^{MAX}, refer to **Appendix H: PUP Data Types**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
      SAGE Calculator (FPemu)

Enter 1st term data type: ?

Valid SAGE data types:
value      means      value      means      value      means
00 hex byte      0E7h      Time HH:MM:SS 0F4h      xxxxx.xxxxx
01 hex word      0E8h/0E9h Bymap/Chmap 0F5h      +xxxxx.xxxxx
02 hex dword     0EAh      .xxxxxxxxxxx 0F6h      xxxxxx.xxxx
03 segment:offset 0EBh      +.xxxxxxxxxxx 0F7h      +xxxxxx.xxxx
04 DOSTime       0ECh      x.xxxxxxxxxxx 0F8h      xxxxxxx.xxx
05 day of week   0EDh      +x.xxxxxxxxxxx 0F9h      +xxxxxxx.xxx
06 asciz string  0EEh      xx.xxxxxxxxxxx 0FAh      xxxxxxxxx.xx
07 no/yes        0EFh      +xx.xxxxxxxxxxx 0FBh      +xxxxxxxx.xx
0E0h float      0F0h      xxx.xxxxxxxx 0FCh      xxxxxxxxx.x
0E4h PUPDate    0F1h      +xxx.xxxxxxxx 0FDh      +xxxxxxxxxx.x
0E5h BCD        0F2h      xxxx.xxxxxxxx 0FEh      xxxxxxxxxx.
0E6h Time HH:MM 0F3h      +xxxx.xxxxxxxx 0FFh      +xxxxxxxxxxx.

Enter 1st term data type :

```

Figure 8-109 The SAGE^{MAX} Calculator Data Type Help Screen

When you are prompted for an operator, you must enter a valid SAGE^{MAX} operator code. If you enter an invalid operator code (or if you enter ? for help) SAGE^{MAX} displays a help screen containing the valid SAGE^{MAX} operator codes. This is illustrated in **Figure 18-110**. These operators have the same functions as their SAGE^{MAX} Programming Language counterparts. For more information about operator types, refer to **Chapter 11: Programming**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
      SAGE Calculator (FPemu)

Enter 1st term data type: 0FFh
Enter 1st term value      : 109

Type operator: ?

Operators Available:

MATH:      + - * / mod ** srt ave
UNARY:     neg abs int rnd
LOGICAL:and or xor not
SHIFT:     shl shr
TRANS:     sin cos tan atn log ln
COMPARE:min max bet cmp
CONVERT:fix flt typ

Type operator:

```

Figure 8-110 The SAGE^{MAX} Calculator Operator Type Help Screen

Figure 8-111. shows a complete example illustrating the use of the SAGE^{MAX} calculator.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
      SAGE Calculator (FPemu)

Enter 1st term data type: 0FDh
Enter 1st term value   : 291.6

      Type operator: /

Enter 2nd term data type: 0FFh
Enter 2nd term value   : 4

FDH/00000B64H / FFH/00000004H = FDH/000002D9H [      72.9]

Enter 1st term data type:

```

Figure 8-111 Using the SAGE^{MAX} Calculator

8.15.14 Format Diskette

The **Format Diskette (F)** option of the **Utility Functions** Submenu is used to format a 1.44M high density diskette in drive **A:**.

NOTE

Although SAGE^{MAX} can read from and write to 720K double density diskettes, only 1.44M high density diskettes can be formatted by the SAGE^{MAX}.

When you select this option, SAGE^{MAX} prompts you with the confirmation message **Format drive A: ? (Y/N):**. If you type **Y**, SAGE^{MAX} prompts you to **Insert disk and press any key to continue**.

Next, SAGE^{MAX} displays the number of tracks to be formatted as well as a dynamic display of the current track being formatted. To abort the format process, press the **ESC** key.

When the formatting is completed, SAGE^{MAX} displays the number of bytes available on the diskette and displays the message **Format another? (Y/N):**. If you type **Y** from this prompt, SAGE^{MAX} repeats the entire process. If you type **N**, the **Utility Functions** Submenu is displayed.

Figure 8-112 shows the process of formatting a diskette in the **A:** drive.

CAUTION

Formatting a diskette will completely erase everything from the disk media.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Format drive A: ? (Y/N): Y
Insert disk and press any key to continue
Formatting drive A:.. Press ESC to cancel.
Formatting track [79] of [ 80]
[ 1457664] bytes available on disk
Format another? (Y/N):
```

Figure 8-112 Formatting a Diskette

8.16 The Backup Utilities Submenu

The **Backup Utilities** Submenu is a SAGE^{MAX} submenu that offers system administration functions such as system backup, hard disk optimization and integrity check, and importing name bindings files.

In order to perform such administrative functions without effecting the operation of the SAGE^{MAX}, it is necessary for the SAGE^{MAX} to be *shut down* when any of these operations are performed. This shutdown occurs automatically when you select the **Prepare System for Maintenance (Z)** option from the **Utility Functions** Submenu.

After you select the **Prepare System for Maintenance (Z)** option, there is a brief pause, and the **Backup Utilities** Submenu is displayed. This is shown in **Figure 8-113**.

NOTE

Before you select the Prepare System for Maintenance (Z) option, you must be sure that there is no diskette in disk drive A: of the SAGE^{MAX}. If you Prepare System for Maintenance with a diskette in drive A:, SAGE^{MAX} will prompt you to remove it before the Backup Utilities Submenu is displayed. This precaution ensures that the SAGE^{MAX} will reboot properly when you exit the Backup Utilities Submenu.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Preparing system for maintenance. Please wait...

Backup Utilities:
key          to do
A           Archive SAGE Files
B           Backup SAGE Database
R           Restore SAGE Database
I           Import Name Bindings
V           Verify Disk Integrity
O           Optimize Disk
Q           Exit Maintenance Shell
Press key for desired action:
```

Figure 8-113 The Backup Utilities Submenu

8.16.1 Archive SAGE^{MAX} Files

The **Archive SAGE^{MAX} Files (A)** option of the **Backup Utilities** Submenu is used to make archival copies of particular SAGE^{MAX} files.

IMPORTANT

After you archive a SAGE^{MAX} file, the source file is deleted.

When this option is selected, you are prompted to enter the source drive, pathname, and filename(s) of the file(s) you wish to archive.

Next, SAGE^{MAX} prompts you for the destination drive, pathname, and filename(s) for the archive file you are creating.

SAGE^{MAX} then confirms that you want to perform the archive function, then displays the **Backup Utilities** Submenu.

Figure 8-114 shows the process of archiving the general alarm file of the SAGE to the **D:** drive of the SAGE^{MAX}.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Preparing system for maintenance. Please wait...

Backup Utilities:
key          to do
A           Archive SAGE Files
B           Backup SAGE Database
R           Restore SAGE Database
I           Import Name Bindings
V           Verify Disk Integrity
O           Optimize Disk
Q           Exit Maintenance Shell
Press key for desired action: A

Enter source [drive:]\pathname: C:\GENERAL.LOG
Enter destination [drive:]\pathname: D:\MAR2091.LOG

Archive C:\GENERAL.LOG ? (Y/N): Y
```

Figure 8-114 Archiving SAGE^{MAX} Files

8.16.2 Backup SAGE^{MAX} Database

The **Backup SAGE Database (B)** option of the **Backup Utilities** Submenu is used to backup files.

When you select this option, SAGE^{MAX} prompts you to enter the destination drive and pathname where you want to save the backup files. Typically, you will backup your SAGE^{MAX} database on one or more 1.44M high density diskettes.

After you specify a valid destination drive and pathname, SAGE^{MAX} displays a series of prompts that ask if you want to include certain components of the SAGE^{MAX} database. You may include each component in the backup by selecting **Y**. If you do not want particular components backed up, type **N** at the appropriate prompts.

The five components of the SAGE^{MAX} database that can be backed up are:

- database configuration files
- engineering units files
- groups
- programs
- trends

The *database configuration files* option backs up binary object files, language-independent message text files, port configuration files, task configuration files, priority queues, front panel menu files and PHP site definition files whose names match the following wild card fragments:

- \CFG*.BOB
- \EXE*.MSG
- \CFG*.CFG
- \CFG*.*
- \CFG*.Q
- \MNU*.MNU
- \SITE*.SDF

The *engineering units files* option backs up all engineering units files on the SAGE^{MAX}. The names of these files match the following wild card fragment:

- \EU*.EU

The *groups* option backs up group files that reside on the **\GROUPS** directory or any of its subdirectories, and whose names match the following wild card fragments:

- **\GROUPS*.GRP**
- **\GROUPS\subdi*.GRP**
- **\GROUPS\subdi\subdi*.GRP**

The *programs* option backs up program files (source files, PLBs, PRBs, INIs and tables). Source files reside on the **\SPL** directory or any of its subdirectories. PLB files reside on the **\LOGIC** directory or any of its subdirectories. PRB files reside on the **\REF** directory or any of its subdirectories. INI files reside on the **\INIT** directory or any of its subdirectories. Table files reside on the **\TABLES** directory or any of its subdirectories. The program file names must match the following wild card fragments:

- **\SPL*.SPL**
- **\SPL\subdi*.SPL**
- **\SPL\subdi\subdi*.SPL**
- **\LOGIC*.PLB**
- **\LOGIC\subdi*.PLB**
- **\LOGIC\subdi\subdi*.PLB**
- **\REF*.PRB**
- **\REF\subdi*.PRB**
- **\REF\subdi\subdi*.PRB**
- **\INIT*.INI**
- **\INIT\subdi*.INI**
- **\INIT\subdi\subdi*.INI**
- **\TABLES*.TBL**
- **\TABLES\subdi*.TBL**
- **\TABLES\subdi\subdi*.TBL**

The *trends* option backs up both active (**.TR\$**) and inactive (**.TRN**) trend files whose names match the following wild card fragments:

- **\TREND*.TRN**
- **\TREND*.TR\$**

Figure 8-115 shows the process of backing up database components to a diskette in the **A:** drive of the SAGE^{MAX}.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                             Port: 07

Enter destination [drive:]\pathname: A:\
Include database files? (Y/N): Y
Include engineering units files? (Y/N): Y
Include groups? (Y/N): N
Include programs? (Y/N): N
Include trends? (Y/N): N

Determining storage requirements...please wait
[ 1] 1.44M disks required for [ 224634] bytes in [ 22] files.
Proceed? (Y/N): Y
Copying File C:\CFG\CLASS.BOB
      to A:\CFG\CLASS.BOB
Copying File C:\CFG\POINT.BOB
      to A:\CFG\POINT.BOB
      .
      .
      .
Copying File C:\EU\PEX.EU
      to A:\EU\PEX.EU

...Press any key to continue:
```

Figure 8-115 Backing up a SAGE^{MAX} Database

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                             Port: 07

Enter source [drive:]\pathname: A:\
Include database files? (Y/N): Y
Include engineering units files? (Y/N): Y
Include groups? (Y/N): N
Include programs? (Y/N): N
Include trends? (Y/N): N
Press any key to begin restore...

Copying File A:\CFG\CLASS.BOB
      to C:\CFG\CLASS.BOB
Copying File A:\CFG\POINT.BOB
      to C:\CFG\POINT.BOB
      .
      .
      .
Copying File A:\EU\PEX.EU
      to C:\EU\PEX.EU

...Press any key to continue:
```

Figure 8-116 Restoring a SAGE^{MAX} Database

8.16.3 Restore SAGE^{MAX} Database

The **Restore Database (R)** option of the **Backup Utilities** Submenu is used to restore SAGE^{MAX} database files that were previously backed up.

When you select this option, SAGE^{MAX} prompts you to enter the source drive and pathname of the files you want to restore. Typically, you will restore your SAGE^{MAX} database from one or more 1.44M high density diskettes.

After you specify a valid source drive and pathname, SAGE^{MAX} displays a series of prompts that ask if you want to include certain components of the SAGE^{MAX} database. You may include each component that you want to be restored by selecting **Y**. If you do not want particular components to be restored, type **N** at the appropriate prompts.

Figure 8-116 shows the process of restoring database components from a diskette in the **A:** drive of the SAGE^{MAX}.

8.16.4 Import Name Bindings

The **Import Name Bindings (I)** option of the **Backup Utilities** Submenu is used to load and convert externally created name bindings text files (**.NBF**) to their appropriate **.BOB** files for the SAGE^{MAX} database. For information on the structure of name binding files, refer to **Appendix J: Name Binding Files**.

When you select this option, SAGE^{MAX} prompts you to enter the name bindings filename. From this prompt you enter the name of the ASCII text **.NBF** file that you want to import to the SAGE^{MAX} database.

Next, SAGE^{MAX} asks if you want to display any name redefinitions. If you select **Y** and the **.NBF** file contains one or more names that already exist in the SAGE^{MAX} database, they will be displayed on the screen.

When the import and conversion are complete, SAGE^{MAX} prompts you to **Press any key to continue:**, after which the **Backup Utilities** Submenu is displayed.

8.16.5 Verify Disk Integrity

The **Verify Disk Integrity (V)** option of the **Backup Utilities** Submenu is a hard drive diagnostic test that checks for hard disk problems such as lost clusters.

When you select this option, SAGE^{MAX} displays the hard drive's volume name, the date the volume was created, any diagnostic messages, and hard drive file statistics.

A sample disk verification is illustrated in **Figure 8-117**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Preparing system for maintenance. Please wait...

Backup Utilities:
key          to do
A           Archive SAGE Files
B           Backup SAGE Database
R           Restore SAGE Database
I           Import Name Bindings
V           Verify Disk Integrity
O           Optimize Disk
Q           Exit Maintenance Shell
Press key for desired action: V
Volume SAGE100      created Jan 31, 1991 12:48p

33419264 bytes total disk space
 55296 bytes in 4 hidden files
 32768 bytes in 16 directories
1171456 bytes in 121 user files
32159744 bytes available on disk

655360 bytes total memory
573824 bytes free

..Press any key to continue:
```

Figure 8-117 Verifying Disk Integrity

8.16.6 Optimize Disk

The **Optimize Disk (O)** option of the **Backup Utilities** Submenu is a feature that re-organizes and compresses the data on your hard drive so that it is accessible in the most efficient manner.

IMPORTANT

Before you attempt to optimize, you should back up your hard disk.

When you select this option, SAGE^{MAX} displays a message warning you to backup your hard disk before compression.

Pressing any key starts the disk optimization compression. During the process, several screens are displayed showing the progress of the optimization. When completed, disk statistics are displayed and the **Backup Utilities** Submenu is displayed.

8.16.7 Exit Maintenance Shell

The **Exit Maintenance Shell (Q)** option of the **Backup Utilities** Submenu is how you exit and return the SAGE^{MAX} to normal operation.

When this option is selected, the SAGE^{MAX} is rebooted and you are prompted with the Sign-on Menu.

8.17 The Virtual Terminal Screen

The **Virtual Terminal (V)** option of the **Main Menu** allows you to connect a virtual terminal to a host or peer unit (e.g., STAR, RCU2, SOLOFone, and SAGE^{MAX}) on a SAGE^{MAX} port.

When you select this option, SAGE^{MAX} prompts you for a port number and a unit number. You enter the SAGE^{MAX} port number and the unit number of the device to which you want to connect.

If a virtual terminal can be established, the sign-on screen of the remote device is displayed. From this point you can sign on to the device just as you would if your terminal were connected directly to its operator interface.

The process of connecting a virtual terminal session is shown in **Figure 8-118**.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                             Port: 07
...Press PF1 to return to previous menu

Port/Unit: 3/31
...Connection Successful, use PF4 or CTRL-Z to Exit Virtual Terminal

Remote RCU2 Number 31v6.1 Station Locked               Mon 23-Sep-96 12:00:00
Opr:                                                    Unit: 31

Enter your User Identification:
```

Figure 8-118 Entering Virtual Terminal Mode

To exit the virtual terminal session, you must type either **PF4** or **CTRL-Z**. This returns you to the SAGE^{MAX} **Virtual Terminal** prompt.

CHAPTER 9 - THE FRONT PANEL MENU SYSTEM

This section describes the user sign-on procedure and operation of the LCD User Menu Program (LUM) in the SAGE^{MAX}. This User Menu Program operates upon a menu file that is structured to conform to the 40-character by 4-line LCD display on the SAGE^{MAX}. Typically, the menu file consists of two lines of the 40-character text description relating to the object that is monitored, followed by the object name.

The LUM Program provides a convenient and easy method for retrieving the description and the real-time values of objects in an installation. The LUM Program also provides you with the ability to change the values of objects in an installation. You may choose to divide your installation into sections and create separate menus to monitor each of these sections.

The keypad/display combination is an option when you purchase a SAGE^{MAX}. For this reason, your SAGE^{MAX} may not have these features, in which case this section may be skipped.

9.1 *The Keypad and Display*

The front panel of the SAGE^{MAX} has a keypad and an LCD display screen through which you can monitor and/or modify preselected attributes of database objects. Refer to **Figure 9-1**.

The display screen is backlit and has four rows with 40 columns in each row (4x40). The contrast of the LCD display screen is adjustable using the LCD display contrast control on the right side of the SAGE^{MAX} cabinet.

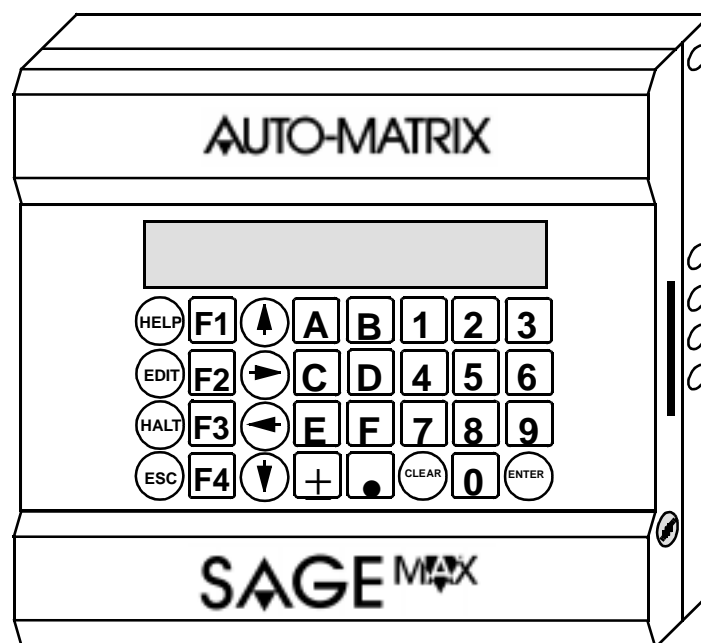


Figure 9-1 Front View of the SAGEMAX Enclosure Showing Keypad and LCD Display

The keypad is below the LCD display and contains 32 keys which include numerals **0-9**, hexadecimal numerals **A-F**, a **CLEAR** key, an **ENTER** key, a **.** (decimal point) key, a **±** (plus/minus) key, four arrow keys (**↑**, **↓**, **←**, **→**), four special function keys (**F1-F4**), a **HELP** key, an **EDIT** key, a **HALT** key and an **ESC** (escape) key (See **Figure 9-1**).

9.2 Description of Operation

The LUM Program prompts you to enter a code number corresponding to a particular menu file, terminating the code with the **ENTER** key.

NOTE

*If the code number is eight digits in length, pressing the **ENTER** key is not required. The SAGE^{MAX} will automatically read the code.*

If the code number is recognized, the appropriate menu file is read and the two-line object text description, together with the specified object value, are displayed. If the code is not recognized, an error message is displayed, prompting you to enter another code.

The LUM Program uses the following keys found on the SAGE^{MAX} front panel to manipulate the display of the object values.

The numerals **0-9** are the valid code number digits that are used to access menu files from the front panel. These numbers are also used to change the values of attributes in the menu files from the front panel.

The **CLEAR** key is used to erase characters that you input and restart the input process. This key can also be used to zero an attribute value or to respond negatively to a yes/no prompt.

The **ENTER** key is used to terminate user input. It is also used to respond positively to a yes/no prompt.

The **.** (decimal point) key is used when you are changing an attribute value that requires a decimal point.

The **±** (plus/minus) key is used when you change an attribute value. This key toggles the sign of the value between “-” for negative and “ ” (blank) for positive for an attribute with a signed data type.

The **↑** (up arrow) key is used to scroll to the previous menu file entry. The **↓** (down arrow) key is used to scroll to the next menu file entry. The **←** and **→** (left and right arrow) keys are used for cursor positioning when changing the values of attributes that use bit map or binary data types.

The **HELP** key is used to display a help screen on the front panel display. Typically, help screens define the uses of the menu keys, including the special function keys **F1-F4**.

After you select an attribute on the display screen, you use the **EDIT** key to enter edit mode, so that the value of the attribute can be modified. Attributes can be configured through menu files such that they cannot be edited (e.g., read only) from the front panel display screen.

The **ESC** key on the front panel of the SAGE^{MAX} is used the same way the **ESC** key is used on a VT100 terminal. **ESC** is used to cancel the current operation or to return to the previous screen.

9.3 Creating Menu (.MNU) Files

Before you can access attributes from the front panel of the SAGE^{MAX}, you must create menu files that contain information about the attributes you wish to be accessible.

Menu files are ASCII text files that must be located on the C:\MNU subdirectory of the SAGE^{MAX}. Menu file names may have up to eight digits (“0”-“9” only) and must have .MNU as the file extension. You may create the menu files with the SAGE^{MAX} text editor or any other ASCII text editor.

The menu file contains a collection of one or more three-line groups or *object entries*. Each object entry consists of two 40-character text description lines followed by the named object in line three. This is shown in **Figure 9-2**. The menu file may also contain optional comment lines which are preceded by a semicolon. Comment lines are not displayed on the LCD.

The following is the syntax for the third line of the object entry (refer to **Figure 9-2**):

- an optional asterisk (*) in column one that gives you the ability to modify the attribute from the front panel
- an optional two-character object type code (refer to **Table 9.1**) that is delimited with backslash (\) characters
- the object name which can be up to 24 characters in length

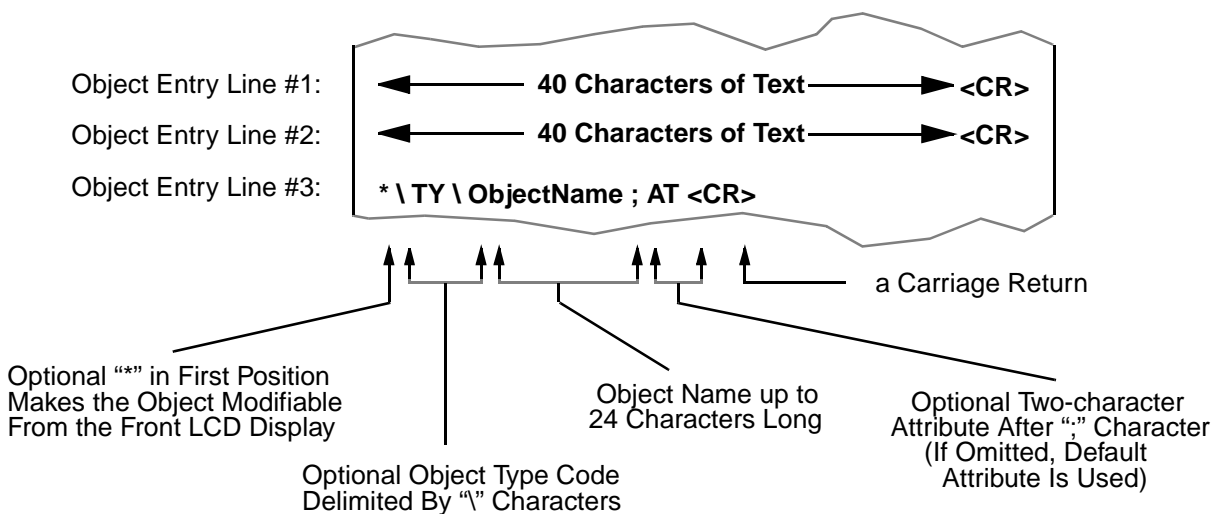


Figure 9-2 Syntax of an Object Entry in a Menu File

- an optional two-character attribute (following a semicolon) which, if left blank, will denote the default attribute
- a carriage return (<CR>).

The text from a sample menu file is shown in **Figure 9-3**.

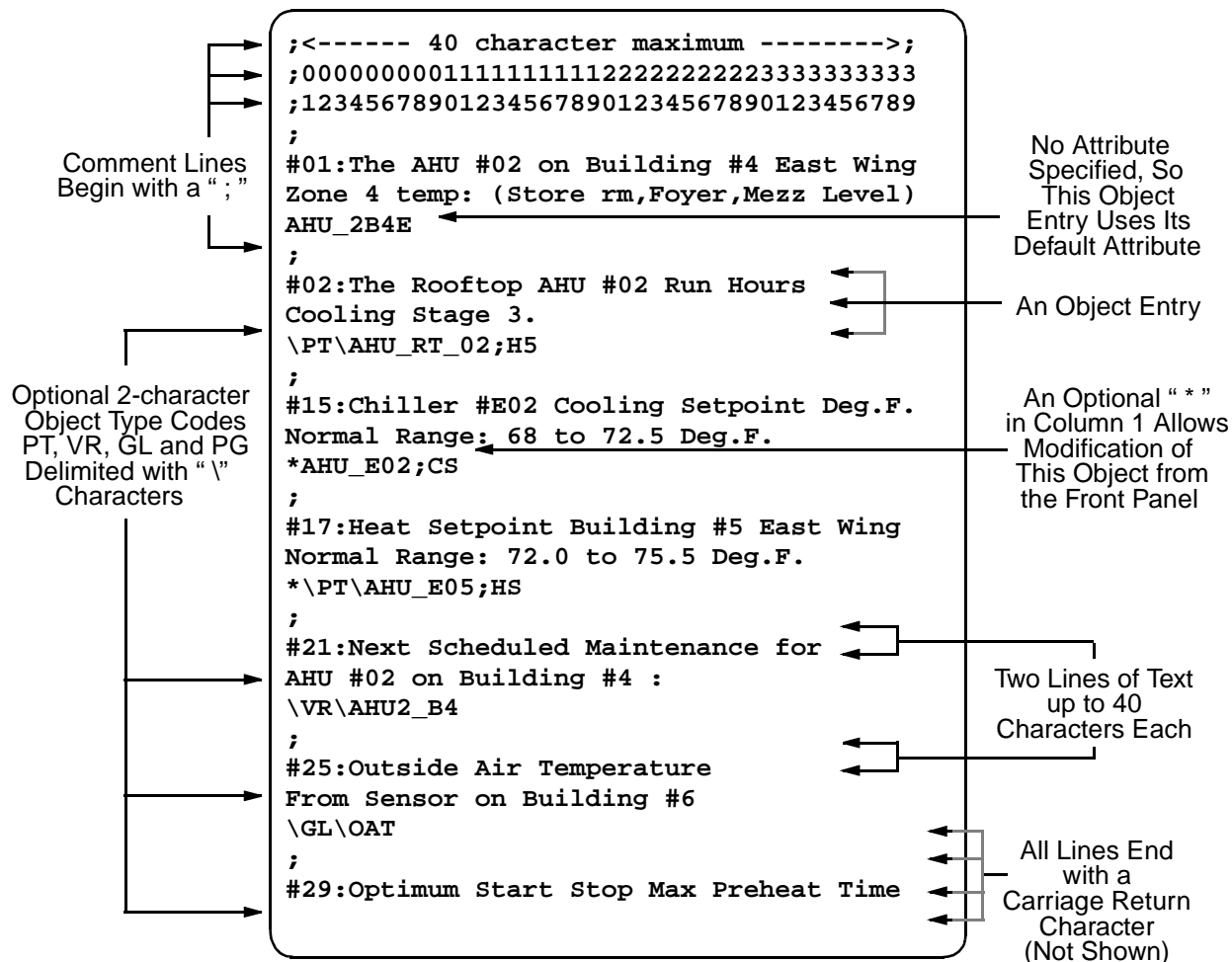


Figure 9-3 Sample Menu Text File

NOTES

Text description lines may not exceed 40 characters in length. Characters beyond the 40 character position are truncated. All lines must end with a carriage return. (<CR>).

The object attribute field is optional. If omitted, the SAGE^{MAX} fetches the default attribute.

OBJECT TYPE	CODE
Point	PT
Program	PG
Variable	VR
Global	GL

Table 9-1 Object Type Codes

9.4 Using the Front Panel Menu System

The LUM Program uses the first line of the LCD to display status fields. An example of the status line is shown in **Figure 9-4**.

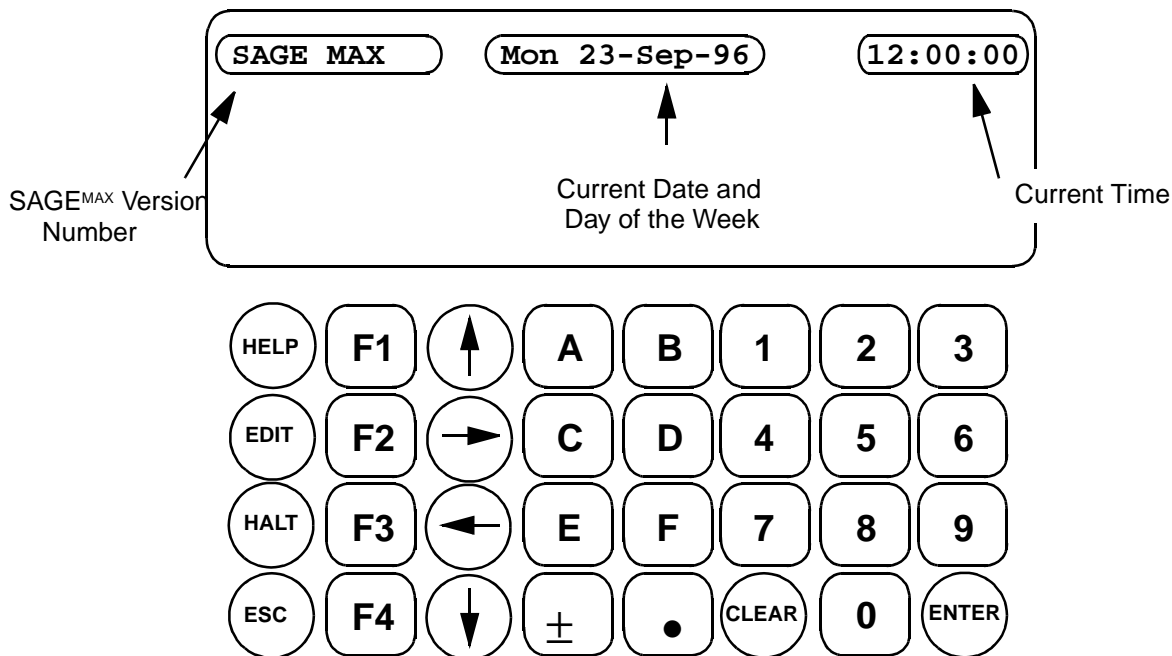


Figure 9-4 LCD and Keypad

Pressing any key on the keypad (see **Figure 9-4**) causes the LCD to prompt you to enter a codeword. *Codeword* refers to the filename of a menu file that you want to access.

After the codeword prompt is displayed, you have 20 seconds to enter a valid filename. If no keys are pressed for 20 seconds, the codeword prompt is erased and the LCD displays the status fields. If valid keys (**0-9** and **ENTER**) are pressed, the characters of the filename are displayed on the LCD as asterisks for security.

If an invalid codeword is entered, the LCD displays the error message

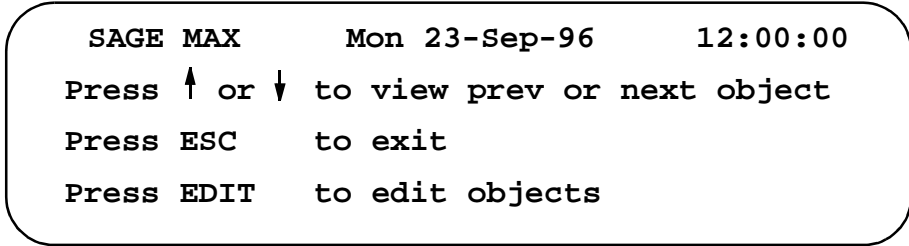
```
#[21]:FileNotFound
```

```
...Press any key to continue:
```

and waits for a key to be pressed. When a key is pressed (or if 20 seconds elapse without a key being pressed), the LCD displays the codeword prompt again.

If a valid codeword is entered (up to 8 digits followed by the **ENTER** key), the first object entry of the specified file is displayed. Line three of the object entry is replaced with the actual live value of the object specified. When an object entry is displayed on the LCD, you can use five of the keys from the SAGE^{MAX} keypad to perform different functions: the up and down arrows, **HELP**, **EDIT**, and **ESC**.

Pressing the **HELP** key when an object entry is being displayed shows a help screen on the LCD. This help screen explains the uses of the **EDIT**, **ESC** and up and down arrow keys. The object entry help screen is shown in **Figure 9-5**. From this screen, you can press any key on the keypad to return to the object entry screen. If no key is pressed for 20 seconds, the LCD displays the codeword prompt again.



```
SAGE MAX      Mon 23-Sep-96      12:00:00
Press ↑ or ↓ to view prev or next object
Press ESC     to exit
Press EDIT    to edit objects
```

Figure 9-5 The Object Entry Help Screen

Pressing the **ESC** key when an object entry is being displayed causes the LCD to display the codeword prompt again.

Pressing the up arrow key when an object entry is being displayed causes the LCD to display the previous object entry from the menu file. If you press the up arrow key from the first object entry of the menu file, the display wraps around to the last object entry of the file.

Pressing the down arrow key when an object entry is being displayed causes the LCD to display the next object entry from the menu file. If you press the down arrow key from the last object entry of the menu file, the display wraps around to the first object entry of the file.

If the displayed object entry can be modified, pressing the **EDIT** key causes the LCD to display the edit object entry screen.

NOTE

To be able to modify the value of an object entry from the front panel of the SAGE^{MAX}, the corresponding menu file must contain an asterisk in column one of the third line of the associated object entry. In addition, the object entry must not be a read-only attribute.

The numeric data type of the displayed object determines the format of the edit object entry screen. The edit value screen and edit value help screen for bitmap values are displayed somewhat differently, because editing values in a bitmap format requires that you maneuver horizontally to select the appropriate bit to be edited.

NOTE

You can exit from a help screen and return to the previous screen by typing any key on the keypad of the SAGE^{MAX}.

9.5 *Sending Output to the LCD*

The LCD on the front panel of the SAGE^{MAX} can be used as an output device by directing output to port 0. This may be useful for alarm logging or custom programs. Programs can print to port 0.

For example, you may want to display three vital zone temperatures on the LCD and refresh the values every 30 seconds. This can be accomplished by using a combination of PRINT (to port 0) and WAIT statements in an SPL program. You can make this display program as elaborate as you like, including conditional logic to display different information on particular days of the week or at particular times of the day. Refer to **Chapter 11: Programming** for more information on SPL.

The LUM Program is still available if the SPL programming alternative is used. If, while an SPL program is sending information to be printed on the LCD, you type a key from the keypad, the LUM Program is activated and any further LCD manipulation from an SPL program is queued. When the LUM Program returns to an *idle* state (i.e., the status fields display), the LCD queue is processed and the SPL program continues.

9.6 *LCD Upgrade Procedure*

You can purchase your SAGE^{MAX} with the optional keypad and LCD display, or you can upgrade your plain front panel in the field.

If you want to upgrade a plain front panel SAGE^{MAX} in the field to include the LCD and keypad display, you must carefully follow the installation procedure explained below.

1. With the SAGE^{MAX} turned off, remove the front panel by removing the 6 hex nuts that connect it to the enclosure. The positions of the nuts is shown in **Figure 9-6**.
2. Next, install the new front panel assembly, securing it to the enclosure using the 6 hex nuts that you just removed. Be careful not to overtighten these nuts.
3. In order to connect the front panel data and power cables to the SAGE^{MAX} motherboard, it is necessary to remove the disk storage module. This process involves working near delicate, static- and shock-sensitive components of the SAGE^{MAX}. **Therefore, extreme care should be taken when performing this procedure.**

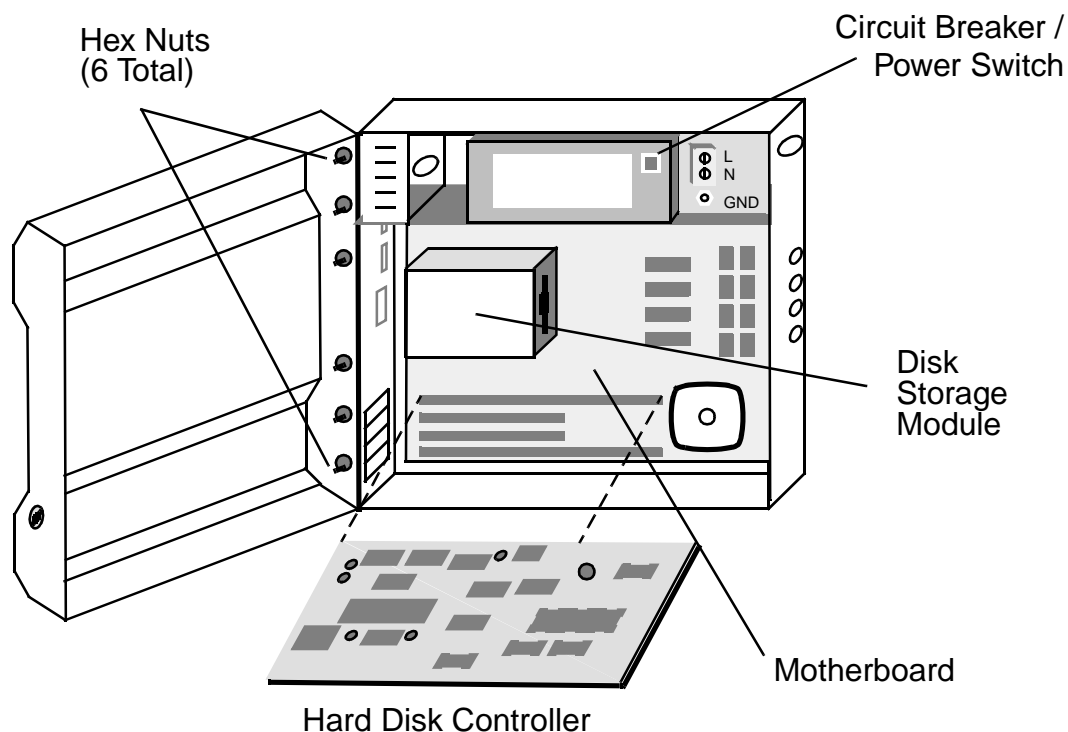


Figure 9-6 Removing the Plain Front Panel

4. The disk storage module has brackets that allow it to be attached to the inner left side of the enclosure and standoffs on the bottom of the enclosure. To remove the disk storage module you must remove 2 hex nuts from the inner left side of the enclosure and then remove 2 screws that connect the disk storage module to 2 standoffs on the base of the enclosure. The disk storage module is shown in **Figure 9-7**.

CAUTION

When removing the disk storage module, you must be cautious not to strain the cables that lead to it. If these cables cannot be held in a strain-free position during the installation of the front panel data and power cables, you must carefully disconnect the cables from the disk storage module, noting their positions for later reference. It may be helpful to remove the hard disk controller card from its card slot. This allows better access to the bottom portion of the disk storage module and minimizes the risk of damaging the controller card.

5. With the disk storage module free, you have access to connector J7 (which connects to the front panel power cables) and connector J13 (which connects to the front panel data cable). Refer to **Figure 9-7**.
6. Carefully insert the keyed end of the front panel power cables into connector J7 located on the motherboard.

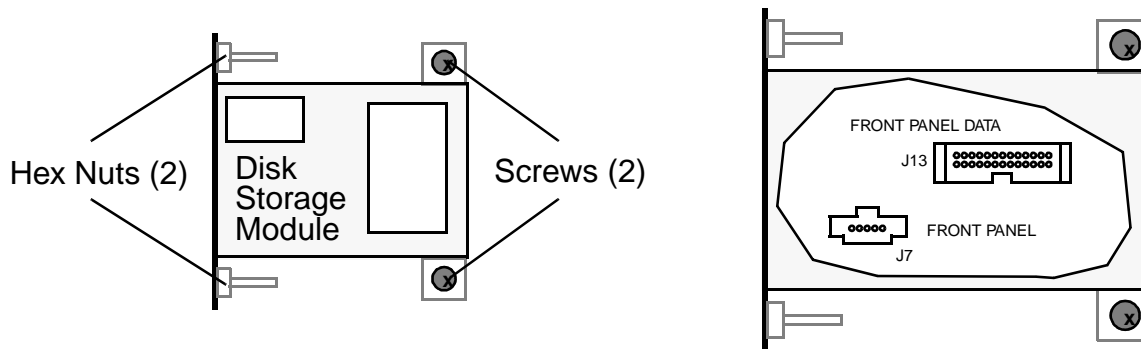


Figure 9-7 Disk Storage Module (left), Cutaway View of J7 and J13 Connectors (right)

7. Next, insert the keyed end of the front panel data cable (a ribbon cable) into connector J13. This connector has a latching mechanism on each side. You must make sure the latches are *opened* (pointing outwards) before installing the front panel data cable. After installation, gently push the latches toward each other to ensure a secure connection.
8. After the front panel power and data cables are connected to the SAGE^{MAX} motherboard, it is necessary to replace the disk storage module. Use the reverse of the procedure that you used to remove it, being careful not to strain any of the front panel or disk controller cables. Be sure that you reconnect any disk control module cables that you may have disconnected earlier.
9. After securing the disk control module with the 2 hex nuts and 2 screws, be sure that you reinstall the hard disk controller card. Refer to **Figure 9-8**.

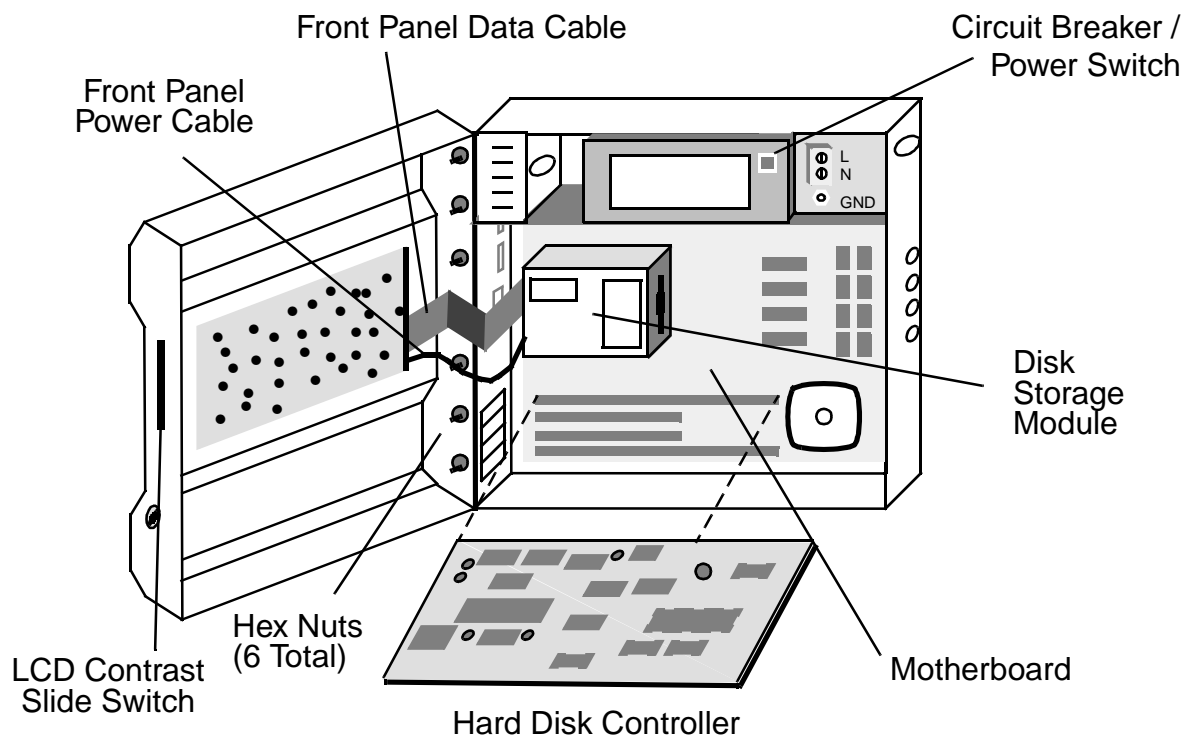


Figure 9-8 SAGE^{MAX} with New Front Panel

10. At this point you can restore power to the SAGE^{MAX} and close the front panel. After the SAGE^{MAX} boots (approximately 15 seconds), the LCD display will show the time and date. If the LCD display on the front panel does not appear to function properly, check the following:

- Is there power to the SAGE^{MAX}?
- Is the LCD contrast control switch **ON** ?
- Is the power cable connected properly?
- Is the data cable connected properly?

If the LCD display still does not appear to be functioning properly, call Auto-Matrix Customer Service.

CHAPTER 10 - ALARM AND EVENT MANAGEMENT

10.1 *Classes and Logging*

A class is a structure that specifies various options for handling incoming alarm and event notification messages. Typically, each incoming alarm and event notification message has a class that is assigned to it by its sender. In cases where the SAGE^{MAX} receives an alarm or event notification message that does not have a class, one is assigned by the SAGE^{MAX} using policies that are explained later in this chapter.

The SAGE^{MAX} has 256 classes. Each class has a 3-character name, which is **000-255** by default, but can be changed to more meaningful names such as **STR** (for STAR alarms), **FIR** (for Fire Alarms), **FL1** (for Floor 1 Alarms), etc.

Although any of the 256 SAGE^{MAX} classes may be used to handle any type of alarm (e.g., PHP alarms, XANP alarms, etc.), some classes have special predefined assignments under normal circumstances.

Classes 000-009 have a specific predefined use. These uses are illustrated below:

<u>Class #</u>	<u>Meaning</u>
000	General Messages
001	Operator Action Required
002	Operator Did Something
003	SAGE ^{MAX} Program Alarms
004	Warning Messages
005	SAGE ^{MAX} Trend Alarms
006	reserved
007	reserved
008	reserved
009	Unit Scan Alarms>Returns

Table 10-1 Predefined Uses for SAGE^{MAX} Classes 0-9

Classes 010-025 are the classes used for incoming STAR alarms that have actions 00-15 respectively. For example, if a STAR reports an alarm using action 5 over the Peernet to a SAGE^{MAX}, the SAGE^{MAX} uses class 015 to redirect the alarm. Since STARS include the action number when alarms are reported over the Peernet, the SAGE^{MAX} can easily convert the incoming STAR action number (00-15) into a SAGE^{MAX} class (010-025) to facilitate the formation of *uniform* SAGE^{MAX} alarms.

SAGE^{MAX} classes 026-255 are the classes used to redistribute PUP alarms that have classes 000-229 respectively. For example, if a mechanical alarm of PUP class 002 is reported to the SAGE^{MAX}, the alarm is usually handled based on SAGE^{MAX} class 028.

Classes specify if the incoming alarm or event notification message is to be recorded to the **GENERAL.LOG** file and/or the **class.LOG** file.

Classes may also specify that incoming alarms require operator acknowledgment. In this case, the alarm is recorded in a disk-based alarm list.

In addition, classes specify up to 7 port destinations (a port number and a unit number) where the alarm may be redirected, based on time-of-day and day-of-week.

Classes also have a privilege bitmap that specifies the user privileges required to acknowledge alarms that use this class.

Log, priority and dial information are also part of each SAGE^{MAX} class.

Dial information includes up to 6 dial destinations, each of which contains a dial type, baud rate, dial string, list of active days, and valid time range. **Figure 10-1** shows the components of SAGE^{MAX} classes.

Classes are used to specify various options for handling incoming messages. These incoming messages are sent to the Alarm Logging Task (ALOG) by the SAGE^{MAX} port drivers (e.g., PUPhost driver, Ethernet driver, PHPHdial driver, VT100 driver, etc.) and internal SAGE^{MAX} tasks (e.g., the Program Task, Trend Task, Job Scheduler Task, etc.).

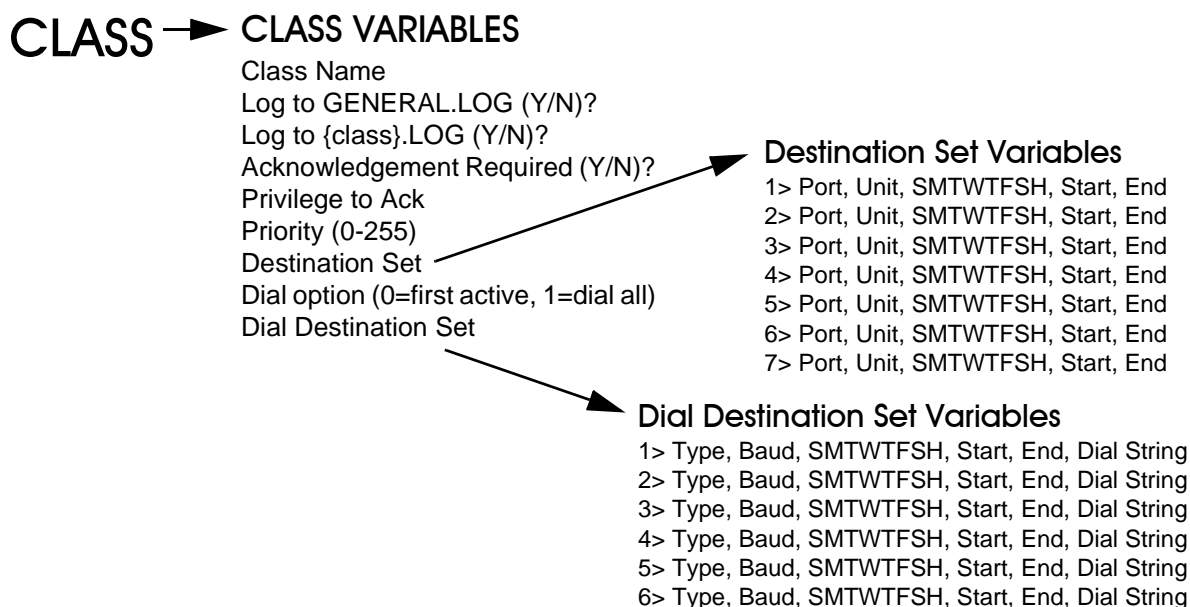


Figure 10-1 Components of SAGE^{MAX} Classes

10.2 How ALOG Works

The Alarm Logging Task (ALOG) is responsible for the recording and distribution of alarm and event notification messages to multiple destinations based on the alarm class and the time and date.

Alarm and event notification messages that are sent to ALOG for processing may come from a variety of sources (e.g., a unit on a PUP network, an XANP device, a PHP slave device, the SAGE^{MAX} Program Executor, a Broadcast Message request, etc.), but regardless of the source, the message reaches ALOG with four pieces of information:

- The originating port number
- Message text
- A class number
- A flag that specifies whether or not the message is a continuation line

Typically, it is the responsibility of the associated drivers to pass along these four pieces of information in a standard SAGE^{MAX} message format. This standard format unifies the appearance of alarms and event notification messages in printed logs and files. In cases where the necessary pieces of information are not directly available from the alarm or event that is received, the missing information must be filled in by the driver according to some policy. Since these policies vary among drivers, they are discussed in the individual driver chapters (**Chapters 14-18**). Refer to **Figure 10-2**. Alarms and event messages that originate internally (e.g., a PRINT statement from an SPL program, SPL program alarms, a broadcast message request from an operator interface, trend alarms, etc.) are formatted into a standard SAGE^{MAX} message internally by the originator of the alarm or event message.

Standard SAGE^{MAX} messages are sent to ALOG where they are routed to the appropriate files and ports based on the class information associated with every alarm or event notification message.

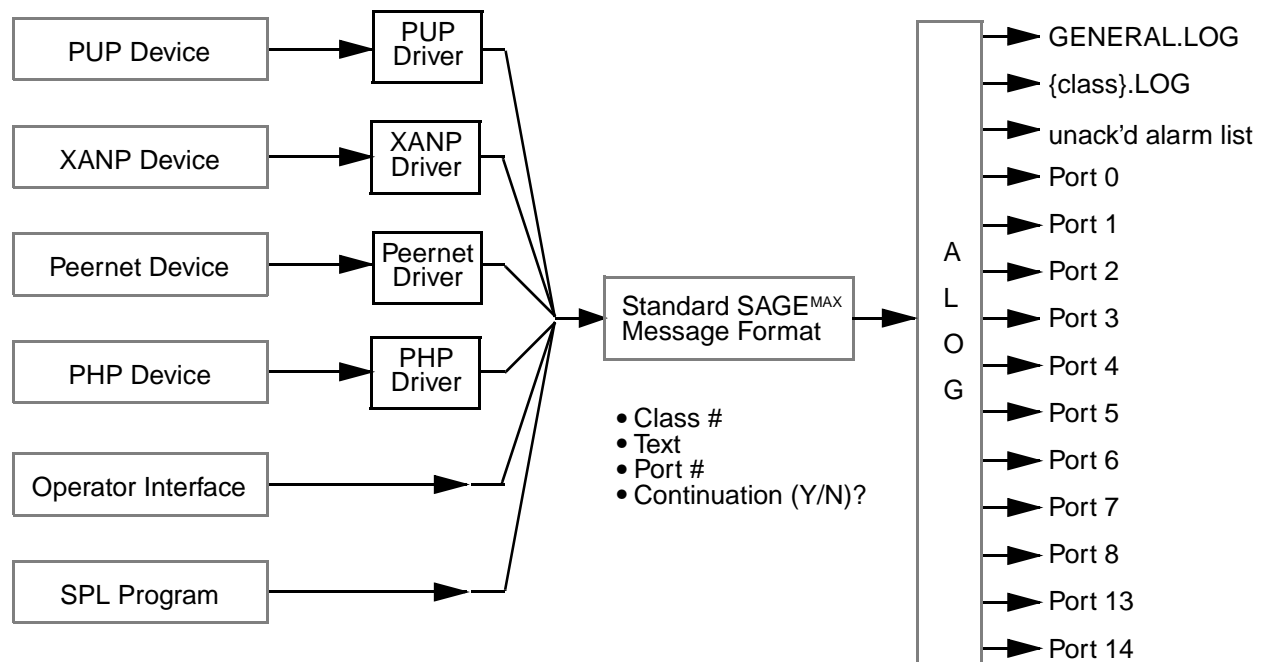


Figure 0-2 Propagation of Alarm and Event Messages from Several Sample Sources

Although the SAGE^{MAX} has 32 ports (or tasks), it may not make sense to use classes to direct alarms or event notification messages to all of them. For example, directing information to port 15 (the ALOG task), port 16 (the SPL Program Executor), port 17 (the SPL Program Timer task), etc., serves no function. Therefore, SAGE^{MAX} classes should redirect alarms and event messages to ports 0-14 (the front panel, network ports, COM1, COM2, local OPI ports, virtual terminal OPI ports, the printer, or Ethernet ports).

Typically, messages from ALOG have a standard format that is shown in **Figure 10-3**. The **tttt** field represents a sequential transaction number that is assigned by ALOG to every transaction that occurs. The **llll** field contains the unit number (0-65,535) on port **p** that generated the alarm. The SAGE^{MAX} class number (or name, if one is assigned) appears in the **ccc** field, which is followed by standard **date and time fields** specifying when the message was processed by ALOG. Time and date information is included in the prevailing language format. In the default language (English), the format includes the abbreviated form of the day of the week (e.g., **MON, TUE**, etc.), the date when the alarm was received by ALOG (e.g., **13-Feb-96**), and the time when the alarm was received by ALOG (e.g., **17:23:45**).

```

tttttp/lllll  ccc  date and time fields
              ccc  message text
              ccc  message text (optional)
              ccc  message text (optional)

```

Figure 10-3 Typical SAGE^{MAX} Alarm/Event Message Format

Typical ALOG messages have a second line which contains the SAGE^{MAX} class number (or name, if one is assigned) and some message text. Depending on the source of the alarm or event, a message from ALOG may contain several of these continuation lines. Usually, continuation lines contain such information as point description text, alarm/return message text, the type of alarm that has occurred, etc.

Actual message formats vary according to the source of the alarm or event. For more information about driver-specific formats, refer to the individual driver section.

10.3 Acknowledgment

Every ALOG transaction has a SAGE^{MAX} class associated with it. Each class defines how to handle and route its respective alarm and/or event notification messages. If the Acknowledge Required parameter is set to **YES** for a particular class, any alarms that use that class are placed in a special unacknowledged alarm file (**C:\LOG\ALARMS.ACK**). The alarms/messages remain in this list until they are acknowledged.

Acknowledgment is performed from the List Unacknowledged Alarms (**U**) option of the Alarm List submenu. In order for you to be able to acknowledge (i.e., remove) an entry in the unacknowledged alarm list, you must have the proper privilege bitmap associated with your SAGE^{MAX} user name. If the privilege bitmap associated with your user name does not include privilege bits that are set in the Privilege to Ack bitmap of the SAGE^{MAX} class associated with the entry, you are not permitted to acknowledge the entry. For more information on privileges, refer to **Chapter 7: Initial Configurations**. For information on the process of alarm acknowledgment, a detailed explanation is included in **Chapter 8: SAGE^{MAX} Menu Operations**, along with examples and illustrations of the menus.

When there are unacknowledged alarms, an **ALARMS** icon flashes in the banner line of the SAGE^{MAX} (below the time field). This is used to notify the operator that unacknowledged alarms exist. In addition, SAGE^{MAX} provides the **\$ALARMS** variable as a means of determining whether or not there are unacknowledged alarms. When you monitor this variable, it displays either **Yes** or **No**. This variable can also be read from an SPL program.

10.4 Dialing

If a dial port (typically port 5) is specified as a destination port within a SAGE^{MAX} class, alarm or event messages of this class are to be reported to a remote location(s) over the modem port of the SAGE^{MAX}.

If alarms are to be dialed out, you must define additional dial variables within the SAGE^{MAX} class. These variables specify up to six remote locations, each of which includes a dial type, a baud rate, a time and day *window* during which the location is considered *active*, and a dial string. Another class variable specifies if the alarm or event message should be sent to the first active destination or all active destinations. Refer to **Figure 10-1**.

Dial Option is a SAGE^{MAX} class variable that specifies if the alarm or event message should be sent to the first active destination (**0**) or all active destinations (**1**). SAGE^{MAX} cycles through the six dial destinations until the dial option is satisfied (the event message is reported to the first active or all active destinations) or until there are no active destinations.

The *Dial Destination Set* is a series of up to six remote destinations, each of which includes a dial type, a baud rate, a time and day *window* during which the location is considered *active*, and a dial string.

The *Dial Type* column provides a means to define the type of dial-out that the SAGE^{MAX} will perform. There are four possible dial types:

- 0 = Unused Destination
- 1 = Ring Only
- 2 = Dial to Printer
- 3 = Automatic

The Ring Only option is used for paging systems. The SAGE^{MAX} will dial the number listed, ring and optionally transmit a programmable numeric code (e.g., a phone number or a location ID).

When the Ring Only option is chosen and multiple alarms of the same class occur, each alarm is dialed out individually. For example, if 16 scan alarms are generated, the SAGE^{MAX} will dial out 16 times.

An “@” character in the dial string marks the beginning of a numeric code (such as a unit ID) to be transmitted to a display pager using the tones of digits 0-9. After the SAGE^{MAX} dials the paging system, it waits for five seconds of silence and then transmits the numbers that follow the “@” character (if any are specified). After this, the SAGE^{MAX} hangs up and the alarm is considered to be satisfied for the destination.

Remember that the total number of characters in the dial string cannot exceed 39 (40 including the “/” character).

The Dial to Printer option causes SAGE^{MAX} to dial the remote number. Once SAGE^{MAX} determines that a connection has been made, it prints an identifying banner line followed by the alarm or event message.

Automatic mode dials to a PHP host (e.g., SPECTRA, a host SAGE^{MAX} or a terminal). The SAGE^{MAX} dials and waits for a connection. Once the connection has been established, the SAGE^{MAX} waits for the PHP “Who are You?” command (;?). If it receives the “Who are You?” command within five seconds, SAGE^{MAX} waits for further PHP instructions. If there are no further PHP instructions within the amount of time specified in the Auto Hang-up Time PHP dial driver variable, the SAGE^{MAX} will hang up.

If a connection is made and the SAGE^{MAX} does not receive the “Who are You?” query, it will report the alarm as ASCII text, assuming it is connected to a printer or terminal. The SAGE^{MAX} then waits the amount of time specified in the Auto Hang-up Time system variable for a response from an operator. If you type three carriage returns before this time elapses, the SAGE^{MAX} will allow terminal operation from the remote terminal by offering the Sign-on Screen.

The Active Days column heading is **SMTWTFSH**. This represents a bitmap for **S**unday, **M**onday, **T**uesday, **W**ednesday, **T**hursday, **F**riday, **S**aturday and **H**oliday. A **Y** beneath any of these letters will allow alarm dial-out operations for that particular day. The holiday setting references the SAGE^{MAX} calendar function to determine if holiday operation should be used.

The Start Time and End Time columns are used to determine the operational time window for each listing. The SAGE^{MAX} will only perform dial-out operations between the start and end times for the destination. These times are given in military format (00:00-23:59).

The Telephone Number column heading specifies the number to be called for each destination. Telephone numbers may be up to 40 characters in length, including spaces, commas, “@” and “/.” The / key (forward slash) must be used to end each listing entry. This is a delimiter that tells the SAGE^{MAX} that it has reached the last digit in the number to be dialed. The use of a “@” character in the dial string causes five seconds of silence which can be followed by other numbers such as the ID for the SAGE^{MAX} or a specific code that is displayed on a display pager.

Commas may be inserted into the telephone number to cause a pause between numbers when dialing out. Be sure that the total number of characters in the Telephone Number column (including numerals, spaces, commas, “@” and the forward slash delimiter) does not exceed 40.

For more information about dial destination set variables, class variables, or SAGE^{MAX} classes, refer to **Chapter 8: SAGE^{MAX} Menu Operations**.

CHAPTER 11 - PROGRAMMING

The SAGE^{MAX} has an extremely versatile Application Programming Subsystem that can be used to make database objects that perform special functions.

The Application Programming Subsystem uses SAGE^{MAX} Programming Language (SPL). SPL offers a “seamless” interface between program objects and other objects of the database. In addition, SPL includes integrated edit/compile/debug/execute mechanisms.

This section reviews the parts of SPL program objects, explains the structure of the runtime system, and lists the components used to create the logic of program source code.

11.1 *The Parts of SPL Programs*

An SPL program is a SAGE^{MAX} database object that consists of the following items:

- a program name
- a Program Logic Block (PLB) file
- an optional Program Reference Block (PRB) file
- an optional attribute Initial Value (INI) file
- program attributes and registers
- an Engineering Units (EU) override file
- a set of program options

These components are explained in the following sections of this chapter..

11.1.1 *Program Names*

SAGE program objects must have an associated name that can be up to 24 characters in length. The valid characters for program names are shown below:

- A-Z (uppercase letters “A” through “Z”)
- a-z (lowercase letters “a” through “z”)
- 0-9 (numbers “0” through “9”)
- _ (under bar)
- (space)
- . (period)
- \$ (dollar sign).

Lowercase letters are treated the same as uppercase letters in program object names (e.g., **abc** is the same as **ABC**).

11.1.2 The Program Logic Block (PLB)

SAGE^{MAX} program objects must reference a Program Logic Block (PLB). The PLB is a binary data file that contains compiled binary pseudocode in a form which can be executed by the SAGE^{MAX} Program Executor (PEX). This file is created by the SPL Compiler after you create, edit and compile an ASCII text file (i.e., an SPL source file) which contains program logic statements that are easily edited and understood by programmers.

ASCII SPL source code files have the extension **.SPL** and are located on the **C:\SPL** subdirectory of the SAGE^{MAX}. Binary Program Logic Block files (PLBs) have the extension **.PLB** and are located on the **C:\LOGIC** subdirectory of the SAGE^{MAX}.

Although program objects can assume names that have up to 24 characters, their associated PLB and SPL source file names are *file fragments* which are restricted to having a maximum of 17 characters (i.e., an optional 8-character maximum subdirectory followed by a “\” character and an 8-character maximum file name). For this reason, it is not always possible to assign matching names for program objects, PLBs and SPL source files.

When you create or modify an SPL program through the SAGE^{MAX} Create/Modify Submenu, and you select the Edit/Compile option followed by the Edit Program Logic Source option, you are presented with an SPL filename prototype that consists of the first 8 characters of the program name with spaces “ ” and periods “.” replaced with tildes “~”. You are then free to rename the SPL source file to any file fragment of up to 17 characters using the MS-DOS file naming convention. There is never an assumed PLB name. It must always be supplied when you create a program object. This permits a single PLB to be shared between multiple programs.

The source file contains SPL program logic statements which are discussed in detail later in this chapter. SPL source code can be created and/or edited by using the SAGE^{MAX} Program Editor. Source code can also be created and/or edited off-line on a PC using any unformatted text editor and transferred to the SAGE^{MAX} using a 3.5” floppy disk. The number of lines in an SPL source file is unlimited.

The source code that you create in ASCII form is converted to a binary pseudocode file (the PLB) by the SAGE^{MAX}-based SPL compiler when you select the compile feature of the Create Program Submenu of the SAGE^{MAX}. Source code can also be compiled off-line on a PC using the PC-based SPL compiler utility program (**SPLC.EXE**) from the Auxiliary Disk supplied with the SAGE^{MAX}. Compiled PLBs cannot exceed 65,535 bytes in size.

NOTE

If any errors are generated during the compiling process, a PLB is not created.

You may choose to have the SPL Compilers optionally create a list file during the compile process. The list file is an ASCII text file that contains the source code statements along with the pseudocode and their respective *relative* locations in memory and any error messages generated by the compiler. List files are useful in debugging the execution of SPL programs. List files have the same name as the SPL source file except that they have the extension **.LST** and are found on the **C:\SPL** subdirectory of the SAGE^{MAX}.

NOTE

The list file does not point out execution or logic errors in your program logic. Only syntax errors (errors due to the improper construction or format of program statements) are flagged by the compiler and shown in the list file.

11.1.3 The Program Reference Block (PRB)

SAGE^{MAX} program objects may contain an optional Program Reference Block (PRB) file. PRBs allow programs to refer to named objects indirectly so that the actual object names do not need to be used in the program logic statements. This allows many different program objects to share the same PLB.

If a PLB uses *references* as part of its logic, the associated program object must contain a PRB. The PRB is an ASCII text file that specifies the object name/reference association.

Each reference in the PRB text file must be on a separate line and must adhere to a specific format so that it may be recognized by the SAGE^{MAX}. References must contain the object name to be referenced, and may optionally contain the object type code (i.e., PT, PG, VR, GL) and/or an attribute. Indices into the PRB are zero-based (e.g., reference 0 is the first reference). The four formats for PRB references are shown below:

- \ objecttype \ objectname ; attribute
- objectname ; attribute
- \ objecttype \ objectname
- objectname

The *objecttype* is a two character mnemonic (PT, VR, GL, PG, etc.). If missing, the correct type will be determined by a search of *all* object types

If no object type is specified in a reference, then the SAGE^{MAX} searches all object types in its database for the first object name that matches. The SAGE^{MAX} performs this database search in the following order: Points, Programs, Variables, and Globals.

11.1.4 The Program Attribute Initial Value (INI) File

A third component of an SPL program is an attribute Initial Value File or INI file. This optional ASCII text file is used to set program attributes to initial values at the start of program execution. If a program attribute is not listed in the INI File, the attribute is initialized to zero. If no INI file is specified for a program, all its program attributes are initialized to zero. SPL provides mechanisms that can be used to save new initial values to this file. Programs with the same attributes that have the same initial values may share the same Initial Value File.

11.1.5 Attributes and Registers

All programs have 16 registers (%A-%P) and 20 program control attributes (\$\$, \$D, \$E, \$\$, \$C, \$W, \$P, \$B, \$N, \$Z, \$L, \$R, \$A, \$I, \$F, \$X, \$M, \$#, \$H and \$1). To an operator, program registers always begin with a percent sign (%) and program control attributes always begin with a dollar sign (\$). Up to 255 additional two-character attributes can be defined for every program. Program registers, program control attributes and user-defined program attributes can be accessed from the Monitor Program Submenu by qualified users. The first user-defined program attribute is the *default attribute*. If no user-defined program attributes are specified in the program, the \$\$ program control attribute is displayed as the default when you monitor the program object.

11.1.6 Engineering Units (EU) File

Another component of SAGE^{MAX} program objects is an Engineering Units override file (refer to **Chapter 7.9: Engineering Units Files**). This ASCII text file may contain direct and/or indexed Engineering Unit assignments for specified program attributes (both program control and user-defined attributes).

By default, all SAGE^{MAX} programs have C:\EU\PEX.EU as an Engineering Units file. A different EU file may be specified. If an EU *override* file is specified, the associated program references *it* rather than C:\EU\PEX.EU for Engineering Unit information. **Figure 11-1** shows the default SPL program EU file C:\EU\PEX.EU. This file contains EU assignments for program control attributes only.

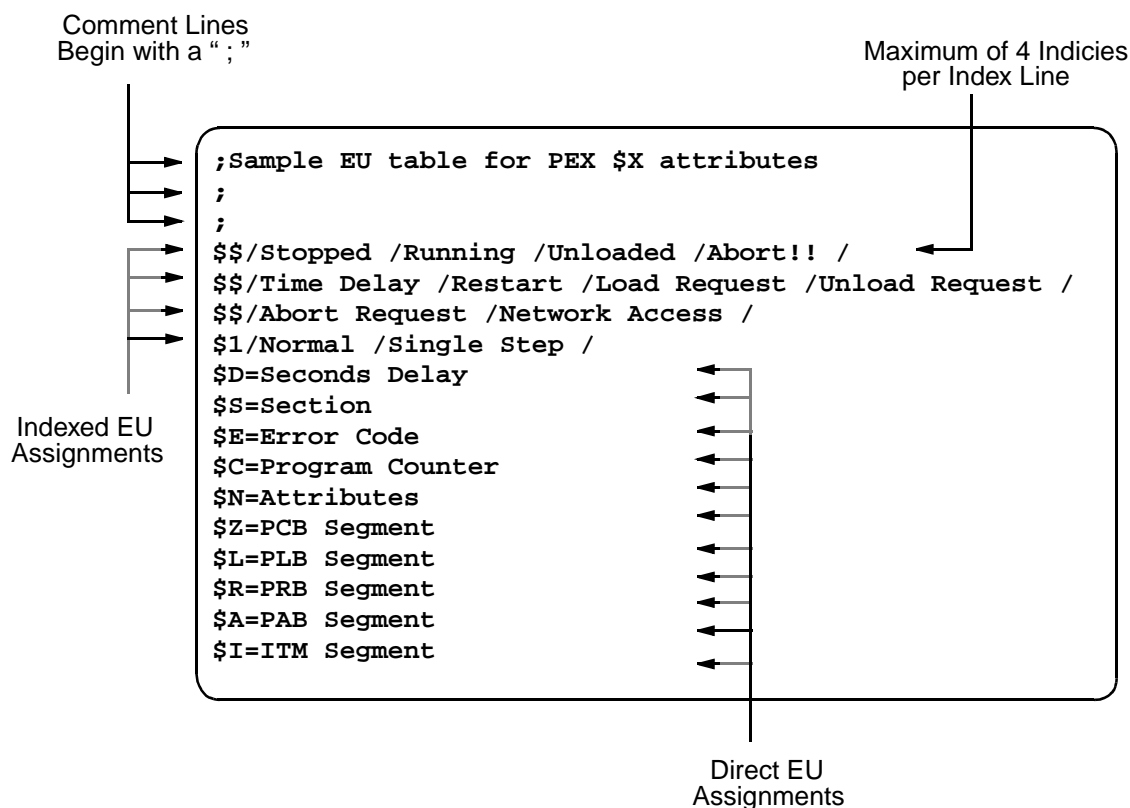


Figure 1-1 PEX.EU - The Default EU File for SPL Programs

11.1.7 Program Options

Every SPL program object has a set of options that are associated with it. These options are available through the Program Options (**O**) choice of the Create/Modify Program Submenu. Program options include the following:

- Famous Option
- Log Object Changes
- Preload Option
- Program Logic Option
- Program Reference Option

Famous Option is a **Y/N** (Yes or No) option which (if set to **Y**) is used to make the program object famous. If this option is set to **N**, the program is considered to be private.

For example, let's suppose you have a network of SAGES^{MAX} connected by an Ethernet. An SPL program object named **PROG1** is created and entered into the database of the first SAGES^{MAX}. If this program's famous option is set to **YES**, then all SAGES^{MAX} on the Ethernet automatically create the global object **PROG1** in their databases. This allows you to access the registers (**PROG1;%A - PROG1%P**), control attributes (**PROG1;\$\$ - PROG1;\$1**) and user-defined attributes of **PROG1** (e.g., **PROG1;CV**, **PROG1;MN**, **PROG1;SP**, etc.) on other SAGES^{MAX} without having to create the global object yourself on the other SAGES^{MAX}.

Log Object Changes is a **Y/N** (Yes or No) option which (if set to **Y**) is used to generate a log anytime one of the program's attributes are changed by an operator through an operator interface (OPI) or by a program other than the one in which the attribute is defined. If this option is set to **N**, changes to the attributes are not logged.

Preload Option defines when the program's PLB is loaded into memory. Through this option an SPL program can be set to **0** to load on demand (e.g., through an OPI by changing the **\$\$** attribute, through a program using the **ACTIVATE** statement) or set to **1** to be loaded automatically when the SAGE^{MAX} is turned on or reset.

Program Logic Option defines how the program's PLB should be treated after it is in memory and an unload is requested. If the Logic Option is *not sticky* (**0**), the program does not "stick in RAM" and is removed from memory when an unload is requested. If the Logic Option is *sticky* (**1**), the program remains in memory once it is loaded.

Sticky programs should be programs that are executed frequently on a regular basis, when it would not be desirable to unload then reload the program every time it needed to be executed. However, sticky programs do require memory space (RAM) regardless of their current state of operation.

Nonsticky programs should be those which are executed less frequently and would not suffer detrimental effects from being reloaded into memory when needed. Once removed from memory, a nonsticky program releases valuable memory space which can be used by other programs and tasks of the SAGE^{MAX}. However, nonsticky programs do require more time to begin executing since they must be loaded into RAM first.

Ultimately, you must choose a Logic Option that suits your application and optimizes your results.

NOTE

If you change a PLB from sticky to not sticky, you must reset the SAGE^{MAX} to remove the sticky PLB from memory. If you modify the logic of a sticky PLB, you must also reset the SAGE^{MAX} for the changes to take effect.

Program Reference Option allows you to define the RAM-resident nature of a PRB. PRBs can be (0) always file-resident, (1) RAM-resident, but able to be unloaded, or (2) sticky (i.e., not unloadable once it has been loaded into RAM).

The advantage of file-resident PRBs is that they do not take up valuable RAM space. The disadvantage is that file access is necessary each time a reference in the PRB is used, thereby slowing down the program's execution. Therefore, file-resident PRBs should be used only for infrequently referenced PRBs or when RAM usage is at a premium.

PLBs and PRBs are pooled among all programs that use them. For example, if two programs both execute the same nonsticky PLB, it will remain in RAM until all users of the PLB are finished. It will then be unloaded.

SPL program components and their relationships are shown below:

- .PLB file (program logic)
- .PRB file (program references)
- .INI file (initial values)
- .EU file (engineering units)
- OPTIONS
- ATTRIBUTES
- RESISTERS

11.2 *The Structure of the Runtime System*

In order to run, a program's PLB must be loaded into the SAGE^{MAX} memory (i.e., RAM). By default, SPL PLBs remain loaded in memory even after they are finished running (i.e., stopped). A PLB may be unloaded when it is finished by using an SPL **UNLOAD** statement in the code. This allows previously allocated RAM to be released and used for other SAGE^{MAX} functions.

PLBs can also be called by SPL programs by using the SPL **CALL** statement. When the called PLB has completed its function, it must execute an SPL **RETURN** statement, at which time it is unloaded from memory by default.

There may be cases when it is more efficient to force a program to remain loaded (in RAM), even though it may not be currently executing. This is the case for programs that perform regular or cyclic functions and then stop. It is the decision of the programmer to determine if the regularity of a cyclic function merits forcing it to remain in RAM after it has completed, otherwise the overhead of loading the program into RAM from the hard disk must be realized. Through the Create Program Submenu, you can specify whether the PLB should remain in RAM by selecting the program logic option to be *sticky*.

Like the PLB, it is possible to define the resident nature of the PRB by selecting the program reference option using the Create Program Submenu.

Program processes are administrated by the SAGE^{MAX} Program Executor Task (PEX). PEX handles all program loading (including PLBs and PRBs), unloading, subtasking and execution. The PEX task executes each active program object's application code (its PLB) in a round-robin manner, one program object at a time. Each *runnable* program object executes (at most) one primitive instruction per *time slice* before the PEX task switches to the next object. Each primitive instruction is represented by a single statement in the SPL source code. SPL statements are explained in the following sections.

11.3 *Comments*

All lines in SPL programs that have a semicolon (;) in the leftmost column are *comments*. The generous use of comment statements within your programs will make them more readable and easier to troubleshoot, especially if *you* are not the person doing the troubleshooting.

As a guideline, the top lines of programs should be reserved for program identification comments. This area may contain information such as:

- the name of the program
- the date the program was written
- the name of the author
- what the program does
- the meaning/use of program attributes
- the meaning/use of program registers
- any assumptions made by the author
- any input variables used by the program
- any output values calculated by the program
- an *edit trail* indicating any modifications made to the program logic, when they were made, and by whom
- in general, any information that may prove useful to someone looking at the program for the first time

In addition to using comments at the beginning of your program logic, it is also helpful to create a series of comment lines prior to any program segments with logic that may need special explaining. The extra effort you take in adding useful comments to your programs is well worth the benefits you (or someone else) will reap in the future.

Figure 11-2 illustrates a sample of how a commented SPL program might look. The styles that programmers choose to comment their work vary significantly according to their personal preferences, their expertise, the complexity of the program, and other factors. It is important for you (the programmer) to adopt your own style. Be consistent and thorough, making clear, *useful* comments where you feel they are necessary. If you will be writing programs with others, develop a technique that you are both comfortable using.

Remember that it is also possible to over-comment your programs. Programs with too many comments about the obvious tend to be cluttered and may be just as difficult to understand as an under-commented program.

11.4 Labels

SPL programs are composed of one or more *statements* which define the actions and logical operations that the program is to take when it is executed. SPL program statements are grouped into *lines* which contain a single program statement. These lines may be labeled with symbolic names to identify them. Typically this is done so that **GOTO** and other branching statements can refer to them.

Labels cannot use any of the reserved names that identify SPL statements. Labels are case-insensitive, meaning that the label **ABC** is treated the same as the label **ABc** or **abc**. Labels must begin in the leftmost column of the line. Labels may contain up to eight characters or up to eight characters and numbers. Labels can consist of the following:

- **A** through **Z**
- **a** through **z**
- **0** through **9** (not as the first character)
- **\$**, **.** and **_** characters.

Labels **cannot** begin with the numbers **0-9**. If a line contains any statement following the label, the statement must be separated from the label by one or more **TABs** or spaces. Labels may optionally end with a colon character (**:**), which is not counted in the length of the label.

Figure 11-2 illustrates the use of labels in an SPL program. In this example, two of the three labels include the optional colons, although they could be omitted.

11.5 Expressions and Terms

Expressions are symbolic formulas which represent a chain of arithmetic calculations on data from various sources. Expressions are used to convey values for parameters in many of the primitive statements in the SPL language. SPL expressions can contain an arbitrary number of *terms* and *operators* which may represent mixed mode arithmetic (i.e., integer, floating point and fixed point). SPL automatically performs type coercion on mixed mode values.

```

;-----
; Program Name : TBL_AVE Author : Dave Date : Nov 30,1991
;
; Description : This program determines the average (;AV), minimum (;MN), and maximum
;               (;MX) values of the elements of a table with ;NE unsigned fixed point elements.
;               Element values will range from 0.00 to 50000.00
;
; Inputs : INPUT_TBL- name of the input table
;           ;NE      - number of elements in the table
;
; Outputs : ;AV- the average value of the elements of table INPUT_TBL
;           ;MN- the current minimum value from table INPUT_TBL
;           ;MX- the current maximum value from table INPUT_TBL
;           A      - used as loop counter
;           B      - used to temporarily store each table value as it is read
;           C      - used as an accumulator for calculating the average
;
; Last Edit :Dec 10, 1991 John Added code to determine MIN and MAX values
;-----
;
; Attribute Declarations
;-----
;           ATTRNE,0FFh
;           ATTRAV,0FAh
;           ATTRMN,0FAh
;           ATTRMX,0FAh
;
;-----
; Set ;MN to a high value that will not be in the table. Set ;MX to the lowest possible table value.
; These are the artificial minimum and maximum values. As the table elements are read, actual
; minimum and maximum values will be saved using conditional statements. Program register C
; is used as the input value accumulator and is initialized to zero.
;-----
;           ;MN = 99999.99
;           ;MX = 0.00
;           C = 0.00
;
;-----
; Program register A is used as a loop counter. (A-1) is used as an index into the table, since table
; indices are zero-based. Within the READ_VAL loop, the value is compared to the current
; minimum and maximum values. If the read value is less than the current minimum value (;MN),
; the read value is saved as the new minimum value. If the read value is greater than the maximum
; value (;MX), the read value is saved as the new maximum value. When all the table entries have
; been processed, the average value can be calculated.
;-----
;           A = ;NE
READ_VALB = &TABLE_01 (A-1)
;           C = C + B

```

Figure 1-2 Example Comments in an SPL Program

Expression evaluation is performed from left to right. Up to six levels of nesting (i.e., the use of parentheses) may be used in expressions to define an order or *precedence* for evaluation. The expression within the innermost set of parentheses is evaluated first from left to right. This procedure continues outward until the expression within the outermost set of parentheses is evaluated from left to right.

Expressions may contain constants, variables, registers, named object attributes, references, tables, built-in functions and arithmetic and logical operators.

In a very general sense, expressions are composed of terms and operators. In the simplest case, an expression is simply a term with no operators. In general, an expression is defined as follows:

expression ::= term or
expression ::= term operator expression

Because expressions may contain terms that are objects on networks, an expression does not necessarily have to be completely resolved before execution is passed to another program. Such network accessing is processed asynchronously, causing only the program using the value to be delayed until the value has been fetched. This provides for a fair method in dealing with network-intensive programs, and not penalizing other programs by waiting for a network device object value.

Terms in expressions may be one of several possible types, indicating one of several possible sources of a value to be used during expression evaluation. In general, each term has a data type as well as a value. Data types identify the way in which the values are represented numerically, and may imply additional hidden operations or coercions to be performed when arithmetic operations are required between dissimilar types. The types of terms that can be used in expressions are as follows:

- constants
- named terms
- registers
- program control attributes
- user-defined program attributes
- named object attributes
- references
- virtual attributes
- tables
- functions
- nested expressions

Each type of expression term is explained in detail in the following sections.

11.6 Constants

Constants specify particular unique values implying a data type by their use. SPL supports the following four basic types of constants:

- integer
- fixed
- float
- time

Integer constants can be expressed in decimal, binary or hexadecimal format. Decimal numbers are a sequence of one or more decimal digits, optionally preceded by a unary minus (-) which indicates the *twos complement* negation of the constant.

Hexadecimal (hex) numbers must be preceded with a leading zero if their most significant digit is **A-F**. Hex constants end in the letter **H** or **h** to denote hexadecimal format. Binary constants end with the letter **B** or **b** and use the digits **1** and **0**.

The range of values for integer types is from 0 to $2^{32}-1$ (4,294,967,295) for unsigned decimal integers and from -2^{31} (-2,147,483,648) to $+2^{31}-1$ (2,147,483,647) for signed decimal integers. For hexadecimal values the limits are from 0 to 0FFFFFFFh. For values that are in binary format, the limits are from 0 to 11111111111111111111111111111111b. Examples of integer constants are shown in below:

A = 27	E = 0FFh
B = 0	F = 12345678h
C = -6	G = 11001110b
D = 378h	H = 11b

Fixed constants are expressed in free-form notation. Fixed constants are distinguished from similar integer constants by the presence of a decimal point (.) in the sequence of decimal digits. The general form of fixed constants is $\pm ddd.ddd$ where *ddd* represents one or more decimal digits, and the \pm represents an optional plus sign or minus sign. The number of significant digits for fixed constants is 10, however the significance cannot exceed 2^{32} for unsigned fixed constants and 2^{31} for signed fixed constants. Note that integer constants are simply fixed constants with the format $\pm ddd$.

Since both fixed and float types can be expressed in free-form notation, the SPL compiler must be told which of the two types to use when it encounters a constant with a decimal point. By default, all free-form constants are fixed types. The **#FIXED** compiler command can also be used so that all free-form constants are expressed as fixed types.

Examples of fixed constants are shown below:

#FIXED	C = 1.6
A = -.45	D = -29.6
B = 1234.5678	E = 0.123456789

Float constants can be expressed in free form or in scientific notation. Float constants are distinguished from similar integer constants by the presence of a decimal point (.) in the sequence of digits, and/or the presence of the exponent indicator (**E** or **e**). The general form of float constants is $\pm ddd.dddE\pm ddd$ or $\pm ddd.ddd$ where *ddd* represents zero or more decimal digits, and the \pm represents an optional plus sign or minus sign. At least one decimal digit must be present before the decimal point. The range of float values is from $\pm 1.175494E-38$ to $3.402823E+38$ with 6-7 significant digits.

Since both fixed and float types can be expressed in free-form notation, the SPL compiler must be told which of the two types to use when it encounters a constant with a decimal point. By default, all free-form constants are fixed types and all scientific form constants are float types. The **#FLOAT** compiler command can be used so that all free-form and scientific form constants are expressed as float types.

Examples of float constants are shown below:

```
#FLOAT
E = 1.6
F = -29.6
G = 0.04
H = -.45
I = 1234.5678
J = +39
```

Time constants can be expressed in *hours:minutes* or *hours:minutes:seconds* forms. They are distinguished from integer constants by the presence of a colon (:) in the string of the decimal digits. Examples of time constants are shown below:

```
A = 12:34:56
B = 1:22
C = 14:42
D = 48:00
```

11.7 *Named Terms*

There are a number of constants which are given names and are recognized by the SPL compiler. These *named terms* can be used as if they were constants anywhere that a term may be used. Named terms are identified by reserved names in SPL and are explained in the following paragraphs. In the figures that follow, sample SPL program segments are included to illustrate the use of the named terms. The segments vary in their range of complexity.

TRUE and **FALSE** are named terms that represent the two logical states *true* and *false*. These named terms are integer types with associated numeric values of 1 and 0 respectively. Program examples of these named terms are shown below:

```
A = FALSE
Check: IF [MX_SEC1;V0] == 1 Then Merge
A = TRUE
IF [MX-SEC1;V1] == 0 Then Merge
A = TRUE
.
.
Merge: IF NOT A Then Check
```

PI is a named term that represents the irrational value of pi (π). This constant is the ratio of the circumference of any circle to its radius and is approximated in the SAGE^{MAX} by using the floating point value 3.141593. An example of a program using the named term **PI** is shown below:

```
A=9.0E0
B=PI*(A*A)
```

Output: A = 9.0 (the radius of the duct)
 B = 254.469 (the area of the duct)

DAY is a named term that represents the current day of the current month as an integer from 1-31.

MONTH is a named term that represents the current month of the year as an integer from 1-12.

YEAR is a named term that represents the current year as an integer, e.g., 1996.

DATE is a named term that represents the current date (e.g., JAN 1, 1996).

The example below shows the **DAY**, **MONTH**, **YEAR** and **DATE** named terms and an SPL programming example using **DAY** and **MONTH**:

```
START:          Wait MONTH == 7
                Wait DAY == 4
                Call HOLIDAYS\JULY4TH
                WAIT DAY <> 4
                Goto START
```

DAYOFWEEK is a named term that represents the current day of the week as an integer from 0-6. The value 0 refers to Monday, 1 refers to Tuesday, etc.

MON, TUE, WED, THU, FRI, SAT and **SUN** are named terms that represent each of the seven days of the week as an integer from 0-6. The value 0 refers to **MON** (Monday), 1 refers to **TUE** (Tuesday), etc. The example below shows the named terms **MON-SUN** and the named term **DAYOFWEEK** in addition to an SPL program using these named terms.

```

                If (DAYOFWEEK==SAT) OR (DAY OF WEEK++SUN) Then WKND Else L2
WKND:           Call WEEK_END
L2:

```

DAYOFYEAR is a named term that represents the current Julian day of the current year as an integer from 1-366. A simple SPL program segment using the named term **DAYOFYEAR** is shown below:

```
A = ((DAY OF YEAR-1) / 7)+1
```

TIME is a named term that represents the current time in long time format (**HH:MM:SS**) from 00:00:00-23:59:59. A simple SPL program segment using the named term **TIME** is shown below:

```

WAIT TIME == 23:00:00
A = [BLDG1_STATUS;BO]

```

HOLIDAY is a named term that reflects whether or not the current day is a holiday as specified by the SAGE^{MAX} calendar. If the current day is a holiday, the value of **HOLIDAY** is **1** (or **YES**). If the current day is not a holiday, the value of **HOLIDAY** is **0** (or **NO**). **HOLIDAY** reflects the state of the special **\$HOLIDAY** system variable.

All of the named terms that are supported by SPL are listed in **Table 1-1**.

NAME	MEANING	VALUE/RANGE	TYPE
FALSE	logical false state	0	integer
TRUE	logical true state	1	integer
PI	value of PI	3.141593	float
DAY	current day of current month	1..31	integer
MONTH	current month of current year	1..12	integer
YEAR	current year	e.g., 1996	integer
DATE	current date	e.g., JAN 1, 1996	date
DAYOFWEEK	current day of the week	0..6 (0=Monday)	integer

Table 11-1 Named Terms Supported by SPL

DAYOFYEAR	current Julian day	1..356	integer
TIME	current time	00:00:00..23:59:59	time
MON	Monday	0	integer
TUE	Tuesday	1	integer
WED	Wednesday	2	integer
THU	Thursday	3	integer
FRI	Friday	4	integer
SAT	Saturday	5	integer
SUN	Sunday	6	integer
HOLIDAY	Is today a holiday?	0=no, 1=yes	integer

Table 11-1 Named Terms Supported by SPL

11.8 Registers

For each program, there are 16 arithmetic registers available for storage of temporary values, counters, loop indices and other applications. The registers are named for the first 16 letters of the alphabet (**A - P**). Each program register has a 32-bit value and a data type that is determined automatically.

When you access a program's registers from within the program itself, only the register name (A-P) is required.

A program's registers are accessible outside the program itself as named attributes of the program object (e.g., **%A**, **%B**, **%C**, etc.). Likewise, if you access (i.e., *read from* or *write to*) another program's registers, you must use the percent sign as in **PROG1;%A**. This is one of four possible formats for accessing named object attributes. The use and format of named object attributes is discussed in **Chapter 11.11: Named Object Attributes**.

The example below shows a sample program segment using program registers from within a program and from other programs

```
A = -5
B = [PROGRAM9;%P]*5.0
C = A + B /10.0
[PROGRAM4;%F] = F
D = PROGRAM3;%D
```

11.9 Program Control Attributes

There are 20 program control attributes associated with each program. Control attributes begin with the dollar sign (\$) character. Each program control attribute has a specific meaning that is summarized in **Table 11-2**.

CONTROL ATTRIBUTE	MEANING
\$	<p>Program status</p> <p>0=Stop program execution is stopped</p> <p>1=Run program is executing</p> <p>2=Unloaded program logic is not loaded into RAM</p> <p>3=Abort program has aborted due to a run-time error</p> <p>4=Time Dealy program is waiting for an WMAIT or SWAIT statement to timeout</p> <p>5=Restart initializes the program and starts executing it from the beginning</p> <p>6=Load Request request to load the program into RAM and begin execution</p> <p>7=Unload Request request to stop the program and unload it from RAM</p> <p>8=Abort Request PEX has encountered a run-time error and is aborting the program</p> <p>9=Network Access program is waiting for the completion of a network access</p> <p>10=Reload Request PEX has received a request to unload, reload and start a program</p>
\$D	Number of seconds remaining in time delay
\$E	Error code (0=no error) (Refer to Appendix B for a complete list of error codes)
\$S	Current section number
\$C	Program location counter in hexadecimal of next statement to be executed
\$W	Wait/Abort on trappable errors such as timeouts and CRC errors (see Appendix B)
\$P	Stack pointer used for the evaluation of nested expressions
\$B	Subroutine stack pointer contains 00h or return address of subroutine
\$N	Number of program attributes
\$Z	Paragraph pointer to Program Context Block (PCB)
\$L	Paragraph pointer to Program Logic Block (PLB)
\$R	Paragraph pointer to Program Reference Block (PRB)
\$A	Paragraph pointer to Program Attribute Block (PAB)
\$I	Paragraph pointer to pending Intertask Message (ITM)
\$F	Program code fetch state, 0=normal, <>0=fetching expressions
\$X	Expression state, 0=preexpr, 1=wait for aterm, 2=wait for bterm
\$M	Term state, 0=normal, 1=aterm, 2=bterm
\$#	Number of expressions in multiple expression term
\$H	Hard disk access counter is incremented for every hard disk read or write
\$1	Single step execution, 0=normal, 1=single step

Table 11-2 Program Control Attributes

The **\$E** control attribute specifies the error code number of the previously executed program statement. Normally this attribute equals zero (no error).

This special control attribute can be read as an attribute from the program and can be used to determine a course of action should an error occur (i.e., **\$E** <> 0). The example below illustrates the use of the **\$E** control attribute in a sample SPL program segment.

```
GetItt:      ;$E = 0
             A = [ZONE_TEMP;CV]
             ONERROR Err

Err:         IF ;$E==5 Then GetItt
             STOP
```

The **\$S** control attribute is another special control attribute which reflects the current section number of the program. The **SECTION** statement is used to set the **\$S** attribute to any integer value (refer to **Chapter 11.46: The SECTION Statement**). This control attribute can be used in diagnosing errors in your program logic. By using **SECTION** statements at strategic locations in the logic (e.g., before and after loops, conditional statements, calls to subroutines, etc.), you can check the progress of the program execution. By examining the **\$S** control attribute through OPI monitor/modify, you can determine if a particular segment of code is getting executed. The example below illustrates the use of **SECTION** statements so that the **\$S** control attribute can be examined.

```
SECTION 1
Call INITIALIZE
SECTION 2
Call CALC_LOOP
SECTION 3
Call PROCESS_LOOP
SECTION 4
Call SUBMIT_JOB
SECTION 5
STOP
```

For logic errors that are especially difficult to diagnose, you may choose to use the single step mode of execution (control attribute **\$1**) in conjunction with the **\$E** and **\$S** control attributes. When set to single step mode (**\$1=1**), the program is executed one line at a time. The program must be set to the **RUN** state (**\$\$=1**) after each line of the program is executed. In some complex programs, this may be a helpful way to determine if your program logic is doing what you really want it to do or if you are encountering a program error. Using single step mode may also make it easier to follow the logic of large programs at a statement-by-statement level.

The **\$S** control attribute reflects the current state of the program. This attribute can assume one of eleven values representing one of eleven possible states for the program. These states are:

- 0 - stop
- 1 - run
- 2 - unloaded
- 3 - abort
- 4 - wait for time
- 5 - restart
- 6 - load request
- 7 - unload request
- 8 - abort request
- 9 - network access
- 10 - reload request

Stop indicates that the program has stopped and is no longer executing its program code. *Run* indicates that the program is in the process of executing its program code. *Unloaded* indicates that the program logic has not yet been loaded into memory. *Abort* indicates that the program has stopped due to a run-time error. *Wait for time* indicates that the program has encountered an **MWAIT** or **SWAIT** statement in its logic and is in a wait state. *Restart* indicates that the program has been initialized and is going to start executing from its beginning. *Load request* indicates a signal for the program to be loaded into memory (i.e., RAM) and to begin execution. Conversely, *unload request* indicates a signal for the program to stop execution and unload (remove) itself from RAM. *Abort request* indicates the state prior to abort when the program executor (PEX) encounters a run-time error and signals that it should be aborted. *Network access* indicates that the program had executed either a network read or network write request and is currently in the process of waiting for the network transaction to be completed. *Reload request* is used to unload a program, then reload and start it.

The **\$D** control attribute reflects the number of seconds remaining in a time delay imposed by either an **SWAIT** or **MWAIT** statement. When **\$D**<>0, **\$\$**=4 (wait for time).

The **\$C** control attribute indicates the program location counter. As the program executes, **\$C** changes, reflecting the relative memory location of the next program statement to be executed. **\$C** is used primarily for low-level troubleshooting and diagnosis of program errors.

The **\$N** control attribute represents the number of user-defined program attributes that have been declared. The **\$N** control attribute will always equal the number of **ATTR** declarations within the program.

The **\$H** control attribute displays the current count of the number of times the SAGEMAX hard disk has been accessed by the program. It can be used to monitor the frequency of hard disk accesses by watching how rapidly **\$H** increases.

The **\$1** control attribute is used to set the single step mode of execution of a program. When set to 0, program execution continues normally. If this attribute is set to 1 (single step mode), program execution stops after each program line is executed. Once program execution is stopped, you can examine the other control attributes, registers and user-defined program attributes. This may be very helpful in troubleshooting and diagnosing hard-to-find errors in your program logic. The next line of program code can be executed by setting **\$\$=1** (the RUN state).

The **\$W**, **\$P**, **\$B**, **\$Z**, **\$L**, **\$R**, **\$A**, **\$I**, **\$F**, **\$X**, **\$M** and **\$#** attributes are used by PEX to control the execution of the program. Although their meanings are summarized in **Table 11-2**, the programmer and/or operator normally does not need to reference them.

Table 11-2 summarizes all of the program control attributes and explains what they represent.

CAUTION

You should *never* change any \$ attributes except **\$\$**, **\$E**, **\$\$**, **\$H** and **\$1**.

11.10 User-defined Program Attributes

Each SAGE^{MAX} program can have up to 255 unique, case-sensitive, user-defined program attributes. Each attribute has a two-character attribute name, a data type and a 32-bit value. Program attribute names can consist of any characters, but cannot begin with % or \$.

When you select an attribute name, choose one that has some mnemonic significance, that is, one that reflects the use or meaning of the attribute (e.g., **CV** for current value, **MN** for minimum, **TI** for time, etc.). This increases the readability of your program and will aid in the process of troubleshooting in the future.

The data type of program attributes can be any of the 256 standardized data types used by the SAGE^{MAX}. See Appendix H for a complete list of available data types supported by the SAGE^{MAX}.

Before you reference a user-defined program attribute within an SPL program, you must first *declare* the attribute by using the **ATTR** statement. The **ATTR** statement contains the attribute name and a data type argument.

See **Chapter 11.19 : The ATTR Statement** for more information on using the **ATTR** statement to declare user-defined program attributes.

A program's own attributes should be referenced in the program logic using their short form (**[;AT]** or **;AT**) since this results in the fastest execution of the program. A program can reference its own attributes by their full program attribute name (**[pgname;AT]** or **pgname;AT**) however this results in much slower execution of the program. The example below shows how to declare user-defined program attributes and illustrates some simple examples using program attributes.

```

ATTR CV,0FDh
ATTR KW,0E0h
ATTR TI,231
ATTR SZ,249
.
;CV = (ZONE_TEMP;CV - 32.0)*5.0/9.0
A= [KW] + 1.0E3
;TI = TIME
;SZ = 16.358
B = ;CV - ([PROG7;SP]/100)

```

11.11 Named Object Attributes

Attributes of other objects including points, variables, globals and other programs may be referenced as terms of expressions. The object name and attribute are specified using the same syntax as the Program Reference Block. The example below illustrates the formats available for named object attributes using SPL program segments.

```

A = \PT\HEAT_VALVE;CV
B = BLDG5_ROOM1;ZT + [\VR\OFFSET VALVE]
C = $MODE
[ 7WEST_SETPT;MN] = (C+[SETPOINT B]) / 2

```

For compatibility with RPL programs, the object name and attribute may be optionally enclosed in square brackets, e.g., **[objectname;attribute]**. Object names that begin with **0** to **9** and/or contain one or more spaces **must** be enclosed in square brackets.

Named object attributes may begin with a 2-character object type code. When used, this code is delimited by backslash characters (\). The named object type codes are:

- PT for points
- VR for variables
- GL for globals
- PG for programs

If a named object type code is not specified, the SAGEMAX performs an exhaustive database search for the named object. The search is conducted in a standard order starting with points and followed by programs, variables and globals. This is the same search hierarchy that was discussed in **Chapter 11.1.3: The Program Reference Block (PRB)** and was mentioned earlier in this chapter.

If no attribute is specified in the named object, the default attribute is assumed. For variables (which have a single value), it is not necessary to specify an attribute, since variables do not have attributes. For points and globals, the default attribute is the first attribute that appears when you monitor the point. For programs, the default attribute refers to the user-defined program attribute that is defined first in the program. If no user-defined program attributes are declared or if the program is unloaded, the **\$\$** control attribute is the default attribute.

11.12 References

If you are writing generic programs, you may wish to defer the decision of which named objects attributes are influenced by the program until the actual execution of that program. Similarly, there are applications when the specific object name or attribute name is unknown at the time the program is written.

SPL provides the PRB as a mechanism for specifying lists of up to 256 named object attributes for this purpose. SPL allows the elements of this list to be referenced indirectly by using the **REF (expression)** term in program statements.

When the **REF (expression)** term is used, the argument expression is evaluated at run time to determine an index from 0-255. The resulting index is used to determine which of 256 possible object names in the PRB to use in place of the **REF (expression)** term. If the result of the index expression is greater than 255, or greater than the actual number of references in the PRB, then an expression evaluation error occurs at run time. The syntax for the reference is shown below.

REF(expression)

NOTE

Reference index values as evaluated in an expression must be valid integer values. Integer data types are 00, 01, 02, E8h, E9h and FEh.

A simple SPL example of the **REF** term is shown below:

```

A = 100
B = 0
Calc:  B = REF (A-1) +B
      LOOP A, Calc
      REF (100) = B/100

```

11.13 Virtual Attributes

In some applications, it may be desirable to read or write attribute values to or from a point which has no database entry. SPL provides a mechanism for using arithmetic expressions to specify each of the necessary parameters for identifying the particular point attribute that is desired. This type of access is called *virtual attribute access*. The **UNS** term makes use of the SAGE^{MAX} unified network services for reading or writing object attributes without having to access the SAGE^{MAX} database. The syntax for the **UNS** term is as follows:

UNS(port,unit,ftype,card,chan,attr)

The arguments used in the **UNS** term are expressions that represent the port, unit, fundamental type, card, channel and attribute of the virtual point you want to access. In this syntax, **port** is an expression whose value must be in the range of 0-31, representing the SAGE^{MAX} port

number where the point is located. **Unit** is an expression representing the unit number (0-65535) on that port where the point is located. **Ftype** is an expression representing the fundamental type code and subchannel code for the point. The fundamental type range is 0-15 and the subchannel range is 0-15. They are combined using the formula

$$\text{ftype} = \text{ftype code} + (\text{subchannel} * 16)$$

Card is an expression representing the card number (0-255) for the point. **Chan** is an expression representing the channel number (0-65535 or 0-0FFFFh). **Attr** is not an expression, but is the literal attribute name as two case-sensitive characters.

The example below illustrates the **UNS** function in an SPL program:

```
ReadIt:      A=UNS(1,1234,0,0,0F900h,SP)
             B=UNS(3,1,04,3,7,HL)
WriteIt:     UNS(2,2037,0,0,0F900h,SP)=A
             UNS(4,17,04,6,1,HL)
```

NOTE

Not all SAGE^{MAX} port types or protocols require all of the UNS parameters to fully qualify a point. In those cases, the other parameters may be specified as zero.

11.14 Tables

Tables are collections of up to 1,073,741,824 data values which are stored as linear, one-dimensional arrays in disk-resident files. These files have a maximum size of 4,294,967,296 bytes.

SPL recognizes two types of tables: SPL tables and non-SPL tables. All SPL table files are stored in the reserved directory **C:\TABLES**. SPL table files may be stored in this directory or in any of its subdirectories and have the extension **.TBL**. A typical table file might have the full path name **C:\TABLES\XXX\YYY.TBL** which would be represented in an SPL statement by the path fragment **XXX\YYY**.

Individual SPL table terms may be referenced as terms in expressions using the syntax below.

&tablefragment(expression)

The argument expression in parentheses is evaluated to determine a zero-based index into the table. The resulting index is used to determine which data value to read from the table file. If the result of the index expression is greater than the actual number of values in the table file, an expression evaluation error occurs at run time. Examples of SPL table references are shown below.

```
&TABLE6(A+5)
&MYTABLE(4)
&LOOKUP\CLAIREX(D)
```

To access non-SPL tables (e.g., trend files), the entire pathname for the table must be specified using the following syntax.

&tablepath(expression)

This is done by placing a backslash character (\) as the first character after the **&** or a colon (:) as the second character after the **&**. Examples of non-SPL table references are shown below.

```
&\TREND\TREND1.TRN (J)
&C:\TREND\TREND2.TRN (K+L)
```

The code below shows an SPL programming example of the use of table references:

```
A=100
B=0
L1: B=B+&TABLE3(A-1)
    LOOP A,L1
```

11.15 Functions

Functions are arithmetic procedures which operate on one or more arguments and produce a single value result. Function calls may be used as terms in any expression. SPL provides a number of built-in arithmetic functions. The arguments to these functions can be integer or fixed expressions (**ix**), floating expression (**fx**), integer, fixed or floating expressions (**x**) or time expressions (**tx**). Each function is explained in detail below. A complete list of the functions available in SPL is shown in **Table 11-3**.

The **RETYPE** function is used to convert the data type of an expression to a specified data type. This function uses two expressions. The first expression represents the value to be converted. The second expression represents the new data type of the conversion. The code below shows examples of the **RETYPE** function in simple SPL program segments.

```
[MX_FE00;MX] = RETYPE (A,0FDh)
OAT1;CV = RETYPE (B,251)
```

The **FIX** function is used to convert the data type of an expression from floating point (E0h) to a fixed type. This function uses two expressions. The first expression represents the floating point value to be converted. If the second expression **ix** evaluates to a number from 1-10, then the result is a signed fixed data type with **ix** digits to the right of the decimal point. If the second expression evaluates to zero, then the result has the appropriate number of digits to represent the floating point value in a fixed format. If the second expression **ix** evaluates to a number greater than 10, it represents the desired data type, e.g., **FIX(XYZ,2)** is the same as **FIX(XYZ,0FAh)**.

FUNCTION	DESCRIPTION	RETURNED DATA TYPE	EXAMPLES
RETYPE(x1 ,x2)	Convert the value of expression x1 to a different data type specified by expression x2	x2	A=RETYPE9B,0FEh)
FIX(fx,ix)	Convert the value of a floating point expression fx to a fixed data type specified by the expression ix	fixed	A=FIX(1E3,0FFh)
FLOAT(ix)	Convert integer expression to floating point data type (0E0h)	float	A=FLOAT(106)
INT(x)	Convert an integer or floating point value to the largest integer value that is less than or equal to the value of expression x	integer	A+INT(3.96E1)
ROUND(x)	Round off the value of an integer or floating point expression	fixed or float	A=ROUND(C+D)
ABS(x)	Absolute value	integer or float	A=ABS((B+C+D)/3)
SIN(x)	Sine value	float	A=C*SIN(PI/6)
COS(x)	Cosine value	float	A=C*COS(PI/4)
TAN(x)	Tangent value	float	A=B*TAN(PI/3)
ARCTAN(x)	Arctangent value	float	A=ARCTAN9B)
LOG(x)	Base 10 logarithm	float	A=LOG(B+10)
LN(x)	Natural logarithm	float	A=LN(B*2/3)
SQRT(x)	Square root	integer or float	C=SQRT((A**2)+(B**2))
BETWEEN(tx1,tx2)	Between two times	integer	IF BETWEEN (A.B) Then L1 Else L2
MEAN(x1,x2...x8)	Mean value from list	integer or float	A=MEAN(B,C,D,E,F)
MAX(x1,x2...x8)	Maximum value from list	integer or float	A=MAX(B=5,C*2,D/4)
MIN(x1,x2...x8)	Minimum value from list	integer or float	A=MIN(B**2,C,D-10)
TODAY(ix)	Compare today's day of the week with a bitmap pattern	integer	IF TODAY(F900;AD) Then L1 Else L2
DATATYPE(x)	Get the data type of expression	integer	A=DATATYPE(B)

Table 11-3 SPL Functions

The code below shows examples of the **FIX** function and examples in simple SPL program segments

A=1.06E4	Output:	A=1.06E4	(float)
B=1		B=1	(integer)
C=FIX (A,B)		C=10600.0	(fixed)
D=3.4567E-1		D=3.4567E-1	(float)
E=FIX (D,B-1)		E=0.34567	(fixed)
F=FIX (D,0FDh)		F=0.3	(fixed)

The **FLOAT** function is used to convert an integer or fixed point value to a floating point value (E0h). This function uses a single expression to represent the integer or fixed point value that is to be converted to floating point. The code below shows an example of the **FLOAT** function using simple SPL statements.

A=421	Output:	A=421	(integer)
B=FLOAT(A=100)		B=5.21E2	(float)

The **INT** function is used to convert a floating or fixed point value to an integer that is less than or equal to the value of the specified expression. The resulting data type after using the **INT** function is always either 0FEh or 0FFh. Simple **INT** SPL program examples are shown in the code below:

A=7.5	Output:	A=7.5
B=-3.1		B=-3.1
C=-1.75E0		C=-1.75E0
D=INT(A)		D=7
E=INT(B*2)		E=-7
F=INT(C)		F=-2

The **ROUND** function is used to round off an integer, fixed or floating point value to the closest whole number. The result is either a fixed (for integer or fixed point values) or a floating point number (for floating point values). This function has a single expression **x** which represents the value to be rounded. The code below shows examples of the **ROUND** function using simple SPL statements.

A=7.5	Output:	A=7.5
B=-3.1		B=-3.1
C=-1.75E0		C=-1.75E0
D=ROUND(A)		D=8.0
E=ROUND(B*2)		E=-6.0
F=ROUND(C)		F=-1.0E0

The **ABS** function is used to get the absolute value of an integer, fixed or floating point value. This function has a single expression which represents the input value to the absolute value function. The result is either an integer, fixed or floating point value. Simple **ABS** SPL program examples are shown below:

A=7.5	Output:	A=7.5
B=-5.1		B=-5.1
C=-3.21E0		C=-3.21E0
D=ABS(A)		D=7.5
E=ABS(B)		E=5.1
F=ABS(C)		F=3.21E0

The **SIN** function is used to calculate the sine value of an expression. This function uses a single expression which represents the input value in *radians* (2π radians = 360°). The result is always in floating point format. The code below shows an example of the **SIN** function:

C=5	Output:	A=2.5E0
A=C*SIN(PI/6)		

The **COS** function is used to calculate the cosine value of an expression. This function uses a single expression which represents the input value in *radians* (2π radians = 360°). The result is always in floating point format. The code below shows an example of the **COS** function:

C=5	Output:	A=2.5E0
A=C*COS(PI/3)		

The **TAN** function is used to calculate the tangent value of an expression. This function uses a single expression which represents the input value in *radians* (2π radians = 360°). The result is always in floating point format. The code below shows an example of the **TAN** function:

Input: angle= 45° (pi/4 radians)	Output:	B=3.0E0
A=3		
B=A*TAN(PI/4)		

NOTES

*As the expression used in the **TAN** function approaches 90° ($1.5707965 = \pi/2$ radians) and 270° ($4.7123895 = 3\pi/2$ radians), the limits of the **TAN** function approach positive infinity ($+\infty$) and negative infinity ($-\infty$) respectively. These values are displayed as **+INF** and **-INF** when you use the **TAN** function.*

*The named term **PI** is most likely used in the expression argument for the **SIN**, **COS** and **TAN** functions.*

The **ARCTAN** function is used to calculate the arctangent (the inverse function of the tangent) value of an expression. This function uses a single expression which represents the tangent (**TAN**) value. The **ARCTAN** function determines an angular value in radians (2π radians = 360 °) whose tangent is the value specified by the input expression. The result is in floating point format and represents an angular value in radians. The code below shows a simple SPL program using the **ARCTAN** function:

Input: A=TANvalue=1.0E0	Output: B=7.853982E-1 (angle in radians)
	C=45 (angle size in degrees)
A=1.0E0	
B=ARCTAN(A)	
C=B/(PI/180)	

The **LOG** function is used to calculate the logarithm (base 10) of an expression. This function uses a single expression in integer, fixed or floating point format. The result of the **LOG** function is in floating point format.

The **LN** function is used to calculate the natural logarithm of an expression. This function uses a single expression in floating point format. The result of the **LN** function is in floating point format.

The code below shows examples of both the **LOG** and **LN** functions.

A=1.0E2	Output: A=1.0E2
B=2.0E0	B=2.0E0 (LOG(X))
C=LN(A)	C=4.6051702E0 (LN(X))

The **SQRT** function is used to calculate the positive square root of positive expressions (values greater than or equal to zero). This function uses one input expression that can be an integer, fixed or floating point type. The type of the result is either integer or floating point depending on the type of the input argument. The code below shows an example of the **SQRT** function in an SPL program:

A=1000.0	Output: A=1.0E3
B=SQRT(A)	B=3.162277E1

The **BETWEEN** function is used to determine if the current time of day is between two specified times. This function has two input expressions which are separated by a comma. These expressions represent the two times which must be in a time format. The integer result is either **0 (false)** if the current time is not between the two times specified, or **1 (true)** if the current time is between the two specified times. An example is shown below:

L20: A=7:00	Output: A=7:00 (1st time argument)
B=21:00	B=21:00 (2nd time argument)
C=BETWEEN (A,B)	C=1 if current time is between A & B
	C=0 if current time is not between A & B

NOTE

*When you use the **BETWEEN** function, the value of argument **tx1** must be less than the value of argument **tx2**.*

The **MEAN** function is used to determine the average value of a list of up to 8 expressions. The mean is calculated by summing the values of the expressions and then dividing by the number of expressions. The expressions used by this function can mix integer, fixed and floating point data types in the same statement. The input expressions are separated by commas. The result is the average value of the expressions listed, but may have a data type that is different than the input expressions. This is due to the calculation of the *average* value, in which the data type coercion rules for addition and division apply. Depending on your application, you may have to use the **RETYPE** function to get the data type that you desire. **RETYPE** was discussed earlier in this section.

The code below shows an example of the **MEAN** function using simple SPL program statements:

```
A=10.0          Output:   G=(A+B+C+D+E+F)/6=63.6
B=98.0
C=75.0          E=91.2
D=62.5          F=44.9

G=MEAN(A,B,C,D,E,F)
```

The **MAX** function is used to determine the maximum value of a list of up to 8 integer, fixed or floating point expressions. The arguments are separated by commas. The resulting data type is the data type of the maximum value. The code below shows an example of the **MAX** function using simple SPL program statements:

```
A=10.0          Output:   G=98.0
B=98.0
C=75.0          E=91.2
D=62.5          F=44.9

G=MAX(A,B,C,D,E,F)
```

The **MIN** function is used to determine the minimum value of a list of up to 8 integer, fixed or floating point expressions. The arguments are separated by commas. The resulting data type is the data type of the minimum value. The code below shows an example of the **MIN** function:

```
A=10.0          Output:   G=10
B=98.0
C=75.0          E=91.2
D=62.5          F=44.9

G=MIN(A,B,C,D,E,F)
```


The **TODAY** function determines whether or not the current day of the week is part of a day-of-the-week bitmap pattern specified by its integer expression. The result of this function is in integer format and is **0 (false)** if there is no match between the current day and the expression bitmap, and **1 (true)** if there is a match between the current day and expression bitmap.

To determine whether or not the match exists, the **TODAY** function logically **ANDs** a day-of-the-week mask with the argument. The day-of-the-week masks are shown below:

- Monday 0000 0001b (1)
- Tuesday 0000 0010b (2)
- Wednesday 0000 0100b (4)
- Thursday 0000 1000b (8)
- Friday 0001 0000b (16)
- Saturday 0010 0000b (32)
- Sunday 0100 0000b (64)
- Holiday 1000 0000b (128)

The code below illustrates the **TODAY** function in an SPL program example:

```

ATTR AD,0E9h
:
;AD=---11111b
Check: IF TODAY(;AD) Then Do_it
MWAIT 1
GOTO Check

Do_it: B=100

```

The **DATATYPE** function is used to determine the data type of an expression that you specify as the argument. The result of this function is in integer format. The code below illustrates an SPL example using the **DATATYPE** function:

```

D=DATATYPE(MX_FE01;CV)
:
C=A*1.035-5
:
MX_FE01;CV=RETYPE(C,D)
:

```

All of the functions available through SPL are summarized in **Table 11-3**. A description, the returned data type and an example are also shown in the table.

11.16 Nested Expressions

An expression may also be nested within parentheses and used as a term anywhere within an expression. Up to six levels of parentheses may be used.

The syntax for a nested expression is shown below.

(expression)

The evaluation of nested expressions occurs first from the innermost set of parentheses and continues outward. Expressions that are at similar levels of nesting are simply evaluated from left to right.

The code below shows some complex SPL programming examples using nested expressions.

```
A=MAX(MEAN(B,C,D), MEAN(E,F,G))
H=SQRT((B**2)+(C**2))
;SV=((;MX-;MN)/255)*;CV+;MN
```

11.17 Expression Operators

An arithmetic expression in SPL may be a simple term, or may be made up of a sequence of terms and operators. SPL provides both unary and binary operators. Unary operators have precedence over binary operators, so any term may be preceded by zero or more unary operators which are evaluated from left to right. Binary operators are always evaluated from left to right with no implied precedence. You must use parentheses to set operator precedence.

Table 11-4 lists all the unary and binary operators available in SPL. Included with each operator is a description of the function of the operator, a sample expression using the operator, and notes relating to the use of the operator.

NOTE

Since binary operators are evaluated from left to right with no precedence, you must use parentheses to indicate precedence for binary operations. For example:

$$12.34 + 45.7 / 2.0 + 2.1 = 31.12 \text{ and}$$

$$12.34 + 45.7 / (2.0 + 2.1) = 14.15609756.$$

NOTE

Logical expressions must be enclosed in parentheses for proper evaluation. For example:

```
IF (DAYOFWEEK==SAT) OR
(DAYOFWEEK==SAT) Then ...
```

Modulo (**MOD**) is simply a mathematical operation in which the result is the remainder of a division operation.

OPERATOR	DESCRIPTION	EXAMPLE	NOTES
-	Unary Negation	- A	Same as 0 - A
NOT	Unary Ones Complement	NOT A	32-bit integer result
**	Exponential	A**B	A to the B power (B is integer)
*	Multiplication	A*B	
/	Division	A/B	
MOD	Remainder After Division	A MOD B	e.g., 7 MOD 3 = 1
+	Addition	A+B	
-	Subtraction	A-B	
==	Equality	A==B	0 if not equal, 1 if equal
<>	Inequality	A<>B	1 if not equal, 0 if equal
>	Greater Than	A>B	1 if A>B, else 0
>=	Greater Than or Equal to	A>=B	1 if A>B or A=B, else 0
<	Less Than	A<B	1 if A<B, else 0
<=	Less Than or Equal to	A<=B	1 if A<B or A=B, else=0
AND	Bitwise Logical AND	A AND B	32-bit integer result
OR	Bitwise Logical OR	A OR B	32-bit integer result
XOR	Bitwise Logical Exclusive OR	A XOR B	32-bit integer result
SHL	Bitwise Shift Left	A SHL B	e.g., 1 SHL 3 = 8 (binary only)
SHR	Bitwise Shift Right	A SHR B	e.g., 256 SHR 7 = 2 (binary only)

Table 11-4 Expression Operators

11.18 *SPL Program Statements*

All of the various SPL program statements are listed in this section. This section is intended to familiarize you with all of the SPL programming statements by organizing them into logical groups based on the functions that they perform. Program statements fall into the following categories:

- program attribute manipulation
- assignment statements
- conditional and unconditional branches
- iteration control
- printing, logging and alarming
- spooling
- job execution
- PLB and subroutine execution
- program execution and control
- trending control
- execution error control
- debugging statements

Program attribute manipulation statements are used to declare user-defined program attributes and save the values of attributes to the program's attribute initial value (INI) file.

Assignment statements are used to assign the value of an expression to a variable. This type of program statement is characterized by the use of an equal sign (=).

Conditional and unconditional branches refer to program statements that change the order in which the logic is executed.

Iteration control refers to the ability to perform a statement or group of statements some number of times.

Printing, logging and alarming refers to statements that give you the ability to print information to a port, log information to a file, or generate formatted alarms of definable alarm classes.

Spooling refers to commands which offer the ability to send specified files to the printer.

Job execution refers to statements that give you the ability to execute any SAGE^{MAX} job from within the PEX environment.

PLB and subroutine execution statements are used to transfer program control to another portion of the program (a subroutine) or to another program's PLB.

Program execution and control refers to statements that can start, stop and prepare programs to be executed.

Trending control refers to program statements that can control the execution of trends from a program.

Execution error control refers to program statements that allow you to define a course of action for PEX when network access errors occur.

Debugging statements refer to programming statements that can be used to aid in the diagnosis of program logic errors.

Each SPL programming statement is individually explained, including sample SPL statements in the following pages. All of the SPL programming statements are summarized in **Table 11-5**.

11.19 *The ATTR Statement*

Local program attribute names and their data types are declared using the **ATTR** statement. Although user-defined program attributes can be declared anywhere within the SPL text, they must be declared before they are used in a statement. It is recommended that all user-defined program attributes be declared before any other SPL statements. User-defined program attributes that are referenced but not declared result in compile-time errors.

The syntax of the **ATTR** statement and SPL programming examples are illustrated in the code below:

```
ATTR CV,jffh
ATTR Z1,OFEH
ATTR KW,224
ATTR os,233
ATTR ti,0E6h
ATTR1A,OFFh
ATTR2B,OFEH
:
```

11.20 *The SAVE Statement*

The **SAVE** statement is an attribute manipulation command that is used to write the values of local program attributes to an initial value (INI) file. The **SAVE** command has two formats to accommodate saving all or selected program attribute values. **SAVE** by itself causes the current value of all program attributes to be written to the INI file. **SAVE** followed by up to 16 program attributes saves the values of only those attributes which are specified. Only those attributes specified will appear in the .INI file.

Regardless of the format used, the current INI file is renamed with a **.\$NI** extension and a new INI file is created with the updated data.

FORMAT	DESCRIPTION
variable=expression	assignment statement
ACTIVATE progname	start a stopped program - load if required
ALARM classexpr, "formatstring", x,x,x...x,x,x	generates an alarm
ATTR progattr,datatype	declare program attribute
CALL PLBname	goto external subprogram's logic block then load,execute and possibly unload it
CALL PLBname,STICK	same, but do not unload the program
DEACTIVATE progname	remove a program from memory (RAM)
ERRORABORT	trap condition - abort on errors
ERRORWAIT	trap condition - wait until no error
GOSUB label	goto internal subroutine
GOTO label	unconditional branch
IF expr THEN label	conditional branch if expr is true
IF expr THEN label1 ELSE label2	conditional branches if expr is true or false
JOB classexpr, "jobstring", x,x,x...x,x,x	request to SAGE ^{MAX} job scheduler
LOG logfilename, "formatstring",x,x,x...x,x,x	print (log) information to a file
LOOP register,label	iteration control
MWAIT expression	wait a certain amount of minutes
NOP	no operation used for debugging
ON expression GOTO label0,label1...labeln	indexed conditional branches
ONERROR label	trap condition - branch if error occurs
PRINT portexpr,classexpr,"formatstring",x,x,x...x,x,x	print information to a port
RESTART progname	start a program from beginning - load if required
RETURN	return from a subroutine
SAVE	copy all program attributes to INI file
SAVE aa,bb,cc,dd...	copy selected attributes to INI file
SECTION number	section marker used for debugging
SPOOL portexpr,pathname	send a file to be printed
SPOOL portexpr,pathname,DELETE	send a file to be printed, then delete it
STARTTREND trendname	activate a trend
STOP	halt execution of this program
STOP progname	halt execution of specified program
STOPTREND trendname	deactivate a trend
SWAIT expression	wait a certain amount of seconds
UNLOAD	remove this program from memory (RAM)
WAIT (expression)	wait until an expression is true, then go on

Table 11-5 Program Statements

The **SAVE** statement behaves like a **NOB** statement if no INI file is specified in the program definition. If an INI path fragment that does not currently exist is specified, PEX will create the file when the first **SAVE** is executed.

The code below illustrates the use of the **SAVE** statement in an SPL programming example:

```
ATTR HL,OFBh
ATTR LL,OFBh
ATTR CS,OFBh
ATTR HS,OFBh
:
;CS=A+5.0
;HS=B-7.0
SAVE CS,HS
:
```

11.21 The Assignment Statement

SPL allows various forms of value assignments. In each case, a variable on the left side of the equal sign (=) is assigned the new value dictated by the expression on the right side of the equal sign. The right side expression produces a value and a data type. Because automatic data type coercion may occur during evaluation, the data type of the expression may not match the data type of the variable on the left side. In this case, the data type and value from the expression may have to be coerced into the variable's data type according to certain rules (see **Chapter 11.49: Mixed-mode Arithmetic and Coercion**). **Table 11-6** summarizes the conversions in general.

left side	right side	effect
fixed	float	left = FIX(right)
float	fixed	left = FLOAT(right)
fixed	fixed	left = RETYPE(right)

Table 11-6 Assignment Conversions

There are 20 distinct types of fixed types, i.e., 10 decimal point positions for each signed and unsigned type.

When the left side variable is one of the program's own attributes, coercion of the expression into the proper data type is done automatically by PEX.

When the left side variable is a register, unless the data type of the result is converted according to the table above, the data type of the value in the register is automatically changed to the data type of the result.

When the left side variable is any other type of object, the result must be converted according to the table, otherwise incorrect values may be assigned to the variable.

For example, if the left side variable is a point whose data type is F9h (#####.###) and the expression has a data type of F7h (#####.#####), a **RETYPE** (*expression*, F9h) must be done so that the value assigned to the variable is not, in this case, 10 times too large.

There are several general forms of assignment statements which may be used in SPL. They are summarized below.

- *register = expression*
- *;programattribute = expression*
- *namedobject = expression*
- *\objecttype\namedobject = expression*
- *namedobject;attribute = expression*
- *\objecttype\namedobject;attr = expr*
- **REF** (*expression*) = *expression*
- *&tablename(expression) = expression*
- **UNS**(*x1,x2,x3,x4,x5,attribute*) = *expr*

These forms of assignment statements are illustrated in the code below:

```
A=B+C
[;CV] = B+C
OAT = B+C
[ZONETEMP] = B+C
[1STFLOOR] = B+C
\VR\OAT = B+C
[LOOP;SP] = B+C
\PT\LOOP;SP = B+C
REF(6) = B+C
&CLAIREX(29) = B+C
UNS(1,0,0,0,0FB00h,CV) = B+C
```

At first, there would appear to be a conflict in syntax between local attribute references that are used as variables on the left side of assignment statements (e.g., **;AT**=*expression*), and comment statements since both begin with a semicolon. The difference in syntax between the two is that comments must begin in the leftmost column, while local program attribute references used as variables must have at least one leading space or **TAB** character. In order to avoid confusion, local program attributes may be optionally enclosed in brackets (i.e., [**;AT**] = *expression*).

11.22 The GOTO Statement

The **GOTO** statement is an unconditional branch statement that causes program logic to jump to some other location that is identified by a label.

The code below illustrates the use of the **GOTO** statement and shows a sample SPL programming example. It may increase the readability of your program logic if you add a blank line after **GOTO** statements.

```
      :  
L1:   C = A+B  
      GOTO L3  
  
L2:   C = B-A  
L3:   D = C*2  
      :
```

11.23 The IF/THEN Statement

The **IF/THEN** statement is a conditional statement that causes the program logic to jump to some other location identified by a label if a certain condition is true. If the condition is false, execution *falls through* to the next sequential statement.

The code below illustrates the use of the **IF/THEN** statement and shows its usage in an SPL programming example.

11.24 The IF/THEN/ELSE Statement

The **IF/THEN/ELSE** statement is a conditional statement that causes the program logic to jump to some other location if a certain condition is true. If the condition is false, execution is passed to a second location.

The code below illustrates the use of the **IF/THEN** and the **IF/THEN/ELSE** statement in an SPL programming example.

```
      IF (Dayofweek==SUN) Then L3  
L0:   IF (A>B) Then L1 Else L2  
L1:   C=A+B  
      GOTO L3  
  
L2:   C=B-A  
L3:   D=C*2  
      :
```

11.25 *The ON/GOTO Statement*

The **ON/GOTO** statement is a conditional statement that identifies a series of indexed labels to which PEX transfers control based on the value of an expression. The code below illustrates the use of the **ON/GOTO** statement.

```

ATTR ER,07
      :
ON INT(B-10) GOTO L0,L1,L2
:ER=1
PRINT 1,0,"Unsuccessful."
GOTO Done

L0:    D = (C+1)/2
      GOTO Merge

L1:    D = (C+20)/2
      GOTO Merge

L2:    D = (C+50)/2
Merge: PRINT 1,0,"Success. D=%?%",D
Done:  STOP

```

The indices of the **ON/GOTO** statement are zero-based. In addition, if an index evaluates to a number that is greater than the number of indices, program execution continues with the next line of the program.

11.26 *The MWAIT and SWAIT Statements*

The **MWAIT** and **SWAIT** statements are used to cause a timed delay in program execution. These statements each have a single argument which represents a number of minutes or seconds (respectively) that must pass before program execution continues. The time delay can be viewed as it counts down from the **\$D** program control attribute. This attribute shows all time delays in seconds. Once the delay reaches zero, the next program statement is executed.

The code below illustrates the proper use of the **MWAIT** and **SWAIT** statements. Sample SPL programming examples are also shown.

```

L0:    IF (SWITCH==1) Then L1
      MWAIT 5
      GOTO L0

L1:    [PROG1;MN]=[PROG2;SP]+5.0

```

11.27 The WAIT Statement

The **WAIT** statement is a conditional statement that halts further program execution until the expression specified in the argument is true.

The code below illustrates the syntax of the **WAIT** statement and shows a sample SPL programming example.

```

LO:      WAIT $ALARMS
        CALL Notify

L1:      CALL Clear, STICK
        IF $ALARMS THEN L1 ELSE LO

```

11.28 The LOOP Statement

The **LOOP** statement is an iteration control statement that performs a “decrement register and jump if not zero” function using a specified register and a program label. The **LOOP** statement is a combination of an assignment statement (e.g., **A = A-1**) and a conditional statement (e.g., **IF A>0 THEN Continue**).

The code below illustrates the proper use of the **LOOP** statement in a sample SPL programming example.

```

        A = 100
        B = 0
Calc:   B = REF (A-1)+B
        LOOP A, Calc
        REF (100)=B/100

```

11.29 The PRINT Statement

The **PRINT** statement is used to send text output to a port on the SAGE^{MAX}. In the syntax of the **PRINT** statement (see **Figure 11-3**), the *portexpr* expression defines which SAGE^{MAX} port (0-31) to print to. The *classexpr* expression represents the alarm class code (0-255) to be applied to the text message. The *formatstring* field of the **PRINT** statement contains a base text string with *format specifiers* that determine the representation of the corresponding expression in the list of expressions (**x,x,x...x,x,x**) which follow the format string.

Figure 11-3 illustrates the syntax and program examples using the **PRINT** statement. **Table 11-7** lists the format specifiers that are supported by the SAGE^{MAX}.

PRINT portexpr, classexpr, "formatstring", x,x,x...x,x,x

Used to send text output to a port on the SAGE according to the class specified by *classexpr* using the format specified by *formatstring*. You can also include the values of expressions in the output.

```

;-----
; This program illustrates a very simple printer program using the PRINT statement.
;-----
      ATTR  PT,0FFh
      ATTR  FP,0E0h
      ATTR  SI,0FFh
      ATTR  FX,0F9h
      ATTR  PC,0E0h

START:;PT = -1
      SWAIT1

WAIT: IF ( (;PT < 0) OR (;PT > 13) ) THEN START

      ;FP = 1.234567E14
      ;SI = -98765
      ;FX = 85.456
      ;PC = 99.2E0

First LOG the formatted print statements to a file.
Print ;PT,0," TSTPRINT Program Variables on %W% %D%-M3%-Y4%"
Print ;PT,0," Signed Integer [;SI] = %I7% (7 digits)%13%%10%";SI
Print ;PT,0," Signed Integer [;SI] = %?% (default format)%13%%10%";SI
Print ;PT,0," Signed Integer [;SI] = %?L3% (left 3 digits)%13%%10%";SI
Print ;PT,0," Signed Integer [;SI] = %?R4% (right 4 digits)%13%%10%";SI
Print ;PT,0," Floating Point [;FP] = %?% (scientific)%13%%10%";FP
Print ;PT,0," Fixed Point [;FX] = %?% (default format)%13%%10%";FX
Print ;PT,0," Percent [;PC] = %F3.1% %% (float with percent symbol%%)%13%%10%",
;PC

Print ;PT,0," Referenced Object Ref (0) %R% = %?% (with name)%13%%10%",0,REF(0)
Print ;PT,0," The first string from \SPL\SPL.TXT (0-based) = %S%",1
GOTO START

```

```

TSTPRINT Program Variables on Thu 06-Feb-1992
Signed Integer [;SI] = -98765 (7 digits)
Signed Integer [;SI] = -98765. (default format)
Signed Integer [;SI] = -98 (left 3 digits)
Signed Integer [;SI] = 765. (right 4 digits)
Floating Point [;FP] = 1.234567E+14 (scientific)
Fixed Point [;FX] = 85.456 (default format)
Percent [;PC] = 99.2% (float with percent symbol %)
Referenced Object Ref (0) TSTLOG;PC = 99.2 (with name)
The first string from \SPL\SPL.TXT (0-based) = User-definable text strings

```

Figure 1-3 The PRINT Statement (top), PRINT Statement Output (bottom)

11.30 The LOG Statement

The **LOG** statement is used to send text output to a file. In the syntax of the **LOG** statement, the *logfile* argument represents a complete pathname of a text file to which the text output will be appended. If *logfile* does not exist, it will be created. The *formatstring* field of the **LOG** statement contains a base text string with *format specifiers* that determine the representation of the corresponding expression in the list of expressions (*x,x,x...x,x,x*) which follow the format string. A list of format specifiers is shown in **Table 11-7**.

Figure 11-4 illustrates the use of the **LOG** statement.

LOG logfile, "formatstring", x,x,x...x,x,x

Used to append text to *logfile* using the format specified by *formatstring*. You can also include the values of expressions in the output.

logfile-DOS file name to receive information
x,x,x...x,x,x-expressions whose values are inserted into the *jobstring* text in place of format specifiers when PEX submits the job to the SAGE job scheduler.

```

;-----
; This program illustrates a very simple logger/spooler.
;-----
      ATTR PT,0FFh
      ATTR FP,0E0h
      ATTR SI,0FFh
      ATTR FX,0F9h
      ATTR PC,0E0h

START:;PT = -1
      SWAIT 1

WAIT:IF ( ;PT < 0) OR ( ;PT > 13) THEN START

      ;FP = 1.234567E14
      ;SI = -98765
      ;FX = 85.456
      ;PC = 99.2E0

;      First LOG the formatted print statements to a file.
LOG \LOG\TSTLOG.LOG," TSTLOG Program Variables on %W% %D%-M3%-Y4%"
LOG \LOG\TSTLOG.LOG," Signed Integer [;SI] = %I7% (7 digits)%13%10%",;SI
LOG \LOG\TSTLOG.LOG," Signed Integer [;SI] = %?% (default format)%13%10%",;SI
LOG \LOG\TSTLOG.LOG," Signed Integer [;SI] = %?L3% (left 3 digits)%13%10%",;SI
LOG \LOG\TSTLOG.LOG," Signed Integer [;SI] = %?R4% (right 4 digits)%13%10%",;SI
LOG \LOG\TSTLOG.LOG," Floating Point [;FP] = %?% (scientific)%13%10%",;FP
LOG \LOG\TSTLOG.LOG," Fixed Point [;FX] = %?% (default format)%13%10%",;FX
LOG \LOG\TSTLOG.LOG," Percent [;PC] = %F3.1% %% (float with percent symbol
%%)%13%10%",;PC
LOG \LOG\TSTLOG.LOG," Referenced Object Ref (0) %R% = %?% (with name)%13%
%10%",0,REF(0)
LOG \LOG\TSTLOG.LOG," The first string from \SPL\SPL.TXT (0-based) = %S%",1

;      Now spool the file to port [;PT] immediately.

SPOOL ;PT, \LOG\TSTLOG.LOG
GOTO START

```

Figure 1-4 The LOG Statement

11.31 The SPOOL Statement

The **SPOOL** statement is used to send log files that were created by **LOG** statements to a specified port. The **SPOOL** statement causes the specified port to begin printing the file as soon as it can.

The *portexpr* argument specifies the port number where the file is to be **SPOOLED**. The *pathname* argument specifies the path and filename of the file to be **SPOOLED**.

If the **DELETE** argument is used with this command, then the file will be deleted after it is **SPOOLED**.

The code below illustrates the use of the **SPOOL** statement in an SPL programming example:

```
LOG C:\ValFile.TXT,"Value is %I2% and",A
LOG C:\ValFile.TXT,"Setpoint is %F2.1%",B
LOG C:\ValFile.TXT,"at %T5% on %W%."
LOG C:\ValFile.TXT,"%13% %10%"
SPOOL 13,C:\ValFile.TXT
```

11.32 The ALARM Statement

The **ALARM** statement is used to send text to the Alarm Log (ALOG) task for alarm processing. In the syntax of the **ALARM** statement (see **Figure 11-5**), the *classexpr* expression represents the alarm class code (0-255) to be applied to the text message. This class code is used by the ALOG task to determine how to process the alarm message. The *formatstring* field of the **ALARM** statement contains a base text string with *format specifiers* that determine the representation of the corresponding expression in the list of expressions (x,x,x...x,x,x) which follow the format string. The code below illustrates examples of the **ALARM** statement in use.

```
A=1234
B=29
C=19
[;KW] = 5.0E3
[;RL] = 300
:
ALARM B,"Peak Usage ---> %F1.5%KW",[;KW]
ALARM C+11, "Maintenance check for unit %I5% at %I3% hours.",A,[;RL]
ALARM 030,"!This is the %I1%nd time around",2
```

Figure 11-5 shows the printed results of the statements shown in the code above. **Table 11-7** lists the format specifiers that are supported by the SAGE^{MAX}.

```

O 00013/00000 029 Mon 09-Feb-91 12:34:56 O
O - 029 Peak Usage --> 5000.00000 KW O
O 00023/00000 030 Mon 09-Feb-91 12:35:01 O
O - 030 Maintenance check for unit 1234 at 300 hours. O
O - 030 This is the 2nd time around O
O O

```

Figure 1-5 Printed Results from the ALARM Statement Example Code

When ALOG processes a message sent to it through the **ALARM** statement, it automatically adds a transaction number, source code, class, day, date and time information in front of the message before routing it, unless the message is preceded by an exclamation point (!).

By using an exclamation point (!) in the format string, an alarm continuation line is generated. Alarm continuation lines begin with a hyphen (-), contain the class and the message text and must follow an alarm line. See the example code shown above.

11.33 The JOB Statement

SPL provides for the execution of any built-in SAGE^{MAX} job function within the SAGE^{MAX} execution environment. SAGE^{MAX} jobs are files with the extension .JOB and include the following:

- RPT.JOB
- SPOOL.JOB
- BC.JOB
- DCS.JOB
- UDL.JOB
- EXPORT.JOB

The syntax of the **JOB** statement is illustrated in **Figure 11-6**. Note that the *classexpr* argument in the **JOB** statement is required, but is currently not used. For now, it should be specified as zero.

The *jobstring* argument varies based on the type of job that is being requested (e.g., report, spool, etc.). The syntax requirements for each specific job application are explained in detail in the following sections. These sections also contain SPL programming examples that illustrate the **JOB** statement for each job type.

The expression arguments (x,x,x...x,x,x) of the **JOB** statement contain expressions whose values are substituted for format specifiers contained in *jobstring* when PEX submits the job string to the SAGE^{MAX} job scheduler. For example, in the following **JOB** statement

```
JOB 0,"BC %I2%/ %I1%/Hello",14,0
```

would be submitted to the job scheduler as

```
BC 14/0/Hello.
```

A list of format specifiers is shown in **Table 11-7**.

JOB *classexpr*, "*jobstring*", *x,x,x...x,x,x*

Used to request a job "as soon as possible." Job statement arguments have the following meanings:

classexpr-class expression which must be set to zero

jobstring-used to specify parameters that are specific to the type of job being requested
 RPT.JOB **RPT** *templatename, valuepathname, reportpathname*
 SPOOL.JOB**SPOOL** *port pathname/D/B/Sbaud/Ntelephonenumber*
 BC.JOB **BC** *port/unit/messagetext*
 DCS.JOB**DCS** *pathname /S/D*
 UDL.JOB **UDL** *t port peername delimiter remotepath localpath*
 EXPORT.JOB**EXPORT** *nbfpath /switch ... /switch*

x,x,x...x,x,x-expressions whose values are inserted into the *jobstring* text in place of format specifiers when PEX submits the job to the SAGE job scheduler.

Figure 1-6 Generic Syntax of the JOB Statement

CAUTION

You must take care to specify the job string correctly, as the SPL compiler and PEX do not check the validity of jobstring.

FORMAT	DESCRIPTION
% %	the character % itself
??%	use the data type of the expression - default characters wide
?R _x %	use the data type of the expression - only show the x rightmost characters
?L _x %	use the data type of the expression - only show the x leftmost characters
F _{x.y} %	for floating point format where x is the number of digits to the left of the decimal point and y is the number of digits to the right of the decimal point (e.g., %F3.2% is ###.##)
I _x %	for integer with field that is x digits wide
U _x %	for unsigned integer with field that is x digits wide
H _x %	for hexadecimal mode with field that is x digits wide
B _x %	for binary mode with field that is x digits wide
S _x %	gets x characters from a specified line in the ASCII text file C:\CFG\SPL.TXT. The associated argument in the SPL statement specifies the line number index of the file (e.g., ALARM 0, "%S20%", 100 means you get 20 characters from line 100).
S%	gets the entire line of characters in the ASCII text file C:\SPL\SPL.TXT. The associated argument in the SPL statement specifies the line number index of the file (e.g., ALARM 0, "%S%", 54 means you get all characters from line 54).
xx%	for control characters using decimal numbers xx (e.g., %10% for <If>, %13% for <cr> or %07% for <bel>)
R%	the 24-character name of the reference that matches the variable
T _x %	current time in HH:MM:SS format (x=8), HH:MM format (x=5) or HH format (x=2)
W%	current day of the week (e.g., MON, TUE, WED, etc.)
D%	current day of the month as two decimal digits
M _x %	current month as two digits (x=2) or as a 3 letter abbreviation (x=3)
Y _x %	current year as last two digits (x=2) or as full year (x=4)

Table 11-7 Format Specifiers Used by ALARM, LOG, JOB and PRINT Statements

11.33.1 The REPORT Job

SPL provides enhanced report generation capabilities through the **REPORT** job using the **JOB** “RPT” statement. The **RPT** format of the **JOB** statement is used to issue an “as soon as possible” request to the SAGE^{MAX} scheduler to generate a report.

The **REPORT** job is used to merge a text template file with a file containing data values to produce a final report file. This report file may be printed or saved for later reference.

Reports are created by first creating the *text template file*. This file contains the basic, underlying text of the report. The text file identifies locations to be filled in with data by including special place holders called *format specifiers* (see **Table 11-7**).

The data file may be either a SAGE^{MAX} table file (**.TBL**) or a trend file. The resulting output file is all text.

JOB 0, "RPT *templatepath, valuepath, reportpath*", x,x,x...x,x,x

Used to request a report job "as soon as possible." Creates the file specified by the path *reportpath* from the template file specified in the path *templatepath* while replacing every %...% token with the next expression in the list x,x,x...x,x,x which is contained in the file specified by the path *valuepath*. Job statement arguments have the following meanings:

template - name of the ASCII text template file
valuepath - data file that contains values in CSV, TBL or TRN format
reportpath - the ASCII text output report file
x,x,x...x,x,x - expressions whose values are inserted into the *jobstring* text in place of format specifiers when PEX submits the job to the SAGE job scheduler.

```

;-----
; This program is a simple report printer using JOB "RPT ..." and JOB "SPOOL ..." assuming
; that the data has already been collected. This program prints either at the start of a new day or
; on demand. This simplifies preset on-demand report generation for an operator by allowing him
; to set a program attribute rather than submit a JOB through the SAGE menuing system. This
; also allows generation of the report through a host such as EtherView.
;
;
; Scheduling of the report generation/spooling can also be accomplished through the SAGE
; menuing system. The report generation at midnight is included here to illustrate the use of the
; BETWEEN statement.
;
;
; Attributes:PT request to print report, if <> -1 , PT is the port to spool to
;              DP default port for auto-printing
;-----
                ATTR PT,0FFh
                ATTR DP,0FFh
START: ;DP = 13
;-----
; If we restarted between midnight and 1:00 AM, then do report now.
;-----
                IF (BETWEEN (0:00, 1:00)) THEN REPORTA
RSTART:;PT = -1
                D = DAYOFYEAR

TOP:      L = 60
WAITMIN:IF (;PT <> -1) THEN REPORTM
                SWAIT 1
                LOOP L,WAITMIN
;-----
; Check for day rollover every 60 seconds. Use D register to keep the current Julian
; day. If D is not the same as the current Julian day, then it is time to auto report.
;-----
                IF (D == DAYOFYEAR) THEN TOP
;-----
; Auto-print. Set ;PT to default port
;-----
REPORTA:;PT = ;DP
;-----
; Submit RPT job request then SPOOL job on port ;PT
;-----
REPORTM:JOB 0,"RPT \LOG\TSTRPT.TMP, \TABLES\TSTRPT.TBL, \LOG\TSTRPT.RPT"
                JOB 0,"SPOOL %I2%, \LOG\TSTRPT.RPT",;PT
                GOTO RSTART

```

Figure 1-7 Program TSTRPT.SPL Illustrating the JOB "RPT" Statement

Figure 11-7 illustrates the syntax of the **JOB** statement for a sample report (**RPT**) job. The template file, table file and output file are shown in **Figure 11-8**, **Figure 11-9**, and **Figure 11-10**, respectively.

```

*****
                        OPTIMIZED START REPORT
Date: %w% %d%-%m3%-%y4%Time: %t5%

TODAY'S AVERAGE OUTSIDE AIR TEMPERATURE : %?% deg F
TODAY'S LOW OUTSIDE AIR TEMPERATURE : %?% deg F
TODAY'S HIGH OUTSIDE AIR TEMPERATURE : %?% deg F

LOCATION      OPT. START  OPT. STOP  RUN HRS  MAINT
AHU 1 - Building A%?? %?? %I5% %I5%
AHU 2 - Building A%?? %?? %I5% %I5%
AHU 3 - Building A%?? %?? %I5% %I5%
*****
    
```

Figure 1-8 Sample Template File C:\LOG\TSTRPT.TMP

Table: C:\TABLES\TSTRTP.TBL contains 19 entries, 5 bytes per entry.

Index	DT	Value
0:	[FDH]	51.0
1:	[FDH]	31.0
2:	[FDH]	59.7
3:	[E6H] 07:30	
4:	[E6H] 16:45	
5:	[FEH]	2010
6:	[FEH]	3000
7:	[E6H] 06:45	
8:	[E6H] 17:00	
9:	[FEH]	1099
10:	[FEH]	2200
11:	[E6H] 06:45	
12:	[E6H] 17:30	
13:	[FEH]	2200
14:	[FEH]	2122

Figure 1-9 Sample Table File C:\TABLES\TSTRPR.TBL

```

*****
                        OPTIMIZED START REPORT
Date: Wed 12-Feb-1992Time: 11:05

TODAY'S AVERAGE OUTSIDE AIR TEMPERATURE : 51.3 deg F
TODAY'S LOW OUTSIDE AIR TEMPERATURE : 31.0 deg F
TODAY'S HIGH OUTSIDE AIR TEMPERATURE : 59.7 deg F

LOCATION      OPT. START  OPT. STOP  RUN HRS  MAINT
AHU 1 - Building A07:30      16:45      2010      3000
AHU 2 - Building A06:45      17:00      1099      2200
AHU 3 - Building A06:45      17:30      2200      2122
*****
    
```

Figure 1-10 Sample Output File C:\LOG\TSTRPT.RPT

11.33.2 The SPOOL Job

The **SPOOL** format of the **JOB** statement is used to issue an “as soon as possible” request to the SAGE^{MAX} scheduler to print out (spool) a particular file. Typically the request will be sent to the line printer task (LPT). Other types of tasks also support **SPOOLing** services. For example, the PHPHDial driver responds to **SPOOL** requests by dialing a particular telephone number and then printing the file once the connection is established.

SPOOL can be executed as a job under SPL. **Figure 11-7** illustrates the syntax of the **JOB** statement for a **SPOOL** job in conjunction with an **RPT** job. **Figure 11-11** shows the syntax of the **SPOOL** job statement.

JOB 0,“SPOOL *port pathname /D /B /Sbaud /Ntelephonenum*”,*x,x,x...x,x,x*

Used to request a spool (print) job “as soon as possible.” Arguments have the following meanings:

port - specifies the SAGE port number (0-31) that provides a spool service

pathname - valid DOS pathname of the file to be spooled

/D - optional switch used to delete *pathname* after printing

/B - optional switch used to add banner line information to printed listing

/Sbaud - optional switch used to specify baud rate for dialout (300, 600, 1200, 2400, 4800, 9600, 19200, 38400). If not used, the default is 2400.

/Ntelephonenum-optional dialout phone number used to specify destination for dialout port. Valid telephone number characters include 0-9, # and *.

x,x,x...x,x,x-expressions whose values are inserted into the *jobstring* text in place of format specifiers when PEX submits the job to the SAGE job scheduler.

; This program segment is taken from the complete program in . It illustrates the
; use of the JOB “SPOOL” statement. Note the use of the ;PT attribute as a variable used by
; the SPOOL statement.

:

```
REPORTM:JOB 0,“RPT \LOG\TSTRPT.TMP, \TABLES\TSTRPT.TBL, \LOG\TSTRPT.RPT”
      JOB 0,“SPOOL %i2%, \LOG\TSTRPT.RPT”,;PT
      :
```

Figure 1-11 The JOB “SPOOL” Statement

The *port* argument refers to any SAGE^{MAX} port number which provides spool services. The *pathname* argument may be any valid DOS pathname.

The */S* and */N* switches are only relevant for dial-out purposes. If the baud rate is not specified for a dial out port, it will default to 38400 baud. The telephone number defaults to *none* which would be ignored by a dial-out port. The telephone number is specified as a sequence of digits, possibly including # and *. You are not permitted to use dashes, parentheses or spaces in the telephone number.

CAUTION

Dashes, parentheses and spaces may not be used within the Ntelephonenum argument.

Note that there may be a significant difference when a file is printed using a **SPOOL** statement versus a **JOB "SPOOL..."** statement. The latter submits a **SPOOL** job to the SAGE^{MAX} job scheduler while the former submits a **SPOOL** command directly to the port task. In the latter case, if there are jobs in the schedule queue, the new **SPOOL** job will not be submitted to the port immediately. If you are generating a report that needs to be **SPOOLED** in the same program, you should use the **JOB "SPOOL..."** statement to make sure that the report is finished before it is **SPOOLED**, i.e., the **SPOOL** job is inserted into the job queue after the **REPORT** job.

11.33.3 The **BROADCAST** Job

The broadcast job (**BC**) is used to issue an "as soon as possible" broadcast of *message* text to the SAGE^{MAX} scheduler to a specified SAGE^{MAX} *unit* number on a specified SAGE^{MAX} *port*.

Figure 11-12 illustrates the syntax of the **JOB** statement for broadcast (**BC**) jobs.

JOB 0,"BC port/unit/message",x,x,x...x,x,x

Used to request a broadcast message job "as soon as possible." Broadcast job statement arguments have the following meanings:

port - SAGE port number of broadcast destination
unit - device unit number of broadcast destination or SAGE peername
message-message text to be broadcast
x,x,x...x,x,x-expressions whose values are inserted into the *jobstring* text in place of format specifiers when PEX submits the job to the SAGE job scheduler.

```

;-----
; This program is a simple broadcaster program that can submit a broadcast request on demand.
; This simplifies the broadcast process and allows it to be done from a host such as EtherView.
;
; Attributes:RQ <>-1 means request a broadcast on this port
;              UN unit (-1 is broadcast to all units on port RQ)
;-----

```

```

ATTR RQ,0FFh
ATTR UN,0FFh

```

```

START: ;RQ = -1
WAITRQ:IF (;RQ <> -1) THEN BCAST
        SWAIT 1
        GOTO WAITRQ

BCAST:IF (;RQ >13) THEN START
        IF (;UN == -1) THEN BCAST2
        JOB 0,"BC %I2%/ %I5%/This is a message for a specific unit";[:RQ],[;UN]
        GOTO START

BCAST2:JOB 0,"BC %I2%/-/This is a critical message for all units";[:RQ],[;UN]
        GOTO START

```

Figure 1-12 The **JOB "BC"** Statement

11.33.4 The DATA CAPTURE /DATA STUFF Job

The data capture/data stuff job (**DCS**) is used to create an “as soon as possible” request to the SAGE scheduler gather real-time values (capture) from a list of *named object attributes*, and/or modify values (stuff) from a list of *named object attribute/value* pairs.

Captured data may be formatted into several forms for later use. The captured data may be stored in Comma Separated Value (CSV) format which is useful for Lotus or Excel. As an alternative, data may be stored in a format suitable for use as a Data Stuff input file for use at a later date. During data stuffing, an *audit file* can be created to record the success or failure of each stuff.

Figure 11-13 illustrates the syntax of the **JOB** statement for data capture/stuff (**DCS**) jobs along with several SPL programming examples.

```

JOB 0,“DCS pathname /S/D”,x,x,x...x,x,x

Used to request a broadcast message job “as soon as possible.” Broadcast job statement arguments have the following meanings:
  pathname-an ASCII text file with a valid DOS name and extension .PDF which is a points
              description file.
  /S        - optional switch to specify captured data in Stuff file format
              (default is Comma Separated Variable or CSV format is blank)
  /D        - optional switch which turns on debug tracing
  x,x,x...x,x,x-expressions whose values are inserted into the jobstring text in place of
              format specifiers when PEX submits the job to the SAGE job scheduler.

;-----
; This program illustrates a simple data collection. It is used to submit a data collection job
; request when it is a holiday. In this program, “holiday” is defined to be whenever the SAGE
; global calendar $MODE variable is 255 and the time is between 00:00 and 00:30. This can
; also be submitted on demand by an operator or host such as EtherView.
;
; Attributes:  RQ<> 0 means request a data collection
;-----
              ATTR  RQ,0FFh
;
;                               If we restarted between midnight and 12:30 AM AND
;                               $MODE = 255, then do collection immediately.

START: IF (($MODE == 255) AND (BETWEEN (0:00,0:30))) THEN COLLECT

RSTART:;RQ = 0
        D = DAYOFYEAR

TOP:    L = 60
WAITMIN:IF (;RQ <> 0) THEN COLLECT
        SWAIT 1
        LOOP L,WAITMIN
;
;                               Check for holiday ($MODE=255) and time between
;                               0:00 and 0:30 every 60 seconds. Use D register to
;                               keep the current Julian day. If D is the same as the
;                               current Julian day, then we've done the collection for
;                               that day.

        IF (D == DAYOFYEAR) THEN TOP
        IF ( ($MODE == 255) AND (BETWEEN (0:00,0:30) ) ) THEN COLLECT ELSE TOP

COLLECT:JOB 0,“DCS \LOG\HOLIDAY.PDF”
        GOTO RSTART

```

Figure 1-13 The Job “DCS” Statement

The Points Definition File (**.PDF** file) contains lines of text which adhere to one of several possible formats. The lines may be up to 128 characters long before the carriage return and linefeed. Extra characters will be ignored. The seven possible line formats for **.PDF** files are:

- *;comment line*
- *>pathname*
- *>>pathname*
- *objectname,description*
- *!pathname*
- *!!pathname*
- *objectname=value*

Lines which begin with a semicolon (;) character are treated as comment lines and are ignored by **DCS**.

The *>pathname Key*”> format causes further data capture list output to be directed to the file *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain *wildcards* as described later in this section.

The *>>pathname* format cause further data capture list output to be appended to the file *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain *wildcards* as described later in this section.

The *objectname,description* format will read (capture) the named object/attribute and output its value to the list output file. If the */S* switch is present in the job string, then the output is formatted in a form that is suitable for subsequent use as a data stuff **.PDF** file. The *description* is any arbitrary text, presumably used to describe the named object in more detail.

The *!pathname* format causes data stuff audit output to be directed to the file *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain wildcards as described in the following sections.

The *!!pathname* causes data stuff audit output to be appended to the file *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain wild cards as described later in this section.

The *object=value* format is used to write (stuff) the named object attribute with the new *value*. The acceptable formats for the new value are described in **Table 11-8**. If there is an open data stuff audit file, then a confirming message is output to it showing the object name, the stuffed value and the success/failure of the stuff operation. If the stuff is successful, then only the object name and value are output to the audit file. If the operation fails, then the line in the audit file also has a trailing error message that explains the error.

The **.PDF** file can contain as many lines as desired. There is no limit to the number of times that the list output may be redirected. The pathname for list/audit output is verified before a new list output file is opened or appended. This means that **DCS** checks for the existence of each sub-directory in the pathname. If a sub-directory does not exist, then it is created automatically by **DCS**.

CAUTION

You must be careful to use the >, >>, ! and !! formats correctly, since they may overwrite existing files!

The >, >>, ! and !! formats allow the percent (%) character to appear in the pathname. If it is used, the % character must be followed by a single letter code which indicates one of the time and date values shown in **Table 11-9**. You can specify any combination of these time and date codes in conjunction with normal pathname characters to form unique time/date-based pathnames for **DCS** files. This allows the generation of time-ordered files by **DCS**. Since **DCS** automatically creates sub-directories, you may freely use these % codes in sub-directory names, if desired.

% CODE	TIME UNIT	DIGITS GENERATED
N	month	01-12
D	day of month	01-31
C	year of century	00-99
Y	year	e.g., 1991
H	hour	00-23
M	minute	00-59
W	week of year	01-52
K	day of week	1-7, 1=SUN
J	day of year	001-366

Table 11-9 Wildcards Used in >, >>, ! and !! Pathnames

For example, the pathname

>LOG\%W%K%H%M.PRN

might be used as the name of a data capture list file which was run every day, perhaps several times a day. The resulting files would show the week of the year (**%W**), day of the week (**%K**), and time (**%H%M**) as the file name, e.g., **14021335.PRN**.

11.33.5 The **UPLOAD / DOWNLOAD FILE** Job

The upload/download file (**UDL**) job is used to transfer files between a SAGE^{MAX} and a network device such as another SAGE^{MAX}, a STAR peer or an XANP or PHP device.

Downloading files from a SAGE^{MAX} to a network device may occur between any existing SAGE^{MAX} **drive:\file.ext** and any valid network device file. The download service is used to open a (source) file on the SAGE^{MAX}, create a (destination) file on the network device, and write each record from the source file to the destination file until all source file records have been read and downloaded.

Uploading files from a network device to a SAGE^{MAX} may occur between any existing network device file and any valid (DOS) SAGE^{MAX} file. The upload service is used to open a network device (source) file, create a SAGE^{MAX} (destination) file, and write each record from the source file to the destination file until all source file records have been uploaded and written.

The syntax of the upload/download file (UDL) job is illustrated in **Figure 11-14** with sample SPL programming examples.

JOB 0, "UDL *t port peername delimiter remotepath localpath*", *x,x,x...x,x,x*

Used to request a file upload/download job "as soon as possible." UDL job statement arguments have the following meanings:

- t*** - type of file service, where **U** is upload and **D** is download.
- port*** - the SAGE port number to which the network device is connected.
- peername***-represents the SAGE Ethernet peername or peer unit number for the network device.
- delimiter***:-, / or \ character
- remotepath***-either a standard DOS pathname if the network device is a SAGE peer or **drive:file.ext** for other peers.
- localpath***-the local DOS pathname of the file in **drive:\path\file.ext** format.
- x,x,x...x,x,x***-expressions whose values are inserted into the *jobstring* text in place of format specifiers when PEX submits the job to the SAGE job scheduler.

```

;-----
; This program illustrates a simple uploader and downloader. This program allows you to submit,
; on demand, a file upload request followed by a file download request. The upload/download is
; from/to a peer on the Ethernet (e.g., EtherView or other SAGEs). This simplifies the file
; upload/download process and allows it to be done from a host such as EtherView.
;
;
; Attributes: RQ<>0 requests an upload/download
;-----
          ATTR  RQ,0FFh

START:IF (;RQ <> 0) THEN UL_DL
          SWAIT 1
          GOTO START

UL_DL:
JOB 0, "UDL U 14 SAGE Peer #12/C:\LOG\SOMEFILE.LOG C:\LOG\LOG1.LOG"
JOB 0, "UDL D 14 EtherView Host Peer #1/C:\DATA\SOMEFILE.LOG C:\LOG\LOG1.LOG"
;RQ = 0
GOTO START

```

Figure 1-14 The JOB "UDL" Statement

The *t* refers to the function type and is either a **U** for upload or a **D** for download. *Port* refers to the SAGE^{MAX} port number to which the network is connected. *Peername* is the SAGE^{MAX} Ethernet peer name or peer unit number for the network device. *Delimiter* is either a front slash (/), a colon (:), or a backslash (\) character. *Remotepath* is either a DOS pathname if the network device is a SAGE^{MAX} peer, or **drive:file.ext** for other devices. *Localpath* is the local DOS pathname of the file in the format **drive:\path\file.ext**.

11.33.6 The EXPORT DATABASE FILE Job

The export database file (**EXPORT**) job is used to export name bindings files (**.NBF**) by reading and translating SAGE^{MAX}-resident binary object files (**.BOB**). Name bindings files are editable ASCII text files that contain text representations of the objects which exist in a binary form within SAGE^{MAX} **.BOB** files. The **EXPORT** job can create the new **.NBF** file on any valid DOS path. **EXPORT** reads **.BOB** files from the **\CFG** directory. **Figure 11-15** illustrates the syntax of the **EXPORT** job statement and shows sample SPL program examples.

JOB 0, "EXPORT nbfpath /switch /switch ... /switch", x,x,x...x,x,x

Used to request an export name bindings files job "as soon as possible." EXPORT job statement arguments have the following meanings:

- nbfpath**- represents the full .NBF pathname which may include an extension, but will be ignored and .NBF will be added.
- /switch** - series of optional switches that represent the type of .BOB files to search for in the \CFG directory. If no switches are listed, then *all* .BOB files are searched.
- x,x,x...x,x,x**-expressions whose values are inserted into the *jobstring* text in place of format specifiers when PEX submits the job to the SAGE job scheduler.

```

;-----
; This program allows you to submit, on demand, a database export request. This may be the
; entire database or the points, variables, programs or globals individually. This simplifies the
; export process and allows it to be done from a host such as EtherView.
;
; Attributes:RQ=0do nothing
;
;           =1 entire database
;           =2 points only
;           =3 variables only
;           =4 programs only
;           =5 globals only
;-----
                ATTR RQ,0FEh

START:  ;RQ = 0
WAITRQ:ON [:RQ] GOTO NONE, ALL, POINTS, VARS, PROGS, GLOBS
NONE:   SWAIT 1
                GOTO WAITRQ

;           Export entire database
ALL:    JOB 0,"EXPORT \BACKUP\DATABASE.NBF"
                GOTO START

;           Export points only
POINTS:JOB 0,"EXPORT \BACKUP\DATABASE.NBF -PT"
                GOTO START

;           Export variables only
VARS:   JOB 0,"EXPORT \BACKUP\DATABASE.NBF -VR"
                GOTO START

;           Export programs only
PROGS:  JOB 0,"EXPORT \BACKUP\DATABASE.NBF -PG"
                GOTO START

;           Export globals only
GLOBS:  JOB 0,"EXPORT \BACKUP\DATABASE.NBF -GL"
                GOTO START

```

Figure 1-15 The JOB "EXPORT" Statement

The *nbfp* path is a full pathname which may include an extension, however the extension is ignored and **.NBF** is added. If no switches are specified, then the SAGE^{MAX} will search in the **\CFG** directory for all of the **.BOB** files understood by **EXPORT**. If switches are specified, SAGE^{MAX} will search for only those **.BOB** files specified. **EXPORT** recognizes the following switches:

- PT points
- VR variables
- PG programs
- GL globals

11.34 The CALL Statement

The **CALL** statement is used to execute a PLB from within another PLB. The **CALL**ed PLB may be a block of logic that is shared among several PLBs and/or may be so infrequently used that it is not required that it be RAM-resident all the time.

PLB names are file fragments as discussed in **Chapter 11.1.2: The Program Logic Block (PLB)**. If a PLB name begins with a digit (**0-9**), it must be enclosed in square brackets.

CALLing a PLB from a program causes the **CALL**er's program counter to be saved. The counter is set to execute the first statement of the **CALL**ed PLB. If the PLB is already loaded into memory, then an *in-use* count for that PLB is increased by one. If the PLB is not in memory, then it will be loaded and its *in-use* count is set to 1.

If the **STICK** argument is used in the **CALL**, then the in-use count is set to FFFFh, preventing the PLB from being unloaded later.

Arguments are passed to the PLB by way of the program registers and user-defined attributes. PLBs which are loaded by way of **CALL** statements ignore the **CALL**ed PLB's attribute declarations because the attribute list is only created when the program is initially loaded. In other words, **CALL**ed PLBs inherit the **CALL**ing program's registers and attributes.

NOTE

*CALL*ed PLBs must have any common attributes defined in the *CALL*ing program. Figure 11-73 illustrates the syntax of the *CALL* statement and shows its use in a sample SPL program.

The code below illustrates the use of the **CALL** statement in a sample SPL program:

```
L1:      D=DAYOFYEAR
L2:      SWAIT 5
         CALL PID,STICK
         [MX_FE01;CV]=0
         IF DAYOFYEAR=D THEN L2
         CALL DAILY_REPORT
         GOTO L1
```

The **CALL**ed PLB must execute a **RETURN** at the end of its execution. When the **RETURN** is executed, the saved program counter of the **CALL**er is restored, and execution begins at the statement following the **CALL**. Once the program **RETURN**s, the **CALL**ed PLB's in-use count is decreased by one, unless it is FFFFh. If the result is zero, then the **CALL**ed PLB is released from memory. If a main program executes a **RETURN** statement while not in an internal subroutine, the program is completely unloaded and its PRB and attribute block are released.

11.35 *The GOSUB Statement*

The **GOSUB** statement is used to call a subroutine *in the current PLB*. A **RETURN** statement is used to terminate the internal subroutine and return execution control to the statement directly following the **GOSUB**. The subroutine name is actually a label for which all the naming conventions apply.

The code below illustrates the syntax of the **GOSUB** statement and shows its use in a sample SPL program segment:

```

                ATTR AR,OFAH
                A=65
Readit:        D=&DuctDiam(A-1)
                GOSUB AreaCalc
                &DuctArea(A-1) = ;AR
                LOOP A, Readit
                :
AreaCalc:     ;AR = PI*(D*D)/4
                RETURN

```

11.36 *The RETURN Statement*

The **RETURN** statement is used in conjunction with the **CALL** and **GOSUB** statements. Refer to **Chapter 11.35: The GOSUB Statement**.

11.37 *The ACTIVATE Statement*

The **ACTIVATE** statement is used to load an inactive program into memory (if necessary), and begin executing its logic. This statement will not restart a program that is already running, but will restart a program that has been aborted. If a program has been stopped, **ACTIVATE** will start the program at that point. The program to be activated is a database object, i.e., a program name as described in **Chapter 11.1.1: Program Names**.

The code below illustrates the use of the **ACTIVATE** statement in an SPL program:

```

                IF (TIME >= 5:00) THEN Wmup
                ACTIVATE Coast
                GOTO End

Wmup:         ACTIVATE Warmup1
End:          STOP

```

11.38 *The DEACTIVATE Statement*

The **DEACTIVATE** statement allows you to remove a program from memory (RAM). This may free memory space for other programs. The program to be **DEACTIVATED** is a database object, i.e., a program name as described in **Chapter 11.1.1: Program Names**.

The code below illustrates the use of the **DEACTIVATE** statement in a simple SPL program segment:

```
ATTR SD,OFeh

L1:      If ;SD=0 Then L1
         DEACTIVATE OSS1
         DEACTIVATE OSS2
         DEACTIVATE OSS3
         STOP
```

11.39 *The RESTART Statement*

The **RESTART** statement is a program control command that is used to restart program execution from the beginning, as if it had just been loaded. **RESTART** can also activate a deactivated program and start it from the beginning. If a program is unloaded, **RESTART** causes it to be loaded then restarted.

The **RESTART** command is also used to start a program that has been suspended by a **STOP** statement. The program that is **RESTARTed** is a database object, i.e., a program name as described in **Chapter 11.1.1: Program Names**.

The code below illustrates the syntax of the **RESTART** statement and shows a simple SPL programming example:

```
LO:      WAIT (DAY==1)
         CALL ENDMONTH
         JOB 0,"RPT kwtemp,kwvals,kw"
         RESTART KW_CALC
         WAIT (DAY<>1)
         GOTO LO
```

11.40 *The STOP Statement*

The **STOP** statement is used to halt program execution, maintaining it in a suspended state. If a program name is not included with the **STOP** statement, the current program is halted. Otherwise, the specified program is halted. To resume program execution from a halted state, you must use the **RESTART** or **ACTIVATE** statements.

The code below illustrates the syntax of the **STOP** statement and shows a sample SPL programming example:

```

                IF (TIME>=5:00) THEN Wmup
                ACTIVATE Coast
                GOTO End

Wmup:          STOP SETBACK
                ACTIVATE Warmup1

End:           STOP

```

11.41 The UNLOAD Statement

The **UNLOAD** statement is used to remove this program from memory (RAM). This frees memory space for other programs.

If the program's PLB is *sticky*, it is not unloaded, but the program is put in the unload state.

The code below illustrates the syntax of the **UNLOAD** statement and shows a sample SPL programming example.

```

                IF (TIME>=5:00) THEN Wmup
                ACTIVATE Coast
                GOTO End

Wmup:          ACTIVATE Warmup1
End:           UNLOAD

```

11.42 The STARTTREND and STOPTREND Statements

The **STARTTREND** and **STOPTREND** statements are used to start or stop data collection for the specified trend. The trend must be created prior to the execution of the **STARTTREND** or **STOPTREND** statements.

The *trendname* argument represents a trend file fragment that can be up to 17 characters long. All trends are file images that are contained in the directory **C:\TREND** and have the file extension **.TRN**. The complete pathname is **C:\TREND\trendname.TRN**.

The code below illustrates the use of the **STARTTREND** and **STOPTREND** statements and shows a sample SPL programming example:

```

LO:           WAIT (DAY==1)
                MWAIT 30
                STOPTREND GET_KW
                JOB 0,"RPT KWTEMP,C:\TREND\GET_KW.TRN,KW"
                JOB 0,"SPOOL 13,C:\RPT\KW.RPT"
                STARTTREND GET_KW
                WAIT (DAY<>1)
                GOTO LO

```

11.43 *The ERRORABORT Statement*

The **ERRORABORT** statement is an error control statement that causes the program executor to abort the program when any trappable or non-trappable error is detected. (See also **Chapter 11.44: The ERRORWAIT Statement** and **Chapter 11.45: The ONERROR Statement**.)

There can be multiple **ERRORABORT** and **ERRORWAIT** statements within a program. This allows the aborting of errors to be turned on and off. Unless an **ERRORWAIT** statement is included in a program, the **ERRORABORT** statement is in effect.

All errors that are not trappable (e.g., no such object name, invalid operation, etc.) will always cause the program to be aborted.

The code below illustrates the syntax for the **ERRORABORT** statement and shows its use in a simple SPL program segment:

```
ERRORABORT
;CV = [PT]Zone Temp;ZT]
Print 13,0, "We got it on the first try."
:
```

11.44 *The ERRORWAIT Statement*

The **ERRORWAIT** statement is an error control statement that allows the programmer to specify what PEX should do when it encounters a trappable error such as a timeout, CRC or checksum error, a NAK response, a data rejection response, a temporary blocked state, a dialer busy state or a failed-to-connect error. If the **ERRORWAIT** statement is included in a program and PEX detects a trappable error, then the statement that caused the trappable error is re-executed forever until the error condition no longer exists. (See also **Chapter 11.43: The ERRORABORT Statement**, **Chapter 11.45: The ONERROR Statement** and **Appendix B**.)

There can be multiple **ERRORWAIT** and **ERRORABORT** statements within a program. This allows the aborting of errors and error waiting to be staggered throughout the program. Unless an **ERRORWAIT** statement is included in a program, the **ERRORABORT** statement is in effect.

The code below illustrates the syntax of the **ERRORWAIT** statement and shows it being used in an SPL program segment:

```
ERRORWAIT
Print 13,0, "Wait till we get a good value."
;CV = [PT]Zone Temp;ZT]
Print 13,0, "We got a good value."
:
```


11.45 The ONERROR Statement

The **ONERROR** statement identifies a label to which PEX transfers control whenever it detects a *trappable* error (see **Appendix B**). The **ONERROR** statement is in effect only for the statement that precedes it. (See also **Section 11.43: The ERRORABORT Statement** and **Section 11.44: The ERRORWAIT Statement**.) When an error is detected, the error code is placed in the program's **\$E** control attribute by PEX. **ONERROR** statements take precedence over **ERRORWAIT** statements. The **\$E** program control attribute should be reset to zero before leaving the error code handler.

The code below illustrates the syntax of the **ONERROR** statement and shows an SPL programming example.

```

Getit:      ;$E = 0
           A = ZONE_TEMP;CV
           ONERROR Err
L1:         B=A+10
           :

Err:        IF (;$E<>5) THEN END
           A=72.0
           GOTO L1

End:        STOP

```

NOTE

*The **ONERROR** statement can only be used with trappable errors such as a timeout, a CRC or checksum error, NAK responses, data rejection, temporarily blocked states, dialer busy states and failed to connect errors. Any other program execution errors cause the program to abort.*

11.46 The SECTION Statement

The **SECTION** statement is a debugging statement that stores the *number* argument in the **\$S** program control attribute of the program. This command can be placed strategically at multiple locations in the program to be debugged. By using unique *numbers* in the statements, you can track the progress of the program through various logical sections by monitoring the **\$S** program control attribute.

The code below illustrates the syntax of the **SECTION** statement and shows an SPL programming example:

```

           SECTION 1
           A = REF (0)
L1:        SECTION 2
           B = B + REF (A-1)
           LOOP L1
           SECTION 3
           :

```

11.47 *The NOP Statement*

The **NOP** statement is used for low-level debugging and is normally not be used in SPL programs. The function of the **NOP**(or No OPeration) statement is to use up time and occupy program space.

11.48 *Compiler Directives*

Compiler directives are non-executable control statements recognized by the SPL compiler. These directives are used to control the generation and format of SPL compiler listings and PLBs. The various compiler directives are listed below:

- #NOLIST
- #PAGE *length,width*
- #TITLE "*titletext*"
- #NOLABELS
- #FIXED
- #FLOAT

Compiler control statements always begin with the pound sign character (#). They may begin in the leftmost column of the source code.

The #NOLIST compiler directive is used to suppress the generation of a compiler list file. Unless otherwise directed by this command, the SPL compiler generates a companion list file *sourcefilename.LST* from the designated SPL program logic source file. The compiler *always* generates a PLB from the source file provided that there are no syntax errors. If the #NOLIST directive is used, it must be the first line of the source file.

The #PAGE compiler directive is used to set the maximum number of characters per line in the listing file to *width* and the maximum number of lines per page to *length*. When the page control statement is used without arguments, a new page is started in the compiler listing by writing a form-feed character to the list file.

The #TITLE compiler directive is used to put the specified *titletext* at the top of each page of the compiler list file in order to help identify the program logic. The text string *titletext* must be enclosed in double quotation marks and can be up to 79 characters long.

The #NOLABELS compiler directive informs the compiler not to generate pseudocode for statement labels in the PLB file. Using this command results in smaller, slightly faster-executing PLBs, but eliminates the ability to visually locate labels in the PLB files during troubleshooting.

The #FIXED compiler directive is a data type command that is used to generate a fixed point data type when the compiler encounters a number with a decimal point (e.g., 1.234). The data type is determined by the number of digits and whether a sign is included. So, 1.234 results in a data type 0F8h and -1.23 results in a data type of 0FBh. Note that numbers that are expressed in exponent form, e.g., 1.2E-2 **always** result in floating point data types. Fixed mode is the default when no data type directive is specified.

The #FLOAT compiler directive is a data type command that is used to generate a floating point data type when the compiler encounters a number with a decimal point (e.g., 1.234). Fixed mode is the default when no data type directive is specified.

11.49 Mixed-mode Arithmetic and Coercion

Mixed-mode arithmetic refers to the SAGE^{MAX} ability to perform arithmetic operations using expressions that have different data types. Although some mixed-mode arithmetic operations are meaningless (e.g., adding a time to a channel map data type), the SAGE^{MAX} allows operations of *similar* types by internally changing data types. This internal transformation is known as *coercion* and is accomplished through a complex series of checks based on the data types of the two terms and the operator between them.

To understand the results of SPL math coercion, you can apply a series of rules that are used when math operations are performed.

11.49.1 General Rules

The following general rules apply for the evaluation of expressions by SPL's Program Executor (PEX).

COERCION RULE #1

The result of any math expression that contains a floating point value is a floating point.

(Ex. 1) **1.23 + 1.2E2 ⇒ 1.2123E2**

In the previous example, the result has a floating point data type because the math expression contains a floating point value.

COERCION RULE#2

The result of any math expression that contains only fixed point values has the most precise data type unless the result is larger (or smaller) than the maximum (or minimum) fixed point magnitude, in which case the result is floating point.

(Ex. 2) **1.2 + 1.23 ⇒ 2.43**

(Ex. 3) **1.2 * 1.23 ⇒ 1.476**

(Ex. 4) **4000000000 * 3 ⇒ 1.2E10**

The most precise data types are used in Examples 2 and 3 above, while Example 4 illustrates the floating point result since it exceeds the largest magnitude for fixed point values.

COERCION RULE #3

Hexadecimal (double words, words and bytes), bit map and channel map data types that are mixed in math expressions with fixed point or floating point data types are considered unsigned integers.

(Ex. 5) **10 + 1011B ⇒ 21**

(Ex. 6) **1.2 + 64H ⇒ 101.2**

Example 5 illustrates a result with an unsigned integer data type. Example 6 illustrates a result with a fixed data type.

COERCION RULE #4

The result of any math expression that contains a day data type is a day data type modulo 7.

(Ex. 7) **MON + 6 ⇒ SUN**

(Ex. 8) **WED + 14 ⇒ WED**

Example 7 illustrates a result from adding a day data type and an integer. Example 8 illustrates a result from adding a day data type and a larger integer. This illustrates how the value of the day data type *wraps around* after every multiple of 7 (i.e., modulo 7 or MOD 7).

11.49.2 Floating Point Rules

The following rules apply for the evaluation of expressions containing floating point data types.

COERCION RULE #5

The results of expressions containing transcendental functions (i.e., SIN, COS, TAN, ARCTAN, LOG, LN, SQRT) are floating point.

(Ex. 9) **SIN (PI/4) ⇒ .70711**

(Ex. 10) **1 + SIN(PI/4) ⇒ 1.70711**

In the above examples using transcendental functions, the results have floating point data types.

COERCION RULE #6

The MOD operation cannot be performed on floating point values.

COERCION RULE #7

Floating point values cannot be mixed with time or day data types.

11.49.3 Time Rules

The following rules apply for the evaluation of expressions containing long time (HH:MM:SS) and short time (HH:MM) data types.

COERCION RULE #8

Times can be added, subtracted and used in the BETWEEN function, but cannot be multiplied or divided. Further, the result of any expression that contains a long time data type.

(Ex. 11) **12:00 + 1:23 ⇒ 13:23**
(Ex. 12) **12:34:56 + 1:20 ⇒ 13:54:56**

In example 11, the result is a short time data type. In Example 12, the result has a long time data type since one of the expressions contains a long time.

COERCION RULE #9

Only signed and unsigned integers can be added/subtracted to/from times. The integer values have seconds units if they are added/subtracted to/from an HH:MM:SS data type, and have minutes units if they are added/subtracted to/from an HH:MM data type.

(Ex. 13) **12:00 + 90 ⇒ 13:30**
(Ex. 14) **12:20:30 + 90 ⇒ 12:22:00**

In Example 13, an integer is added to a short time, yielding a short time result. In Example 14, an integer is added to a long time, yielding a long time result.

11.49.4 Unary Operation and Function Rules

The following rules apply for the evaluation of expressions containing unary operations and functions.

COERCION RULE #10

The NOT operator is valid only for unsigned integer, bit map, channel map and hexadecimal data types.

COERCION RULE #11

The unary - operator is valid for signed fixed point and floating point data types. It has no effect on unsigned integer, bit map, channel map and hexadecimal data types.

- | | |
|----------|--------------------------------|
| (Ex. 15) | $-12345 \Rightarrow -12345$ |
| (Ex. 16) | $-1.2E2 \Rightarrow -1.2E2$ |
| (Ex. 17) | $-(10101B) \Rightarrow 10101B$ |
| (Ex. 18) | $-1234H \Rightarrow 1234H$ |
| (Ex. 19) | $-12345 \Rightarrow 12345$ |

In Example 15, the value of the signed integer data type is negated. In Example 16, the value of the floating point expression is negated. In The unary - operator has no effect on the binary, hexadecimal or unsigned integer values in Examples 17-19.

COERCION RULE #12

The shift operators (SHL and SHR) are valid for any data type. However, the shift count is restricted to signed and unsigned integer, bit map, channel map and hexadecimal data types.

COERCION RULE #13

*The exponential operation ($x^{**}y$) is valid for fixed point, floating point, bit map, channel map, day and hexadecimal data types. The power to which the value is being (y) raised must be an unsigned integer. The resulting precision is the same as doing the same number of multiply operations as the power minus one.*

- | | |
|----------|-----------------------------------|
| (Ex. 20) | $1.2^{**}3 \Rightarrow 1.728$ |
| (Ex. 21) | $1.2E2^{**}3 \Rightarrow 1.728E6$ |

In Example 20, the result is a fixed point data type. In Example 21, the result is a floating point data type.

COERCION RULE #14

The arguments BETWEEN functions must have a short time or long time data type.

COERCION RULE #15

Logical operations (AND, OR, XOR) are valid for all data types.

CHAPTER 12 - SAGE^{MAX} NETWORKING

This section illustrates common configurations of SAGE^{MAX} networks. These configurations are intended as typical network examples and should not be considered the only possible configurations available for a given application.

For simplicity, the network examples are divided into sections based on SAGE^{MAX} ports and then further divided into sections based on network types and protocols.

12.1 SAGE^{MAX} Ports 1-4

Ports 1-4 on the SAGE^{MAX} are dual trunk EIA-485 ports. EIA-485 is the Electronic Industries Association standard for multidrop or multipoint communications lines. **Figure 12-1** shows the terminal blocks for SAGE^{MAX} ports 1-4. **Table 12-1** shows possible configurations for SAGE^{MAX} ports 1-4.

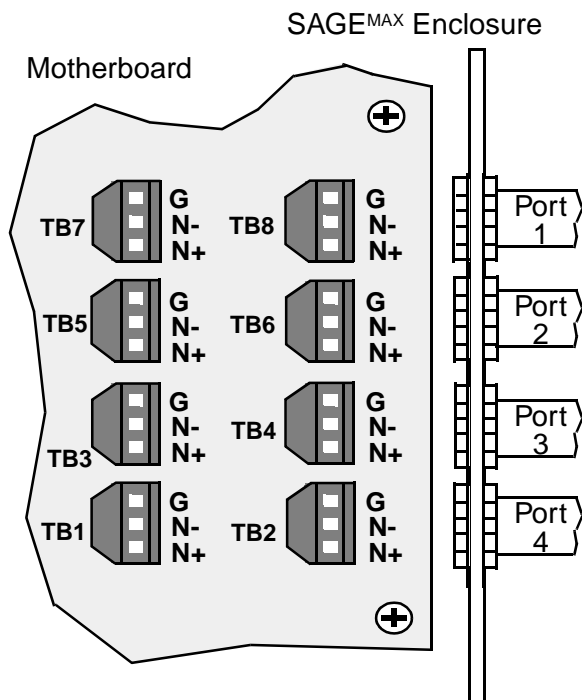


Figure 2-1 Terminal Blocks for SAGE^{MAX} ports 1-4

PUPHOST DRIVER	up to 64 PUP devices networked on Port A up to 64 PUP devices networked on Port B
PHPDCON DRIVER (SLAVE)	up to 31 PHP devices networked on Ports A (TX) and B (RX), connected via 4-wire multidrop to a SPECTRA host using CC/PC
PHPDCON DRIVER (HOST)	Up to 64 PHP devices networked on Ports A (TX) and B (RX)
XANP DRIVER	Up to 32 XANP devices, networked on Port A
PEERNET DRIVER	Up to 32 peers, networked on Port A
VT100 OR DUMB DRIVER	4-wire connection to CC/485, which is connected via EIA-232D cable to VT100 or Dumb Terminal
PRINTER DRIVER	4-wire connection to CC/485, which is connected via EIA-232D cable to Serial Printer

Table 12-1 Hardware Configurations for Ports 1-4

12.1.1 PUPhost Networks

SAGE^{MAX} ports 1-4 can be configured to use Public Unitary Protocol (PUP). The PUPhost driver is used by SAGE^{MAX} ports that are connected to PUP devices. PUPhost networks use 2-wire configurations only. Each dual-trunk SAGE^{MAX} port (i.e., ports 1-4) can accommodate up to 64 PUP units per trunk (128 total units maximum). Each trunk uses twisted shielded pair cable that may extend up to 5000 feet.

Token passing networks can exist on each trunk of a port configured as a PUPhost. The only restriction is that peers on one trunk of the SAGE^{MAX} port cannot pass the token to peers on the other trunk of the SAGE^{MAX} port.

For more information on the PUPhost driver, refer to **Chapter 14: Public Unitary Protocol (PUPhost) Driver**.

12.1.2 XANPhost Networks

SAGE^{MAX} ports 1-4 can be configured to use eXtended Automation Network Protocol (XANP). The XANP driver is used by SAGE^{MAX} ports that are connected to XANP devices. XANP networks use 2-wire configurations only. Each dual-trunk SAGE^{MAX} port (i.e., ports 1-4) can accommodate up to 32 XANP units across the two trunks of each port. Each trunk uses twisted shielded pair cable that may extend up to 5000 feet.

For more information on the XANPhost driver, refer to **Chapter 16: XANPhost Driver**.

12.1.3 STAR Peernet Networks

SAGE^{MAX} ports 1-4 can be configured to use STAR Peernet Protocol. The STAR Peernet driver is used by SAGE^{MAX} ports that are connected to SAGES^{MAX} and STARS. Peernet networks use 2-wire configurations only. Each dual-trunk SAGE^{MAX} port (i.e., ports 1-4) can accommodate up to 32 Peernet units on trunk “A” only. Trunk “A” uses twisted shielded pair cable that may extend up to 5000 feet.

For more information on the STAR Peernet driver, refer to **Chapter 17: STAR Peernet Driver**.

12.1.4 PHPdcon Networks

SAGE^{MAX} ports 1-4 can be configured to use Public Host Protocol (PHP). The PHPdcon driver is used by SAGE^{MAX} ports 1-4 in direct connection applications (i.e., no modem) when the SAGE^{MAX} is to act as a slave to a direct connect host system such as SPECTRA MouseView.

NOTE

PHPdcon, when used with ports 1-4, can only use 4-wire configurations.

For more information on the PHPdcon driver, refer to **Chapter 15: Public Host Protocol (PHP) Drivers**.

12.1.5 PHPHdcon Networks

SAGE^{MAX} ports 1-4 can be configured to use Public Host Protocol. PHPHdcon is used to configure the SAGE^{MAX} for direct connection applications (i.e., no modem) when the SAGE^{MAX} is to act as a host to a direct connect PHP slave network consisting of field panels such as STARS, SAC3s and even slave SAGES^{MAX} (i.e., SAGES^{MAX} connected by way of a PHPdcon network).

NOTE

PHPdcon, when used with ports 1-4, can only use 4-wire configurations.

For more information on the PHPHdcon driver, refer to **Chapter 15: Public Host Protocol (PHP) Drivers**.

12.1.6 Dumb/VT100 Networks

Although the SAGE^{MAX} has two local operator interface ports (ports 7 and 8), SAGE^{MAX} ports 1-4 can be configured to use XON/XOFF protocol for additional local operator interfaces. Such a configuration requires the modification of the EIA-485 network to an EIA-232D network. This can be accomplished using a CC/485 (an EIA-232D to EIA-485 communications converter). Dumb or VT100 driver types are then used by the SAGE^{MAX} in this type of a 4-wire configuration.

12.1.7 Serial Printer Networks

Although the SAGE^{MAX} has two local operator interface ports (ports 7 and 8), either of which can be configured for a serial printer, SAGE^{MAX} ports 1-4 can be configured to support a serial printer. Such a configuration requires the modification of the EIA-485 network to an EIA-232D network. This can be accomplished using a CC/485 (an EIA-232D to EIA-485 communications converter). The printer driver type is then used by the SAGE^{MAX} in this type of a 4-wire configuration.

12.2 SAGE^{MAX} Port 5

SAGE^{MAX} port 5 is the modem port which is typically used for SAGE^{MAX} dial applications. Two types of network configurations are possible using SAGE^{MAX} port 5. **Figure 12-2** illustrates hardware configurations for SAGE^{MAX} port 5.

Due to the unique nature of SmartmodemTM communications, it is unlikely that you would ever use any port other than port 5 (or port 6) for dial applications. SAGE^{MAX}, however, does not prevent you from configuring other ports this way. **Figure 12-2** illustrates SAGE^{MAX} port 5.

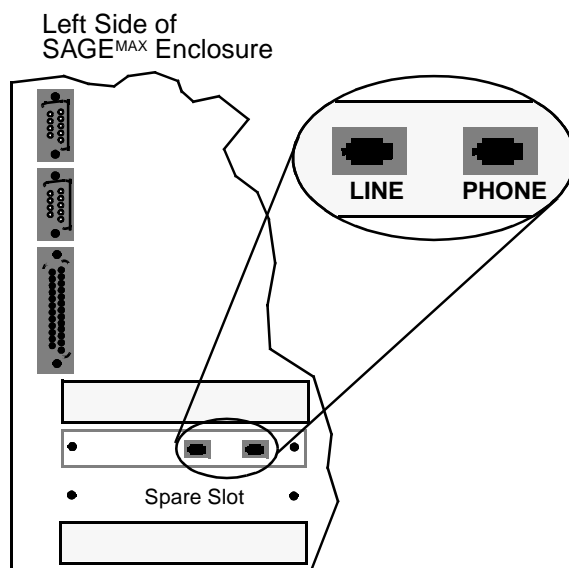


Figure 2-2 SAGE^{MAX} Port 5

12.2.1 PHPdial Networks

SAGE^{MAX} port 5 can be configured to use Public Host Protocol (PHP) for dial-in or dial-out applications (i.e., using the modem). The PHPdial driver type is used to configure the modem port (port 5) when the SAGE^{MAX} acts as a slave to a host system such as SPECTRA.

PHPdial networks typically use the built-in SAGE^{MAX} modem to link the SAGE^{MAX} to a host system or to a terminal, printer or paging system by way of the telephone line connection. Both dial-in (from a host to the slave SAGE^{MAX}) and dial-out (from the slave SAGE^{MAX} to a host) connections can be realized.

For more information on the PHPdial driver, refer to **Chapter 15: Public Host Protocol (PHP) Drivers**.

12.2.2 PHPdial Networks

SAGE^{MAX} port 5 can be configured to use Public Host Protocol (PHP) for dial-in or dial-out applications (i.e., using the modem). The PHPdial driver type is used to configure the modem port (port 5) when the SAGE^{MAX} acts as a host system. With the PHPdial configuration, you can dial out to slave PHP devices (e.g., a SOLOFone, a STAR, an RCU or RCU2, an MCU, a SAC3 or even another SAGE^{MAX}) or have them dial the SAGE^{MAX} host. In addition, a host system can dial in to a PHPdial SAGE^{MAX} and cause it to become a slave. This allows PHP hosts (like other SAGE^{MAX} field panels) to dial in to a PHP host SAGE^{MAX}.

PHPdial networks typically use the built-in SAGE^{MAX} modem to link the SAGE^{MAX} to slave PHP networks by way of a telephone line connection. Both dial-in (from slave PHP devices to the host SAGE^{MAX}) and dial-out (from the host SAGE^{MAX} to slave PHP devices) connections can be realized.

For more information on the PHPdial driver, refer to **Chapter 15: Public Host Protocol (PHP) Drivers**.

12.3 SAGE^{MAX} Port 6

SAGE^{MAX} port 6 is a port that can be configured by adding one of two optional cards to the SAGE^{MAX} hardware platform. An optional internal modem card can be added to the SAGE^{MAX} to increase the dial capabilities of the SAGE^{MAX}. An optional EIA-232D card can be added to the SAGE^{MAX} to supply an additional EIA-232D interface for such operations as leased line modems, a serial printer, or an additional operator interface. In addition, an EIA-232D card can add all the capabilities of ports 1-4 if a CC/485 is used to convert to an EIA-485 network.

NOTE

There is only one spare slot in the SAGE^{MAX} that can accept an optional card. This card can be an additional modem card (port/task 6), an EIA-232D card (port/task 6), or an Ethernet card (port/task 14).

Table 12-2 illustrates network configurations for SAGE^{MAX} port 6 with the optional modem card or EIA-232D card.

PHPDIAL DRIVER (SLAVE)	modem connection to SPECTRA host system
PHPDIAL DRIVER (SLAVE)	modem connection to Terminal or Serial Printer
PHPDIAL DRIVER (HOST)	modem connection to PHP Host
PHPDIAL DRIVER (HOST)	modem connection to PHP Slave

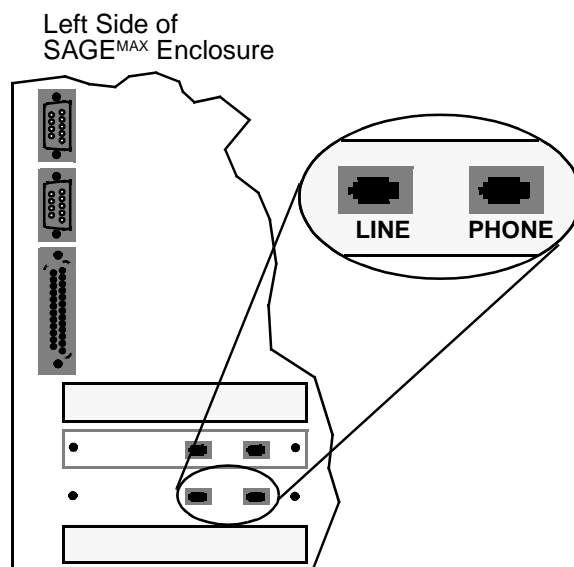
Table 12-2 Hardware Configurations for Port 6

VT100 OR DUMB DRIVER	EIA-232D connection to VT100 or Dumb Terminal
PRINTER DRIVER	EIA-232D connection to Serial Printer
XANP DRIVER	EIA-232D connection to leased line modems, connected to RCU2/G filed panels
PHPHDCON DRIVER	EIA-232D connection to STAR field panel
PHPHDCON DRIVER	EIA-232D connection to RCU2 field panel
PHPHDCON DRIVER	EIA-232D connection to PHP Slave Device
PHPHDCON DRIVER	EIA-232D connection to SPECTRA Host
PHPHDCON DRIVER	EIA-232D connection to PHP Host Device

Table 12-2 Hardware Configurations for Port 6

12.3.1 Additional Internal Modem

An optional internal modem card can be placed in the spare slot in the SAGE^{MAX} to provide increased dial capabilities (see **Figure 12-3**). SAGE^{MAX} port 6 is used by the optional modem card when it is installed in the spare slot of the SAGE^{MAX}.

Figure 2-3 SAGE^{MAX} Port 6 as a Modem

When configured as a modem port (i.e., PHPdial or PHPHdial), port 6 has all the capabilities of SAGE^{MAX} port 5. For more information on the capabilities of port 5, refer to the previous section.

Due to the unique nature of SmartmodemTM communications, it is unlikely that you would ever use any port other than port 6 (or port 5) for dial applications. SAGE^{MAX}, however, does not prevent you from configuring other ports this way.

For more information on the PHPdial and PHPHdial drivers, refer to **Chapter 15: Public Host Protocol (PHP) Drivers**.

12.3.2 Optional EIA-232D Card

An optional EIA-232D card can be placed in the spare slot in the SAGE^{MAX} to provide a full handshake EIA-232D interface for leased line modem support as well as other possible configurations. Refer to **Figure 12-4** and **Table 12-2**. SAGE^{MAX} port 6 is used by the optional EIA-232D card when it is installed in the spare slot of the SAGE^{MAX}.

When the EIA-232D card is connected to leased line modems, a long distance multipoint connection is established for any of various configurations.

Usually the leased line modem would only be used with the XANP driver. When the EIA-232D card is used with port 6, it can also serve as an additional operator port (e.g., VT100 or Dumb) or serial printer port.

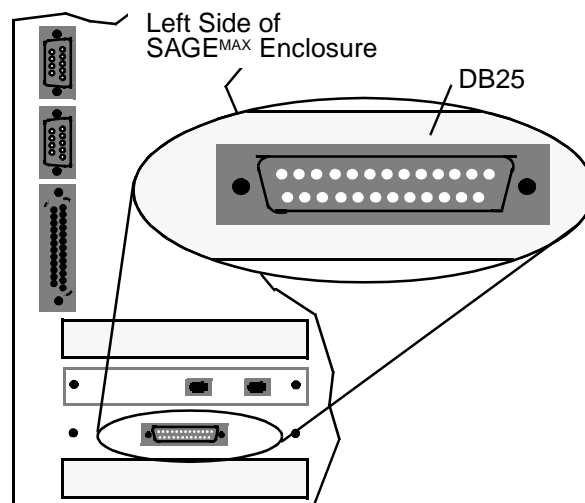


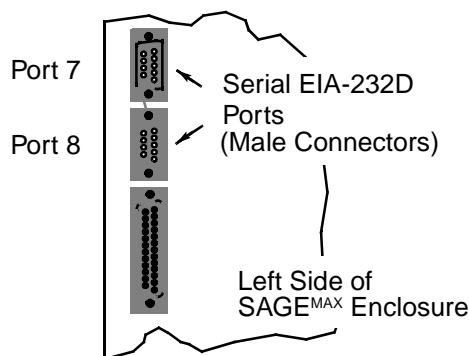
Figure 2-4 SAGE^{MAX} Port 6 as an EIA-232D Card

12.4 SAGE^{MAX} Ports 7 and 8

SAGE^{MAX} ports 7 and 8 are the local operator interface ports (see **Figure 12-5**). These ports can be configured in a variety of ways and can use any of the available direct connect driver types of the SAGE^{MAX} (i.e., no dial driver types). **Table 12-3** illustrates possible configurations for ports 7 and 8.

12.4.1 Dumb/VT100 Terminal and Serial Printer Configuration

Ports 7 and 8 are the primary operator interfaces to the SAGE^{MAX}. These ports can be configured to use XON/XOFF protocol and can support Dumb and VT100 driver types. In addition, these ports have the ability to support a serial printer using the printer driver type.

Figure 2-5 SAGE^{MAX} Ports 7 and 8

VT100 OR DUMB DRIVER	connection to VT100 or Dumb Terminal
PHPDCON OR PHPDCON DRIVER	connection to SPECTRA Host
PRINTER DRIVER	connection to Serial Printer
APPROPRIATE SAGE ^{MAX} DRIVER (PUPHOST, XANPHOST, PEERNET, PHPDCON, PHPDCON)	connection to CC/485, 4-wire connection to EIA-485 Network

Table 12-3 Hardware Configurations for Ports 7 and 8

12.4.2 EIA-485 Configurations

SAGE^{MAX} ports 7 and 8 can support a variety of other configurations with the addition of a CC/485 communications converter. This converts all appropriate signals and converts the EIA-232D connection to an EIA-485 connection. This application gives ports 7 and 8 many of the configuration options available to ports 1-4, but should be used only if the standard EIA-485 network ports (ports 1-4) are already being used.

Using the CC/485, SAGE^{MAX} ports 7 and 8 can be configured to use Public Host Protocol (PHP). The PHPdcon driver is used by SAGE^{MAX} ports 7 and 8 in direct connection applications (i.e., no modem) when the SAGE^{MAX} is to act as a slave to a direct connect host system such as SPECTRA MouseView.

Using the CC/485 as above, SAGE^{MAX} ports 7 and 8 can be configured to use Public Host Protocol in a host application. (Note that PHP can also be used without the CC/485 in an EIA-232D configuration.) PHPdcon is used to configure the SAGE^{MAX} for direct connection applications (i.e., no modem) when the SAGE^{MAX} is to act as a host to a direct connect PHP slave network consisting of field panels such as STARS, SAC3s and even slave SAGES^{MAX} (i.e., SAGES^{MAX} connected by way of a PHPdcon network).

For more information about EIA-485 configurations and applications, refer to **Chapter 12.1: SAGE^{MAX} Ports 1-4.**

12.5 SAGE^{MAX} Port 13

SAGE^{MAX} port 13 is a dedicated parallel printer port (see **Figure 12-6**). Although the use of a parallel printer is optional, port 13 cannot be used for any other function. Any printer which uses the Centronics-style parallel interface can be connected to the SAGE^{MAX} with the appropriate cable.

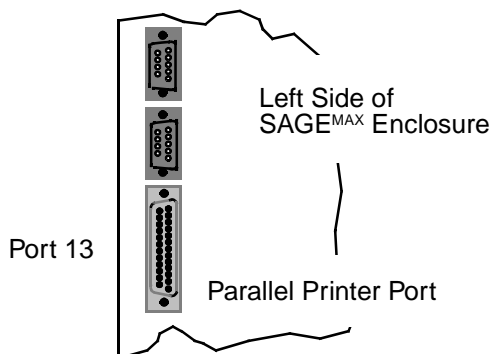


Figure 2-6 SAGE^{MAX} Port 13

12.6 SAGE^{MAX} Port 14

SAGE^{MAX} port 14 is a dedicated Ethernet port on the SAGE^{MAX}. If the optional Ethernet card is installed in the SAGE^{MAX} spare slot, SAGE^{MAX} port 14 is used (see **Figure 12-7**).

NOTE

There is only one spare slot in the SAGE^{MAX} that can accept an optional card. This card can be an additional modem card (port/task 6), an EIA-232D card (port/task 6), or the Ethernet card (port/task 14).

The SAGE^{MAX} can be connected to the Ethernet network using several media. If connecting the SAGE^{MAX} to a thick Ethernet (10Base5), you use the AUI connector of the SAGE^{MAX}. If connecting the SAGE^{MAX} to a thin Ethernet (10Base2), you use the BNC connector of the SAGE^{MAX}. If connecting the SAGE^{MAX} to a Universal Twisted Pair (UTP) Ethernet, you must use a UTP transceiver.

For more information on Ethernet topologies, refer to **Chapter 13: High Speed LANs**.

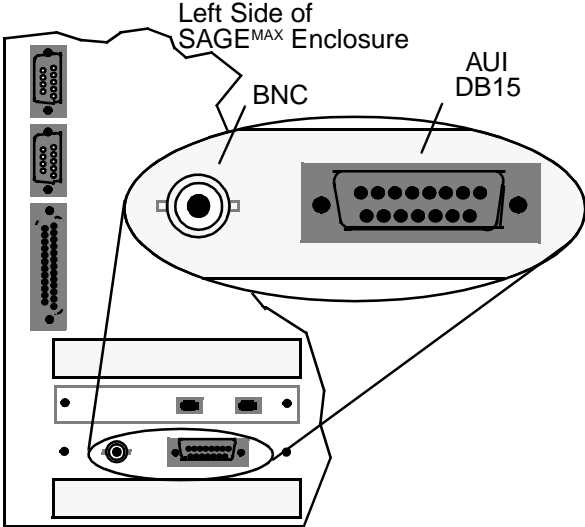


Figure 2-7 SAGE^{MAX} Port 14

CHAPTER 13 - HIGH SPEED LANs

13.1 *Understanding LANs*

LAN is an acronym for Local Area Network-- a communication system that interconnects two or more computers within a relatively limited (local) geographic area such as a single building or small group of buildings.

All of the building automation systems in existence today use some type of LAN architecture. In recent years, the cost of high-performance LAN technology has decreased, making High-Speed LAN (HSLAN) technology available to everyone, including the building automation industry, at a relatively low cost. We can now take advantage of this high-performance, low-cost technology and apply it to areas which have traditionally been accustomed to far more modest networking capabilities.

The three most common office automation LANs today are:

- Ethernet
- ARCNET
- Token Ring

Ethernet is currently the most popular LAN technology. Developed by Xerox Corporation, it transmits information at a rate of 10 megabits per second. Ethernet was originally designed to run on a physical bus topology that supports 1024 nodes. (Bus topologies have nodes attached linearly along a main cable.) New advances enable Ethernet to run over twisted-pair cables configured in a star formation.

Ethernet includes four basic types of wiring options. There is standard Ethernet coaxial cable (sometimes referred to as *Thicknet* or *10Base5*), thin wire Ethernet RG58C/U coaxial cable (sometimes called *Thinnet* or *10Base2*), universal twisted-pair (sometimes called *UTP* or *10BaseT*), and fiber optic cable. Ethernet is the most widely used LAN (estimated at 46% in 1991) and is recognized as an international standard (ISO 8802.3). Ethernet uses a bus or star topology and a collision detection media access protocol.

ARCNET was the first commercially available LAN technology. Developed by Datapoint Corporation, it is based on the use of coaxial cable, but now offers twisted-pair and fiber optic options as well. ARCNET transmits information at a rate of 2.5 megabits per second. While this is nearly 250 times the speed of peernets used in present building automation systems, it is significantly slower than either Ethernet or Token Ring. It also supports fewer nodes (up to 255) than either of its competitors, making it most appropriate for small applications. ARCNET uses a physical star topology or a series of linked stars and a token-passing media access protocol.

CHAPTER 14 - PUP HOST DRIVER

14.1 Features

The PUPhost driver is used by ports that have PUP (Public Unitary Protocol) configurations. PUPhost networks use 2-wire configurations only. Therefore, each dual-trunk SAGE^{MAX} port (i.e., ports 1-4) can accommodate up to 64 PUP units (128 total units maximum). Each trunk uses twisted shielded pair cable that may extend up to 5000 feet.

Token passing networks can exist on each trunk of a port configured as a PUPhost. The only restriction is that peers on one trunk of the SAGE^{MAX} port cannot pass the token to peers on the other trunk of the SAGE^{MAX} port.

Figure 14-1 illustrates PUPhost network constraints for token passing and non-token passing configurations. If token passing is not implemented, up to 128 PUP units can be recognized. If token passing is implemented, up to 128 PUP units can still be recognized, but token passing may not occur from one trunk to the other.

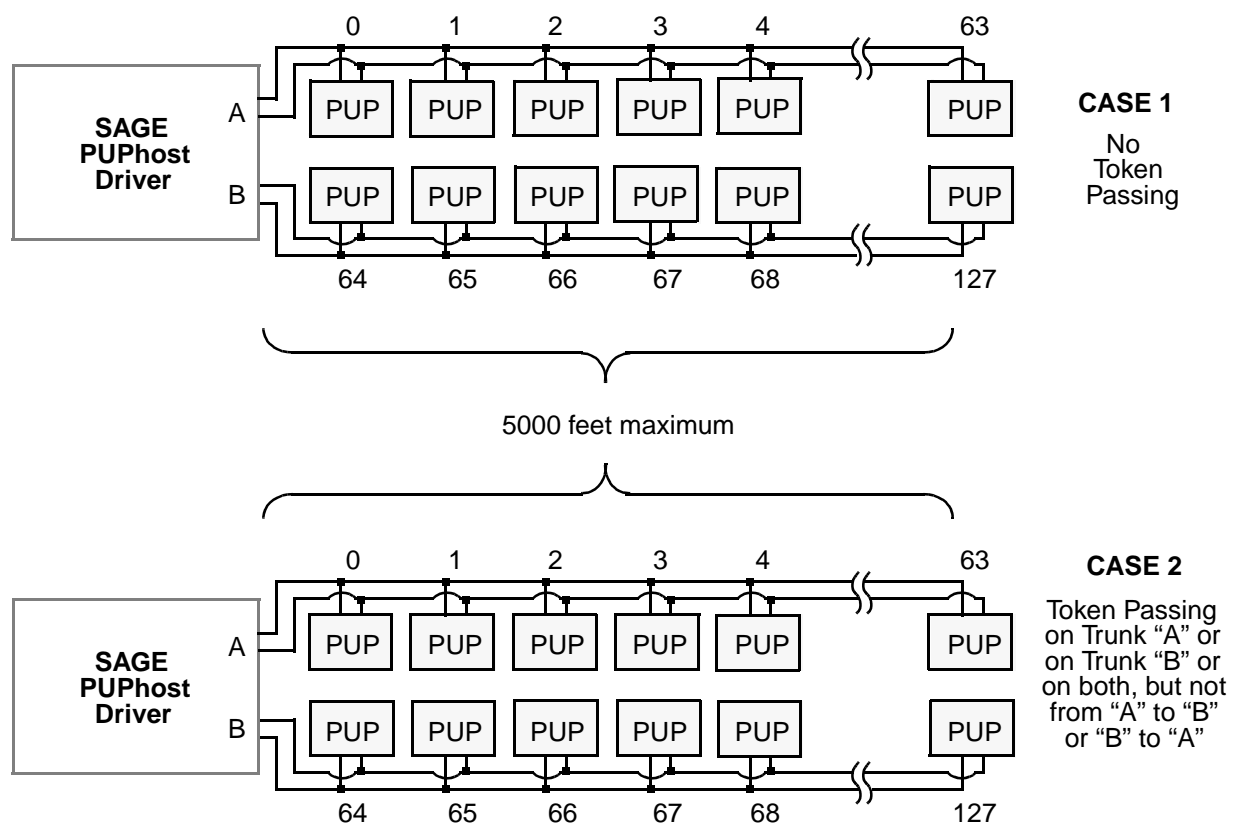


Figure 4-1 PUPhost Constraints for Token Passing and Non-token Passing Networks

When a port is configured as a PUPhost, the port only supports *attribute reads* and *attribute writes*. Other functions that may be directed to the PUPhost port (e.g., broadcast messages, spooled files, virtual terminal mode requests, file transfers, etc.) are ignored by the PUPhost driver.

The SAGE^{MAX} PUPhost driver performs time and date synchronization of network devices every five minutes. This ensures time/date uniformity among the PUP devices and is vital when local PUP schedules are used.

14.2 *Communication Parameters*

The baud rate, parity, number of data bits, number of stop bits and the language used by PUPhost ports are set up by using the Change Port Communications Parameters (C) option of the Ports Status and Setup Submenu.

The baud rate of a PUPhost network on a SAGE^{MAX} serial port can range from 110-38,400 bps. Be sure to select a baud rate that is available to all of the PUP devices on the network. Typically you will use the default PUPhost baud rate of 9600 bps -- lower in environments with large amounts of electrical and/or magnetic *noise* (these environments can cause communications problems), and higher if there is little noise and all the PUP devices support the higher baud rate.

PUPhost networks communicate using Public Unitary Protocol which requires the use of no parity, 8 data bits and 1 stop bit for serial communications.

The default language of PUPhost ports is 001 (English). The range for this communication parameter is 0-255 and is definable in your SAGE^{MAX}.

14.3 *Driver Variables*

Driver configuration variables are used to configure certain operations of the driver. In the case of the PUPhost driver, there are seven driver variables that define PUPhost operations. These variables are:

- PUP Unit Number for this SAGE^{MAX}
- Max Transactions before Token Pass
- Request Interleave
- Alarm Poll Interleave
- Max Tries per Transaction
- Broadcast Time Synch
- Unit #/ Peer / Alarm Poll

PUP Unit Number for this SAGE^{MAX} refers to the PUP network address that is used by this SAGE^{MAX}. The value can range from 0-32767 and defaults to 100.

Max Transactions before Token Pass is a PUPhost driver variable that is only used when the PUP network is configured for token passing. This variable represents the maximum allowable number of PUP network transactions that may occur when the SAGE^{MAX} has the token. SAGE^{MAX} may voluntarily relinquish the token before the maximum number of transactions is reached. This variable defaults to 10, which means that after a maximum of 10 PUP transactions (but maybe fewer) on the PUP token passing network, the SAGE^{MAX} must relinquish the token.

Request Interleave is a PUPhost driver variable that represents the number of internal SAGE^{MAX} requests to read or write PUP attributes per “cycle.”

Alarm Poll Interleave is a PUPhost driver variable that represents the number of times that the SAGE^{MAX} polls for alarms on the PUP network per “cycle.”

Both *Request Interleave* and *Alarm Poll Interleave* should be set according to the type of system configuration of your network. For example, if your system requires a relatively high number of attribute *reads* and/or *writes* over the network, you should use a relatively high number for the *Request Interleave* variable. If, on the other hand, your system requirements are primarily the determination and/or reporting of PUP alarms, your *Alarm Poll Interleave* variable should be set to a higher value.

In token passing configurations, the SAGE^{MAX} monitors the PUP network for alarms even when SAGE^{MAX} does not possess the token. PUP peer alarms that occur when SAGE^{MAX} does not have the token are queued into a 1024-byte buffer which can hold alarms until the SAGE^{MAX} regains the token and reports the PUP alarms. During this time, it is possible for the SAGE^{MAX} to *miss* alarms. This can occur if another peer polls for more alarms than can be queued into the SAGE^{MAX} buffer. In order to minimize the possibility of the loss of PUP alarms in token passing configurations, you should limit the number of times that PUP peers poll for alarms. Typically, you should not have more than two PUP peers polling for alarms. The *Request Interleave* defaults to 9 and the *Alarm Poll Interleave* defaults to 1.

When the sum of the *Request Interleave* and the *Alarm Poll Interleave* equals the *Max Transactions before Token Pass*, then every time the SAGE^{MAX} gets the token it has the potential to perform *r* network requests followed by *p* polls for alarms, where *r* is the *Request Interleave* and *p* is the *Alarm Poll Interleave*. If there are no peers to pass the token to, then *Request Interleave* and *Alarm Poll Interleave* can be thought of as the ratio of importance between internal requests and alarm polling.

If the sum of the *Request Interleave* and the *Alarm Poll Interleave* does not equal the *Max Transactions before Token Pass*, then the SAGE^{MAX} remembers its place within the request/ alarm poll interleaving until it possesses the token again. At that time, it begins where it left off within the request/alarm poll interleaving.

Max Tries per Transaction is a PUPhost driver variable that specifies the number of times that the SAGE^{MAX} will transmit a transaction across the PUP network before the SAGE^{MAX} considers the transaction to have failed. This driver variable defaults to 3, which means that a PUP transaction is repeated three times (without receiving a proper response) before the SAGE^{MAX} considers it a failed transaction.

Broadcast Time Synch is a PUPhost driver variable that specifies whether or not it is the responsibility of the SAGE^{MAX} to broadcast time and date information on a routine basis to PUP devices on the network. If this variable is set to **YES**, time and date information are broadcast over the PUP network every five minutes. If this variable is set to **NO**, the SAGE^{MAX} assumes that some other *host* device is responsible for updating time and date information for the networked devices. This driver variable defaults to **NO**.

Unit #/ Peer / Alarm Poll is a series of 128 driver variables, each of which specifies a PUP unit ID number of a device on the network, a **P** if the device is a peer (i.e., gets passed the token), and an **A** if the device should be polled for alarms.

To maneuver and edit fields within a *Unit #/ Peer / Alarm Poll* variable, you use the left and right arrow keys. Press **RETURN** for the new value to take effect. Press **PF1** to return to the configuration menu.

The *Unit #/ Peer / Alarm Poll* variables are zero-based (i.e., units 0-127) and default to a value of 0 (i.e., no unit number specified, no **A** and no **P**).

All of the PUPhost driver variables are summarized in **Table 14-1**.

VARIABLE	DESCRIPTION (Default)
PUP Unit Number for this SAGE^{MAX}	The PUP network address for this SAGE ^{MAX} from 0-32767 (default=100)
Max Transaction before Token Pass	The maximum number of PUP network transactions that can occur before the SAGE ^{MAX} passes the token to the next PUP device on the network. SAGE ^{MAX} may voluntarily relinquish the token before this number of transactions is reached. (default=10)
Request Interleave	The maximum number of PUP network requests that can occur when the SAGE ^{MAX} gets the token (default=9)
Alarm Poll Interleave	The maximum number of times the SAGE ^{MAX} polls for alarms on the PUP network when the SAGE ^{MAX} gets the token (default=1)
Max Tries per Transaction	The maximum number of times a transaction is transmitted across the PUP network before it is considered a failed transaction (default=3)
Broadcast Time Synch?	If YES, enables SAGE ^{MAX} to broadcast current time at power up and every five minutes thereafter (default=NO)
Unit # / Peer / Alarm Poll	Unit P A (Peer, Alarmpoll) The unit number of another PUP device on the network. This variable also specifies if this unit is a peer (i.e., gets passed the token) and if this unit should be polled for alarms by entering P and/or A in the appropriate fields. (default is blank, i.e., no units, no peers, no alarm polling) Use the up and down arrow keys to scroll through all 128 units. Use the left and right arrow keys to maneuver through the field while editing. Press RETURN for new values to take effect. Press PF1 to return to the configuration menu.

Table 14-1 PUPhost Driver Type Variables

14.4 Dynamic Status of PUP Units

From the Watch Performance Statistics (**W**) option of the Edit PUPhost Driver Variables Submenu, you are presented with a “big picture” of which units are responsive (*) and which units are not (?). **Figure 14-2** shows a sample Dynamic Status Screen for PUP units.

If a PUP peer unit displays “*” beside its unit number, this does not necessarily mean that the unit is responsive. In order for a PUP unit to be marked as unresponsive, it must not respond for *Max Tries per Transaction* tries, and only then is it marked as unresponsive. If the status is checked before the maximum number of tries has occurred, the unit may only appear to be responsive, but may actually be on the verge of timing out.

SAGE MAX	Banner	Line	Text					Mon 23-Sep-96 12:00:00
Opr:				Port:	07			
2588=*	1790=*	405=*	341=*	213=*	3210=*	1260=*	449=*	
1499=*	998=*	1888=*	1199=*	441=*	2044=*	1234=?	3331=?	
4949=?	3123=*	228=*	3308=?					

Figure 4-2 Sample Dynamic Status Screen

14.5 Message Handling

Alarms which are received by the PUPhost during alarm polling are structured by the PUPhost driver into one or more messages which are forwarded to the Alarm Logging task (ALOG).

When the alarm is first detected, the PUPhost driver searches for a point in the SAGE^{MAX} database that corresponds to the unit number and channel of the PUP alarm. If the PUPhost driver cannot find an associated point in the SAGE^{MAX} database, the driver calculates a default alarm class by adding 26 to the PUP class number that was part of the original PUP alarm text. Refer to **Figure 14-3**.

Next, a point name is constructed based on the unit number and the PUP channel of the PUP alarm (e.g., [00002:FE03]). If there is alarm text sent by the PUP unit that is in alarm, this text will be passed along in the alarm message.

If the PUPhost driver can find an associated point in the SAGE^{MAX} database and the point’s class override equals zero, then the driver calculates a default alarm class by adding 26 to the PUP class number that was part of the original PUP alarm text.

If a point name exists, but the class override is non-zero, then it is used as the class for the messages sent to ALOG.

If the point name has associated alarm and return message text fields defined (i.e., they are non-blank), the appropriate field (alarm or return) is passed on as a continuation line. Otherwise, the text associated with the original PUP alarm is passed along to the ALOG.

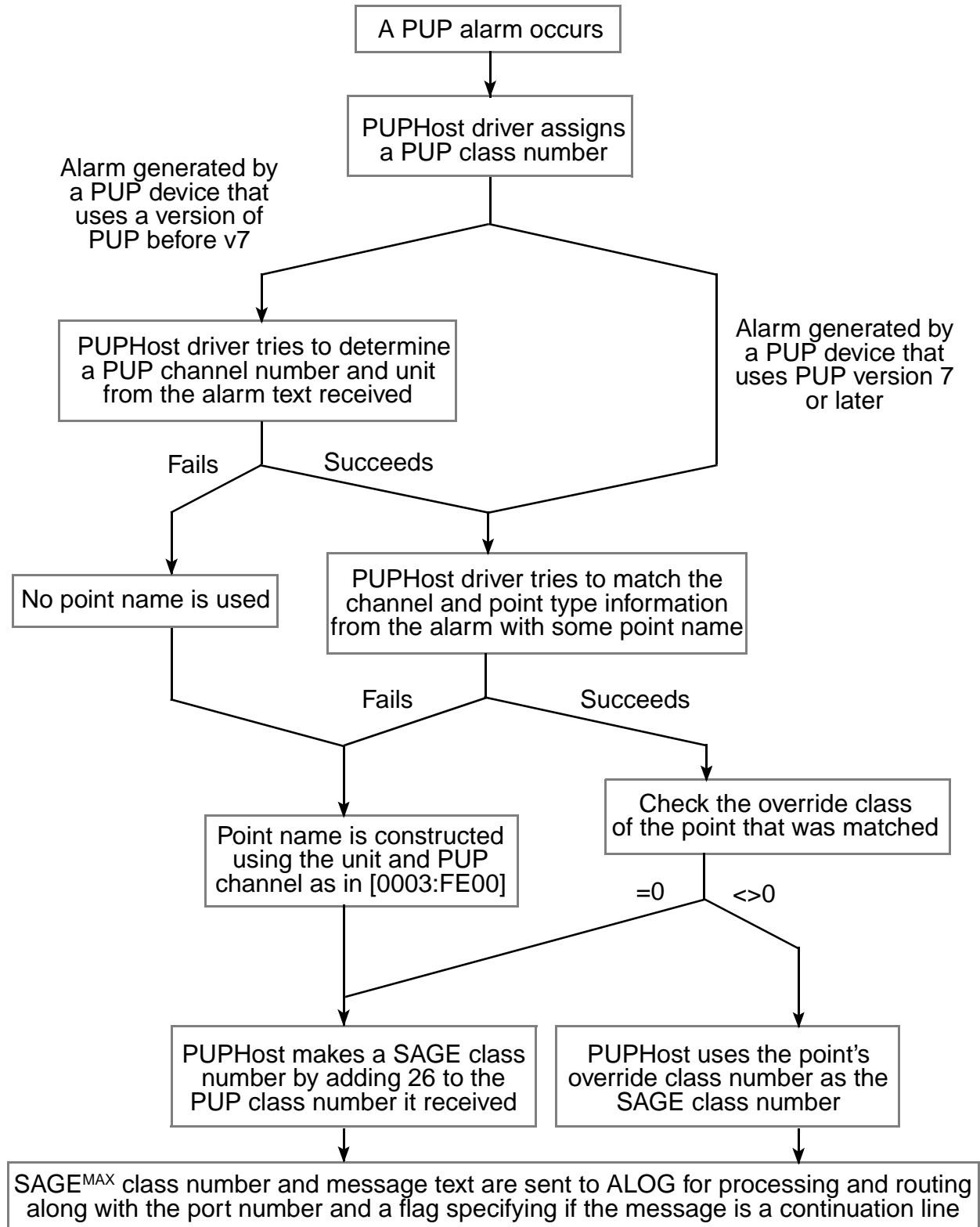


Figure 4-3 PUPHost Alarm Handling Flowchart

When messages are received from PUP devices, they contain a transaction number, a class code and optional message text. Beginning with PUP version 7, the message also contains the PUP channel number that caused the alarm, a standardized alarm type, and a flag indicating an alarm or a return to normal. The class code is remapped into contiguous SAGE^{MAX} classes 26-255. There is no standard structure to the message text.

There are three formats for PUP alarms. The format depends on the type of PUP message which is received by the PUP driver:

- Before PUP v7
- PUP v7 or later, with no point association
- PUP v7 or later, with a point association

The first case occurs when a message is received from a pre-PUP v7 device and no good guess can be made about the source of the alarm. The typical output from ALOG is shown in **Figure 14-4**.

In **Figure 14-4**, **p** represents the port number where the pre-PUP v7 alarm occurred. The field **tttt** represents a sequential transaction number that is assigned by ALOG to every alarm transaction that occurs. The **lllll** field contains the unit number on port **p** that generated the alarm. The SAGE^{MAX} class number (or class name if one is defined) is in the **ccc** field. ALOG includes standard time and date information in the **date** field. Time and date information is included in the prevailing language format. In the default language (English), the format includes the abbreviated form of the day of the week (e.g., **MON, TUE**, etc.), the date when the alarm was received by ALOG (e.g., **13-Feb-96**), and the time when the alarm was received by ALOG (e.g., **17:23:45**).

The actual text message that was sent by the PUP unit is routed by ALOG as a continuation line. The continuation line starts with a hyphen (-) in the first column, and is followed by the class code and the text message.

In some cases, the PUP driver *can* make a good guess about the type of alarm that it has received, based on pattern matching of certain key words within the alarm message text. If a guess can be made, then the PUP driver passes the alarm along as if it were a PUP v7 alarm, and ALOG treats it as such.

```
tttpp/lllll ccc -----date-----
-          ccc message from PUP device (pre v7)
```

Figure 4-4 PUP Message Format - Case 1

The second case occurs when a message is received from a PUP v7 (or later) device, but an association between the port/unit/PUP channel and a particular SAGE^{MAX} point name cannot be made. This happens if no point name is found to match the port, unit and channel of the reported alarm. ALOG formats this type of incoming message as shown in **Figure 14-5**.

In this case, a *fake* point name is created in lieu of the real point name. The fake point name consists of the decimal unit number and hexadecimal channel number (e.g., **[00721:FE00]**). This fake name is included as part of a continuation line, following the standard hyphen (-) in column one and the class number or class name in the **ccc** field as shown in **Figure 14-5**. This first continuation line also contains the type of alarm that was generated by the version 7 PUP device (e.g., High Limit, Low Limit, Fire, etc.) and the status (i.e., Alarm or Return). This information is passed along to the SAGE^{MAX} from the version 7 PUP device.

The third line of a Case 2 PUP message is another continuation line. This continuation line is the message text from the original PUP alarm which is in standard continuation line format (i.e., hyphen in column one and class number or name in the **ccc** field followed by the message text).

```

ttttp/lllll ccc -----date-----
-          ccc [uuuuu:cccc]          alarm type  status
-          ccc message from PUP device (v7, no point assoc.)

```

Figure 4-5 PUP Message Format - Case 2

The third and final possible ALOG format for PUP alarms occurs with PUP version 7 (or later) devices when a SAGE^{MAX} point name can be determined. In this case, the alarm format is the same as in Case 2 for lines 1 and 2. Refer to **Figure 14-6**.

The contents of line 3 (continuation line 2) are determined based on the alarm and return message text fields that are associated with the matching SAGE^{MAX} database point that was found. If the appropriate message text field (alarm or return) contains text (**Figure 14-6** case 3a), this text is used as the text message in continuation line 2. If the appropriate message text field (alarm or return) does not contain text (**Figure 14-6** case 3b), the message text that was passed along from the original PUP alarm is used in continuation line 2.

```

ttttp/lllll ccc -----date-----
-          ccc point name      alarm type  status
-          ccc point alarm/return msg (v7, pt assoc case 3a)

ttttp/lllll ccc -----date-----
-          ccc point name      alarm type  status

```

Figure 4-6 PUP Message Format - Cases 3a & 3b

CHAPTER 15 - PUBLIC HOST PROTOCOL (PHP) DRIVERS

This section explains the functions, setup configurations and uses of the Public Host Protocol (PHP) drivers of the SAGE^{MAX}. Four PHP drivers are supported. They are:

- PHPdcon
- PHPdial
- PHPHdcon
- PHPHdial

PHPdcon is used to configure the SAGE^{MAX} for direct connection applications (i.e., no modem) when the SAGE^{MAX} is to act as a slave to a direct-connect host system such as SPECTRA MouseView.

PHPdial is used to configure the SAGE^{MAX} for dial-in or dial-out applications (i.e., using a modem) when the SAGE^{MAX} acts as a slave to a host system such as SPECTRA MouseView Professional Extension or SPECTRA DUX. This driver type is also used for applications where the SAGE^{MAX} dials out alarms to a remote terminal or printer.

PHPHdcon is used to configure the SAGE^{MAX} for direct connection applications (i.e., no modem) when the SAGE^{MAX} is to act as a host to a direct-connect PHP slave network consisting of field panels such as STARS, SAC3s and even slave SAGES^{MAX} (i.e., SAGES^{MAX} connected by way of a PHPdcon network).

PHPHdial is used to configure the SAGE^{MAX} for dial-in or dial-out applications (i.e., using a modem) when the SAGE^{MAX} acts as a host system. With the *PHPHdial* configuration, you can dial-out to slave PHP devices (e.g., a SOLOFone, a STAR, an RCU/D or RCU2/D, an MCU/D, a SAC3 or even another SAGE^{MAX}) or have them dial the SAGE^{MAX} host.

The following sections explain the features available for each of the four PHP driver types and include explanations of driver variables and communication parameters.

15.1 *PHPdcon -- The PHP Direct Connect (Slave) Driver*

The PHPdcon driver is used by SAGE^{MAX} ports in direct connection applications (i.e., no modem) when the SAGE^{MAX} is to act as a slave to a direct connect host system such as SPECTRA MouseView.

15.1.1 *Features (PHPdcon)*

PHPdcon networks use full-duplex configurations only. Each port of the SAGE^{MAX} can be part of a network of up to 64 PHP slave devices (e.g., STARS, SAC3s and even other slave SAGES^{MAX}) and a PHP host such as SPECTRA. The maximum number of devices may be further limited by the type of media used with the PHPdcon port. EIA-485 ports operate in 4-wire mode and must not exceed 5000 feet in distance. EIA-232D ports connecting to other media types may have different limitations. Refer to **Figure 15-1**.

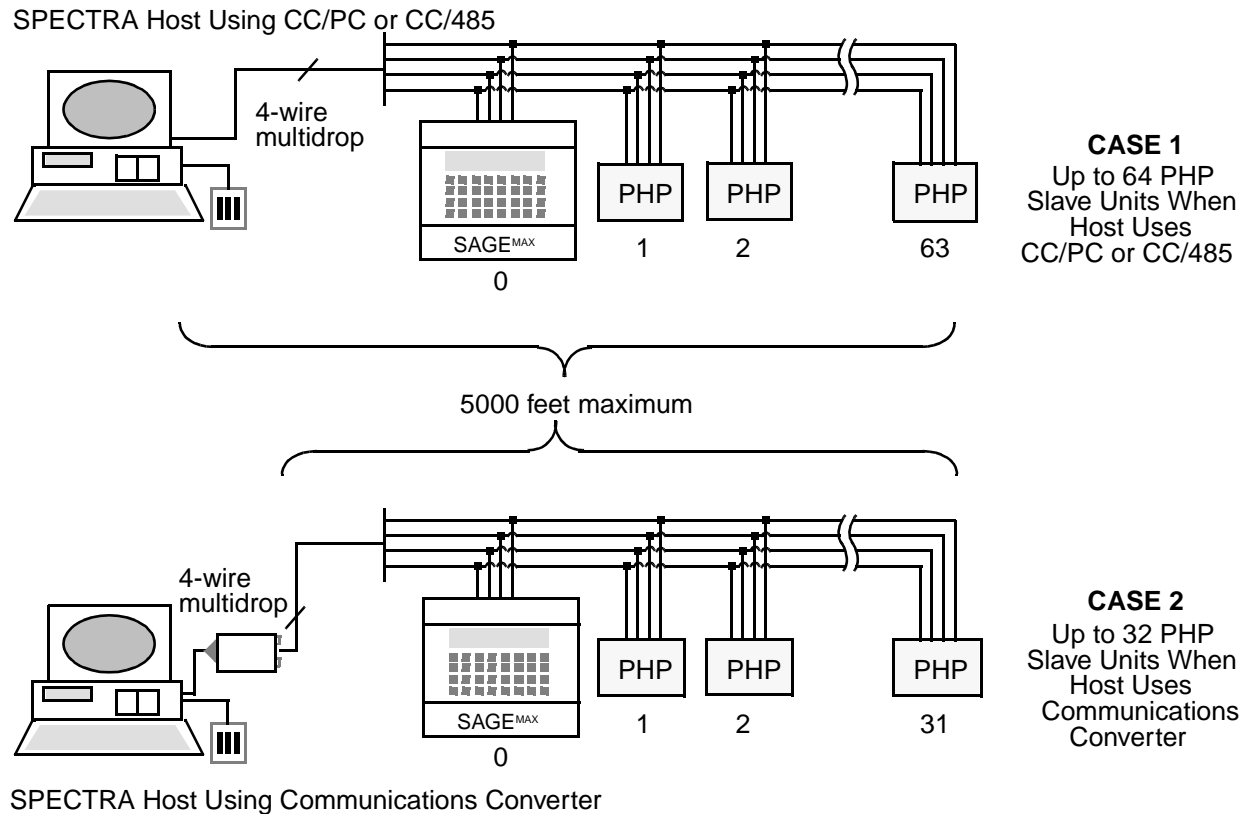


Figure 5-1 Constraints for PHPdcon Networks

NOTE

For EIA-485 networks, up to 64 slave PHP devices can exist on the network only if the host computer uses a CC/PC or CC/485 communications converter. Host computers that are equipped with an old-style communications converter are limited to 32 slave PHP devices.

When a SAGEMAX port is configured as a PHPdcon network, the SAGEMAX supports *attribute reads*, *attribute writes*, *PHP uploading of files*, *PHP downloading of files*, *alarming to a host* (such as SPECTRA MouseView), and *virtual terminal capabilities* to the SAGEMAX from the host.

15.1.2 Communications Parameters (PHPdcon)

The baud rate, parity, number of data bits, number of stop bits and the language used by PHPdcon ports are set up by using the Change Port Communications Parameters (**C**) option of the Ports Status and Setup Submenu.

The baud rate of a PHPdcon network on a SAGEMAX serial port can range from 110-38,400 bps. Be sure to select a baud rate that is available to all of the PHP devices on the network. Typically you will use the default PHPdcon baud rate of 9600 bps--lower in environments with large amounts of electrical and/or magnetic *noise* (these environments can cause

communications problems), and higher if there is little noise and all the PHP devices support the higher baud rate.

PHPdcon networks communicate using Public Host Protocol which requires the use of no parity, 8 data bits and 1 stop bit for serial communications.

The default language of PHPdcon ports is 001 (English). The range for this communications parameter is 0-255.

15.1.3 Driver Variables (PHPdcon)

Driver configuration variables are used to configure certain operations of the driver. In the case of the PHPdcon driver, there are nine driver variables that define PHP slave operations. These variables are:

- PHP Unit Number
- Turnaround Delay (25 ms ticks)
- Terminal Mode Delay (25 ms ticks)
- Auto Hangup Delay (secs)
- Extended Site ID
- Redial Interval (secs)
- Dial Out Tries before Quiet Time
- Object Modify Privileges
- PHP Site Banner Line Text
- PHP Upload/Download Path
- PHP Reboot Password
- Default Operator Name

PHP Unit Number is a PHPdcon driver variable that specifies the unit number for this SAGE^{MAX}. The *PHP Unit Number* can be a number from 48-57 or 62-255. The default value for this attribute is 50.

Turnaround Delay is a PHPdcon driver variable that represents a time delay in increments of 25 ms that is imposed by the SAGE^{MAX} before it responds to a PHP command.

Terminal Mode Delay is a PHPdcon driver variable that specifies a time delay in 25 ms increments that is imposed by the SAGE^{MAX} before it responds to a PHP terminal mode request. The SAGE^{MAX} multiplies the value of this variable by 25 ms to determine the actual delay. This delay gives the host system (e.g., SPECTRA) time to load terminal mode software (e.g., VTERM) before it establishes virtual terminal mode communication.

Auto Hangup Delay is a driver variable that is used by the PHPdial driver and is explained in **Chapter 15.2.3: Driver Variable (PHPdial)**.

Extended Site ID, *Redial Interval*, *Dial Out Tries before Quiet Time*, and *PHP Site Banner Line Text* are driver variables that are used by the PHPdial driver and are explained in **Chapter 15.2.3: Driver Variable (PHPdial)**.

Object Modify Privileges is a PHPdcon driver variable that specifies an object modification bit map. This variable defaults to 00000000B.

PHP Upload/Download Path is a PHPdcon variable that specifies a 53-character (maximum) MS-DOS path prefix for the filename used by the host system for uploading and downloading. This variable provides a mechanism by which *any* file on the SAGE^{MAX} may be transferred to the host, even if the file is on a subdirectory or even a different drive.

For example, there is a trend file located on **D:\SAGE5\TREND2.DAT** of a slave SAGE^{MAX} that you want to upload to a PHP host (e.g., SPECTRA MouseView). The file name you request from the host is **0:TREND2.DAT**. In this example, **0:** represents a drive number (which is part of Public Host Protocol). The drive number is a requirement of PHP and it is ignored by the SAGE^{MAX}. For a successful upload, the *PHP Upload/Download Path* driver variable must be set to **D:\SAGE5**.

PHP Reboot Password is a PHPdcon driver variable that is used by the *PHP reboot* command as an added measure of security. Anytime the *PHP reboot* command is issued, it must include a reboot password as part of the command message. If the password in the command message does not match the *PHP Reboot Password* driver variable, the command is not honored.

Default Operator Name is a PHPdcon driver variable that is a 24-character (maximum) case-insensitive operator name that is used in SAGE^{MAX} *log modification* alarms that are generated when attributes of SAGE^{MAX} points are modified from a host (e.g., SPECTRA). If the attributes are modified after you enter terminal mode, the operator name used to sign on in terminal mode is used in the *log modification* alarms.

Table 15-1 explains the PHPdcon and PHPdial driver variables with their default values. These driver variables are used in dial and direct connect *slave* applications of the SAGE^{MAX}.

15.2 *PHPDial -- The PHP Dialup (Slave) Driver*

PHPdial is used to configure the SAGE^{MAX} for dial-in or dial-out applications (i.e., using a modem) when the SAGE^{MAX} acts as a slave to a host system such as SPECTRA for Windows. This driver type is also used for applications where the SAGE^{MAX} dials out alarms to a remote terminal or printer. Refer to **Figure 15-2**.

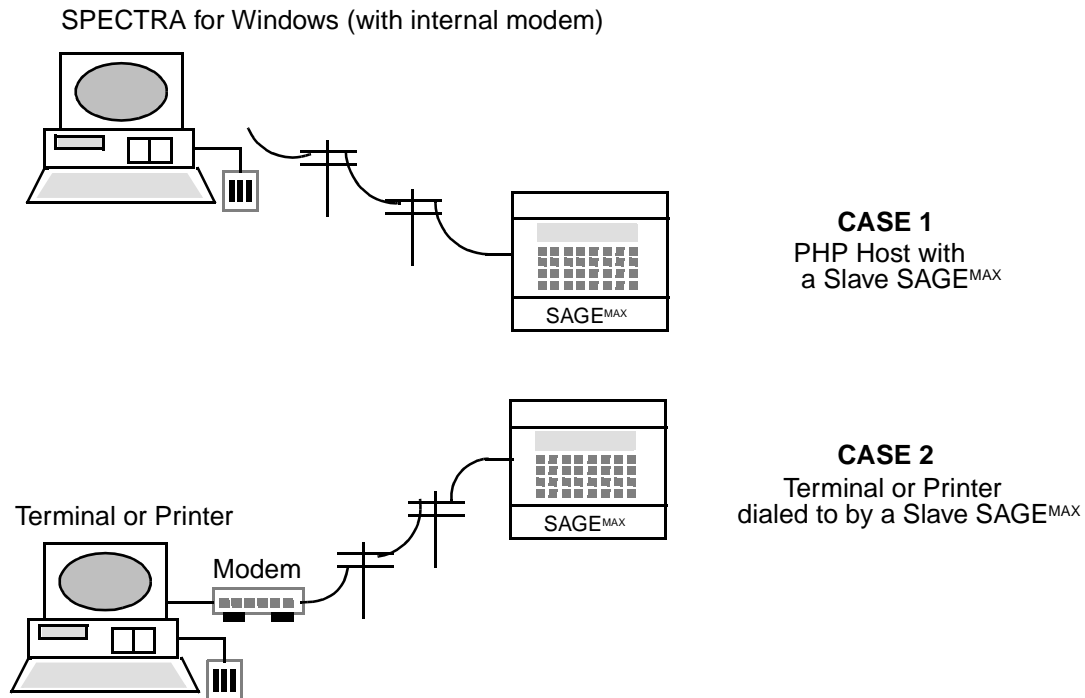


Figure 5-2 Constraints for PHPdial Networks

15.2.1 Features (PHPdial)

PHPdial networks typically use the built-in SAGE^{MAX} modem to link the SAGE^{MAX} to a host system or to a terminal, printer or paging system by way of a telephone line connection. Both dial-in (from a host to the slave SAGE^{MAX}) and dial-out (from the slave SAGE^{MAX} to a host) connections can be realized.

NOTE

Due to the unique nature of SmartmodemTM communications, it is unlikely that you would ever use any port other than ports 5 and 6 for PHPdial applications. SAGE^{MAX}, however, does not prevent you from configuring other ports this way.

When a SAGE^{MAX} port is configured as PHPdial, the SAGE^{MAX} supports *attribute reads*, *attribute write*, *PHP uploading of files* (from the slave SAGE^{MAX} to a host), *PHP downloading of files* (from a host to the slave SAGE^{MAX}), *alarming to the host* (alarm polls from the host) and *virtual terminal capabilities* (from the host to the slave SAGE^{MAX}).

As a PHPdial slave, a SAGE^{MAX} port (typically port 5) can dial out SAGE^{MAX} alarms to a printer, a paging system, a terminal or a PHP host system providing that the alarm class for the SAGE^{MAX} object in question specifies the PHPdial port in the class.

In the case of a PHPdial slave that dials out alarms to a remote printer, once a connection is established, the alarm message is printed. Afterwards, the SAGE^{MAX} remains connected to the printer for a length of time specified by the Auto Hangup Delay driver variable. If no other alarms need to be reported, the SAGE^{MAX} hangs up.

In the case of a PHPdial slave that dials out to a paging system, the slave will cause the paging system to ring only.

A PHPdial slave can be configured to dial out in automatic (*auto*) mode. In this mode, the SAGE^{MAX} waits 5 seconds after a connection has been established to determine if the connection is to a host or to a terminal.

If, during this 5 seconds, the SAGE^{MAX} receives the PHP command ;? (Who Are You?), the SAGE^{MAX} knows that it is connected to a host device and begins responding to the host's alarm polls. Once the alarms have been reported, the connection is maintained until the host hangs up or the host stops polling and the amount of time specified by the Auto Hangup Delay driver variable has elapsed. During this time, the host can enter terminal mode with the SAGE^{MAX}, and perform the same functions as a direct connect host (i.e., *attribute reads*, *attribute writes*, *PHP uploading of files*, *PHP downloading of files*, *alarming to a host* (such as SPECTRA MouseView), *virtual terminal capabilities* to the SAGE^{MAX} from the host) or hang up.

If the SAGE^{MAX} does not receive the PHP command ;? (Who Are You?) within the first 5 seconds after the dial connection has been established, the SAGE^{MAX} assumes that it is connected to a terminal. In this case the SAGE^{MAX} prints its alarms to the terminal. After the alarms have been reported, the connection is maintained to allow an operator at an attended terminal to enter virtual terminal mode with the SAGE^{MAX}. If you type three consecutive carriage returns (<CR>) from the terminal within the period of time specified by the Auto Hangup Delay, you enter terminal mode with the slave SAGE^{MAX}.

A SAGE^{MAX} port that is configured as PHPdial is not limited to dialing out alarms. PHPdial ports also have the ability to spool SAGE^{MAX} files and broadcast messages to the dialer port.

In addition to *dialing out*, a PHP host system or terminal can *dial in* to the slave SAGE^{MAX} by way of the PHPdial port to establish a virtual terminal connection with the SAGE^{MAX}, to perform file upload/download functions and to perform read attribute and write attribute functions.

As in dial-out applications, the SAGE^{MAX} waits 5 seconds after a dial-in connection is established so SAGE^{MAX} can determine who is dialing in to it.

If, during this 5 seconds, the SAGE^{MAX} receives the PHP command ;? (Who Are You?), the SAGE^{MAX} knows that it is connected to a host device and begins responding to the host's alarm polls. Once the alarms have been reported, the connection is maintained for an amount of time specified by the Auto Hangup Delay driver variable. During this time, the host can enter terminal mode with the SAGE^{MAX}, and perform the same functions as a direct connect host (i.e., *attribute reads*, *attribute writes*, *PHP uploading of files*, *PHP downloading of files*, *alarming to a host* (such as SPECTRA), *virtual terminal capabilities* to the SAGE^{MAX} from the host) or hang up.

If the SAGE^{MAX} does not receive the PHP command ;? (Who Are You?) within the first 5 seconds after the dial connection has been established, the SAGE^{MAX} assumes that it is connected to a terminal. The connection is maintained to allow an operator at an attended terminal to enter virtual terminal mode with the SAGE^{MAX}.

15.2.2 Communications Parameters (PHPdial)

The baud rate, parity, number of data bits, number of stop bits and the language used by PHPdial ports are set up by using the Change Port Communications Parameters (C) option of the Ports Status and Setup Submenu.

The baud rate of a PHPdial network on a SAGE^{MAX} serial port can range from 110-38,400 bps, but defaults to 38,400 bps -- the baud rate of the internal modem.

PHPdial networks communicate using Public Host Protocol which requires the use of no parity, 8 data bits and 1 stop bit for serial communications.

The default language of PHPdial ports is 001 (English). The range for this communications parameter is 0-255.

15.2.3 Driver Variables (PHPdial)

Driver configuration variables are used to configure certain operations of the driver. In the case of the PHPdial driver, there are twelve driver variables that define PHP slave operations. These variables are:

- PHP Unit Number
- Turnaround Delay (25 ms ticks)
- Terminal Mode Delay (25 ms ticks)
- Auto Hangup Delay (secs)
- Extended Site ID
- Redial Interval (secs)
- Dial Out Tries before Quiet Time
- Object Modify Privileges
- PHP Site Banner Line Text
- PHP Upload/Download Path
- PHP Reboot Password
- Default Operator Name

PHP Unit Number is a PHPdial driver variable that specifies the unit number for this SAGE^{MAX}. The *PHP Unit Number* can be a number from 48-57 or 62-255. The default value for this attribute is 50.

Turnaround Delay is a PHPdial driver variable that represents a time delay in increments of 25 ms that is imposed by the SAGE^{MAX} before it responds to a PHP command.

Terminal Mode Delay is a PHPdial driver variable that specifies a time delay in 25 ms increments that is imposed by the SAGE^{MAX} before it responds to a PHP terminal mode request. The SAGE^{MAX} multiplies the value of this variable by 25 ms to determine the actual

delay. This delay gives the host system (e.g., SPECTRA MouseView) time to load terminal mode software (e.g., VTERM) before it establishes virtual terminal mode communication.

Auto Hangup Delay is a PHPdial driver variable that specifies the amount of time SAGE^{MAX} waits between network/terminal activity before hanging up and breaking the connection. This variable defaults to 70 seconds.

Extended Site ID is a PHPdial driver variable that specifies an extended host ID number for dial-based host systems. This variable defaults to zero.

Redial Interval is a PHPdial driver variable that specifies an amount of time the SAGE^{MAX} waits between dial-out attempts. This variable defaults to 60 seconds.

Dial Out Tries before Quiet Time is a PHPdial driver variable that represents the number of times the SAGE^{MAX} will dial out alarms before initiating a *quiet time*. This *quiet time* is a window during which no dial-outs occur. This window gives remote operators and hosts the opportunity to dial in to the SAGE^{MAX} (to connect in virtual terminal mode, for example).

Object Modify Privileges is a PHPdial driver variable that specifies an object modification bit map. This variable defaults to 00000000B.

PHP Site Banner Line Text is a PHPdial driver variable that specifies a 50-character maximum ASCII text string. If the SAGE^{MAX} has dialed out to a terminal or printer, this banner line is printed to the device followed by the alarm text. This driver variable is blank by default.

PHP Upload/Download Path is a PHPdial variable that specifies a 53-character (maximum) MS-DOS path prefix for the filename used by the host system for uploading and downloading files over a dialed connection. This variable provides a mechanism by which *any* file on the SAGE^{MAX} may be transferred to the host over the modem, even if the file is on a subdirectory or even a different drive.

For example, there is a trend file located on **D:\SAGE5\TREND2.DAT** of a slave SAGE^{MAX} that you want to upload to a PHP host (e.g., SPECTRA). The file name you request from the host is **0:TREND2.DAT**. In this example, **0:** represents a drive number (which is part of Public Host Protocol). The drive number is a requirement of PHP and it is ignored by the SAGE^{MAX}. For a successful upload, the *PHP Upload/Download Path* driver variable must be set to **D:\SAGE5**.

PHP Reboot Password is a PHPdial driver variable that is used by the PHP *reboot* command as an added measure of security. Anytime the PHP *reboot* command is issued, it must include a reboot password as part of the command message. If the password in the command message does not match the *PHP Reboot Password* driver variable, the command is not honored.

Default Operator Name is a PHPdial driver variable that is a 24-character (maximum) case-insensitive operator name that is used in SAGE^{MAX} *log modification* alarms that are generated when attributes of SAGE^{MAX} points are modified from a remote host (e.g., SPECTRA for Windows). If the attributes are modified after you enter terminal mode, the operator name used to sign on in terminal mode is used in the *log modification* alarms.

Table 15-1 explains the PHPdcon and PHPdial driver variables with their default values. These driver variables are used in dial and direct connect *slave* applications of the SAGE^{MAX}.

VARIABLE	DESCRIPTION (Default)
PHP Unit Number	PHP unit number for this SAGE ^{MAX} from 48-57 or 62-255 (default=48).
Turnaround Delay	The delay imposed by the SAGE ^{MAX} before it responds to a PHP command. The SAGE ^{MAX} multiplies the number you enter by 25 ms (default=0).
Terminal Mode Delay	The delay imposed by the SAGE ^{MAX} before it responds to a PHP terminal mode request. The SAGE ^{MAX} multiplies the number you enter by 25 ms (default=2).
Auto Hangup Delay	For dial-up connections, this is the amount of time the SAGE ^{MAX} will wait between keystrokes before hanging up and breaking the dial-up connection (default=70).
Extended Site ID	An extended ID number (0-65535) for PHP dial-up systems (default=0).
Redial Interval	For dial-up connections, the amount of time the SAGE ^{MAX} waits between dial-out attempts (default=60 seconds).
Dial Out Tries Before Quiet Time	For dial-up connections, the number of times the SAGE ^{MAX} will dial out alarms before initiating a "quiet time", during which no dial-outs occur, so that remote sites have an opportunity to dial in to the SAGE ^{MAX} (default=5).
Object Modify Privileges	An object modification bitmap used by the PHP slave drivers (default=00000000B).
PHP Site Banner Line Text	For dial-up connections, 50-character ASCII text string displayed as banner line text when host connects via virtual terminal mode to this unit (default is blank).
PHP Upload/Download Path	SAGE ^{MAX} pathname where uploaded/downloaded PHP files are stored (default is blank).
PHP Reboot Password	The password which is required for the PHP reboot command (default is SYSTEM).
Default Operator	24-character case-insensitive operator name used in alarm messages when log modifications is enabled (default is PHPSlave).

Table 15-1 PHPdcon and PHPdial Driver Type Variables

15.2.4 Spooling Files to the Dialer (PHPdial)

SAGE^{MAX} ports that are configured as PHPdial ports have the ability to spool SAGE^{MAX} files after dial-out so that they may be printed on a remote terminal or printer.

This is done by making a Spool File for Printing (**S**) Job Request from the Job Scheduler Submenu. The Job Scheduler then prompts you for a port number. At this time you must enter the port number of the PHPdial port (i.e., the modem port -- typically port 5).

The SAGE^{MAX} then prompts you for information that is necessary to spool a requested file to a remote terminal or printer. You must enter the path and file name of the file you wish to print. In addition, you must specify the dial-out baud rate (typically this will be 28.8K) and the actual telephone number to dial.

SAGE^{MAX} also asks if you wish to include the banner line text. This text is a header title which identifies the file being printed. After responding, the SAGE^{MAX} asks if you wish to delete the file after it is spooled. The spool request is now completed. The file you selected will be printed to the location specified by the dial string you entered depending on when you requested the spool to occur (e.g., as soon as possible or scheduled for some later time), and provided that the remote location is a terminal or printer, the modem is available and a connection can be established.

When you spool a file to a PHPdial port, the SAGE^{MAX} dials the number you specify and attempts to establish a connection. If successful, the SAGE^{MAX} waits for 5 seconds to determine if the connection is to a host. If the SAGE^{MAX} connects to a host, the spool request is ignored and removed from the queue. If a connection is made to a terminal or printer, SAGE^{MAX} prints the file to the device then hangs up.

CAUTION

For spool requests made to a dial port, the SAGE^{MAX} must connect to a terminal or printer in order for the file to be printed. If the connection is made to a PHP host device (i.e., the PHP "Who Are You?" command is seen by the SAGE^{MAX}), the file is not spooled and the spool request is removed from the queue.

15.3 PHPHdcon -- The PHP Direct Connect Host Driver

PHPHdcon is used to configure the SAGE^{MAX} for direct connection applications (i.e., no modem) when the SAGE^{MAX} is to act as a host to a direct connect PHP slave network consisting of field panels such as STARS, SAC3s and even slave SAGES^{MAX} (i.e., SAGES^{MAX} connected by way of a PHPdcon network).

15.3.1 Features (PHPHdcon)

PHPHdcon networks use full-duplex configurations only. Each PHPHdcon port on the SAGE^{MAX} allows the SAGE^{MAX} to act as a PHP host to a network of up to 64 slave PHP devices (e.g., STARS, SAC3s and even slave SAGES^{MAX}). Refer to **Figure 15-3**.

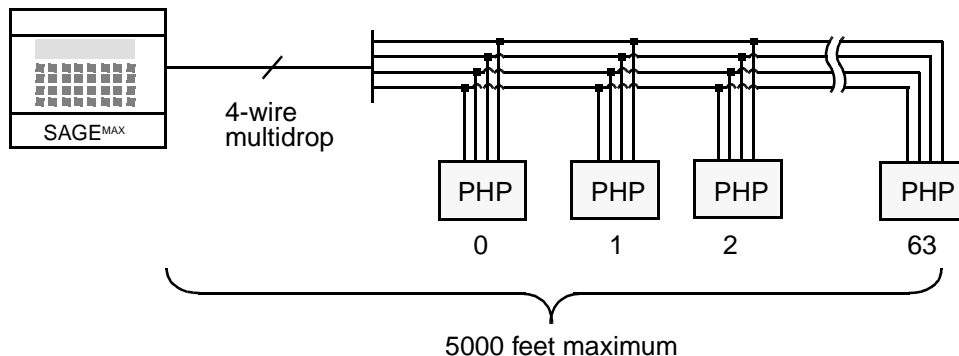


Figure 5-3 Constraints for EIA-485 PPHdcon Networks

When a SAGE^{MAX} port is configured as a PPHdcon network, the SAGE^{MAX} supports *reading and writing attribute values to PHP slave devices, PHP uploading of files, PHP downloading of files, polling for alarms from slave PHP devices and virtual terminal capability* from the SAGE^{MAX} to slave PHP units.

15.3.2 Communications Parameters (PPHdcon)

The baud rate, parity, number of data bits, number of stop bits and the language used by PPHdcon ports are set up by using the Change Port Communications Parameters (C) option of the Ports Status and Setup Submenu.

The baud rate of a PPHdcon network on a SAGE^{MAX} serial port can range from 110-38,400 bps. Be sure to select a baud rate that is available to all of the PHP devices on the network. Typically you will use the default PPHdcon baud rate of 9600 bps -- lower in environments with large amounts of electrical and/or magnetic *noise* (these environments can cause communications problems), and higher if there is little noise and all the PHP devices support the higher baud rate.

PPHdcon networks communicate using Public Host Protocol which requires the use of no parity, 8 data bits and a single stop bit for serial communications.

The default language of PPHdcon ports is 001 (English). The range for this communications parameter is 0-255.

15.3.3 Driver Variables (PPHdcon)

Driver configuration variables are used to configure certain operations of the driver. In the case of the PPHdcon driver, there are nine driver variables that define PHP host operations. These variables are:

- PHP Unit Number
- Extended Site ID
- Terminal Mode Delay (25 ms ticks)
- Auto Hangup Delay (secs)

- Response Timeout (secs)
- Object Modify Privileges
- Default Alarm Class
- PHP Upload/Download Path
- Max Tries per Transaction
- PHP Reboot Password
- Default Operator Name
- Unit # / Peer / Alarm Poll

PHP Unit Number, Turnaround Delay, Terminal Mode Delay, Auto Hangup Delay, Extended Site ID, Redial Interval, Dial Out Tries before Quiet Time, and PHP Site Banner Line Text are driver variables that are used by the PHPDial driver and are explained in **Chapter 15.4.3: Driver Variable (PHPdial)**.

Object Modify Privileges is a PHPDcon driver variable that specifies an object modification bit map. This variable defaults to 00000000B.

PHP Upload/Download Path is a PHPDcon variable that specifies a 53-character (maximum) MS-DOS path prefix for the filename used by the host system for uploading and downloading. This variable provides a mechanism by which *any* file on the SAGEMAX may be transferred to the host, even if the file is on a subdirectory or even a different drive.

For example, there is a trend file located on **D:\SAGE5\TREND2.DAT** of a slave SAGEMAX that you want to upload to a PHP host (e.g., SPECTRA). The file name you request from the host is **0:TREND2.DAT**. In this example, **0:** represents a drive number (which is part of Public Host Protocol). The drive number is a requirement of PHP and it is ignored by the SAGEMAX. For a successful upload, the *PHP Upload/Download Path* driver variable must be set to **D:\SAGE5**.

Max Tries per Transaction is a PHPDcon driver variable that specifies the number of times that the SAGEMAX will transmit a transaction across the PHP network before the SAGEMAX considers the transaction to have failed. This driver variable defaults to 3, which means that a PHP transaction is repeated three times (without receiving a proper response) before the SAGEMAX considers it a failed transaction.

PHP Reboot Password is a PHPDcon driver variable that is used by the PHP *reboot* command as an added measure of security. Anytime the PHP *reboot* command is issued, it must include a reboot password as part of the command message. If the password in the command message does not match the *PHP Reboot Password* driver variable, the command is not honored.

Default Operator Name is a PHPDcon driver variable that is a 24-character (maximum) case-insensitive operator name that is used in SAGEMAX *log modification* alarms that are generated when attributes of SAGEMAX points are modified from a host (e.g., SPECTRA for Windows). If the attributes are modified after you enter terminal mode, the operator name used to sign on in terminal mode is used in the *log modification* alarms.

Unit #/Peer/Alarm Poll is a series of 64 PHPHdcon driver variables, each of which specifies a PHP unit ID number of a device on the network and an **A** if the device should be polled for alarms.

To maneuver and edit fields within a *Unit #/Peer/Alarm Poll* variable, you use the left and right arrow keys. Press **RETURN** for the new value to take effect. Press **PF1** to return to the configuration menu.

The *Unit #/Peer/Alarm Poll* variables default to zero (i.e., no unit number specified, no **A** and no **P**).

Table 15-2 explains the PHPHdcon and PHPHdial driver variables with their default values. These driver variables are used in dial and direct connect *host* applications of the SAGE^{MAX}.

15.3.4 Special Host Naming Conventions

When you create a point on the SAGE^{MAX}, there is a special naming convention that you must follow if the physical point resides on a PHP slave device (e.g., STAR, RCU2, etc.). This naming convention allows you to create unique SAGE^{MAX} point names for point names that are common among several PHP slave devices. An example is illustrated in **Figure 15-4**.

The host naming convention requires that the SAGE^{MAX} name must contain a space. The characters (up to 12) that follow the space in the SAGE^{MAX} point name represent the PHP slave point name. The characters (up to 11) that precede the space in the SAGE^{MAX} point name represent a user-defined *prefix* that you use to make the SAGE^{MAX} point names unique.

In the example shown in **Figure 15-4**, three slave STARs are on a PHPHdcon network that is connected to a SAGE^{MAX}. Each of the slave STARs has defined a database point **ZONETEMP**. The SAGE^{MAX} database has three unique points, each of which consists of a *unique prefix*, a *space character* and the *common point name used by the slave STARs*.

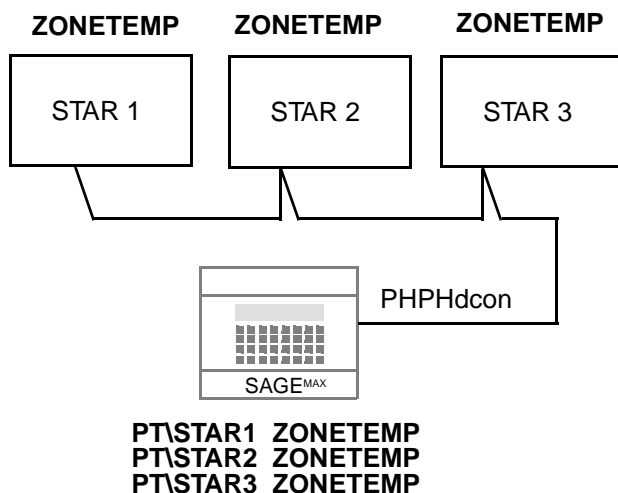


Figure 5-4 Naming Conventions for PHPHdcon Ports

VARIABLE	DESCRIPTION (Default)
PHP Unit Number	PHP unit number for this SAGE ^{MAX} from 48-57 or 62-255 (default=48).
Extended Site ID	An extended ID number (0-65535) for PHP dial-up systems (default=0).
Terminal Mode Delay	The delay imposed by the SAGE ^{MAX} before it responds to a PHP terminal mode request. The SAGE ^{MAX} multiplies the number you enter by 25 ms (default=2).
Auto Hangup Delay	For dial-up connections, this is the amount of time the SAGE ^{MAX} will wait between keystrokes before hanging up and breaking the dial-up connection (default=70).
Response Timeout	An intercharacter timeout that is reset every time a valid PHP character is received (default=5 seconds).
Object Modify Privileges	An object modification bitmap used by the PHP slave drivers (default=00000000B).
Default Alarm Class	If nonzero, this PHP alarm class override is used by the SAGE ^{MAX} to process incoming alarms and override their class. If zero, the incoming alarm uses its own alarm class (default=0).
PHP Upload/Download Path	SAGE ^{MAX} pathname where uploaded/downloaded PHP files are stored (default is blank).
PHP Reboot Password	The password which is required for the PHP reboot command (default is SYSTEM).
Default Operator	24-character case-insensitive operator name used in alarm messages when log modifications is enabled (default is blank).
Unit # / Peer / Alarm Poll	Unit P A (Peer, Alarmpoll) The unit numbers of other PUP devices on the network. If this unit should be polled for alarms, enter A in the appropriate fields. Use the left and right arrow keys to maneuver through the field while editing. Press RETURN for new values to take effect. Press PF1 to return to the configuration menu (default is blank, i.e. no units, no peers, no alarm polling).

Table 15-2 PHPHdcon and PHPHdial Driver Type Variables

When you monitor one of these SAGE^{MAX} point names, the PHPHdcon driver passes along only the portion of the SAGE^{MAX} point name that follows the space character. This is the name (up to 12 characters max) that is referenced by the slave device to honor your monitor request.

CAUTION

When creating SAGE^{MAX} points for host network devices, you must use a space character in the SAGE^{MAX} point name. Failure to follow this naming convention will keep the SAGE^{MAX} from finding the associated point on the slave device.

15.4 PHPHdial -- The PHP Dialup Host Driver

PHPHdial is used to configure the SAGE^{MAX} for dial-in or dial-out applications (i.e., using a modem) when the SAGE^{MAX} acts as a host system. With the *PHPHdial* configuration, you can dial-out to slave PHP devices (e.g., a SOLOFone, a STAR, an RCU or RCU2, an MCU, a SAC3 or even another SAGE^{MAX}) or have them dial the SAGE^{MAX} host. In addition, a host system can dial in to a PHPHdial SAGE^{MAX} and cause it to become a slave. This allows PHP hosts (like other SAGES^{MAX}) to dial in to a PHP host SAGE^{MAX}. Refer to **Figure 15-5**.

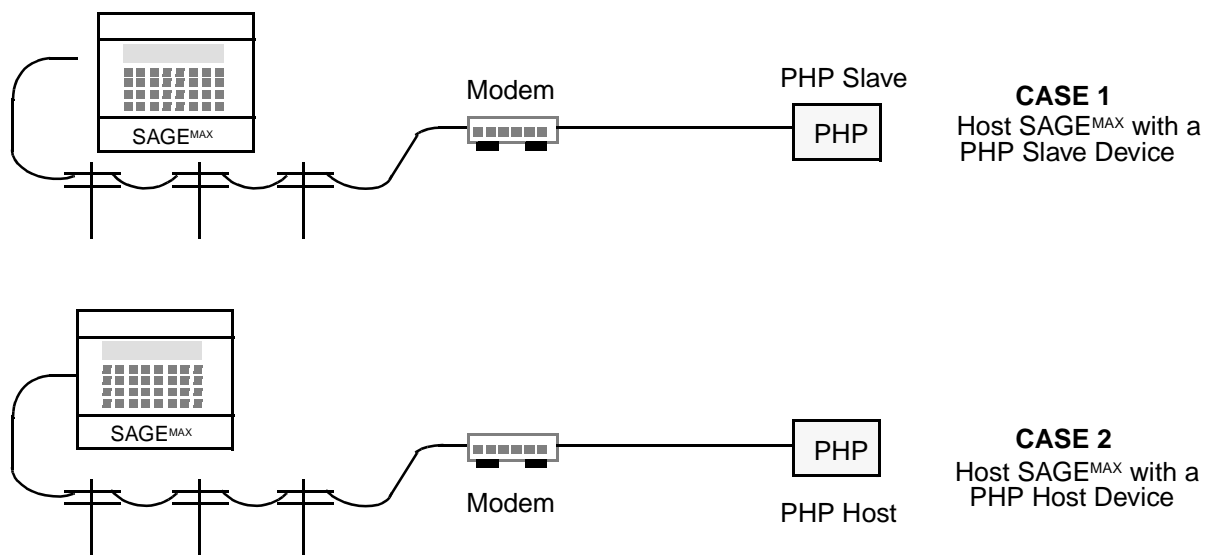


Figure 5-5 Constraints for PHPHdial Networks

15.4.1 Features (PHPHdial)

PHPHdial networks typically use the built-in SAGE^{MAX} modem to link the SAGE^{MAX} to slave PHP networks by way of a telephone line connection. Both dial-in (from slave PHP devices to the host SAGE^{MAX}) and dial-out (from the host SAGE^{MAX} to slave PHP devices) connections can be realized.

NOTE

Due to the unique nature of Smartmodem™ communications, it is unlikely that you would ever use any port other than ports 5 and 6 for PHPdial applications. SAGE^{MAX}, however, does not prevent you from configuring other ports this way.

When a SAGE^{MAX} port is configured as PHPHdial, the SAGE^{MAX} supports *Reading and Writing Attribute Values from PHP slave devices, PHP Uploading of Files, PHP Downloading of Files, Alarm Dial-In from slave PHP devices and virtual terminal capability* from the SAGE^{MAX} to dialed slave PHP units.

With the modem port configured as a PHPHdial port, the SAGE^{MAX} can dial out to remote PHP slave devices and read or write attribute values as well as perform PHP file uploading and/or downloading. The SAGE^{MAX} can also connect to PHP slave devices (e.g., STARS, RCUs, RCU2s, MCUs, SOLOFones, SAC3s and SAGES^{MAX}) in virtual terminal mode.

In addition to dial-out capabilities, PHP slave devices can dial in to the SAGE^{MAX} to report alarms. Once the slave device connects to the SAGE^{MAX}, the corresponding Site Definition File (SDF) on the SAGE^{MAX} can be programmed to cause JOB requests to take place. These job requests might include data capture/stuff, broadcasts, etc. Afterwards, a virtual terminal connection may be established with the slave, giving the SAGE^{MAX} all the capabilities of a direct connect host (i.e., *reading and writing attribute values to PHP slave devices, PHP uploading of files, PHP downloading of files, polling for alarms from slave PHP devices and virtual terminal capability* from the SAGE^{MAX} to slave PHP units).

As in dial-out applications, the SAGE^{MAX} waits 5 seconds after a dial-in connection is established so the SAGE^{MAX} can determine who is dialing in to it.

If, after the dial-in connection is established, the SAGE^{MAX} receives the PHP command ;? (Who Are You?), the SAGE^{MAX} knows that the initiator of the call is a host device. At this time the answering SAGE^{MAX} becomes a slave SAGE^{MAX} with the initiator of the call being the PHP host. The PHP host is then able to enter virtual terminal mode with the SAGE^{MAX}. If the SAGE^{MAX} does not receive an initial PHP command, then it remains a PHP host.

After the dial-in connection is established and the SAGE^{MAX} determines that it is not another PHP host that dialed in, the SAGE^{MAX} queries the slave unit with the PHP command ;? (Who Are You?) and requests the extended host ID number (XID). The PHP slave unit that dialed in responds with its XID number. The SAGE^{MAX} polls the unit for alarms and reports them if necessary.

Once the alarms are reported, the SAGE^{MAX} opens the SDF file whose name consists of the XID of the slave unit and the extension **.SDF**. All SDFs reside on the **!SITE** directory of the SAGE^{MAX}. The SDF is an ASCII text file with extension **.SDF** and a name equivalent to the extended site identification number (XID) of the remote PHP device (e.g., 00048.SDF). This file may contain job information to be executed when the corresponding slave unit dials in.

After the alarms have been reported and the SDF jobs have been queued, the connection is maintained for a period of time specified by the Auto Hangup Delay driver variable. If no further alarms are received or no further action is taken (i.e., entering virtual terminal mode with the slave unit, etc.), the SAGE^{MAX} hangs up.

15.4.2 Communications Parameters (PHPHdial)

The baud rate, parity, number of data bits, number of stop bits and the language used by PHPHdial ports are set up by using the Change Port Communications Parameters (**C**) option of the Ports Status and Setup Submenu.

The baud rate of a PHPHdial network on a SAGE^{MAX} serial port can range from 110-38,400 bps, but defaults to 28,800 bps -- the baud rate of the internal modem.

PHPHdial networks communicate using Public Host Protocol which requires the use of no parity, 8 data bits and a single stop bit for serial communications.

The default language of PHPHdial ports is 001 (English). The range for this communications parameter is 0-255.

15.4.3 Driver Variables (PHPHdial)

Driver configuration variables are used to configure certain operations of the driver. In the case of the PHPHdial driver, there are nine driver variables that define PHP host operations. These variables are:

- PHP Unit Number
- Extended Site ID
- Terminal Mode Delay (25 ms ticks)
- Auto Hangup Delay (secs)
- Response Timeout (secs)
- Object Modify Privileges
- Default Alarm Class
- PHP Upload/Download Path
- PHP Reboot Password
- Default Operator Name
- Unit # / Peer / Alarm Poll

PHP Unit Number, *Extended Site ID* and *Terminal Mode Delay* variables are used by the PHPHdcon driver are used when a host SAGE^{MAX} acts as a slave.

NOTE

When a host SAGE^{MAX} dials another host SAGE^{MAX}, the dialing SAGE^{MAX} remains a host and the answering SAGE^{MAX} acts as a slave SAGE^{MAX}. The PHP Unit Number, Turnaround Delay and Terminal Mode Delay variables of host SAGES^{MAX} should be properly configured to ensure proper slave operation in the event a host SAGE^{MAX} dials in.

PHP Unit Number is a PHPHdial driver variable that specifies the unit number for this SAGE^{MAX}. The *PHP Unit Number* can be a number from 48-57 or 62-255. The default value for this attribute is 48. This unit number is used if another PHP host dials into this port.

Extended Site ID is a PHPHdial driver variable that is used as an additional identification for dial-up applications. This driver variable defaults to 0.

Terminal Mode Delay is a PHPHdial driver variable that specifies a time delay in 25 ms increments that is imposed by the SAGE^{MAX} before it responds to a PHP terminal mode request. The SAGE^{MAX} multiplies the value of this variable by 25 ms to determine the actual delay.

Auto Hangup Delay is a PHPHdial driver variable that represents the amount of time the SAGE^{MAX} waits between network/terminal activity before hanging up and breaking the dial-up connection. This variable defaults to 70 seconds.

Response Timeout is a PHPDial driver variable that represents an inter-character timeout which is reset every time a valid PHP character is received. This driver variable defaults to 5 seconds.

Object Modify Privileges is a PHPDial driver variable that specifies an object modification bit map. This variable defaults to 00000000B.

Default Alarm Class is a PHPDial driver variable that is used by the SAGE^{MAX} to process incoming PHP alarms. If non-zero, this driver variable is used if the incoming class does not match an existing class in the SAGE^{MAX}.

PHP Upload/Download Path is a PHPDial variable that specifies a 53-character (maximum) path prefix for the filename used by a host system for uploading and downloading. This variable provides a mechanism by which *any* file on the SAGE^{MAX} may be transferred to the host, even if the file is on a subdirectory or even a different drive.

For example, there is a trend file located on **D:\SAGE5\TREND2.DAT** of a host SAGE^{MAX} (which becomes a slave when another host SAGE^{MAX} dials in to it) that you want to upload to the PHP host SAGE^{MAX} that initiated the call. The file name you request from the host is **0:TREND2.DAT**. In this example, **0:** represents a drive number (which is part of Public Host Protocol). The drive number is a requirement of PHP and it is ignored by the SAGE^{MAX}. For a successful upload, the *PHP Upload/Download Path* driver variable of the SAGE^{MAX} being dialed in to must be set to **D:\SAGE5**.

PHP Reboot Password is a PHPDial driver variable that is used by the PHP *reboot* command as an added measure of security. Anytime the PHP *reboot* command is issued, it must include a reboot password as part of the command message. If the password in the command message does not match the *PHP Reboot Password* driver variable, the command is not honored.

Default Operator Name is a PHPDial driver variable that is a 24-character (maximum) case-insensitive operator name that is used in SAGE^{MAX} *log modification* alarms that are generated when attributes of SAGE^{MAX} points are modified from a host (e.g., another host SAGE^{MAX}). If the attributes are modified after you enter terminal mode, the operator name used to sign on in terminal mode is used in the *log modification* alarms.

Unit #/Peer/Alarm Poll is a driver variable that is used by the PHPDcon driver and is explained in **Chapter 15.3.3: Driver Variables (PHPDcon)**.

15.4.4 Dial-out Point Monitoring, Site Definition Files and Park Requests (PHPDial)

To read and write attributes that reside on remote PHP slave field panels, the SAGE^{MAX} uses a *Site Definition File* (SDF). When you read or write such an attribute, for example, from the Monitor/Change Point Attributes (**M**) option of the Main Menu or from a SAGE^{MAX} program, the SAGE^{MAX} uses the SDF to determine the phone number to dial in order to reach the remote location for the specified object attribute.

When you create objects on the SAGE^{MAX} that physically reside on remote PHP slave field panels, you must specify the XID of the remote site as the unit number. Then, when you access an attribute of the object, the SAGE^{MAX}:

- checks the object definition to determine the port where the point is located
- notices that it resides on a PHPdial port
- checks the unit number (XID or site number) in the object definition
- uses the unit number and forms a Site Definition File (SDF) name with the extension **.SDF**
- reads the phone number and baud rate associated with the object from the SDF
- attempts to dial
- connects to remote device
- fetches the requested object attribute

In the event that the modem is busy servicing other requests when the attempt to access a remote object is made, the SAGE^{MAX} responds with a **Dialer Busy** error. If you were using the Monitor Menu, SAGE^{MAX} then asks if you want to issue a *Park Request*. If you respond with **Y (YES)**, the SAGE^{MAX} queues (or *parks*) your dial request until the modem becomes available.

Once the SAGE^{MAX} connects to the remote location after a park request has been issued, you are notified that the SAGE^{MAX} has finally connected to the site. The SAGE^{MAX} also displays the XID number of the site at this time. You then have an amount of time (specified by the Auto Hangup Delay driver variable) to make a request (i.e., read an attribute, write an attribute, etc.), otherwise the SAGE^{MAX} will hang up.

The SDF is an ASCII text file with extension **.SDF** and a 5-digit name equivalent to the extended site identification number (XID) of each remote PHP device (e.g., 00048.SDF). The SDF is used in dialer applications to find the site phone number. It can optionally contain scheduled job strings which can be started when the site dials in to the SAGE^{MAX}. The structure of the SDF is shown in **Figure 15-6**. A sample SDF is shown in **Figure 15-7**.

NOTES

When a PHP slave device dials in to the host SAGE^{MAX} to report alarms, the Site Definition File for that slave device is referenced by the SAGE^{MAX}. The optional job lines within the SDF are then queued on the SAGE^{MAX}.

When a PHP host device (such as a PHPdial SAGE^{MAX}) dials in to a PHPdial SAGE^{MAX}, the initiator of the call is considered the master or host, while the SAGE^{MAX} that answers the phone acts as a slave device. This allows PHPdial SAGES^{MAX} to communicate with each other (e.g., virtual terminal connection).

```

;Statistics go here
;The name of this file is C:\site\00001.sdf
Site 00001 description for alarm reporting
2400555-1234
;Start a data capture job if Mon-Fri between 8:00 and 18:00.
Mon Tue Wed Thu Fri 08:00 18:00
DCS\site\site1

```

Figure 5-6 Structure of the Site Definition File (SDF)

```

;This is the required statistics comment line
site description
site baud rate and phone number
;Job 1 description (comment lines are ignored)
Job 1 active days, start time, end time
Job 1 job string (this string is sent to the job executor)
;Job 2 description (comment lines are ignored)
Job 2 active days, start time, end time
Job 2 job string (this string is sent to the job executor)
      :
      :
;Job n description (comment lines are ignored)
Job n active days, start time, end time

```

Figure 5-7 Sample SDF for Site 00001

15.4.5 Special Host Naming Conventions

When you create a point on the SAGE^{MAX}, there is a special naming convention that you must follow if the physical point resides on a PHP slave device (e.g., STAR, RCU2, SOLOFone, etc.). This naming convention allows you to create unique SAGE^{MAX} point names for point names that are common among several PHP slave devices. An example is illustrated in **Figure 15-8**.

The host naming convention requires that the SAGE^{MAX} name must contain a space. The characters (up to 12) that follow the space in the SAGE^{MAX} point name represent the PHP slave point name. The characters (up to 11) that precede the space in the SAGE^{MAX} point name represent a user-defined *prefix* that you use to make the SAGE^{MAX} point names unique.

In the example shown in **Figure 15-8**, three slave devices can be dialed by a SAGE^{MAX} from the PHPHdial port (modem port) on a SAGE^{MAX}. Each of the slaves has defined a database point **MXFE00**. The SAGE^{MAX} database has three unique points, each of which consists of a *unique prefix*, a *space character* and the *common point name used by the slave devices*.

When you monitor one of these SAGE^{MAX} point names, the PHPHdial driver passes along only the portion of the SAGE^{MAX} point name that follows the space character. This is the name (up to 12 characters max) that is referenced by the slave device to honor your monitor request.

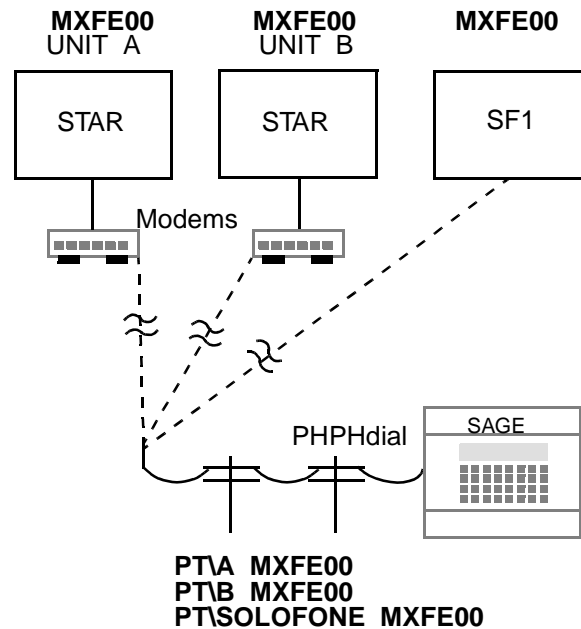


Figure 5-8 Naming Conventions for PHPdial Ports

CAUTION

When creating SAGE^{MAX} points for host network devices, you must use a space character in the SAGE^{MAX} point name. Failure to follow this naming convention will result in the SAGE^{MAX} inability to find the associated point on the slave device.

15.5 Message Handling

Messages received from PHP devices can be in one of four possible formats (refer to **Figure 15-9**). The simplest and oldest form is the unstructured message (line d) which begin with a space (20h) character. The next form is a pre-PHP version 8 format called a transaction message (line c). Beginning with PHP version 8, there are structured transaction (line a) and continuation messages (line b).

The type of message being used is characterized by the following set of rules. If the leading character is a hyphen (-), then the message is a continuation message (line b). If the leading character is a digit ("0"- "9") and the sixth character is a slash (/), then the message is a version 8 structured message (line a). If the leading character is a digit ("0"- "9") and the sixth character is not a slash (/), then the message is a pre-version 8 transaction message (line c). If none of these cases are true, then the message is unstructured (line d).

In **Figure 15-9**, the **ttttt** field represents a 4-digit transaction number. The **sssss** field represents a 5-digit unit number. The **ccc** field represents the 3-character class number/name.

Incoming alarm and event notification messages that have an unstructured format (**Figure 15-9 d**) have no time or date stamp associated with them. In addition, neither a class name or number is supplied in alarms that use an unstructured format. These types of alarms consist of simple text.

With unstructured alarm messages, ALOG uses the class override driver variable associated with the port through which the alarm was reported to the SAGE^{MAX} as the class.

Transaction message format (line c) is very similar to the unstructured format, except that it includes a 5-digit transaction code. This code, however, has no relevance from the SAGE^{MAX} standpoint and therefore is not used by ALOG.

With structured transactions (line a) and continuations (line b), ALOG uses the class override driver variable associated with the port through which the alarm was reported to the SAGE^{MAX} as the class.

```

ttttt/sssss ccc -----date-----  some message text  (a)
-          ccc some continuation message                (b)
ttttt some message text                                (c)
  some message text                                    (d)

```

Figure 5-9 Types of PHP Message Formats

The PHP message format from ALOG is shown in **Figure 15-10**. In this figure, **p** represents the port number where the PHP alarm occurred. The field **ttttt** represents a sequential transaction number that is assigned by ALOG to every transaction that occurs. The **sssss** field contains the extended site identification number (XID) of the PHP unit that originated the alarm. The **ccc** field contains the SAGE^{MAX} class number (or class name if one is defined). ALOG includes standard time and date information in the **date** field. Time and date information includes the abbreviated form of the day of the week (e.g., **MON, TUE**, etc.), the date when the alarm was received by ALOG (e.g., **13-Feb-96**), and the time when the broadcast was received by ALOG (e.g., **17:23:45**).

The first continuation line of a PHP alarm message from ALOG is optional and contains the standard hyphen in column one and class number (or name if one is defined) in the **ccc** field. The text in this continuation line is the description text from the optional Site Definition File (SDF) associated with the site XID.

The next sequential continuation line that may be present is the alarm message text. This is the PHP message text that is sent from the PHP device.

If additional continuation lines are sent from the PHP device, they are appended to the alarm as continuation lines.

```

ttttt/sssss ccc -----date-----
-   ccc description text (optional, from SDF file)
-   ccc alarm message text
-   ccc continued message text (v8 continuation)

```

Figure 5-10 PHP Alarm Message Format from ALOG

CHAPTER 16 - XANP DRIVER

16.1 Features

The XANP driver is used by ports that have XANP (eXtended Automation Network Protocol) configurations. XANP networks use 2-wire configurations only. Therefore, each dual-trunk SAGE^{MAX} port (i.e., ports 1-4) can accommodate up to 32 XANP units across the two trunks of each port. Refer to **Figure 16-1**. Each trunk uses twisted shielded pair cable that may extend up to 5000 feet.

The SAGE^{MAX} broadcasts time and date information over the XANP network every five minutes. This ensures synchronization of all units on the XANP network.

When a SAGE^{MAX} port is configured as an XANP network, the SAGE^{MAX} supports *attribute reads*, *attribute writes*, *remote file copying* and *virtual terminal* capabilities.

When using the remote file copy feature on a SAGE^{MAX} XANP network, up to four simultaneous file sessions may exist (i.e., up to four units may be uploaded/downloaded simultaneously).

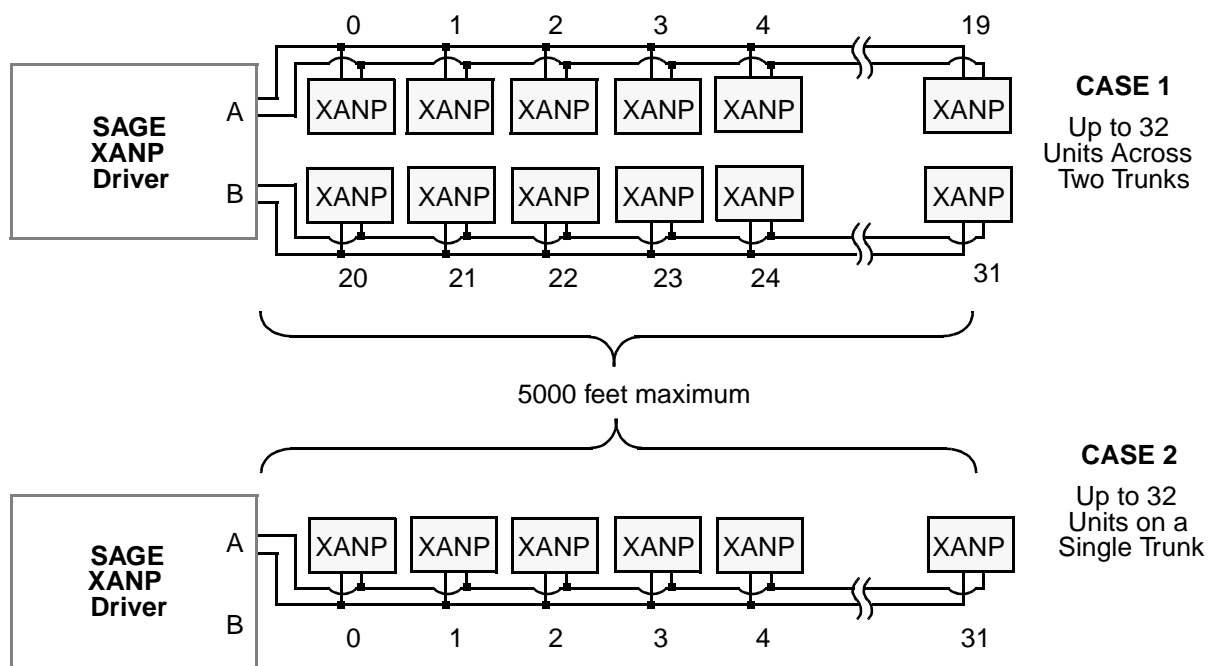


Figure 6-1 Constraints for XANP Networks

16.2 *Communications Parameters*

The baud rate, parity, number of data bits, number of stop bits and the language used by XANP ports are set up by using the Change Port Communications Parameters (**C**) option of the Ports Status and Setup Submenu.

The baud rate of an XANP network on a SAGE^{MAX} serial port can range from 110-38,400 bps. Be sure to select a baud rate that is available to all of the XANP devices on the network. Typically you will use the default XANP baud rate of 9600 bps -- lower in environments with large amounts of electrical and/or magnetic *noise* (these environments can cause communications problems), and higher if there is little noise and all the XANP devices support the higher baud rate.

XANP networks communicate using eXtended Automation Network Protocol which requires the use of no parity, 8 data bits and 2 stop bits for serial communications.

The default language of XANP ports is 001 (English). The range for this communications parameter is 0-255 and is definable in your SAGE^{MAX}.

16.3 *Driver Variables*

Driver configuration variables are used to configure certain operations of the driver. In the case of the XANP driver, there are nine driver variables that define XANP operations. These variables are:

- Leased Line Modem Control (Y/N)
- Seek Factor
- Request Interleave
- Alarm Poll Interleave
- File Session Interleave
- Max Tries per Transaction
- Block Tries to Unresponsive Units (Y/N)
- Normal Response Timeout (25ms)
- Peer Units

Leased Line Modem Control (Y/N) is an XANP driver variable that is used to configure the SAGE^{MAX} XANP port for leased line modem applications.

In leased line modem applications, DCD (data carrier detect) and CTS (clear to send) signals are used. Before the SAGE^{MAX} will transmit an XANP message, DCD must be **FALSE**. At that time, the SAGE^{MAX} (via the XANP driver) asserts RTS (i.e., request to send = **TRUE**) and waits for CTS to be asserted (i.e., CTS = **TRUE**).

If leased line modem control is not used, DCD and CTS signals are not used by the XANP driver, however, RTS is asserted in either case. Leased line modem control defaults to disabled (**NO**).

Seek Factor is an XANP driver variable that represents the number of transactions that must occur before the SAGE^{MAX} retries unresponsive XANP units. This driver variable is set to 100 by default.

If you are viewing the status of XANP units from the Watch Performance Statistics Submenu, a unit may be marked as unresponsive (?), but may actually be a potentially responsive unit that has not yet been retried due to the *Seek Factor*.

Request Interleave is an XANP driver variable that specifies how XANP network requests are interleaved with alarm polls and file transfer transactions.

Alarm Poll Interleave is an XANP driver variable that specifies how XANP alarm poll transactions are interleaved with network requests and file transfer transactions.

File Session Interleave is an XANP driver variable that specifies how XANP file transfer transactions are interleaved with network requests and file transfer transactions.

With values of 3, 3, and 2 for Request Interleave, Alarm Poll Interleave and File Session Interleave, the sequence of XANP message would be the following:

- 3 network requests
- 3 alarm polls
- 3 network requests
- 3 alarm polls
- 1 file transfer transaction

Max Tries per Transaction is an XANP driver variable that specifies the maximum number of times an XANP transaction is transmitted before it is considered a failed transaction. This variable defaults to 3.

Block Tries to Unresponsive Units (Y/N) is an XANP driver variable that allows you to specify whether or not attempts to access attributes on unresponsive units should be blocked (i.e., not allowed).

This mode prevents unnecessary timeouts from occurring when unresponsive units are accessed. This variable defaults to **YES**, meaning that access to unresponsive units is blocked.

If this variable is set to **NO**, access to unresponsive units is permitted. This mode may potentially cause numerous timeouts for each request to a truly unresponsive unit, and is therefore not recommended.

Normal Response Timeout (25ms) is an XANP driver variable that specifies the amount of time (in 25 ms increments) that the SAGE^{MAX} waits for a response from another unit on the XANP network before assuming the transaction failed. This variable defaults to 12 which represents 300 ms (12 times 25 ms = 300 ms).

Peer Units is an XANP driver variable that identifies which units (0-31) are on the XANP network. Using the left and right arrow keys, you position the cursor below the appropriate unit number and type “*” (an asterisk) if you want the SAGE^{MAX} to attempt to scan that unit.

Instead of an asterisk, you can use the letter “A”. In addition to attempting to scan the unit, this also enables the auto-download feature for the specified unit. Thereafter, a *born again* request from the XANP device will automatically cause downloading.

The auto-download feature is available for RCUs, RCU2s and MCUs and involves downloading a previously uploaded “unit data” file.

NOTE

SAGE^{MAX} auto-download commands are ignored by STAR field panels. STARs have other mechanisms for memory backup.

The binary data files used in auto-downloads are stored on the **C:\CFG** subdirectory of the SAGE^{MAX}. The naming convention for file names is **Pport UNITunit.DAT**, where **port** is the SAGE^{MAX} port number of the associated XANP network, and **unit** is the unit number (00-31) of the XANP device in question. For example, the file name **C:\CFG\P1UNIT08.DAT** represents the auto-download file for unit number 8 on SAGE^{MAX} port 1 -- an XANP port. These files are uploaded to the SAGE^{MAX} using the Upload (**U**) function of the XANP Driver Configuration Menu.

Table 16-1 lists all of the XANP driver type variables, gives a brief explanation and shows the default value for each variable.

16.4 Dynamic Status of XANP Units

The Watch Performance Statistics (**W**) option of the XANP Driver Configuration Submenu, presents you with a “big picture” of which units are responsive and which units are not. See **Figure 16-2**.

SAGE MAX Banner Line Text				Mon 23-Sep-96 12:00:00			
Opr:		Port: 07					
0=*	1=*	2=?	3=1	4=1	5=*	6=*	7=*
8=*	9=*	10=*	11=*	12=*	13=*	14=*	15=*
16=*	17=*	18=*	19=*	20=?	21=*	22=*	23=*
24=*	25=*	26=*	27=*	28=*	29=*	30=1	31=1

Figure 6-2 Sample Dynamic Status Screen

If an XANP unit is unresponsive, it displays a “?” beside its unit number. For responsive units, the character that is displayed reflects the type of unit that is being scanned. An “*” represents a STAR, a “2” represents an RCU2 and a “1” represents an RCU. In addition, a “U” and a “D” indicate that unit data files are being uploaded or downloaded to/from the associated XANP unit.

If an XANP unit displays something other than “?” beside its unit number (i.e., *, 1 or 2), this does not necessarily mean that the unit is responsive. In order for an XANP unit to be marked as unresponsive, it must fail to respond for *Max Tries per Transaction* tries. If the status is checked before the maximum number of tries has occurred, the unit may only appear to be responsive, but may actually be on the verge of timing out.

VARIABLE	DESCRIPTION (Default)
Leased-line Modem Control	If YES, configures port for use with leased-line modems (default=NO).
Seek Factor	The number of transactions before retrying unresponsive units (default=100).
Request Interleave	The maximum number of XANP requests that can occur when the SAGE ^{MAX} has the token (default=9).
Alarm Poll Interleave	The maximum number of times the SAGE ^{MAX} polls for alarms on the XANP network when the SAGE ^{MAX} gets the token (default=1).
File Session Interleave	The maximum number of file transfer session transactions which are interleaved with requests (default=1).
Max Tries per Transaction	The maximum number of times a transaction is transmitted across the Peernet before it is considered a failed transaction (default=3).
Block Tries to Unresponsive Units	If YES, access to unresponsive units will be blocked. If NO, you may be able to access a previously unresponsive unit if it had come back on line, but has not yet been polled (default=YES).
Normal Response Timeout	The amount of time (in 25 ms increments) the SAGE ^{MAX} waits for a response from another unit on the XANP network before assuming the transaction failed. The SAGE ^{MAX} multiplies the number you enter by 25 for the actual response timeout in milliseconds (default=12).
Peer Units	<pre> 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 ----- </pre> <p>Identifies which units (0-31) are on the XANP network (default is blank, i.e. no units)</p> <p>Use the left and right arrow keys to position the cursor below the appropriate unit number and type * (an asterisk) to identify units on the Peernet. An "A" can be entered instead of an asterisk for automatic download. Press RETURN for new values to take effect. Press PF1 to return to the configuration menu.</p>

Table 16-1 XANP Driver Type Variables

16.5 Message Handling

When an alarm is first detected, the XANP driver searches for a point in the SAGE database that corresponds to the unit number and channel of the XANP alarm. If the XANP driver cannot find a matching point in the SAGE^{MAX} database, a fictitious name is constructed in the form:

[UU:FF:DDD:CCC:SS].

UU represents the unit number (00-31). **FF** represents the fundamental type of the alarm (e.g., SY, AI, BO, etc.). **DDD** represents the card number (000-127). **CCC** and **SS** represent the channel number (000-127) and subchannel number, respectively.

If no point name is found, the general class (000) is used. If a point name is found, its class override determines the class number to use. If the point specifies an alarm or return message (i.e., the alarm and return message text fields are non-blank), then the appropriate message will be forwarded to ALOG as a continuation line along with the alarm or return. Refer to **Figure 16-3**.

If a point name exists, but the class override is non-zero, then it is used as the class for the messages sent to ALOG.

Messages received from XANP devices contain a fundamental type, card, channel, subchannel, a standardized alarm type, and a flag indicating an alarm or a return to normal. The SAGE^{MAX} XANP driver constructs all text messages from this information.

In **Figure 16-4**, **p** represents the port number where the XANP alarm occurred. The field **tttt** represents a sequential transaction number that is assigned by ALOG to every alarm transaction that occurs. The **IIII** field contains the unit number on port **p** that generated the alarm. The SAGE^{MAX} class number (or class name if one is defined) is in the **ccc** field. ALOG includes standard time and date information in the **date** field. Time and date information is included in the prevailing language format. In the default language (English), the format includes the abbreviated form of the day of the week (e.g., **MON, TUE**, etc.), the date when the alarm was received by ALOG (e.g., **23-Sep-96**), and the time when the alarm was received by ALOG (e.g., **17:23:45**).

The second line of the message format shown in **Figure 16-4** is a continuation line. This continuation line contains the standard hyphen in column one and the class number (or class name if defined) in the **ccc** field. In addition, the associated point name of the alarm is included with the alarm type (e.g., contact, supervisory, high limit, low limit, etc.) and, when appropriate, the alarm status (i.e., alarm or return).

If a point name cannot be determined, a *fake* point name is created in lieu of the real point name. The fake point name is constructed using unit number, fundamental type, card, channel, and subchannel information that is supplied. Refer to **Figure 6-5**. This information is used in the same way as the information in **Figure 16-4**.

The third line of the message format shown in **Figure 16-4** is another continuation line. This second continuation line is an optional line that contains the alarm or return message text (if text is specified) from the point object.

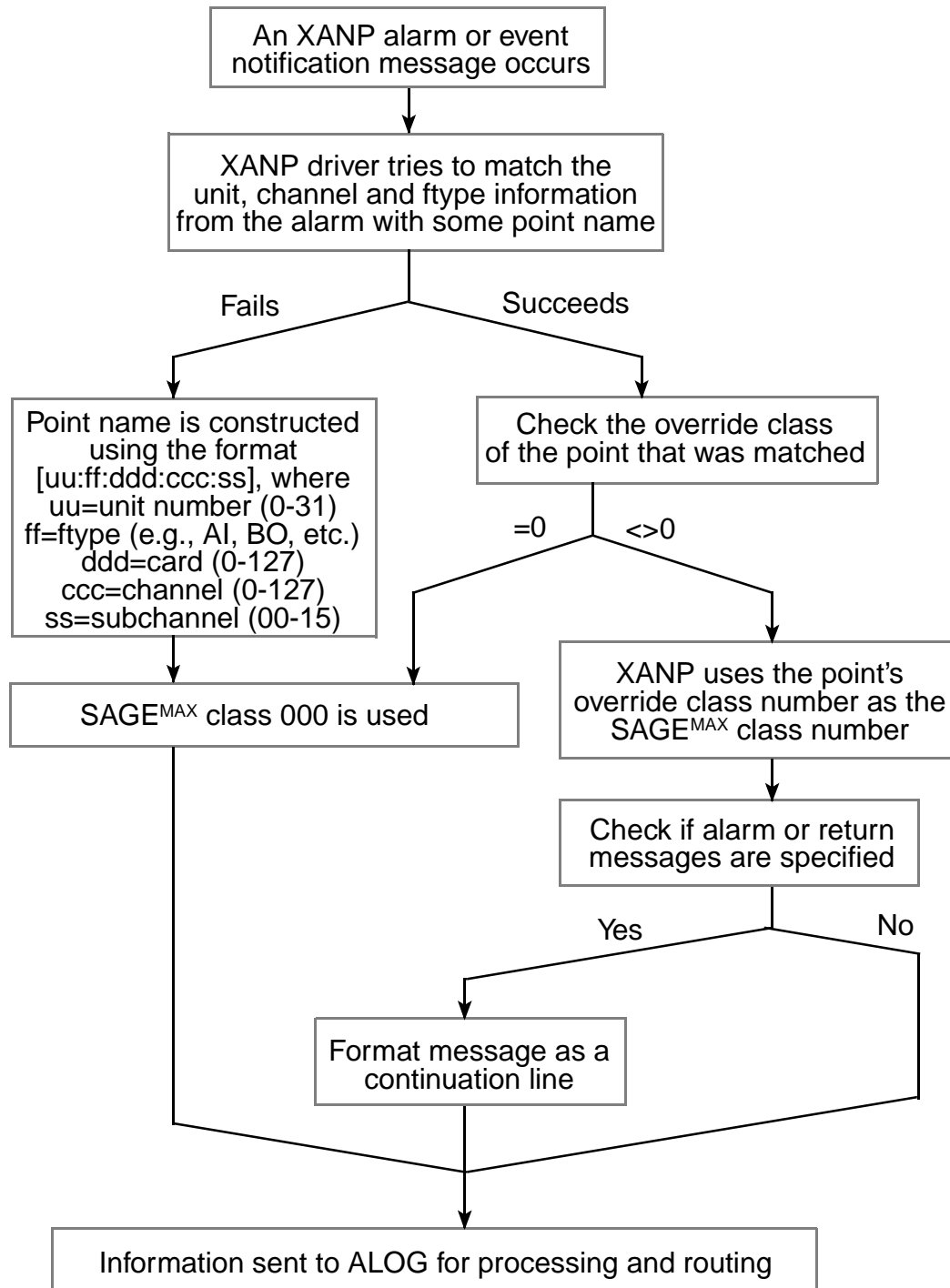


Figure 6-3 XANP Alarm and Message Handling Flowchart


```

ttttp/lllll ccc -----date-----
-          ccc point name          alarm type  status
-          ccc point alarm/return msg (optional)

```

Figure 6-4 XANP Message Format - Case 1

```

ttttp/lllll ccc -----date-----
-          ccc [uu:ff:ddd:ccc:ss]    alarm type  status

```

Figure 6-5 XANP Message Format - Case 2

16.6 Remote Path Syntax

Remote file copying is supported over XANP networks. This feature allows you to upload files from XANP network devices to the SAGE^{MAX} and download files from the SAGE^{MAX} to XANP network devices using the Remote File Copy Option in the Utility Functions Submenu (see **Chapter 8.15.11: Remote Copy**), by making job requests directly (see **Chapter 8.8.5: Make Job Request Directly**), or by using the JOB statement in SPL programs (see **Chapter 11.33: The JOB Statement**).

When using the appropriate upload or download option from the Remote File Copy Submenu, you are prompted to enter all of the necessary information separately in order to perform the upload or download.

However, when files are manipulated using SAGE^{MAX} jobs that are created either directly or through an SPL program, you must specify the *pathname* of the file using the special syntax:

uuuu/dd:file.ext

where **uuuu** is the unit number of the XANP device, **dd** is the drive number, and **file.ext** is the file to be uploaded/downloaded (e.g., **SYSTEM.SYS**, **VARIABLE.SYS**, **DATABASE.SYS**, **trendname.TRN**, **MODULEn.IOM**).

The drive number field is included to allow the uploading and downloading of files between a SAGE^{MAX} and STAR through an intermediate STAR on a Peernet as shown in **Figure 16-6**).

In this scenario, the unit number field (**uuuu**) represents the unit number of the STAR that is connected to the SAGE^{MAX}. The drive number field (**dd**) represents the unit number of the STAR on the Peernet whose files are to be uploaded/downloaded by the SAGE^{MAX}.

When you specify the XANP file syntax for Remote File Copy and Job Scheduler functions, SAGE^{MAX} displays separate prompts for the unit number and drive/filename.

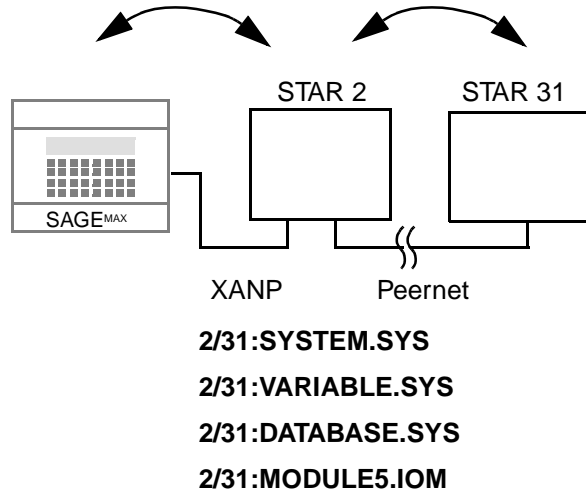


Figure 6-6 Sample Files that can be Downloaded to or Uploaded from a STAR on a Peernet

CHAPTER 17 - STAR PEERNET DRIVER

17.1 Features

The Peernet driver is used by ports that have SAGE^{MAX}/STAR peernet configurations. Peernet networks use 2-wire configurations only. Therefore, each dual-trunk SAGE^{MAX} port (i.e., ports 1-4) can accommodate up to 31 peernet units across one of the two trunks of each port. Refer to **Figure 17-1**. Each trunk uses twisted shielded pair cable that may extend up to 5000 feet.

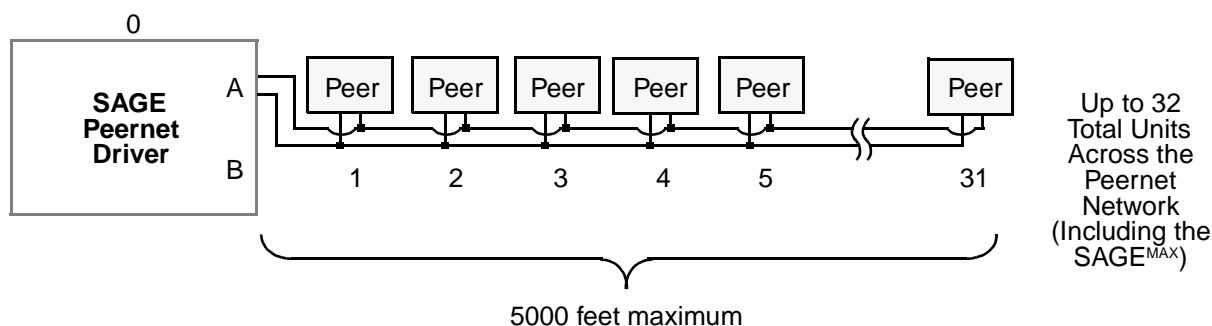


Figure 7-1 Constraints for Peernet Networks

The SAGE^{MAX} broadcasts time and date information over the peernet network every five minutes. This ensures synchronization of all units on the peernet network.

When a SAGE^{MAX} port is configured as a peernet network, the SAGE^{MAX} supports *attribute reads*, *attribute writes*, *remote file copying*, *broadcasting* and *virtual terminal* capabilities. Except for file copying, these same functions are available to STARs on the SAGE^{MAX} peernet network.

NOTE

Remote file copying is not available between SAGES^{MAX} on a peernet. This feature is, however, available for SAGES^{MAX} on an Ethernet network.

When using the remote file copy feature on a SAGE^{MAX} peernet network, up to four simultaneous file sessions may exist (i.e., up to four units may be uploaded/downloaded simultaneously).

NOTE

The SAGE^{MAX} does not poll for alarms on peernet networks, although alarm messages may be broadcast between peernet units.

When a Peernet driver on the SAGE^{MAX} or the SAGE^{MAX} itself is restarted, the Peernet driver transmits a message over the peernet network to all units. This message is broadcast to the units to inform them that a peer on the network is restarting. This is also a very quick method of getting the unit number of the restarted unit into the unit maps of the other network peers.

17.2 *Communications Parameters*

The baud rate, parity, number of data bits, number of stop bits and the language used by peernet ports are set up by using the Change Port Communications Parameters (**C**) option of the Ports Status and Setup Submenu.

The baud rate of a peernet network on a SAGE^{MAX} serial port can range from 110-38,400 bps. Be sure to select a baud rate that is available to all of the peernet devices on the network. Typically you will use the default peernet baud rate of 9600 bps--lower in environments with large amounts of electrical and/or magnetic *noise* (these environments can cause communications problems), and higher if there is little noise and all the peernet devices support the higher baud rate.

Peernet networks communicate using a proprietary network protocol which requires the use of no parity, 8 data bits and 1 stop bit for serial communications.

17.3 *Driver Variables*

Driver configuration variables are used to configure certain operations of the driver. In the case of the Peernet driver, there are nine driver variables that define peernet operations. These variables are:

- Peer Unit Number for this SAGE^{MAX}
- Max Transactions before Token Pass
- Request Interleave
- File Session Interleave
- Max Tries per Transaction
- Normal Response Timeout (25 ms)
- Broadcast Time Sync
- Peer Units

Peer Unit Number for this SAGE^{MAX} is a Peernet driver variable that refers to the peernet unit number for this SAGE^{MAX} from 0-31. The default value for this variable is 0. This value is analogous to the ID number set through SW4 on STAR field panels.

Max Transactions before Token Pass is a Peernet driver variable that refers to the maximum number of peernet transactions that can occur before the SAGE^{MAX} passes the token to the next device on the peernet. SAGE^{MAX} may voluntarily relinquish the token before this number of transactions is reached. This variable defaults to five, which means that a maximum of five peernet transactions may occur before the SAGE^{MAX} will pass the token.

Request Interleave is a Peernet driver variable that specifies the maximum number of peernet requests that can occur when the SAGE^{MAX} has the token. This variable defaults to four.

File Session Interleave is a Peernet driver variable that specifies the maximum number of file session transactions which are interleaved with peernet network requests. This variable defaults to 1, which means that for every group of *Request Interleave* peernet requests, a single file transfer session transaction may occur.

Max Tries per Transaction is a Peernet driver variable that specifies the maximum number of times a transaction is transmitted across the peernet without a valid response before the transaction is considered a failed transaction. The variable defaults to three retries.

Normal Response Timeout (25 ms) is a Peernet driver variable that specifies an amount of time in 25 ms increments that the SAGE^{MAX} waits for a response from another unit on the peernet network before assuming that the transaction failed. The SAGE^{MAX} multiplies the number you enter by 25 ms for the actual response timeout. This variable defaults to 18 (=450 ms or approximately 0.5 seconds) and should not normally be changed.

Broadcast Time Sync is a Peernet driver variable that specifies whether or not it is the responsibility of the SAGE^{MAX} to broadcast time and date information on a routine basis to peernet devices on the network. If this variable is set to **YES**, time and date information are broadcast over the peernet network every five minutes. If this variable is set to **NO**, the SAGE^{MAX} assumes that some other peer is responsible for updating time and date information for the network devices. This driver variable defaults to **NO**.

Peer Units is a Peernet driver variable that identifies which units (0-31) are on the network. Using the left and right arrow keys, you position the cursor below the appropriate unit number and type "*" (an asterisk) if you want the SAGE^{MAX} to attempt to token pass with that peer.

Peernet driver type variables are listed in **Table 17-1**.

17.4 Message Handling

Messages received from the Peernet are forwarded to the alarm logging task (ALOG) with one of 17 classes. Normal broadcast messages are converted to the general SAGE^{MAX} class (000). Alarm messages with actions 0-15 are converted to SAGE^{MAX} classes 10-25.

Messages directed to a Peernet port are either broadcast or sent to a specific unit. The SAGE^{MAX} class number (modulo 16) is used as an action number. Refer to **Figure 17-2**.

Messages received from STAR peernet devices contain an action code 0-15 and message text. For peernet alarms, the action code is remapped by the Peernet driver into 16 contiguous SAGE^{MAX} classes 10-25. For other peernet message transactions (i.e., broadcasts), the SAGE^{MAX} uses class 000 (the general class) to handle incoming messages. There is no structure to the message text.

VARIABLE	DESCRIPTION (Default)
Peer Unit Number for this SAGE^{MAX}	The Peer unit number for this SAGE ^{MAX} from 0-31 (default=0).
Max Transaction before Token Pass	The maximum number of Peernet transactions that can occur before the SAGE ^{MAX} passes the token to the next device on the Peernet. SAGE ^{MAX} may voluntarily relinquish the token before this number of transactions is reached (default=5).
Request Interleave	The maximum number of Peernet requests that can occur when the SAGE ^{MAX} has the token (default=4).
File Session Interleave	The maximum number of file transfer session transactions which are interleaved with requests (default=1).
Max Tries per Transaction	The maximum number of times a transaction is transmitted across the Peernet before it is considered a failed transaction (default=3).
Normal Response Timeout	The amount of time (in 25 ms increments) the SAGE ^{MAX} waits for a response from another unit on the Peernet before assuming the transaction failed. The SAGE ^{MAX} multiplies the number you enter by 25 for the actual response timeout in milliseconds. This variable should not be changed (default=18).
Broadcast Time Synch?	If YES, enables SAGE ^{MAX} to broadcast current time every hour, when first powered up and if requested by a Peer unit (default=NO).
Peer Units	<pre> 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 - - - - -</pre> <p>Identifies which units (0-31) are on the Peernet (default is blank, i.e. no Peer units)</p> <p>Use the left and right arrow keys to position the cursor below the appropriate unit number and type * (an asterisk) to identify units on the Peernet. Press RETURN for new values to take effect. Press PF1 to return to the configuration menu.</p>

Table 17-1 Peernet Driver Type Variables

Peernet alarm messages are displayed in one of two ways by ALOG. The format is determined by the source of the peernet alarm -- namely a STAR or another SAGE^{MAX}. In either case (and in the case of broadcasts as well), the alarm contains two components.

Whether it is a broadcast or an alarm transaction, the first part of the transaction is a standard line supplied by ALOG which lists the transaction number, port number, unit number that caused the alarm, SAGE^{MAX} class number, and the date and time the alarm was received by ALOG.

The second part of the transaction is made up of one or more continuation lines that contain Peernet alarm or broadcast message text. In alarm transaction cases, alarm message text (if available) is used. In the case of broadcasts, the broadcast message text is used.

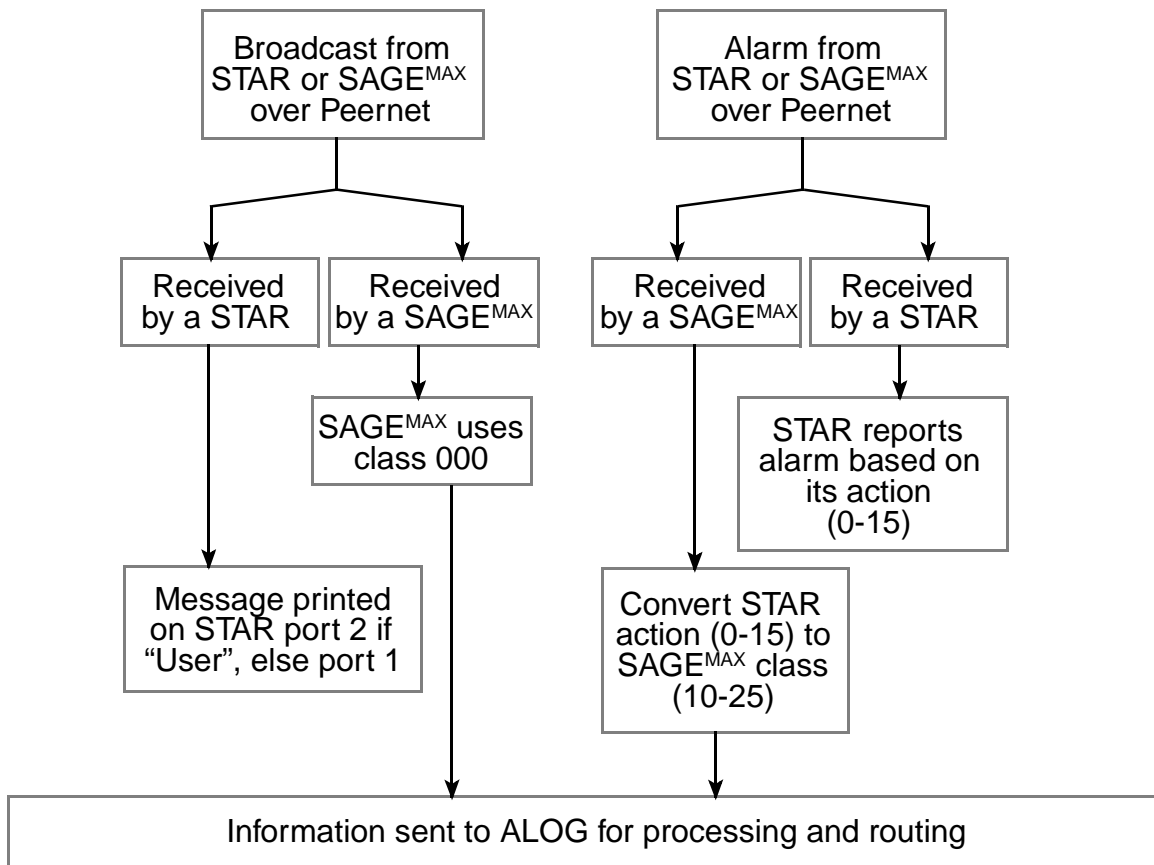


Figure 7-2 Peernet Alarm and Message Handling Flowchart

In **Figure 17-3**, **p** represents the port number where the peernet alarm occurred. The field **tttt** represents a sequential transaction number that is assigned by ALOG to every alarm transaction that occurs. The **lllll** field contains the unit number on port **p** that generated the alarm. The SAGE^{MAX} class number (or class name if one is defined) is in the **ccc** field. ALOG includes standard time and date information in the **date** field. Time and date information includes the abbreviated form of the day of the week (e.g., **MON, TUE**, etc.), the date when the alarm was received by ALOG (e.g., **23-Sep-96**), and the time when the alarm was received by ALOG (e.g., **17:23:45**).

Continuation lines consist of a hyphen in column one, the SAGE^{MAX} class number (or class name if one is defined), and peernet alarm message text. The peernet alarm message text has no particular format, but may contain such information as the point name, the alarm type (contact, supervisory, high limit, low limit, etc.), the alarm status if applicable (alarm or return), the associated point's message text field, alarm/return message text, and associated action message text if one is specified. Refer to **Figure 17-3**.

NOTE

Peernet alarms that originate on a STAR can only report to peer units 0-7. Alarm destination units (STARs and SAGES^{MAX}) must have peer unit numbers between 0 and 7 inclusive, if they are to receive such alarms.


```

tttpp/lllll ccc -----date-----
-          ccc first   line of peernet alarm text
-          ccc second line of peernet alarm text (if needed)
-          ccc third  line of peernet alarm text (if needed)
-          ccc fourth line of peernet alarm text (if needed)

```

Figure 7-3 Peernet Alarm Message Format from ALOG

Figure 17-4 illustrates peernet broadcast message format from ALOG. In this figure, **p** represents the port number where the Peernet broadcast occurred. The field **tttt** represents a sequential transaction number that is assigned by ALOG to every transaction that occurs. The **llll** field contains the unit number on port **p** where the broadcast originated. Either class 000 or the SAGE^{MAX} class name associated with class 000 is in the **ccc** field. ALOG includes standard time and date information in the **date** field. Time and date information includes the abbreviated form of the day of the week (e.g., **MON**, **TUE**, etc.), the date when the alarm was received by ALOG (e.g., **23-Sep-96**), and the time when the broadcast was received by ALOG (e.g., **17:23:45**).

The continuation line of a broadcast message from ALOG contains the standard hyphen in column one and class number (or name if one is defined) in the **ccc** field. The text of the continuation line is the message text of the broadcast, preceded by the port number of the initiator of the broadcast in braces (e.g., {07}).

```

tttpp/lllll ccc -----date-----
-          ccc {pp} broadcast message text

```

Figure 7-4 Peernet Broadcast Message Format from ALOG

17.5 Remote Path Syntax

Remote file operations are supported over peernet networks. This feature allows you to copy files between peernet network devices using the Remote File Copy Option in the Utility Functions Submenu (see **Chapter 8.15.11: Remote Copy**), by making job requests directly (see **Chapter 8.8.5: Make Job Request Directly**), or by using the JOB statement in SPL programs (see **Chapter 11.33: The JOB Statement**).

When using the appropriate upload or download option from the Remote File Copy Submenu, you are prompted to enter all of the necessary information separately in order to perform the upload or download.

However, when files are manipulated using SAGE^{MAX} jobs that are created either directly or through an SPL program, you must specify the *pathname* of the file using the special syntax **uuuu/dd:file.ext**

where **uuuu** is the unit number of the peernet device, **dd** is the drive number, and **file.ext** is the file to be uploaded/downloaded (e.g., **SYSTEM.SYS**, **VARIABLE.SYS**, *trendname.TRN*, **MODULEn.IOM**, **READ.ME**, etc.).

The drive number field is included to allow the uploading and downloading of files to and from SAGES^{MAX} on the Peernet. Refer to **Figure 17-5**.

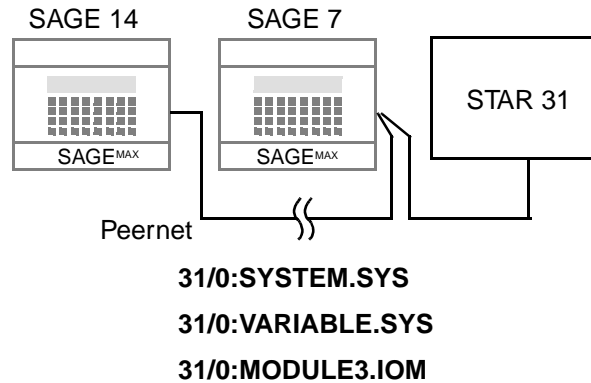


Figure 7-5 Sample Files that can be Downloaded to or Uploaded from a Peer on a Peernet

The drive letter field (**dd**) represents the drive letter of the SAGE^{MAX} on the peernet. When you perform remote file copies of STARS on the peernet, the drive field should be zero.

When you specify the Peernet file syntax for Remote File Copy and Job Scheduler functions, SAGE^{MAX} displays separate prompts for the unit number and drive/filename.

CHAPTER 18 - LPT AND PRINTER DRIVERS

18.1 Features

The LPT and printer drivers are used by ports that have printers connected to them. There are two types of printers that the SAGE^{MAX} uses: parallel printers and serial printers.

Port 13 of the SAGE^{MAX} is the only SAGE^{MAX} port that can be connected to a *parallel printer*. The parallel printer uses a standard 36-pin Centronics interface. Since the use of port 13 is dedicated to a parallel printer, this port is automatically configured with the LPT driver (i.e., the parallel printer driver).

NOTE

Port 13 only supports the LPT driver and cannot be used for any other purpose. If your SAGE^{MAX} does not use a parallel printer, then this port cannot be used.

Figure 18-1 shows a typical parallel printer connection to SAGE^{MAX} port 13 using a Centronix-to-DB25(M) cable.

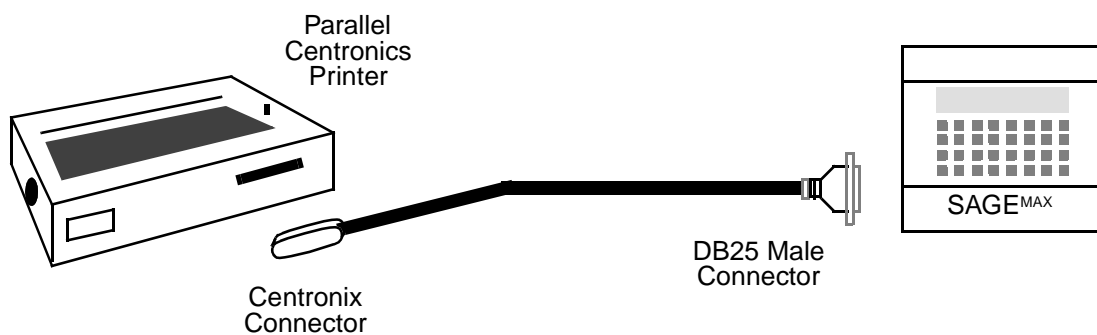


Figure 8-1 Typical Parallel Printer Connection on Port 13

Ports 1-12 on the SAGE^{MAX} can be configured to accommodate a *serial printer*, although, port 7 or 8 is typically used. If you are using a serial printer on a SAGE^{MAX} port, you must configure the port driver type as Printer.

Figure 18-2 shows a typical serial printer connection to SAGE^{MAX} port 8 using a serial DB25(M)-to-DB9(F) cable.

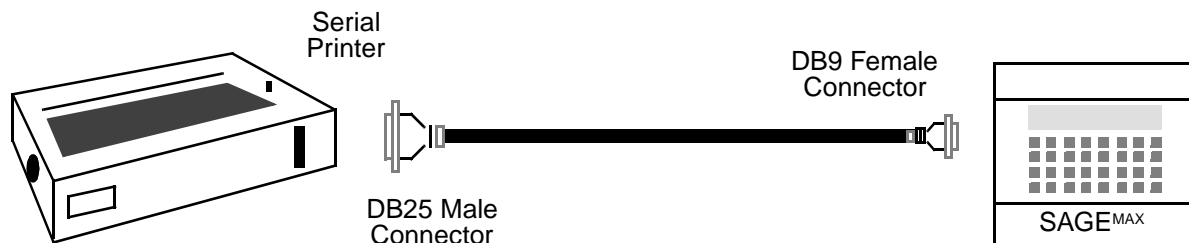


Figure 8-2 Typical Serial Printer Connection on Port 8

Parallel and serial printer interfaces on the SAGE^{MAX} are useful for spooling reports, listing or printing files and logging alarms and events.

For information about creating SPOOL JOBS and listing files to a printer, refer to **Chapter 8: SAGE^{MAX} Menu Operations**. For information on logging alarms to a printer port, refer to **Chapter 10: Alarm and Event Management**.

18.2 *Communications Parameters*

The baud rate, parity, number of data bits and the number of stop bits used by serial printer ports are set up by using the Change Port Communications Parameters (**C**) option of the Ports Status and Setup Submenu.

The baud rate used by a serial printer on a SAGE^{MAX} serial port can range from 110-38,400 bps. Be sure to select a baud rate that is available for the type of serial printer that you are using.

The use of parity, number of data bits and number of stop bits varies based on the type of serial printer that you are using. Refer to your serial printer's manual for information on the communications parameters it uses. These parameters default to no parity, 8 data bits and 1 stop bit.

The parallel printer port (port 13) does not require the setup of communications parameters.

18.3 *Spooling*

Spooling is a service provided by several drivers within the SAGE^{MAX}, notably LPT and Printer drivers.

There are places within the SAGE^{MAX} menuing system from which you can spool a file to a printer port (see **Chapter 8: Menu Operations**). At these places you are prompted with **Spool file to printer port? (Y/N):**.

By selecting **Y** from this prompt, you can request that a selected file be sent to a printer port that you specify. The port that you specify must be either port 13 (if a parallel printer is connected) or any other SAGE^{MAX} port that is configured as a Printer driver type (if a serial printer is connected).

NOTE

If you specify a spool port that is not a printer port (either parallel or serial), the spool request is ignored.

CHAPTER 19 - HISTORICAL DATA TRENDING

19.1 *What Are Trends?*

A *trend* is a historical record that contains data values of up to eight *references* (A-H) recorded over a specific period of time.

The Trend Task of the SAGE^{MAX} is responsible for administrating the dynamic loading into memory, unloading from memory, and execution of trends created on the SAGE^{MAX}.

Trend objects are binary data files which contain information about database objects (points, programs, globals and variables) that are to be monitored and the way they are to be monitored. These object files are created from the **Trend** Submenu of the SAGE^{MAX}. The structure of trend object files is explained in detail in Appendix I.

The Trend Task assumes that all trend files are located on the **C:\TREND** subdirectory. Trends that are currently running have the extension **.TR\$**, while trends not currently running have the extension **.TRN**.

19.2 *Trend Types and Terms*

There are three different types of trends that can be created on the SAGE^{MAX}: *infinite*, *circular*, and *time-anchored*.

Infinite trends (type 00) begin sampling data whenever they are started, and continue sampling data until they are either stopped or the number of samples collected reaches the maximum number of samples defined for the trend. In the latter case, the trend is aborted by the Trend Task and an alarm is generated.

The trend object file for an infinite trend is variable in length, since sample records are always appended to the file.

Circular trends (type 01) are similar to infinite trends in that they begin sampling data whenever they are started, but are a fixed length in size.

When the number of samples collected reaches the maximum allowable for the trend, the next sample is written into the first sample slot. In this way, circular trends act like a “sliding window” of data that has been sampled over some period of time. This period of time is the maximum number of samples that you specify when you create the trend.

Time-anchored trends (types 02-21) are based on an anchor interval. Anchor intervals can be *daily*, *weekly*, *monthly*, or *yearly* and produce trends which contain a history of data which has been sampled over a day, week, month, and year respectively.

Each time-anchored trend allows you to sample data in time intervals of between one minute and one month.

Since time-anchored trends are circular in nature, their associated trend object files are always a fixed length in size, based on the maximum number of samples for that particular trend.

NOTE

You do not specify the maximum number of samples for time-anchored trends. It is determined by the sample and anchor intervals.

Regardless of type, there are several terms that are used frequently when dealing with trends. These terms are *sample interval*, *current sample*, and *max samples*.

The *sample interval* is an amount of time in minutes between each recorded trend sample that you specify when creating an infinite or circular trend. The sample interval for time-anchored trends is implied as part of the anchor interval (e.g., daily, every 15 minutes, monthly every 60 minutes, etc.)

Current sample refers to the record number of the most recently collected sample that was recorded by a trend.

Max samples refers to the largest number of samples that can be recorded for the trend. For time-anchored trends, max samples is defined automatically, based on the anchor interval.

For circular and time-anchored trends, max samples specifies the number of samples that will be taken before the trend *wraps* (i.e., stores subsequent samples starting at sample 1).

For infinite trends, max samples is used as a clamp to prevent the trend from taking too many samples and wasting disk resources.

Figure 19-1 shows the components of an infinite SAGE^{MAX} trend.

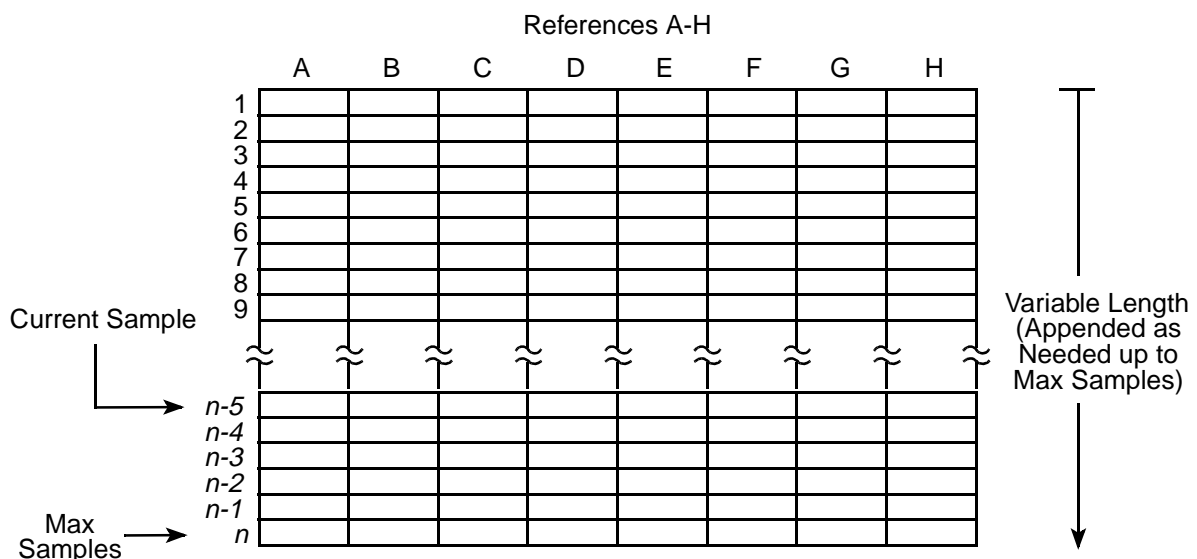


Figure 9-1 Components of an Infinite Trend

Figure 19-2 shows the components of a circular SAGE^{MAX} trend.

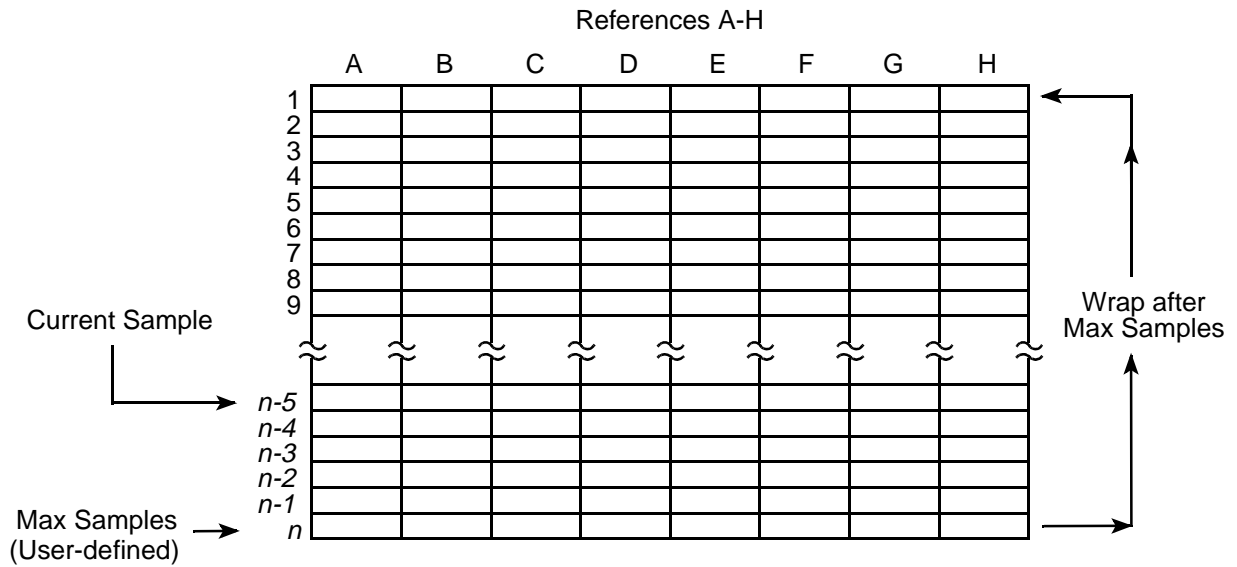


Figure 9-2 Components of a Circular Trend

Figure 19-3 shows the components of a time-anchored trend.

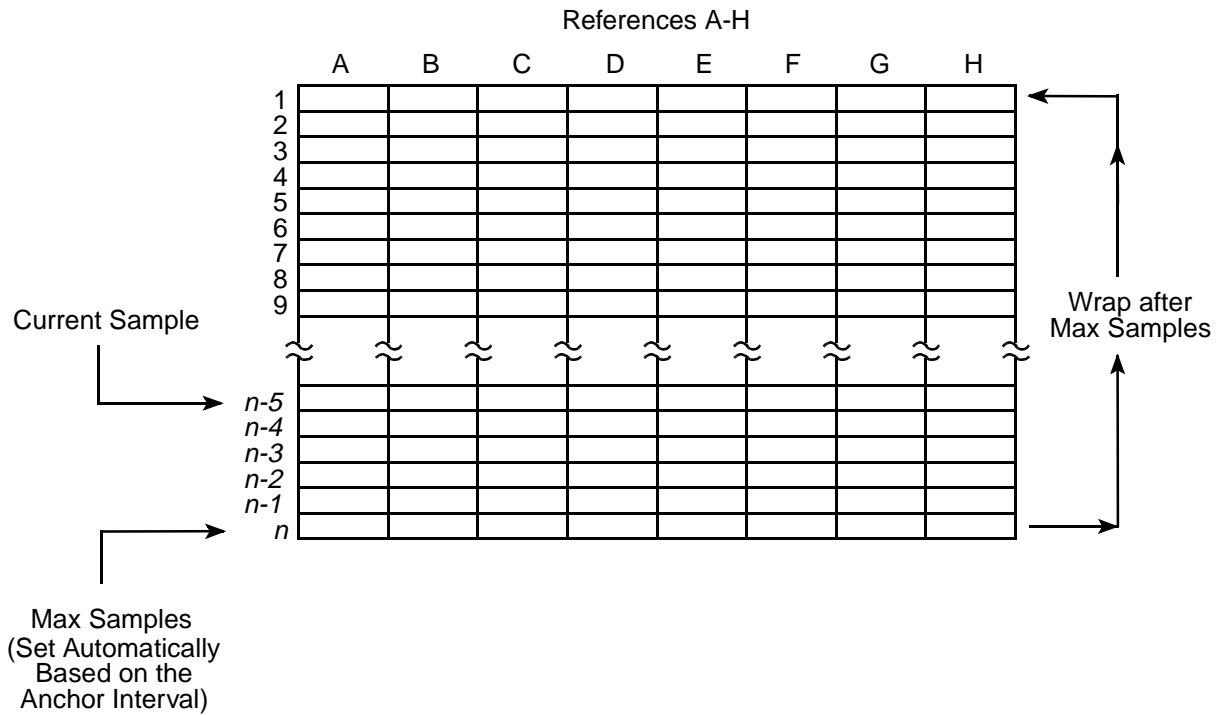


Figure 9-3 Components of a Time-Anchored Trend

19.3 Types of Time-anchored Trends

When you create a time-anchored trend, you must specify one of 20 possible *anchor intervals*. Conveniently, time-anchored trends have names that reflect their anchor intervals. These intervals are listed in **Table 19-1**.

There are four daily time-anchored trends with sample intervals of 1, 15, 30 and 60 minutes.

There are five weekly time-anchored trends with sample intervals of 1, 15, 30, 60 and 1440 (daily) minutes.

There are five monthly time-anchored trends with sample intervals of 1, 15, 30, 60 and 1440 minutes.

There are six yearly time-anchored trends with sample intervals of 15, 30, 60, 1440, 10080 (weekly) and 44640 (monthly) minutes.

Table 19-1 shows the trend type codes for each type of time-anchored trend supported by the SAGE^{MAX}, the names for the anchor intervals, how often references are sampled, how often the trend wraps (i.e., goes back to slot 1 after max samples have been taken), and the maximum number of samples for each type.

Type Code	Name	Wrapping Frequency	Frequency of Sampling	Max Samples
02	Daily_01_Min	Daily	Every minute	1440 (60x24)
03	Daily_15_Min	Daily	Every 15 minutes	96 (4x24)
04	Daily_30_Min	Daily	Every 30 minutes	48 (2x24)
05	Daily_60_Min	Daily	Every hour	24 (1x24)
06	Weekly_01_Min	Weekly	Every minute	10080 (60x24x7)
07	Weekly_15_Min	Weekly	Every 15 minutes	672 (4x24x7)
08	Weekly_30_Min	Weekly	Every 30 minutes	336 (2x24x7)
09	Weekly_60_Min	Weekly	Every hour	168 (1x24x7)
10	Weekly_1440_Min	Weekly	Every day	7 (1x7)
11	Monthly_01_Min	Monthly	Every minute	44640 (60x24x31)
12	Monthly_15_Min	Monthly	Every 15 minutes	2976 (4x24x31)
13	Monthly_30_Min	Monthly	Every 30 minutes	1488 (2x24x31)
14	Monthly_60_Min	Monthly	Every hour	744 (1x24x31)
15	Monthly_1440_Min	Monthly	Every day	31 (1x31)
16	Yearly_15_Min	Yearly	Every 15 minutes	35136 (4x24x366)
17	Yearly_30_Min	Yearly	Every 30 minutes	17568 (2x24x366)
18	Yearly_60_Min	Yearly	Every hour	8784 (1x24x366)
19	Yearly_1440_Min	Yearly	Every day	366 (1x366)
20	Yearly_week	Yearly	Every week	52 (1x52)
21	Yearly_month	Yearly	Every month	12 (1x12)

Table 19-1 Types of Time-Anchored Trends

Time-anchored trends are more sophisticated than circular trends because of their behavior after power failures or restarts of the SAGE^{MAX}. Time-anchored trends “snap” to the proper position in the fixed length trend table based on the time and date at the time the sample is taken. Circular trends which are not time-anchored simply wait for an elapsed sample interval which is not the same effect.

19.4 Trend Averaging

Trend averaging is a feature of SAGE^{MAX} trends that provides you with *averaged* data rather than a *snapshot* of the reference’s value at a given instant in time.

For example, if you create an averaged time-anchored trend with a sample interval of 15 minutes, SAGE^{MAX} takes a sample every minute for 15 minutes and calculates the average of the 15 samples. The average value is then used as the first data sample in the trend.

The same trend in snapshot (non-averaged) mode would simply take a snapshot of the reference’s value every 15 minutes.

Like infinite trends, time-anchored trends that use averaging start sampling data whenever they are started.

19.5 Trend Alarms

SAGE^{MAX} alarm class 005 is reserved for trend alarms. Trend alarms occur when a trend is unsuccessful at fetching reference values some programmable number of times. This variable is called the *sample failure alarm limit*. An alarm limit of zero disables sample failure alarming.

For example, if you set the sample failure alarm limit to 10 when you create a particular trend, a trend alarm of class 005 will be generated if the trend is unsuccessful at making 10 total fetches, regardless of whether they occur as 10 fetches from a single reference or as one fetch from 10 references. A trend may be unsuccessful at fetching a reference due to a communication problem such as a unit being off-line.

SAGE^{MAX} keeps track of the current number of unsuccessful fetches internally. If a trend is shut down and restarted, this number is reset to zero.

Infinite trends also generate an alarm of class 005 when they reach max samples.

19.6 Displaying Trend Output

After a trend collects data, it may be desirable to display the data. SAGE^{MAX} provides three ways to output trend data: *numerical display*, *stripchart display* and *listed to a file*. These options are selectable from the **Trends** Submenu.

The **Numerical Display (N)** option shows the time, date and value of the information that was referenced for each of the eight possible references. This option also shows the name of the trend, when it was activated, the type of trend, how many samples have been collected so far, the sample interval, how many failures have occurred and the low and high values of all

samples that have been collected so far. If there is more information than can fit on a single screen, SAGE^{MAX} prompts you with instructions on how to page between screens.

A sample **Numerical Display (N)** of a trend is shown in **Figure 19-4**.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Trend SAMPLE beginning on 03-20-91 11:30 is Daily.
  1440 samples were collected every 1 minutes with 0 failures.
A = GL\OATEMP;CV           Low = 64.5      High = 79.5
B = ZONE1;SP              Low = 66.8      High = 74.6
C = $MODE                 Low = 5         High = 9
D = PG\RESET;SP          Low = 67.8      High = 69.2
E = RESET;ER             Low = 0.4       High = 4.2
-----
03-20-91 11:30 A = 65.1
03-20-91 11:30 B = 72.0
03-20-91 11:30 C = 5
03-20-91 11:30 D = 68.0
03-20-91 11:30 E = 0.4
03-20-91 11:31 A = 64.8
03-20-91 11:31 B = 71.8
03-20-91 11:31 C = 5
03-20-91 11:31 D = 68.5
03-20-91 11:31 E = 0.8

Press N(+) for next page, P(-) for previous, PF1 to escape

```

Figure 9-4 Numerical Display of a Sample Trend

The **Stripchart Display (S)** option shows the trended data in a graphic format. Stripchart display includes all the information shown in numerical displays, except that the referenced data values are replaced with a scaled grid and a marker that shows where each referenced value lies within the scale.

The stripchart display prints a separate scale for each reference that is used in the trend. For example, if a trend uses five references, then five scales will be printed.

NOTE

Stripchart displays are only valid for references that have data types other than time or date data types.

A sample **Stripchart Display (N)** of a trend is shown in **Figure 19-5**.

The **List to File (F)** option allows you to list a trend in either numerical or stripchart display format to an ASCII text file. In addition, you can spool the file to a printer port on the SAGE^{MAX}.

For more information on creating, using and displaying trends, refer to **Chapter 8: SAGE^{MAX} Menu Operations**.

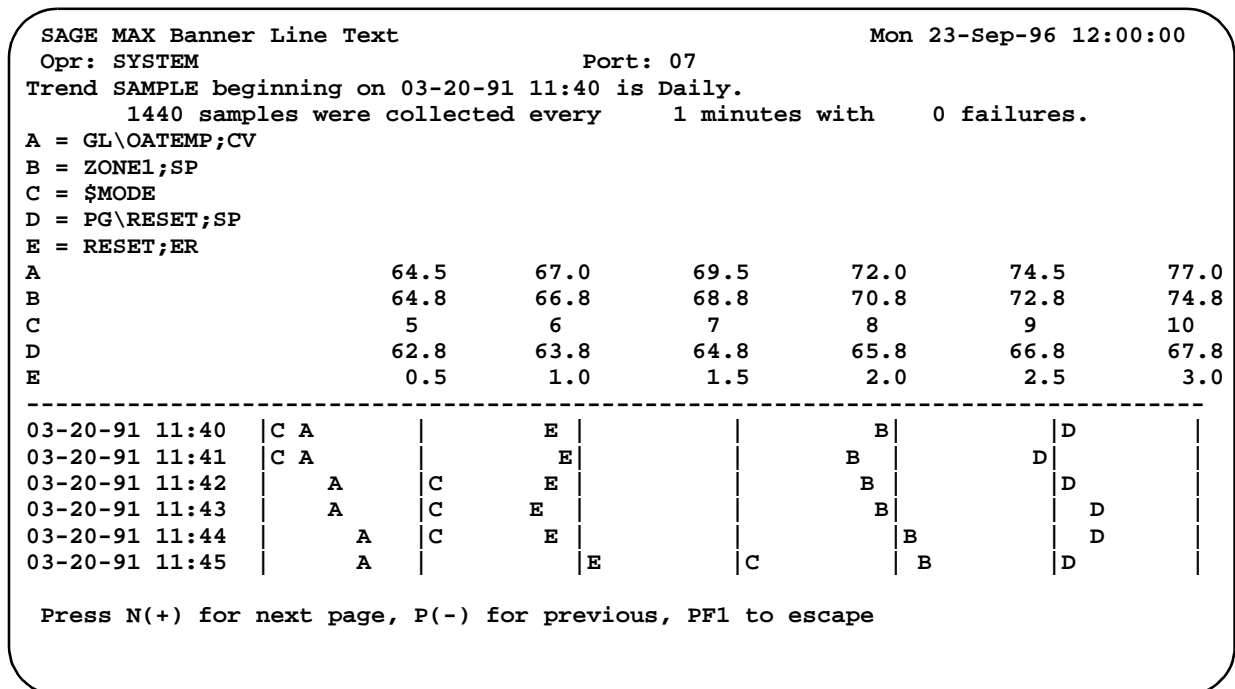


Figure 9-5 Stripchart Display of a Sample Trend

19.7 Translating PHP Trend Objects

Trend objects that have been created using a PHP device such as an RCU2 or STAR can be converted into a format that is recognizable by the SAGE^{MAX}. This is done through the **Translate PHP Trend Object (X)** menu selection of the **Trends** Submenu.

This procedure requires that the PHP trend file (*name.TRN*) is accessible by the SAGE^{MAX} (e.g., on a diskette in the **A:** drive of the SAGE^{MAX} or uploaded to the SAGE^{MAX} hard disk). After selecting the **X** option to start the translation, you supply the appropriate file names when prompted. The trend object can be listed or displayed as if it had been created on the SAGE^{MAX}.

For information on creating, activating, deactivating, listing, or initializing trends, refer to **Chapter 8: Menu Operations**.

CHAPTER 20 - SYSTEM ADMINISTRATION

20.1 SAGE^{MAX} File System Structure

The SAGE^{MAX} comes from the factory with a file system that is already installed on the hard drive. Initially, all SAGE^{MAX} files reside in one of the required SAGE^{MAX} directories. A list of these directories is shown in **Table 20-1**

DIRECTORY	CONTENTS
C:\BAK	backup
C:\CFG	configuration files
C:\EU	engineering units
C:\EXE	executable files
C:\GROUPS	groups
C:\INIT	initial values
C:\LOG	logging files
C:\LOGIC	PLB files
C:\MNU	front panel menus
C:\REF	REF files
C:\SITE	SDF files
C:\SPL	SPL sources
C:\TABLES	table files
C:\TREND	trend files
C:\UTIL	utilities

Table 20-1 Required SAGE^{MAX} Directories

The **C:\BAK** directory is the backup directory and can be used to store backup copies of files. Although the contents of this subdirectory are not backed up automatically, the directory itself is created as a convenience to the user.

The **C:\CFG** directory contains files that are used in configuring the operation of the SAGE^{MAX}. Binary object (**.BOB**) files of the database objects are found in this directory. In addition, port configuration files (e.g., **PORTS.CFG**, **ETH.CFG**) and task configuration files (e.g., **PUP.1**, **XANP.2**, **PEER.3**, etc.) are found in this directory.

The **C:\EU** directory contains engineering units files. Initially, this directory contains the file **PEX.EU**, which is the default engineering units file used by SPL programs. Save other engineering units files in this directory as you create them.

The **C:\EXE** directory contains the executable files that the SAGE^{MAX} uses to run its software. These files have the extensions **.EXE**, **.OBL** and **.JOB**. This directory also contains the language-independent message text files. These files have the extension **.MSG**.

The **C:\GROUPS** directory contains group files. These files have the extension **.GRP** and can contain up to 24 characters maximum. The **C:\GROUPS** directory may contain other group subdirectories. These subdirectories are created automatically (based on fragments of the group object names) when you create/edit group objects.

The **C:\INIT** directory contains SPL program initial values files. These files have the extension **.INI**.

The **C:\LOG** directory contains logged files and log priority queue files. Log files have the extension **.LOG** and log priority queue files have the extension **.PDQ**.

The **C:\LOGIC** directory contains SPL program logic block files. These files have the extension **.PLB**. The **C:\LOGIC** directory may contain other logic block subdirectories. These subdirectory names are embedded in the PRB name, which is a *fragment*, and are created automatically after you compile the program.

Fragments are pieces of file names and/or path names that are appended to a specific directory (e.g., **\LOGIC**, **\REF**, **\INIT**, etc.). Fragments can contain a maximum of 8 characters which can optionally be followed by a backslash (\) and 8 more characters. This provides a mechanism for defining subdirectories within the fragment name. For example, the PLB fragment **OSSPROGS\BLDG7** refers to the file **C:\LOGIC\OSSPROGS\BLDG7.PLB**.

The **C:\MNU** directory contains menu files that you create for accessing objects through the optional front panel display and keypad of the SAGE^{MAX}. Front panel menu files are ASCII text files that have the extension **.MNU**.

The **C:\REF** directory contains SPL program reference files. These files contain references that are used in SPL programs, and have the extension **.PRB**. The **C:\REF** directory may contain other subdirectories. These subdirectory names are embedded in the PRB name (a fragment) and are created automatically when you edit the ASCII PRB file from the **Edit Program References (R)** option of the **Program Edit/Compile** Submenu.

For example, you can create PRBs with the names **BLDG1\OISS**, **BLDG2\LIGHTS** and **BLDG5\DEMAND**. In these cases, the PRB name consists of a subdirectory fragment and the name of the actual PRB file.

The **C:\SITE** directory contains PHP site definition files (SDFs). SDFs are ASCII text files that contain baud rates and phone numbers to dial when you monitor a remote point from the SAGE^{MAX}.

The **C:\SPL** directory contains SPL source files. These files have the extension **.SPL**. The **C:\SPL** directory may contain other source file subdirectories. These subdirectory names are embedded in the source name (a fragment) and are created automatically when you edit the ASCII source file from the **Edit Program Logic Source (S)** option of the **Program Edit/Compile** Submenu.

For example, you can create SPL source files with the names **DEMAND\ZONE5**, **OSS\BLDG6** and **LIGHTS\LIGHTS1**. In these cases, the program source name consists of a subdirectory fragment and the name actual source file.

The **C:\TABLES** directory contains table files. Typically, these files have the extension **.TBL**, although you may define a table with a different extension. The **C:\TABLES** directory may contain other table subdirectories. These subdirectory names are embedded in the table name (a fragment or a full path name) and are created automatically when you edit the table from the SAGE^{MAX} table editor.

For example, you can create SAGE^{MAX} tables with the names **DEMAND\BLDG4\ZONE5.TBL**, **SETPTS\LOSS.TBL** or **ACME\SENSOR7.TBL**. In these cases, the table name consists of a subdirectory fragment (or fragments) and the actual table name.

The **C:\TREND** directory contains active and inactive SAGE^{MAX} trends. Active SAGE^{MAX} trends have the extension **.TR\$**, and inactive trends have the extension **.TRN**.

The **C:\UTIL** directory contains utility programs that can be used with the SAGE^{MAX}. These utilities include **FORMAT.COM**, **FDA.COM** and **DEBUG.COM**. Other utilities include the maintenance shell program **MSHELL.EXE** with the disk optimization program **DOG.EXE**, the import name bindings files utility **INBF.EXE**, and the verify disk integrity program **CHKDSK.COM**.

20.2 SAGE^{MAX} Backup Utilities

The **Backup Utilities** Submenu is a SAGE^{MAX} submenu that offers system administration functions such as system backup, hard disk optimization and integrity check, and importing name bindings files.

In order to perform such administrative functions without effecting the operation of the SAGE^{MAX}, it is necessary for the SAGE^{MAX} to be *shut down* when any of these operations are performed. This shutdown occurs automatically when you select the **Prepare System for Maintenance (Z)** option from the **Utility Functions** Submenu.

NOTES

*Before you select the **Prepare System for Maintenance (Z)** option, you must be sure that there is no diskette in disk drive A:/ of the SAGE^{MAX}.*

*If you **Prepare System for Maintenance** with a diskette in drive A:/, SAGE^{MAX} will prompt you to remove it before the **Backup Utilities** Submenu is displayed. This precaution ensures that the SAGE^{MAX} will reboot properly when you exit the **Backup Utilities** Submenu.*

After you select the **Prepare System for Maintenance (Z)** option, there is a brief pause, and the **Backup Utilities** Submenu is displayed. This is shown in **Figure 20-1**.


```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                             Port: 07
Preparing system for maintenance. Please wait...

Backup Utilities:
key          to do
A           Archive SAGE Files
B           Backup SAGE Database
R           Restore SAGE Database
S           Series Backup
I           Import Name Bindings
V           Verify Disk Integrity
O           Optimize Disk
D           Shell to DOS
Q           Exit Maintenance Shell
Press key for desired action:
```

Figure 0-1 The Backup Utilities Submenu

20.2.1 Archive SAGE Files

The **Archive SAGE Files (A)** option of the **Backup Utilities** Submenu is used to make archival copies of particular SAGE^{MAX} files.

IMPORTANT

After you archive a SAGE^{MAX} file, the source file is deleted.

When this option is selected, you are prompted to enter the source drive, pathname, and filename(s) of the file(s) you wish to archive.

Next, SAGE^{MAX} prompts you for the destination drive, pathname, and filename(s) for the archive file you are creating.

SAGE^{MAX} then confirms that you want to perform the archive function, then displays the **Backup Utilities** Submenu.

Figure 20-2 shows the process of archiving the general alarm file of the SAGE^{MAX} to the **D:** drive of the SAGE^{MAX}.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Preparing system for maintenance. Please wait...

Backup Utilities:
key          to do
A           Archive SAGE Files
B           Backup SAGE Database
R           Restore SAGE Database
S           Series Backup
I           Import Name Bindings
V           Verify Disk Integrity
O           Optimize Disk
D           Shell to DOS
Q           Exit Maintenance Shell
Press key for desired action: A
Enter source [drive:]\pathname: C:\GENERAL.LOG
Enter destination [drive:]\pathname: D:\MAR2091.LOG
Archive C:\GENERAL.LOG ? (Y/N): Y
```

Figure 0-2 Archiving SAGE^{MAX} Files

20.2.2 Backup SAGE^{MAX} Database

The **Backup SAGE Database (B)** option of the **Backup Utilities** submenu is used to backup files.

When you select this option, SAGE^{MAX} prompts you to enter the destination drive and pathname where you want to save the backup files. Typically, you will backup your SAGE^{MAX} database on one or more 1.44M high density diskettes.

After you specify a valid destination drive and pathname, SAGE^{MAX} displays a series of prompts that ask if you want to include certain components of the SAGE^{MAX} database. You may include each component in the backup by selecting **Y**. If you do not want particular components backed up, type **N** at the appropriate prompts.

Figure 20-3 shows the process of backing up database components to a diskette in the **A:** drive of the SAGE^{MAX}.

20.2.3 Restore SAGE^{MAX} Database

The **Restore SAGE Database (R)** option of the **Backup Utilities** Submenu is used to restore SAGE^{MAX} database files that were previously backed up.

When you select this option, SAGE^{MAX} prompts you to enter the source drive and pathname of the files you want to restore. Typically, you will restore your SAGE^{MAX} database from one or more 1.44M high density diskettes.

After you specify a valid source drive and pathname, SAGE^{MAX} displays a series of prompts that ask if you want to include certain components of the SAGE^{MAX} database. You may include each component that you want to be restored by selecting **Y**. If you do not want particular components to be restored, type **N** at the appropriate prompts.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07

Enter destination [drive:]\pathname: A:\
Include database files? (Y/N): Y
Include engineering units files? (Y/N): Y
Include groups? (Y/N): N
Include programs? (Y/N): N
Include trends? (Y/N): N

Determining storage requirements...please wait
[ 1] 1.44M disks required for [ 224634] bytes in [ 22] files.
Proceed? (Y/N): Y
Copying File C:\CFG\CLASS.BOB
           to A:\CFG\CLASS.BOB
Copying File C:\CFG\POINT.BOB
           to A:\CFG\POINT.BOB
.
.
.
Copying File C:\EU\PEX.EU
           to A:\EU\PEX.EU

...Press any key to continue:

```

Figure 0-3 Backing up a SAGE^{MAX} Database

Figure 20-4 shows the process of restoring database components from a diskette in the **A:** drive of the SAGE^{MAX}.

```

SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07

Enter source [drive:]\pathname: A:\
Include database files? (Y/N): Y
Include engineering units files? (Y/N): Y
Include groups? (Y/N): N
Include programs? (Y/N): N
Include trends? (Y/N): N
Press any key to begin restore...

Copying File A:\CFG\CLASS.BOB
           to C:\CFG\CLASS.BOB
Copying File A:\CFG\POINT.BOB
           to C:\CFG\POINT.BOB
.
.
.
Copying File A:\SITE\00000.SDF
           to C:\SITE\00000.SDF

...Press any key to continue:

```

Figure 0-4 Restoring a SAGE^{MAX} Database

20.2.4 Series Backup

The **Series Backup (S)** of the **Backup Utilities** Submenu can be used to spread the SAGE^{MAX} database files over multiple floppy diskettes. The Series Backup command informs you ahead of time how many diskettes you will need to complete the backup procedure, and prompts you to insert new disks as needed.

20.2.5 Import Name Bindings

The **Import Name Bindings (I)** option of the **Backup Utilities** Submenu is used to load and convert externally created name bindings text files (**.NBF**) to their appropriate **.BOB** files for the SAGE^{MAX} database. For information on the structure of name binding files, refer to **Appendix J : Name Binding Files**.

When you select this option, SAGE^{MAX} prompts you to enter the name bindings filename. From this prompt you enter the name of the ASCII text **.NBF** file that you want to import to the SAGE^{MAX} database.

Next, SAGE^{MAX} asks if you want to display any name redefinitions. If you select **Y** and the **.NBF** file contains one or more names that already exist in the SAGE^{MAX} database, they will be displayed to the screen.

When the import and conversion is complete, SAGE^{MAX} prompts you to **Press any key to continue:**, after which the **Backup Utilities** Submenu is displayed.

20.2.6 Verify Disk Integrity

The **Verify Disk Integrity (V)** option of the **Backup Utilities** Submenu is a hard drive diagnostic test that checks for hard disk problems such as lost clusters.

When you select this option, SAGE^{MAX} displays the hard drive's volume name, the date the volume was created, any diagnostic messages, and hard drive file statistics.

A sample disk verification is illustrated in **Figure 20-5**.

20.2.7 Optimize Disk

The **Optimize Disk (O)** option of the **Backup Utilities** Submenu is a feature that re-organizes and compresses the data on your hard drive so it is accessible in the most efficient manner.

IMPORTANT

Before you attempt to optimize, you should back up your hard disk.

When you select this option, SAGE^{MAX} displays a message warning you to backup your hard disk before compression.

```
SAGE MAX Banner Line Text                               Mon 23-Sep-96 12:00:00
Opr: SYSTEM                                           Port: 07
Preparing system for maintenance. Please wait...

Backup Utilities:
key          to do
A           Archive SAGE Files
B           Backup SAGE Database
R           Restore SAGE Database
S           Series Backup
I           Import Name Bindings
V           Verify Disk Integrity
O           Optimize Disk
D           Shell to DOS
Q           Exit Maintenance Shell
Press key for desired action: V
Volume SAGE100      created Jan 31, 1991 12:48p

33419264 bytes total disk space
  55296 bytes in 4 hidden files
  32768 bytes in 16 directories
 1171456 bytes in 121 user files
32159744 bytes available on disk

 655360 bytes total memory
 573824 bytes free

...Press any key to continue:
```

Figure 0-5 Verify Disk Integrity

Pressing any key starts the disk optimization compression. During the process, several screens are displayed showing the progress of the optimization. When completed, disk statistics are displayed and the **Backup Utilities** Submenu is displayed.

20.2.8 Shell to DOS

The **Shell to DOS (D)** option of the **Backup Utilities** Submenu enables you to use DOS commands (copyfile, del, etc.) from a DOS prompt. Type exit to return to the **Backup Utilities** Submenu.

20.2.9 Exit Maintenance Shell

The **Exit Maintenance Shell (Q)** option of the **Backup Utilities** Submenu is how you exit and return the SAGE^{MAX} to normal operation.

When this option is selected, the SAGE^{MAX} is rebooted and you are prompted with the Sign-on Menu.

20.3 SAGE^{MAX} Rebuild Procedure

If possible, make a complete up-to-date backup of the SAGE^{MAX} database before starting the rebuilding procedure.

To rebuild your SAGE^{MAX} hard drive, place the SAGE^{MAX} Builder disk in the SAGE^{MAX} floppy drive and reboot the SAGE^{MAX}. You will be presented with the following options:

Select Build Function to Perform

1. Select Hard Drive Type
2. Partition Hard Drive
3. Format Hard Drive
4. Copy SAGE Files to Hard Drive

NOTE

You may choose any of the 4 selections, but you must always follow the choices in order, and you must always complete the sequence. For example, if you start with step 1, you must follow with steps 2, 3, and 4. If you start with step 2, you must follow with steps 3 and 4. If you start with step 3, you must follow with step 4.

20.3.1 Select Hard Drive Type

Answer all of the prompts that ask for information about the hard drive: number of cylinders, sectors, and heads. The SAGE^{MAX} will then reboot and start the building process again.

20.3.2 Partition Hard Drive

You will receive a confirmation warning, telling you that this action will erase all information on the hard drive. Proceed with partitioning the hard drive. The SAGE^{MAX} will then reboot and start the building process again.

20.3.3 Format Hard Drive

You will receive a confirmation warning, telling you that this action will erase all information on the hard drive. Proceed with formatting the hard drive. The SAGE^{MAX} will then reboot and start the building process again.

20.3.4 Copy SAGE Files to Hard Drive

Answer the prompt that asks what kind of network card you have. SAGE^{MAX} will copy all of the files from the SAGE^{MAX} Builder disk in the floppy drive. When this is finished, remove the disk from the floppy drive and reboot the SAGE^{MAX} floppy drive.

20.4 The Auxiliary Disk

The SAGE^{MAX} auxiliary disk contains utility programs that allow you to create/edit a SAGE^{MAX} database, generate a report, compile an SPL program, and convert a STAR database to a SAGE^{MAX} database, all offline on a PC.

20.4.1 XNBF.EXE and INBF.EXE

To create or edit a SAGE^{MAX} database offline, there are two conversion utilities that you may need. These utilities are *XNBF.EXE* and *INBF.EXE*. These utilities are used to convert database object files (the files that contain information about points, programs, variables and globals) between two formats.

The *binary object file* is a database object file that is used internally by the SAGE^{MAX}. There is a binary object file for each type of database object that you create (e.g., points, programs, variables, globals). These files have the extension **.BOB**, and cannot be edited in their binary form.

The *name bindings file* is a database object file that is used when you want to create or edit database objects offline on a PC. You must create a separate name bindings file for each type of database object that you want to create/edit. These ASCII text files have the extension **.NBF**, and cannot be used by the SAGE^{MAX} in their ASCII format.

The INBF.EXE utility is used to convert **NBFs** to **BOB** files. You use this utility to convert ASCII database files that you create/edit offline on a PC to their binary equivalents for use on the SAGE^{MAX}.

The XNBF.EXE utility is used to convert **BOB** files to **NBFs**. You use this utility convert binary object files to their ASCII equivalents so that they can be edited offline on a PC using a standard text editor.

The conversion process between **BOBs** and **NBFs** is illustrated in **Figure 20-6**.

The XNBF utility program (**XNBF.EXE**) is used to export Name Bindings Files (NBFs) by reading and translating Binary Object (**.BOB**) files. NBFs are editable text files containing text representations of the objects which exist in a binary form within the **.BOB** files.

The XNBF program can create the NBF on any valid DOS path. XNBF can read **.BOB** files from any directory, but all **.BOB** files must reside in the same directory. The XNBF program is executed from the DOS command line using one of the three possible formats shown below.

```
>XNBF bobdir nbfp  
>XNBF bobdir nbfp /switch ... /switch  
>XNBF bobdir nbfp -switch ... -switch
```

In the above formats, **bobdir** is the directory where the **.BOB** files can be found. You do not have to specify a “/” character at the end.

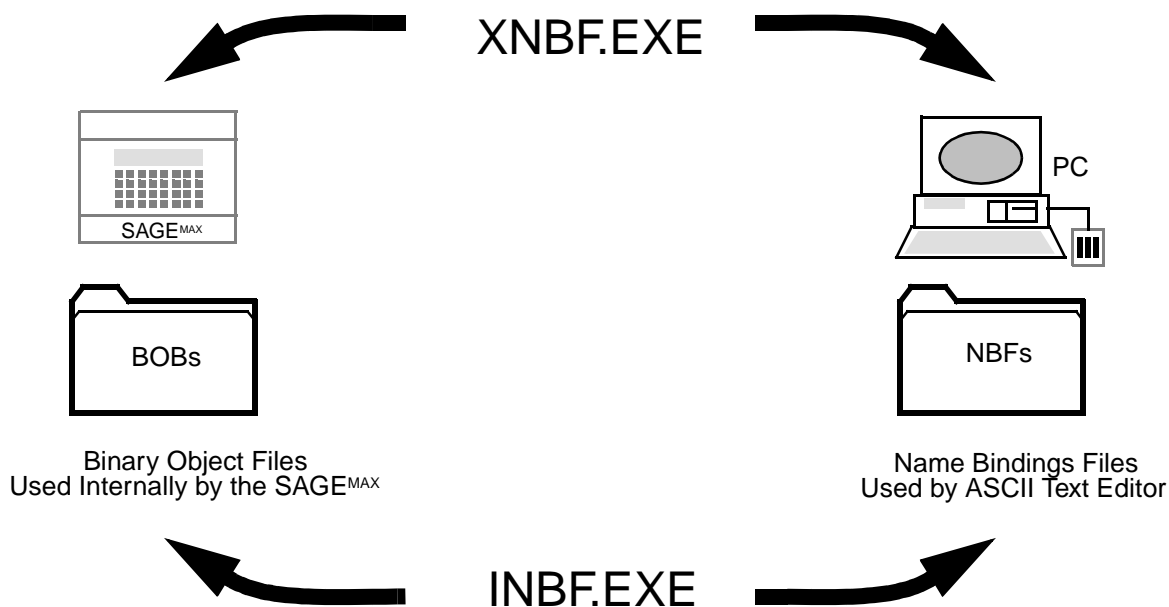


Figure 0-6 Conversion between BOBs and NBFs

The **nbpath** path is a full pathname with may include an extension, however, the extension is ignored and **.NBF** is added.

The switches allow you to specify certain object types. If no switches are specified, then all of the **.BOB** files which XNBF understands will be searched for in the directory **bobdir**. Valid switches are **PT** for points, **VR** for variables, **PG** for programs and **GL** for globals.

Typically, you use the XNBF program to translate **.BOB** files to NBFs so that you can edit your SAGE^{MAX} database *offline* at your PC using a standard text editor and the formats for Name Bindings Files (refer to **Appendix J**). Next, you convert the NBFs back to **.BOB** files and transfer them back to the SAGE^{MAX}.

Offline database editing/creation is particularly useful if, for example, you want to duplicate a SAGE^{MAX} database on a second SAGE^{MAX}. Refer to **Figure 20-7** and **Figure 20-8**. If, for example, the names of the database objects on the primary SAGE^{MAX} are **SAGE1_OAT**, **SAGE1_MX2FE00**, etc., (because of a chosen naming convention that uses the prefix **SAGE1_** before all database objects), you can easily perform a “find and replace” command that will change all occurrences of **SAGE1_** to **SAGE2_**. This feature is available in most text editors and greatly simplifies the process of creating a “duplicate” database.

The function of the XNBF program can also be performed from the Backup Utilities Menu of the SAGE^{MAX} after you Prepare System for Maintenance. This method is preferred since it shuts down the SAGE^{MAX} while you manipulate database object files.

The INBF utility program (**INBF.EXE**) is used to import Name Bindings Files (NBFs) and translate them into Binary Object (**.BOB**) files. NBFs are editable text files containing text representations of of the objects which exist in a binary form within the **.BOB** files.

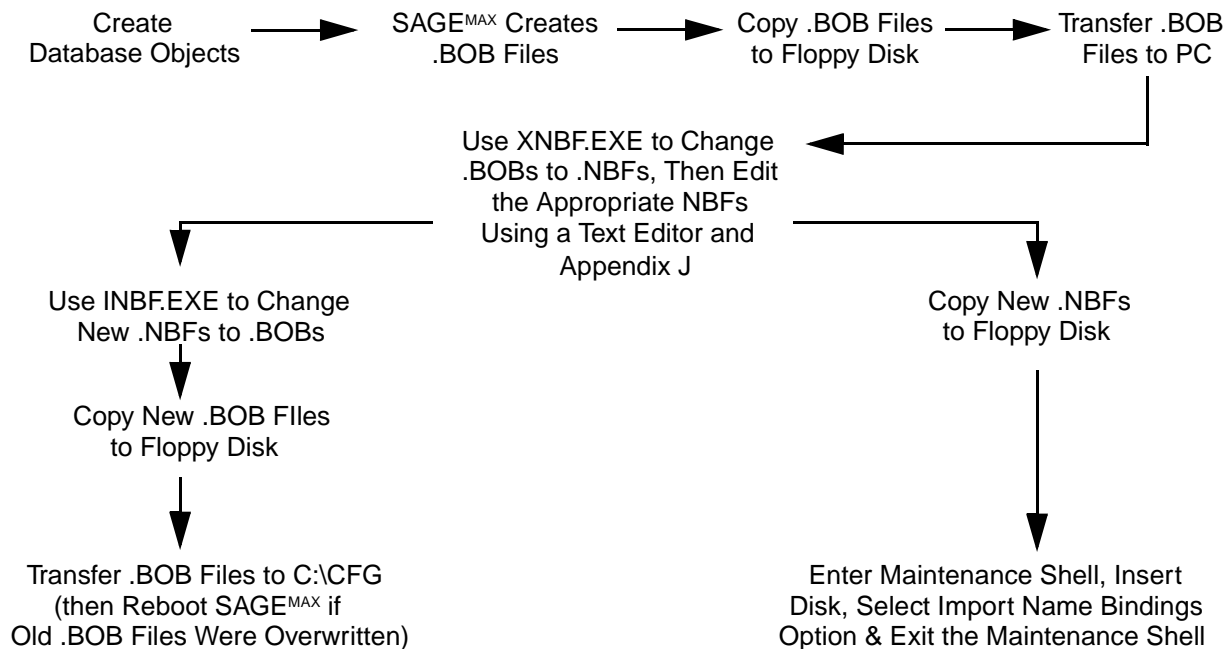


Figure 0-7 Offline Database Editing Process

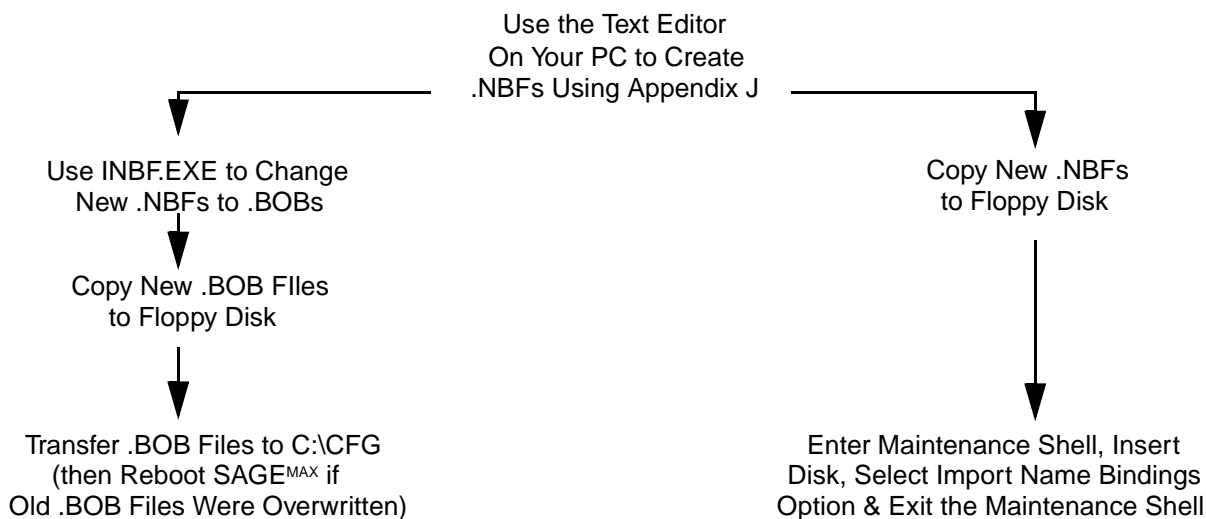


Figure 0-8 Offline Database Creation Process

The INBF program can read the input NBF from any valid DOS path. The **.BOB** files are assumed to exist on the path **C:\CFG**. The INBF program is executed from the DOS command line using one of the two formats shown below.

>INBF pathname

>INBF pathname U

If any given **.BOB** file does not exist, then it is created automatically by INBF. If the pathname is followed by **U**, the INBF will check each name for uniqueness, and report any redefinitions which are encountered. Otherwise, the default is to simply append the new definition onto the end of the **.BOB** file.

CAUTION

Do not use INBF without the U option if the .BOB file already exists.

20.4.2 SPLC.EXE

The SPLC program is an offline compiler that can be used in conjunction with an ASCII text editor to create SPL programs on a PC. You should understand the Application Programming Subsystem and SAGE^{MAX} Programming Language (SPL) before using this utility. Refer to **Chapter 8: SAGE^{MAX} Menu Operations** and **Chapter 11: Programming** for more information.

The SPLC program is executed from the DOS command line using one of the formats shown below.

>SPLC

>SPLC \dir\source.SPL,PLBfrag

>SPLC \dir\source.SPL,PLBfrag /NL

>SPLC \dir\source.SPL,NUL

>SPLC \dir\source.SPL,NUL /NL

If you enter **SPLC** from the DOS command line, the program prompts you for all the necessary information.

First you are prompted to **Type the SPL Source Pathname:**. This is the name of the ASCII text file that contains the source code that you want to compile. You may specify a file fragment if the source file does not live on the root directory of your PC, although this is not required. The source filename must have the extension **.SPL**.

Next, SAGE^{MAX} asks you to **Type SPL PLB Fragment:**. From this prompt you enter the name of the Program Logic Block (PLB) that you want to create. The name that you enter may be a file fragment, but must be located on the **\LOGIC** subdirectory of your PC (e.g., the PLB filename **\LOGIC\BLDG1\OSS.PLB** is represented using the file fragment **\BLDG1\OSS**).

If you specify the PLB name **NUL**, SPLC does not create a PLB. This option can be used if you want to see if a source program compiles correctly without generating a PLB file.

Next, SAGE^{MAX} displays the prompt **Type /NL for no list or press Enter:**. Normally, a list file is generated when you compile an SPL program. If you do not want a list file to be created, type **/NL** as directed at the prompt. Typing the **ENTER** key causes the usual list file to be generated (i.e., **sourcename.LST**).

At this point, SPLC compiles the source file that you specified. The compile results are displayed in the same way that they are displayed when you compile a program on the SAGE^{MAX} (refer to **Chapter 8: SAGE^{MAX} Menu Operations**). If errors are encountered, you can view the **.LST** file and make changes in the **.SPL** file using a standard ASCII text editor before re-compiling.

You can avoid the SPLC prompts by entering a full command line when you use the SPLC utility.

The general format is the **SPLC** program name followed by a space, then the source filename (including any file fragments and using the **.SPL** extension). A comma (,) must follow the source filename. Next, the PLB file fragment is included. The PLB fragment is assumed to be located on the subdirectory **\LOGIC** and is assumed to have the extension **.PLB**. Do not include **\LOGIC** and **.PLB** in the PLB file fragment.

If you do not want to create a PLB (i.e., you simply want to see if the program compiles correctly), you must use the dummy PLB filename fragment **NUL**.

Normally, a list file is generated when you compile an SPL program. If you do not want a list file to be created, type the switch **/NL** (note the initial space) following the PLB file fragment. Omitting this switch causes the usual list file to be generated (i.e., **sourcename.LST**).

After you compile your SPL programs offline, you will eventually want to transfer them back to the SAGE^{MAX} and make them part of your SAGE^{MAX} database.

The transfer portion of this process involves copying three files from your PC to a diskette, and then copying the files from the diskette to the proper directories of the SAGE^{MAX}.

The program source file must be copied to the **\SPL** subdirectory of the SAGE^{MAX}. Also, it must have the extension **.SPL**.

If you created a list file (i.e., did not use the **/NL** option), you may want to copy it to the **\SPL** subdirectory. This is not required. List files have the extension **.LST**.

The PLB must be copied to the **\LOGIC** subdirectory of the SAGE^{MAX}. This process cannot be completed if you used the dummy filename **NUL** for your PLB. The PLB filename should include any file fragments that you specified and must have the extension **.PLB**.

After you transfer the necessary files to the SAGE^{MAX}, you must create a program object and link it to the new PLB. This is accomplished by creating a point object within your database and then specifying the PLB filename that you created (with any fragments) as the new program's PLB.

20.4.3 *REPORT.EXE*

The REPORT utility is an offline program that is used to merge a text file (**.TXT**) with a file containing data values to produce a final report (**.RPT**). This report file may be printed or saved for later reference.

The input text file identifies locations to be filled in with data, by including special token placeholders enclosed with percent signs, e.g., **%token%**. The resulting output report file is also a text file, but has the extension **.RPT**. For a complete list of tokens that are recognized by the REPORT utility, refer to **Table 8-1**.

The REPORT program is called from the DOS command line using the following statement.

```
>REPORT txtpath datapath reportpath
```

If no extensions are specified with text, data and report file path names, then the extensions **.TXT**, **.TBL** and **.RPT** are added.

The *datapath* file may be one of three different types: comma separated value text, table (**.TBL**) or trend (**.TRN**).

Comma Separated Value text files contain lines of text with values separated by commas. When these files are used in reports, the text value is copied exactly as it appears between the single or double quotes.

Table files are produced either by SPL programs or from the **Create/Edit Table** menu selection. Tables can be full (a data type for each value) or sparse (a single data type for all the values of the table).

Trend files are produced by SAGE^{MAX} trends and use a special binary data format.

For more information on the components of reports, refer to the Report JOB portion of **Chapter 11: Programming**.

20.4.4 *S2SAGEPC.EXE*

The S2SAGEPC.EXE utility is used to convert STAR databases to SAGE^{MAX} databases. This conversion process is also a built-in feature of SAGE^{MAX} jobs, and is explained in detail in **Chapter 8-8: The Job Scheduler Submenu**.

20.5 User Privileges

Each user of the SAGE^{MAX} must be assigned a set of user privileges. These privileges are represented internally in a 32-bit bitmap that is unique to every user. The 32 bits are divided into three groups: Menu Privileges, User Object Safe Pattern and a third group that is reserved for future use.

Bits 0-15 of the most significant word of the bitmap represent Menu Privileges.

Bits 0-7 of the least significant word of the bitmap represent the User Object Safe Pattern.

Bits 8-15 of the least significant word are reserved for future use.

Menu Privileges provide users access to groups of menus which are arranged functionally in a “non-level-oriented” manner. In order to have access to (and for that matter, even be able to see) a menu selection, a user’s corresponding menu privilege bit must be set.

Access to menus that require perceived *low-level* privileges is not guaranteed for users with perceived *high-level* privileges (e.g., a user with administrative privilege, which are perceived as high-level, is not automatically granted all operation privileges, which are perceived as low-level privileges).

Menu penetration to certain menu levels may be permitted for multiple privileges. For example, one user may be able to access the **Database Functions** Submenu from the **Main** Menu in order to Create, Modify, Erase *and* List SAGE^{MAX} objects, while a second user with different privileges may access the **Database Functions** *only* for the purpose of Listing objects.

SAGE^{MAX} user privileges are divided into nine functional groups. These groups are:

- Everyone
- Operator
- Administrator
- Object Modifier
- Scheduler
- Installer
- File Access
- File Expert
- User Administrator

The first three groups (Everyone, Operator and Administrator) are *base set* Menu Privileges. Each of these groups defines a basic collection of menu options and features that are available to that group. The remaining six groups (Object Modifier, Scheduler, Installer, File Access, File Expert and User Administrator) are optional features that provide specialized privileges which can be added to any of the base set privileges.

Using a menu access scheme of user privileges (rather than simply a *privilege level* scheme) permits access to menu functions based on the user's job. Typical users include, but are not limited to the following:

- an installer
- a casual operator
- a file expert operator
- a file expert, user administrator
- an operator/user administrator with scheduling and modify capabilities
- a basic user (everyone)

Before you can plan users, you must have an understanding of what is offered through the Menu Privilege groups.

The *Everyone* base set privilege allows very limited access to the SAGE^{MAX} Menuing System. The only menu options available are:

- Alarm List (All Alarms, Unacknowledged Alarms and Alarms by Class Name only)
- Broadcast Message (except on the Ethernet port)
- Quit

The *Operator* base set privilege allows access to menu options that are considered typical *operator* functions. Operators have access to the following menus:

- Alarm List (All Alarms, Unacknowledged Alarms, Alarms by Class Name and alarm acknowledgment only)
- Broadcast Message (on all ports)
- Database Functions (List Objects only)
- Find Objects (Find Objects only)
- Monitor/Change Point Attributes (Monitor only)
- Table Menu (List Objects only)
- Ports Status and Setup (Watch Dynamic Port Status and Watch Driver Status only)
- Trend (List Trend Objects, Numerical Display and Stripchart Display only)
- Virtual Terminal
- Quit

The *Administrator* base set privilege allows access to menu options that are considered typical *administrator* functions. Administrators have access to the following menus:

- Alarm List (All Alarms, Unacknowledged Alarms, Alarms by Class Name, Archive Alarms and Delete Alarms only)
- Calendar Edit
- Database Functions (all functions except Create User, Erase User and List Users)
- Find Objects (all functions except Find and Stuff)
- Monitor/Change Point Attributes (Monitor only)
- Job Scheduler (Import/Export Database Files and STAR/SAC to SAGE^{MAX} Translations)
- System Variables
- Trend (all functions)
- Utility Functions (Directory, Make Directory, Delete Directory, Display File as Hex/ASCII, Format Diskette, Prepare System for Maintenance, Type File, Search for File, Math Calculator, Edit File, Copy File, Delete File and Remote Copy of certain files)
- Table Menu (List, List to File, and Edit)
- Quit

NOTE

Some of the available functions of the three base set Menu Privileges overlap.

Every user of the SAGE^{MAX} is assigned a base set privilege. In addition, it is possible to assign both Operator and Administrator base set privileges to the same user. This gives the user access to both operator and administrator functions.

There are six additional *special privileges* that can be assigned to any base privilege user. These privileges can be added in any combination to supply additional privileges to the user.

The *Scheduler* privilege adds *all* Job Scheduler functions to the user's group of privileges.

The *Installer* privilege adds *all* Ports Status and Setup functions to the user's group of privileges.

The *File Access* privilege adds the following list of Utility Functions to the user's group of privileges:

- Directory
- Edit File
- Copy File
- Delete File
- Remote Copy

For users with the File Access privilege, the features listed above only apply to the following directories on the SAGE^{MAX}:

- **D:**(all files)
- **C:\CFG** (all **.BOB** files except **USER.BOB**)
- **C:\EXE** (all files)
- **C:\GROUPS** (all files)
- **C:\INIT** (all files)
- **C:\LOGIC** (all files)
- **C:\LOG** (all files)
- **C:\REF** (all files)
- **C:\SPL** (all files)
- **C:\TABLES** (all files)
- **C:\TREND** (all files)

The *File Expert* privilege is broader than the File Access privilege. Functions are not limited to specific directories. The File Expert privilege adds the same five features but their capabilities extend to the entire **D:** and **C:** drives of the SAGE^{MAX}, including *all* directories and subdirectories.

The *User Administrator* privilege adds the following Database Functions and Utility Functions that relate to User objects:

- Database Functions (Create Users, List Users and Erase Users)
- Utility Functions (Directory, Edit, Copy, Delete and Remote Copy of the file **C:\CFG\USER.BOB**)

The *Object Modifier* privilege allows certain objects to be changed under certain conditions. Users with Object Modifier privileges have the following additional capabilities:

- Monitor/Change Point Attributes (with some restrictions)
- Job Scheduler (Data Capture/Stuff with some restrictions)
- Find Objects (Find and Stuff with some restrictions)

In your database, it may be the case that you define several users with the Object Modifier privilege. This gives each of these users modification capabilities for *every* point, program and variable in the database. Although this may be the desired effect, SAGE^{MAX} also gives you the ability to further restrict the modification scope of each of those users.

For example, assume that USER1-USER9 are Object Modifiers, each of whom is responsible for maintaining a single floor of a building. SAGE^{MAX} has the ability to restrict the modification privilege of each user to points/programs/variables that are specific to his/her respective area.

These Object Modifier restrictions are based on *object privileges*. User Object Privilege Patterns are 8-bit patterns that are definable when the user is created/modified. Similarly, an *Object Privilege Pattern* must also be defined when you create points, programs and variables.

The User Object Privilege Pattern allows users with modify privileges to have different levels of modification abilities.

The ability to modify an object is determined through a binary operation of the Privilege Pattern of the object (i.e., the point, program or variable) and the Privilege Pattern of the user attempting to modify the object. If the result of logically ANDing the User and Object Privilege Patterns is the same as the Object Privilege Pattern in question, the user is permitted to change the value of the object.

NOTE

If a database object (i.e., point, program or variable) has an Object Privilege Pattern of all zeros, then any user with the Object Modifier privilege is permitted to modify the object's value.

The User Privilege Pattern has an additional function. It is also used to determine whether or not a particular user is permitted to acknowledge alarms that originate from a given point or program. The determination of alarm acknowledgment ability is similar to the process used to determine modification ability. The determination of acknowledgment is made using the Object Privilege Pattern of the *alarm class*. This pattern is logically ANDed with the User Privilege Pattern. The result of this operation determines if the user *is* or *is not* permitted to acknowledge the alarm.

With an understanding of the privileges available to users, who will be using the system and what privileges should be made available to each user, you can plan your users by using an *information table* that contains all necessary information until you are ready to enter the information into your database (see **Table 7-6**). These user information tables will also be useful when you plan the Object Privilege Patterns of points, programs, variables and classes.

For each user you plan to create, you must have a username, an associated codeword, the required *base set* privilege, any optional privileges, a user timeout and a Privilege Pattern (if required) for object modification and alarm acknowledgment. This information should be accumulated in an information table as shown in **Table 7-6**.

The username consists of up to 24 characters and identifies the user. You enter the username from the Sign-on Screen of the SAGE^{MAX}.

The codeword consists of up to 12 characters and is a password that is unique to each username. The codeword is entered after you enter the username to gain access to the system.

The base set privileges and the optional *special* privileges are stored as a 16-bit Menu Privilege Pattern which can be modified when you create/modify the user. Currently, only 8 bits in the 16-bit Menu Privilege Pattern are used.

For users with multiple privileges, you simply set the appropriate menu privilege bit numbers (0-16) corresponding to the desired privileges. The Menu Privilege bits default to zeros. **Figure 7-7** shows the User's Menu Privilege Bitmap.

The User Object Privilege is stored as an 8-bit pattern which can also be changed when you create/modify the user.

Each user must be assigned a User Object Privilege Pattern which ranges from 00000000B to 11111111B (0-255 decimal). As previously explained, this pattern is used in conjunction with the privilege pattern of database objects to test whether or not a user (who is an Object Modifier) is permitted to modify a particular database object (i.e., a point, program or variable).

NOTE

If a user is an Object Modifier and has an Object Privilege Pattern of 11111111B, then the user is able to make modifications to all points/programs/variables, regardless of their Object Privilege Patterns.

The User Object Privilege Pattern is also used to test whether or not a user is permitted to acknowledge alarms of certain classes (refer to **Chapter 7.6.7: Planning Classes**). Therefore, you should plan User Object Privilege Patterns very carefully.

When planning a User Object Privilege Pattern, it is important to:

- decide that you (1) want to selectively limit a user's modification capabilities to certain points, programs and/or variables, and/or (2) want to allow alarm acknowledgment capabilities for this user on an object-by-object basis
- group the database objects based on areas where you desire selective modification (e.g., between buildings, floors, zones, wings, etc.)
- assign users to the appropriate areas
- specify Object Privileges for each user, based on modify and acknowledge patterns that you define
- define appropriate Object Privilege Patterns for each point, program, variable and class in each group you defined.

NOTE

Due to their interrelationship, you may choose to plan User Privilege Patterns and the Object Privilege Patterns of points, programs, variables and classes simultaneously, after you have gathered the other pieces of information about these database objects. In general, this may simplify the process of planning Privilege Patterns.

APPENDIX A - GLOSSARY

This Appendix defines words used in this manual.

acknowledge is a term that refers to the manual operation of removing an alarm or event transaction from the unacknowledged alarm list.

ANDing is an operation involving two binary numbers in which the result of the operation equals 1 if both binary numbers equal 1 (i.e., **1 AND 1** equals 1), and equals 0 for all other cases (i.e., **1 AND 0** equals 0, **0 AND 1** equals 0 and **0 AND 0** equals 0).

ASCII American Standard Code for Information Interchange is a standard which is used to represent text characters and control characters inside a computer system. ASCII codes range from 000-127.

ASCII text file is a disk-resident file of information containing only ASCII character codes. These files are structured in “lines” which end with carriage return (0Dh), line feed (0Ah) pairs. If the file contains a Control-Z (1Ah), then it is assumed to be an end-of-file marker.

asynchronous refers to unsynchronized events, such as in asynchronous transmission, in which the time intervals between transmissions are not necessarily the same. Asynchronous transmission of data involves the transmission of discrete packets or *bytes* of information. Each discrete packet of information contains start and stop bits, and the time between these packets may be uneven.

attribute is a two-character name that represents a property of an object and forms the interface between the object and users of the object.

AUI (Adapter Unit Interface) is a type of cable connector used for Ethernet network connections. (See *DB15*)

averaging is a method of gathering trended information (unlike snapshot) in which the data is gathered every minute during the sample interval, averaged, and recorded as a single value for that sample interval.

AWG stands for American Wire Gage, a series of wire thicknesses and diameters established as a standard in the United States. It is recommended that the SAGE^{MAX} use 18-22 AWG twisted shielded pair for EIA-485 communication lines.

baseband is a network electrical connection scheme that limits the cable to carrying only one channel of information (contrast with *broadband*). Baseband local area network (LAN) cables (such as those used in Ethernet networks) carry data as digital signals.

baud rate refers to the speed (measured in bits per second) at which serial information is transmitted and received. 2400 baud means 2400 bits per second while 10 M baud (mega baud) means 10,000,000 bits per second.

biasing resistance is a resistance that is used to maintain proper line voltage levels on an

EIA-485 network. The SAGE^{MAX} allows biasing resistors to be switched into the circuit to hold the + and - lines at their proper levels. If more than 2 driver devices on a 5000-foot multidrop have biasing resistances in place, the resulting bias of the network may become too great for proper switching of the transceivers, and may cause communication failures.

bit is a term that refers to a single binary digit, i.e., 0 or 1. A bit is the smallest unit of storage in a binary computer.

BNC is a type of connector (used with coaxial cable) that is cylindrical with two short pins on the outer edge and on opposite sides. The outer plug is turned to secure the connection.

boot is a term which refers to the process of turning on the SAGE^{MAX} and loading/running its operating software.

bridge is a component of a local area network that is used to connect two identical networks and filter communications between them.

broadband is a network communications scheme in which the network cable can carry many channels of information (e.g., a cable television hookup carries many TV channels). Broadband LAN cables (contrast with *baseband*) can be used to carry video and voice information as well as computer network information.

byte is a term that refers to a number consisting of eight binary digits or bits ranging from 00000000 to 11111111. A single character of information (e.g., a letter, a number, a punctuation mark, etc.) is stored as a byte.

CC/485 is an EIA-232D-to-EIA-485 communications converter that can be configured for full- or half-duplex communication.

CC/PC is an IBM PC ISA bus-compatible interface card which provides serial network-to-EIA-485 conversion.

Centronics interface is a standard 36-pin parallel interface used for connecting a printer to the SAGE^{MAX}.

checksum is a numeric code that is used in communications for error checking purposes.

circular trend is a trend of fixed length that begins sampling data whenever it is started, and continues sampling data until it is either stopped or the number of samples collected reaches the maximum number of samples defined for the trend. When the number of samples collected reaches the maximum allowable for the trend, that sample is written into the first sample slot. In this way, circular trends act like a “sliding window” of data that has been sampled over some period of time. This period of time is the maximum number of samples that you specify when you create the trend.

coaxial cable is a type of cable that can be used for Ethernet networks. Coaxial cable consists of a central metal conductor covered by two layers of insulator with a grounding shield between them. Coaxial cable is usually more expensive than twisted pair cable, but offers less signal loss and higher noise immunity.

coercion refers to the SAGE^{MAX} ability to perform internal transformations of data types to permit certain mixed-mode arithmetic operations.

COM1 (or COMmunications port 1) is a label that represents serial port 5 of the SAGE^{MAX}. COM1 is reserved for the internal modem of the SAGE^{MAX}.

COM2 (or COMmunications port 2) is a label that represents serial port 6 of the SAGE^{MAX}. COM2 is reserved for an optional EIA-232D card or a second internal modem.

comments are lines used in SAGE^{MAX} Programming Language (SPL) programs to document programs. Comment lines must begin with a semicolon (;) in the leftmost column of the program logic.

compiler is a SAGE^{MAX} software program that is used to convert SAGE^{MAX} Programming Language (SPL) programs into machine language format so they can be executed by the SAGE^{MAX}.

constant is a type of term used in SPL expressions. Constants specify particular unique values implying a data type by their use.

CTS stands for Clear To Send which is pin number 5 of a standard EIA-232D DB25 connector or pin number 8 of a DB9 connector.

current sample refers to the latest sample record number that was taken by a trend.

daisy chain refers to a network configuration in which the network devices are connected linearly (one after another). Communication signals are transmitted over the network and each device hears them at the same time.

DB9 is a 9-pin connector typically used in PCs for a subset of the EIA-232D standard.

DB15 is a 15-pin connector typically used in PCs for a subset of the EIA-232D standard.

DB25 is a 25-pin connector for the EIA-232D standard.

DCD stands for Data Carrier Detect which is pin number 8 of a standard EIA-232D DB25 connector or pin number 1 of a DB9 connector.

delimit means the surrounding, enclosing or following of a discrete object with a unique character in helping to define its boundaries more clearly. The text string **Occupied** is delimited by backslash (\) characters in the example **\Occupied**.

DIP switch refers to a block of 7 switches on the SAGE^{MAX} that are used to set the termination resistances, biasing resistances and duplex operational mode of the SAGE's EIA-485 ports.

diskette is a removable medium used in a floppy disk drive to store files. The SAGE^{MAX} has a disk drive which uses 3.5" diskettes.

distributed control refers to the ability to control different pieces of equipment locally (e.g., from unit controllers) rather than having one point of all control operations.

dynamic RAM refers to a common type of random access memory architecture. Dynamic RAM is sometimes referred to as *DRAM*.

driver device (EIA-485) refers to specific devices that have the ability to bias the EIA-485 network to which it is connected. *See Biasing Resistance.*

DSR stands for Data Set Ready which is pin number 6 of a standard EIA-232D DB25 and DB9 connectors.

DTR stands for Data Terminal Ready which is pin number 20 of a standard EIA-232D DB25 connector or pin number 4 of a DB9 connector.

Dumb is a driver type used by the SAGE^{MAX} to configure a SAGE^{MAX} port for use with a dumb terminal.

dumb terminal refers to any interactive device that sends and receives ASCII characters and supports carriage return, line feed and non-destructive backspace characters.

duplex *See full-duplex*

EIA-232D (formerly RS232) is a communications standard that uses ± 12 V signals over a 25 pin D-type connector, and is used for data communications between data communications equipment (DCE, such as a modem) and data terminal equipment (DTE, such as a terminal or printer).

EIA-485 (formerly RS485) is a communications standard that uses ± 5 V signals for communications over two-wire or four-wire, multipoint networks using transmitters and receivers.

Ethernet is a baseband high speed local area network (HSLAN) developed by Digital Equipment Corp., Intel Corp. and Xerox Corp. Ethernet networks can transmit data at rates of 10,000,000 bits per second.

embedded computer refers to a single *chip* that contains all the necessary components of a computer.

expansion card is a printed circuit card (e.g., an Ethernet card, an internal modem card, an EIA-232D card, etc.) that plugs into an expansion slot of the SAGE^{MAX} to increase its capabilities.

expansion slot is a socket on the SAGE^{MAX} motherboard that accepts an expansion card.

expression is a symbolic formula used by SPL to represent a chain of arithmetic calculations on data from various sources. Expressions consist of *terms* and *operators*.

expression operators are elements of SPL that are used to link terms in an expression. Expression operators include the standard arithmetic operators (+, -, /, and *) as well as comparison operators (<, >, ==, <>, <=, >=), Boolean operators (AND, OR, XOR, NOT), and others.

FCC (Federal Communications Commission) is a Federal agency that regulates and sets standards for communication by wire and radio.

FiberDrop is an interface device for asynchronous serial, two-wire EIA-485 networks which allows any part of the EIA-485 network to be converted to optical fiber.

fiber optic cable is a type of communications cable that uses light instead of electricity to transfer information. Fiber optic cables have high immunity to noise, but are generally more expensive than coaxial or twisted pair cables.

field panel is a generic term that refers to an MCU, RCU2, STAR or SAGE^{MAX}. Field panels are the intermediate step between unit controllers and host systems.

file is a collection of data that is treated as a single unit on a peripheral device such as a hard disk drive.

floating point arithmetic refers to a method for storing and calculating numbers in which the decimal points do not line up as in fixed point numbers. The significant digits are stored as a number called the mantissa, and the location of the decimal point is stored in a separate unit called the exponent. In SPL, a floating point number can be represented in a fixed-style format (i.e., 12300.00) or in scientific notation (i.e., 12.3E3).

When using the fixed-style format in SPL programs, the #FIXED and #FLOAT compiler directives are used to specify the intended data type of the suspect number.

floppy disk drive is a hardware device on the SAGE^{MAX} that is used to save and retrieve information to/from a diskette. The SAGE^{MAX} has a built-in 3.5" floppy disk drive.

four-wire mode *See full-duplex*

full-duplex is a communications scheme by which transmitting and receiving are done concurrently over two separate communication channels.

functions are arithmetic procedures which operate on one or more arguments and produce a single value result. Functions are used as terms in SPL expressions and include INT, ROUND, ABS, SQRT, MIN, MAX, BETWEEN, and others.

half-duplex is a communications scheme by which transmitting and receiving are done in one direction at a time over one communication channel. (Compare with full-duplex which uses a channel to transmit and another channel to receive.) This requires that transmit and receive hardware be switched between transmit and receive periods.

hard disk is a disk drive that stores information on a rapidly spinning hard platter, which provides fast access and dense storage capabilities. The hard disk on the SAGE^{MAX} can hold 44 megabytes (MB) of information.

host system refers to the high-level component of the AI2100 system. A host system uses PHP (Public Host Protocol) as a communications scheme.

HVAC is an abbreviation for Heating, Ventilation and Air Conditioning.

infinte trend is a trend that begins sampling data whenever it is started, and continues sampling data until it is either stopped or the number of samples collected reaches the maximum number of samples defined for the trend.

INI File refers to the INItial value file, an ASCII text file that is used to set SPL program attributes to initial values prior to program execution. Many programs may share the same INI File.

ISA Bus is a de facto standard for peripheral interface modules which are used with IBM PCs and compatible computers.

job is a SAGE^{MAX} background application program (i.e., report generation, spooling files to be printed, broadcasting messages, performing data capture and data stuff functions, performing file uploading and downloading, and exporting database files). Jobs can be run using SPL and the JOB statement.

K represents 1000, as in 200 K baud, which means 200,000 baud.

kilobyte refers to 1,024 bytes of information storage.

labels are optional symbolic names that begin in the leftmost column of SPL programs. Labels may contain up to 8 characters and/or numbers, may be followed by an optional colon (:), and must be separated from an expression by one or more **TAB**s or spaces.

LAN is an acronym for Local Area Network, a system that allows communication between computers or similar devices within a limited geographic area.

LED is an abbreviation for Light-Emitting Diode, a semiconductor diode that converts electric energy into visible electromagnetic radiation.

LPT1 is a SAGE^{MAX} parallel port (port 13) that is reserved exclusively for a parallel printer.

magnetic isolation is a term that describes a circuit which uses a transformer-coupled DC-DC power supply to power communications devices so that they are magnetically isolated from the computer which uses them. This is one method by which a device may be protected from electrical transients.

max samples is the largest number of samples that can be recorded for the trend.

megabyte is 1,048,576 bytes (1,024 * 1,024) of information storage.

mixed-mode arithmetic refers to the SAGE^{MAX} ability to perform arithmetic operations using expressions that have different data types, e.g., integer and float.

modem is a communications device that converts digital data signals into modulating tones so that they can be transmitted over telephone lines, and then converted back into digital signals so that they can be understood by a digital device.

multidrop configuration refers to a network configuration in which devices are connected across a twisted pair of wires, like rungs of a ladder.

multitasking is a software technique which allows a microprocessor to perform several tasks at the same time.

named constant is a type of term used in SPL expressions. Named constants have reserved names that are recognized by the SPL compiler and can be used anywhere a constant can be used. Some SPL named constants are PI, MONTH, YEAR and TIME.

named object attributes are properties or parameters of database objects and can be used as terms in SPL expressions.

nested expressions are SPL expressions that are within another expression. Nested expressions are contained in parentheses and may be up to six levels deep. Evaluation of nested expressions is done from the innermost set of parentheses and works outward.

network is a physical communications line that connects two or more devices that (usually) share the same protocol.

noise refers to any type of electrical interference, usually caused by motors, fluorescent lights, etc., which can be reduced by using fiber optic, coaxial, or shielded twisted pair cables.

null modem is a device that is used to switch transmit, receive and handshake signals of an EIA-232D DB25 connection.

object-oriented database is a collection of items that is organized so that information about the items can be examined and modified in the same basic way. In an object-oriented database, all control points are represented by named objects.

operating system is a set of programs and routines which guides a computer in the performance of its tasks, routes requests, assists the programs (and the programmers) with supporting functions and increases the usefulness of the computer's hardware. One of the jobs of the operating system is to coordinate the multitasking of the SAGE^{MAX}.

operator (1) refers to user who has signed on to the system.

operator (2) refers to an element of SPL that is used to link terms in an expression. Operators include the standard arithmetic operators (+, -, /, and *) as well as comparison operators (<, >, ==, <>, <=, >=), Boolean operators (AND, OR, XOR, NOT), and others.

optical isolation is a term that describes the function of an optoisolator, a coupling device in which an LED, which is energized by an input signal, is optically coupled with a photodetector to provide two coupled circuits that are electrically isolated. This is one method by which a device may be protected from electrical transients.

parallel port is a communications channel through which information is sent/received using eight parallel data wires, enabling eight signals (or eight bits) to be sent/received at any instant. These eight signals are interpreted as a single data byte.

PC is an abbreviation for personal computer. Typically this computer has a microprocessor of the 80x86 family of microprocessors.

peer refers to an element of a network in which each element has equal status and equal network capabilities without having to use an intermediate controlling element.

Peernet is a type of network used by STAR and SAGE^{MAX} field panels to share information. Peernet also refers to the driver type used by the SAGE^{MAX} to configure a SAGE^{MAX} port for such a network.

Peernet Protocol is a proprietary communications scheme that was developed by American Auto-Matrix and is used by STAR field panels.

PHP refers to Public Host Protocol, a communications protocol in the public domain. Public Host Protocol was developed by American Auto-Matrix.

PHPdcon (or PHP Slave direct connect) is a driver type used by the SAGE^{MAX} to configure a SAGE^{MAX} port for connection to a PHP network of slave devices. In a PHPdcon configuration, the SAGE^{MAX} acts as a slave on the PHP network with SPECTRA, for example, acting as the PHP network host.

PHPdial (or PHP Slave dial-up) is a driver type used by the SAGE^{MAX} to configure a SAGE^{MAX} port (typically the modem port -- port 5) for dial-up capabilities. In a PHPdial configuration, the SAGE^{MAX} acts as a slave on the PHP network with SPECTRA DUX or SPECTRA MouseView Professional Extension acting as the PHP dial-up host.

PHPHdcon (or PHP Host direct connect) is a driver type used by the SAGE^{MAX} to configure a SAGE^{MAX} port for connection to a PHP network of slave devices. In a PHPHdcon configuration, the SAGE^{MAX} acts as the host.

PHPHdial (or PHP Host dial-up) is a driver type used by the SAGE^{MAX} to configure a SAGE^{MAX} port (typically the modem port -- port 5) for dial-up capabilities. In a PHPHdial configuration, the SAGE^{MAX} acts as a Host system and dials to PHP slave devices.

PID is an acronym for Proportional+Integral+Derivative and refers to a direct digital control method which responds to changes in a measured variable by producing a control output based on the difference between the measured variable and a setpoint (error), the history of the error and the rate-of-change of error.

pin-outs refer to functional descriptions of the pins on a connector or circuit component.

PLB is an acronym for Program Logic Block, a binary data file that contains compiled SPL program code in a form which can be executed by the SAGE^{MAX}.

port is a communications channel on a field panel or host system that is used to connect peripheral equipment or other devices.

PRB is an acronym for Program Reference Block, an ASCII text file that qualifies references that are made in the Program Logic Block (PLB) of a SAGE^{MAX} program.

Printer is a driver type used by the SAGE^{MAX} to configure a SAGE^{MAX} port for connection to a serial printer.

program attributes are terms used in SPL expressions. Program attributes consist of two-character, user-defined mnemonic codes and are used as storage locations in SPL programs. Program attributes are similar to attributes of other named database objects. Each SPL program in the SAGE^{MAX} can have up to 255 uniquely defined program attributes.

program control attributes are terms used in SPL expressions. Control attributes begin with the dollar sign (\$) character and specify detailed information about the program such as the program status (\$\$), error code (\$E), current section number (\$S), the number of seconds remaining in a time delay (\$D) and others.

protocol is a set of rules that governs how a device communicates with other devices. PHP, PUP and XANP are all protocols developed by American Auto-Matrix.

PTC stands for positive temperature coefficient, a resistive device whose resistance varies based on temperature.

pull-up resistance is a biasing resistance that is used on the +5 V side of an EIA-485 network to hold it at that level.

pull-down resistance is a biasing resistance that is used on the GND side of a network to hold the line at GND potential.

PUP refers to Public Unitary Protocol, a communications protocol in the public domain. Public Unitary Protocol was developed by American Auto-Matrix.

PUP device refers to any unitary controller that supports Public Unitary Protocol (PUP).

PUPHost is a driver type used by the SAGE^{MAX} to configure a port for use by PUP devices. Since the SAGE^{MAX} acts as the front end or *host* of the PUPnet, the name PUPHost is used.

PUPnet is a network of devices that uses Public Unitary Protocol (PUP) for communicating.

RAM is an acronym that refers to Random Access Memory, an area of temporary information storage within the SAGE^{MAX}.

references are named objects that are indirectly represented in SPL program logic (using the REF statement) and are associated in the Program Reference Block (PRB) of the program. References can be used as terms in SPL expressions.

registers are terms used in SPL expressions and are used as temporary storage locations for calculations in SPL programs. Every SPL program has 16 program registers. Program registers begin with a percent (%) sign and are identified as %A-%P.

repeater is a component of a local area network that is used to increase the electric signal so that greater distances can be achieved.

RFI stands for Radio-Frequency Interference, and is a form of interference that comes from energy sources within a system. Unless a system is properly protected to prevent this undesirable interference, televisions, radios and other radio-frequency dependent devices may not operate optimally.

RJ-11 is a type of connector typically used for phone line connections.

RPL is an acronym for REX Programming Language, a programming language for RCU2s and STARS used to write application programs.

RS232 *see EIA-232D.*

RS485 *see EIA-485.*

RTS stands for Request To Send which is pin number 4 of a standard EIA-232D DB25 connector or pin number 7 of a DB9 connector.

RXD stands for Received Data which is pin number 3 of a standard EIA-232D DB25 connector or pin number 2 of a DB9 connector.

SAGE is a communications-intensive field panel of the AI2100 system that offers powerful programming and networking capabilities.

sample count is the largest number of samples in a circular or time-anchored trend. For infinite trends, the sample count is the same as the current sample.

sample interval is an amount of time in minutes between each recorded trend sample that is specified when you create a trend.

scrolling region is an area on a display terminal that defines where displayed information scrolls upward without affecting any information on the screen that is outside this area.

serial port is a communications channel through which information is sent/received in a one-bit-at-a-time fashion. A device with a serial port must wait until at least ten bits (ten high/low signals) are sent or received before the data byte can be interpreted or used.

shield refers to a wire that is wrapped around the twisted shielded pair. The shield is used to carry transients away from components. For this reason, the shield must be electrically connected at only one end of a network to properly dissipate transients.

slave is a generic term that refers to a network device that is controlled by a higher-level device.

snapshot is a method of gathering trended information (unlike trend averaging) in which the data is taken like a snapshot and then recorded for that sample interval.

SPL is an acronym for SAGE^{MAX} Programming Language, the rich control programming language of the SAGE^{MAX} which has a superset of RPL features.

superset is a term which describes the features of SPL in reference to the features of its predecessor REX Programming Language (RPL). SPL contains all the features of RPL plus an additional cluster of new features.

table is a collection of up to 1,073,741,824 data values that are stored as linear, one-dimensional arrays in disk-resident files. Table references can be used as terms in SPL expressions.

task refers to a single function with a sole purpose or goal (i.e., alarm handling, operator interface, etc.). The SAGE^{MAX} has 20 individual tasks.

term is a component of an SPL expression. Terms include constants, named constants, registers, program control attributes, user-defined program attributes, named object attributes, references, virtual attributes, tables, functions and nested expressions.

terminal is a video display and keyboard combination that can be used to interface with the SAGE^{MAX}.

terminal block refers to the EIA-485 network connectors that are located inside the SAGE^{MAX} and are used to accept shielded, twisted pair wiring.

termination resistance is a resistance that is used to terminate a communications line on the end units of a multidrop network.

thick wire is a term that refers to an Ethernet networking configuration that uses AUI connectors. Also called Thick Net, standard Ethernet and 10Base5 Ethernet.

Thin Net is an Ethernet network that uses a small diameter coaxial cable that is less expensive than standard Ethernet coaxial cable. Also called 10Base2 Ethernet.

time-anchored trend is a circular trend trend that is based on an anchor interval. Anchor intervals can be *daily*, *weekly*, *monthly*, or *yearly* and produce trends that contain a history of data which has been sampled over a day, week, month, and year respectively. Each time-anchored trend allows you to sample data in time intervals of between one minute and one month. Time-anchored trends are synchronized to the nearest anchor point after power failures.

transceiver is a hardware device that can be used to transmit and receive data.

transient is an undesirable electromagnetic pulse or other temporary electrical interference.

tranzorb is a semi-conductor device used to protect against transients.

trend is a historical record that contains data values of up to eight references recorded over a specific period of time.

trunk is a two-wire or four-wire communication line.

TXD stands for Transmitted Data which is pin number 2 of a standard EIA-232D DB25 connector or pin number 3 of a DB9 connector.

two-wire mode *See half-duplex.*

unit controller is the lowest component of the Auto-Matrix system. Unit controllers provide distributed control and use PUP (Public Unitary Protocol).

UTP is an acronym for Universal Twisted Pair, a type of Ethernet network that uses standard telephone-style twisted pair wiring to connect a device to a hub.

virtual attributes are terms that can be used in SPL expressions. Virtual attributes represent attributes of network devices that may or may not have a name associated with them. Virtual attributes are specified using the Unified Network Services (UNS) function which requires information such as the port number, fundamental type, card number, channel and subchannel numbers, and the attribute name.

VT100 refers to a terminal that can emulate the functions of the Digital Equipment Corporation's (DEC) VT100 family. VT100 terminals have special features such as programmable scrolling regions and cursor positioning capabilities not available in dumb terminals. VT100 also refers to the SAGE^{MAX} driver type that is used to configure SAGE ports for communication with VT100 terminals.

VT220 refers to a terminal that is similar to a VT100 with a superset of VT100 features. SAGE^{MAX} can take advantage of some of these features if a VT220 is used.

XANP stands for eXtended Automation Network Protocol, a proprietary communications scheme developed by American Auto-Matrix and used for communications between field panels. XANP also refers to the driver type used by the SAGE^{MAX} to configure such networks.

APPENDIX B - SUMMARY OF ERROR MESSAGES

This Appendix lists the error codes that are used by the SAGE^{MAX}. A brief description follows error codes whose meanings may be unclear.

Error numbers followed by an asterisk are *trappable errors*. When trappable errors occur (due to a read or write attribute request, for example) from within a SAGE^{MAX} program, you can use certain SAGE^{MAX} Programming Language (SPL) statements to handle the error condition in special ways. The uses of these programming statements are explained in detail in **Chapter 11: Programming**.

#	CODE	MEANING
0	#Success	successful (no errors)
1	#RequestInvalid	cannot do this request
2	#Argument Error	an argument is bad
3	#InvalidResponse	bad return from a query
4*	#CRCErrror	CRC Error
5*	Timeout	timed out waiting
6	#UnknownDatatype	not a standard data type
7*	#NAKResponse	some special error occurred
8	#InvalidChannel	bad channel number specified
9	#NoSuchAttribute	specified attribute does not exist
10	#NC@RecordDeleted	attempt to read a deleted record
11	#NC@NoSuchName	name was not found in search
12	#NC@TempFlushFailed	wild search file flush failed
13	#NC@TempCreateFailed	wild search file could not be created
14	#NC@LseekCacheFailed	wild search file seek failed
15	#NC@ReadCacheFailed	wild search file read failed
16	#NC@ObjectFileSeekFailed	database file rewind failed
17	#NC@ObjectFileOpenFailed	database file open failed
18	#NC@ObjectFileReadFailed	database file read failed
19	#NC@RecordLocked	could not delete record
20	#NC@NoSuchRecord	bad record number
21	#NoSuchObjectType	bad database type
22	#NC@NoFreeLocks	could not lock record

Table B-1 SAGE^{MAX} Error Messages

#	CODE	MEANING
23	#NCNC@ObjectFileWriteFailed	could not write record
24*	#TemporarilyBlocked	
25	#InvalidObjectName	
26	#InvalidTask	invalid task parameter specified
27	#OB@InvalidTimer	invalid timer handle specified
28	#OB@NoMoreTimers	no more timers available
29	#ObjectTypeNotInitialized	object type is being initialized
30	#UnknownPeer	unknown peer name
31	#InvalidDriver	invalid driver type
32	#NoSuchClass	class does not exist
33	#DOSTempCreateFailed	temporary file could not be created
34	#InvalidPort	invalid port number specified
35	#InvalidUnit	invalid unit number specified
36	#InvalidSession	invalid virtual terminal session
37	#InvalidService	protocol service unknown
38	#AlreadyAcknowledged	transaction already acknowledged
39	#NoSuchCard	card does not exist
40	#BadFtypeForCard	ftype invalid for card
41	#BadDatumSize	data size was rejected
42	#ConversionError	data conversion error
43*	#DataRejected	data was rejected
44	#BadUserOrCodeword	invalid user name or codeword
45	#NoFreeSessions	no free VT or file sessions
46	#PrivilegeViolation	attempted to perform a privileged operation
47	#VTAlreadyOpen	virtual terminal already open
48	#PortBusy	port is busy
49	#BadDrive	invalid drive number
50	#EndOfFile	end of file was reached
51	#NoSuchFile	file name specified does not exist
52	#BadFileHandle	invalid file handle number
53	#BadFileName	invalid file name format
54	#BadRecordNumber	invalid record number

Table B-1 SAGE^{MAX} Error Messages

#	CODE	MEANING
55	#FileHandleNotOpen	a specified file handle is not open
56	#FileInUse	specified file is currently being used
57	#FileHandleInUse	file number already assigned
58	#TransientFileTooBig	Job or OBL file was > 64K
59	#QueueFull	
60	#NoMatchFound	
61	#NoSuchName	
62	#MathConversionError	
63	#Underflow	
64	#Overflow	
65	#NotANumber	
66	#InvalidFormat	
67	#InvalidDataType	
68	#NoCompare	
69	#BadSampleInterval	
70	#InvalidTrendSignature	
71	#SampleLimitExceeded	
72	#TrendAlreadyEnabled	
73	#TrendNotFound	
74	#InvalidTrendFormat	
75	#NoAvailTmPackets	
76	#SampleFailureAlarm	
77	#NotCollected	
78	#NameUnknownToPeer	
79	#ChecksumError	
80	#InvalidProgramRegister	
81	#ProgramUnloaded	
82	#StackError	
83	#InvalidPcode	
84	#InvalidTerm	
85	#InvalidOperator	
86	#InvalidState	

Table B-1 SAGE^{MAX} Error Messages

#	CODE	MEANING
87	#TermError	
88	#ExpressionError	
89	#UnsuccessfulUnload	
90	#InvalidStateCharge	
91	#ProgramNotFound	
92	#IndexTooLarge	
93	unused	
94	unused	
95	unused	
96	#XOPRInvalidPassword	
97	#NotAnAVLFile	
98	#InvalidMessageNumber	
99	#InvalidLanguage	
100*	#DialBusy	
101	#NoPhoneNumber	
102*	#FailedToContact	
103	#NoResponseFromSite	
104 thru 199	unused	
200	#21@Success	DOS success
201	#21@InvalidFunction	
202	#21@FileNotFound	
203	#21@PathNotFound	
204	#21@NoFreeHandles	
205	#21@AccessDenied	
206	#21@InvalidHandle	
207	#21@MCBsDestroyed	
208	#21@OutOfMemory	
209	#21@InvalidAddress	
210	#21@InvalidEnvironment	

Table B-1 SAGE^{MAX} Error Messages

211	#21@InvalidFormat	
212	#21@InvalidAccessCode	
213	#21@InvalidData	
214	#21@ReservedError14	
215	#21@InvalidDrive	
216	#21@AttemptToRemoveCD	
217	#21@NotSameDevice	
218	#21@NoMoreFiles	
219	#21@DiskWriteProtected	
220	#21@UnknownDiskUnit	
221	#21@DriveNotReady	
222	#21@UnknownCommand	
223	#21@DataCRCError	
224	#21@BadRequestStructureLength	
225	#21@SeekError	
226	#21@UnknownMediaType	
227	#21@SectorNotFound	
228	#21@PrinterOutOfPaper	
229	#21@WriteFault	
230	#21@ReadFault	
231	#21@GeneralFailure	
232	#21@CannotCopyFileToItself	
233	#21@InvalidFileName	
234		
thru	unused	
255		

Table B-1 SAGE^{MAX} Error Messages

APPENDIX C - PHP ERROR CODES

This Appendix lists the possible PHP error codes that may be returned by a PHP slave device such as SAGE^{MAX}.

#	CODE	MEANING
00xx	BadCommand	xx is the invalid command ncode
01xx	BadSubUnit	xx is the invalid subunit number
0202	CRCErrror	subunit transaction had CRC error
0203	NoResponse	no response from subunit
0204	NoCard	card was not present
0205	BadType	card does not have fundamental type
0206	BadChannel	card does not have specific channel
0207	AttrNotFound	attribute requested could not be found
0208	BadData	data was rejected
0209	InvalidType	channel returned an unsupported data type
0302	NameNotFound	could not find named datum
0408	EOF	end of file (record number too big)
0500	BadID	access denied to this operator
0700	NoCache	invalid command (not all targets can cache)
0800	NoVirtual	target has no virtual terminal
0801	VirtualInUse	target's virtual terminal is busy
0900	RequestInvalid	request not recognized by target
0901	NAKResponse	message not received (negative acknowledgment)
0902	InvalidResponse	target's response was invalid
0903	InvalidService	requested service is not available

Table C-1 PHP Error Codes

APPENDIX D - PUP ERROR CODES

This Appendix lists the possible PUP error codes that may be returned by PUP devices.

CODE	MEANING
FFFFh	generic negative acknowledgment (NAK)
FFFEh	PUP command received correctly, but not supported
FFFDh	no such channel
FFFC	no such attribute
FFFBh	value for attribute not accepted
FFFAh - 8000h	reserved for generic NAK codes
7FFFh - 0000h	reserved for custom NAK codes

Table D-1 PUP Error Codes

APPENDIX E - XANP ERROR CODES

This Appendix lists the possible XANP error codes that may be returned by XANP devices.

CODE	MEANING
00h	card specified was not in configuration
01h	type specified was not appropriate for specified card
02h	channel specified was not appropriate for specified type
03h	attribute was not found
04h	datum size provided was not appropriate for attribute
05h	data conversion type was unsupported
06h	ISR data block CRC error
07h	data rejected
08h	invalid exception channel
09h	Peernet access error
0Ah	no port available for virtual terminal
0Bh	not in virtual terminal mode
0Ch	virtual terminal mode prevented by system variable
0Dh	virtual terminal already open
0Eh	port is busy
0Fh	bad close virtual terminal
10h	drive not ready
11h	bad drive number
12h	media not formatted
13h	disk not initialized
14h	media is unusable
15h	end of file encountered
16h	device full
17h	no such file(s)
18h	file not open
19h	disk write protected
1Ah	no room in directory

Table E-1 XANP Error Codes

CODE	MEANING
1Bh	bad filename
1Ch	device not mounted
1Dh	disk write-protected
1Eh	data CRC error
1Fh	seek error
20h	ID CRC error
21h	privilege violation
22h	FDC reset
23h	controller busy
24h	record not found
25h	FDC RAM test failed
26h	block number too large
27h	illegal directory or extent
28h	bad file number
29h	file name in use
2Ah	file number already assigned

Table E-1 XANP Error Codes

APPENDIX F - PEERNET ERROR CODES

This Appendix lists Peernet errors that are returned from the SAGE^{MAX}.

CODE	MEANING
xx01h	illegal command (xx is command code)
xx02h	point errors:
0002h	no such card
0102h	no such fundamental type
0202h	no such channel
0302h	no such attribute
0702h	data rejected
xx03h	access denied errors:
0203h	no port available

Table F-1 Peernet Error Codes

APPENDIX G - FUNDAMENTAL TYPES

This Appendix lists the fundamental point type codes used by XANP and Peernet networks on the SAGE^{MAX}.

NUMERIC CODE	TYPE CODE	POINT TYPE
00	SY	System Point
01	BI	Binary Input
02	BO	Binary Output
03	SS	Start/Stop Scheduling
04	AI	Analog Input
05	AO	Analog Output
06	AC	Analog Control
07	BC	Binary Control
08	MC	Motor Control
09	PG	Program
10	TR	Trend
11	TT	Terminal Type
12	TF	Terminal Flow
13	TC	Terminal Control
14	TD	Terminal Default
15		reserved

Table G-1 Fundamental Point Type Codes

APPENDIX H - PUP DATA TYPES

This Appendix lists the hexadecimal and decimal PUP numeric data types codes that are used by the SAGE^{MAX}.

The hexadecimal codes are followed by **h** and the decimal codes are provided in parentheses.

CODE	DIGIT FORMAT	MEANING
FFh (255)	XXXXXXXXXX.	signed 10 digit
FEh (254)	XXXXXXXXXX	unsigned 10 digit
FDh (253)	XXXXXXXXXX.X	signed 9.1 digit
FCh (252)	XXXXXXXXXX.X	unsigned 9.1 digit
FBh (251)	XXXXXXXXXX.XX	signed 8.2digit
FAh (250)	XXXXXXXXXX.XX	unsigned 8.2 digit
F9h (249)	XXXXXXXX.XXX	signed 7.3 digit
F8h (248)	XXXXXXXX.XXX	unsigned 7.3 digit
F7h (247)	XXXXXX.XXXX	signed 6.4digit
F6h (246)	XXXXXX.XXXX	unsigned 6.4 digit
F5h (245)	XXXXX.XXXXX	signed 5.5 digit
F4h (244)	XXXXX.XXXXX	unsigned 5.5 digit
F3h (243)	XXXX.XXXXXX	signed 4.6 digit
F2h (242)	XXXX.XXXXXX	unsigned 4.6 digit
F1h (241)	XXX.XXXXXXX	signed 3.7 digit
F0h (240)	XXX.XXXXXXX	unsigned 3.7 digit
EFh (239)	XX.XXXXXXXX	signed 2.8 digit
EEh (238)	XX.XXXXXXXX	unsigned 2.8 digit
EDh (237)	X.XXXXXXXXXX	signed 1.9 digit
ECh (236)	X.XXXXXXXXXX	unsigned 1.9 digit
EBh (235)	.XXXXXXXXXXX	signed .10 digit
EAh (234)	.XXXXXXXXXXX	unsigned .10 digit

Table H-1 PUP Data Types

CODE	DIGIT FORMAT	MEANING
E9h (233)	channel map	one bit per channel
E8h (232)	bitmap or text	one bit per text field
E7h (231)	BCD (H/M/S)	hours is LSB
E6h (230)	BCD (H/M/S)	hours is LSB
E5h (229)	packed BCD	8 BCD digits as 4 bytes
E4 (228)	BCD date	MSW is year LSW/MSB is month LSW/LSB is day
E3h (227)	Binary date	MSW is year LSW/MSB is month LSW/LSB is day
E2h (226)	reserved	
E1h (225)	reserved	
E0h (224)	IEEE 784 32-bit floating point	
Dfh - 80h (233 - 128)	reserved	
7Fh (127)	bogus data sample	LSW is SAGE ^{MAX} error code MSW is reserved
7Eh - 09h (126 - 9)	reserved	
08h (8)	1991-2117 1-12 1-31 0-23 0-59 1-7 reserved	year (bits 25-31 in 32-bit Dword) month (bits 21-24) day (bits 16-20) hour (bits 11-15) minute (bits 5-10) day of week (bits 2-4, 1=Sun) (bits 0-1)
07h (7)	0 or 1	0=no, 1=yes (MSBs=0)
06h (6)	ASCIIZ string	value is seg:offset of string
05h (5)	Sunday, etc.	DOS Day-of-week
04h (4)	HH:MM:SS	DOS-style time
03h (3)	XXXX:XXXX	LSW is offset MSW is segment
02h (2)	XXXXXXXXXh	hexadecimal double word
01h (1)	XXXXh	hexadecimal word
00h (0)	XXh	hexadecimal byte

Table H-1 PUP Data Types

APPENDIX I - TREND OBJECT FILES

This Appendix explains the structure of trend object files used by the SAGE^{MAX}.

Trend objects that are created on the SAGE^{MAX} are binary data files which contain information about database objects which have been monitored at a known fixed interval. Up to eight points may be monitored together in one trend. Trend object file names may contain up to eight characters, reside implicitly on the **C:\TRENDS** directory and are given the extension **.TRN**.

A trend object file contains a *header record* which is 512 bytes long and describes the database objects to be monitored. The structure of the header record is shown in **Table I-1**.

The header record begins with the text **TREND** and is followed by a trend type code (see **Table I-2**).

After the header record is a series of *data sample records*, each of which contains the values of up to eight database objects at a particular sample interval. Each data sample record also contains a four-byte time/date stamp, the structure of which is explained in **Table I-3**.

HEADER RECORD (512 bytes)	"TREND"	(5 bytes)
	Trend Type (see Table I-2)	(1 byte)
	Sample interval (minutes)	(2 bytes)
	Sample Count	(4 bytes)
	Number of Bogus Samples (Failures)	(2 bytes)
	Bogus Sample Alarm List	(2 bytes)
	Current Sample Number	(4 bytes)
	Maximum Allowable Samples for Infinite Trends	(4 bytes)
	Reserved	(232 bytes)
	32-byte ASCIIZ Variable Definition #1	(32 bytes)
	32-byte ASCIIZ Variable Definition #2	(32 bytes)
	32-byte ASCIIZ Variable Definition #3	(32 bytes)
	32-byte ASCIIZ Variable Definition #4	(32 bytes)
	32-byte ASCIIZ Variable Definition #5	(32 bytes)
	32-byte ASCIIZ Variable Definition #6	(32 bytes)
	32-byte ASCIIZ Variable Definition #7	(32 bytes)
32-byte ASCIIZ Variable Definition #8	(32 bytes)	
DATA SAMPLE #1	Special Data Type (08h) of Time/Date Stamp	(1 byte)
	Time/Date Stamp (32 bits, see Table I-3)	(4 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #1	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #2	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #3	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #4	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #5	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #6	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #7	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #8	(5 bytes)

Table I-1 Header Record and Sample Data Record for Trend Object Files

	TREND TYPE	TYPE CODE
	infinite = circular = averaged =	0 (00h) 1 (01h) 128 (80h)
DAILY TIME-ANCHORED	every 1 minute = every 15 minutes = every 30 minutes = every 60 minutes =	2 (02h) 3 (03h) 4 (04h) 5 (05h)
WEEKLY TIME-ANCHORED	every 1 minute = every 15 minutes = every 30 minutes = every 60 minutes = every day =	6 (06h) 7 (07h) 8 (07h) 9 (09h) 10 (0Ah)
MONTHLY TIME-ANCHORED	every 1 minute = every 15 minutes = every 30 minutes = every 60 minutes = every day =	11 (0Bh) 12 (0Ch) 13 (0Dh) 14 (0Eh) 15 (0Fh)
YEARLY TIME-ANCHORED	every 1 minute = every 15 minutes = every 30 minutes = every 60 minutes = every day = every week = every month =	16 (10h) 17 (11h) 18 (12h) 19 (13h) 20 (14h) 21 (15h) 22 (16h)

Table I-2 Trend Types

FIELD	RANGE	BIT NUMBERS IN 32-BIT DWORD
Year	1991-2117	31-25
Month	1-12	21-21
Day	1-31	20-16
Hour	0-23	15-11
Minute	0-59	10-5
Day of Week	1-7	4-2 (1=Sunday, 7=Saturday)
reserved		1-0

Table I-3 Structure of the Time/Date Stamp

APPENDIX J - NAME BINDING FILES

Name Binding Files (NBFs) are ASCII text files (with the extension **.NBF**) that you use for off-line database creation of points, programs, variables and globals. NBFs are translated into Binary Object (BOB) files using a SAGE^{MAX} file conversion utility.

There are three basic types of lines in a Named Binding File. *Comment lines* are lines that begin with a semicolon (;) and are ignored by the SAGE^{MAX} conversion program. *Binding Definition lines* are used to specify an object type and a series of parameters to define the object. *Continuation lines* begin with **TABs** and are available when it is impractical to fit the entire Name Binding definition on one line.

Binding Definition Lines can contain parameters that are optional. These optional parameters do not have to be included in Binding Definition Lines. The position of these parameters in the Binding Definition Line is very important in determining the proper syntax for the line.

For example, if only the first two parameters of a line are used, the remainder of the line can remain blank. If, however, the first three and last two parameters of a Binding Definition Line are used, you must use commas to indicate parameters that have been omitted.

The tables in this Appendix describe the required format of the Binding Definition lines for Name Binding Files (NBFs) and illustrate several examples of each type.

All numeric values shown in the following tables are assumed to be decimal unless followed by “H” or “h” which indicates hexadecimal.

Format: GL *name*=*peername*,*euname*

where

GL	is a two-letter mnemonic code used to identify global objects
<i>name</i>	is the name of the global object
<i>peername</i>	is the peername associated with the global object
<i>euname</i>	is the name of the optional Engineering Units file for the global object.

Examples: GL OutsideAirTemp=Main Building SAGE^{MAX},STDAI
GL OAT= Main Building SAGE^{MAX}

Table J-1 Binding Definition Format for Global Objects

Format:VR *name=type,value,famous,log,objprivs*

where

VR	is a two-letter mnemonic code used to identify variable objects
<i>name</i>	is the name of the variable object
<i>type</i>	is the numeric data type code (See Appendix H: PUP Data Types)
<i>value</i>	is the value of the variable object
<i>famous</i>	is the optional famous flag used to specify if the variable object is famous FAMOUSglobally announces changes to this variable (blank)no global announcement of changes
<i>log</i>	is the optional log flag used to specify whether or not changes made to this variable by operators should be logged as operator events LOGlog attribute changes (by operators) as operator events (blank)do not log changes
<i>objprivs</i>	is the optional object privilege bitmap associated with this variable.

Examples:VR hexbyte=00h,1Bh
 VR hexword=01h,FE00h
 VR hexdword=02h,12345678h
 VR DOStime=04h,12:34:56
 VR happyday=05h,friday
 VR occupied=07h,false
 VR holiday=07h,true
 VR Summer=07,Yes,Famous,Log,0001011b
 VR emergency=07,No
 VR pi=E0h,3.14159
 VR noon=E6h,12:00
 VR MidNight=E7h,00:00:00
 VR flags=E8h,00001011b
 VR HumanTemp=FEh,98.6,,01011111b

Table J-2 Binding Definition Format for Variable Objects

Format:PG *name=PLB,PRB,INI,lflag,rflag,preload,euname,famous,log,objprivs*

where

<i>PG</i>	is a two-letter mnemonic code used to identify program objects
<i>name</i>	is the name of the program object
<i>PLB</i>	is a path fragment that identifies the Program Logic Block
<i>PRB</i>	is a path fragment that identifies the Program Reference Block
<i>INI</i>	is a path fragment that identifies the program attribute Initial Values File
<i>lflag</i>	is a flag used to identify the nature of the PLB FILEprogram is file-resident STICKYprogram cannot be unloaded once it is loaded into RAM
<i>rflag</i>	is a flag used to identify the nature of the PRB FILEPRB is always file-resident RAMPRB is RAM-resident, but unloadable STICKYPRB cannot be unloaded once it is loaded into RAM
<i>preload</i>	is a flag used to specify when the program should be loaded PRELOADload program after SAGE reboot (blank)load program on demand
<i>euname</i>	is the name of the Engineering Units file for the program object
<i>famous</i>	is the famous flag used to specify if the program object is famous FAMOUSglobally announces changes to this program (blank)no global announcement of changes
<i>log</i>	is the log flag used to specify whether or not changes made to this program's attributes by operators should be logged as operator events LOGlog attribute changes (by operators) as operator events (blank)do not log changes
<i>objprivs</i>	is the optional object privilege bitmap associated with this program

Examples:

PG AHU39OSS=OSS,AHU39,File,File

PG PumpB=LeadLag,BPumps,,Sticky,RAM,Preload,LeadLag,Famous,Log

Table J-3 Binding Definition Format for Program Objects

Format:

PT *name*=*port,unit,chan,ftype,card,subchan,euname,famous,log,objprivs,ovrclass*
 (tab) *point description message*
 (tab) *alarm message*
 (tab) *return message*

where

PT is a two-letter mnemonic code used to identify point objects
name is the name of the point object
port is the port number of the network to be accessed
unit is the unit number of the network device containing the point
chan is the network channel number
ftype * is the fundamental type code (or numeric equivalent) of the point
 (see **Appendix G:Fundamental Types**)
card * is the card associated with the point
subchan * is the subchannel associated with the point
euname is the name of the Engineering Units file for the point object.
famous is the famous flag used to specify if the point object is famous
 FAMOUSglobally announces changes to this point
 (blank)no global announcement of changes
log is the log flag used to specify whether or not changes made to this
 point's attributes by operators should be logged as operator events
 LOGlog attribute changes (by operators) as operator events
 (blank)do not log changes
objprivs is the optional object privilege bitmap associated with this point
ovrclass specifies a SAGE^{MAX} class number to be used to override all alarm
 classes determined by the driver for this point. Zero means use the
 underlying alarm class.

* *ftype*, *card* and *subchan* parameters are only required for XANP and Peernet networks.

Table J-4 Binding Definition Format for Point Objects

APPENDIX K - SPL PSEUDOCODES

This Appendix lists and explains the pseudocodes (Pcodes) used by the SAGE^{MAX} Program Language (SPL) for statement, term and operator encoding. Refer to **Table K-1**, **Table K-2**, & **Table K-3**.

PCODE(S)	SOURCE FORM AND COMMENTS
00	Stop (this program)
01tt..tt00	tt..tt: (label "tt..tt")
02xxxx	Section xxxx
03iexpr	Swait iexpr
04iexpr	Mwait iexpr
05expr	Wait expr
06pp..pp00	Call plb "pp..pp"
07pp..pp00	Call plb "pp..pp" and stick
08	Return or Unload (this program)
09xxxx expr	If expr then xxxx (xxxx is offset to "jump to")
0Axxxxyyyy expr	If expr xxxx else yyyy
0Bxxxx	Goto xxxx
0Cnn aaaa bbbb cccc ... expr	On expr Goto aaaa, bbbb, cccc... (nn=# of choices)
0D0rxxxx	Loop register r, label xxxx
0Eaabbexpr	program attribute "aabb"=expr
0Fpp..pp00expr	object attribute path "pp..pp"=expr
1xexpr	Register=expr (for register A, x=0) (for register B, x=1) (for register C, x=3) . . (for register O, x=E) (for register P, x=F)
20expr1 expr2	Ref (expr1)=expr2 (expr1=0-255)
21tt..tt00 expr1 expr2	&tab;e (expr1)=expr2 ("tt..tt" is table fragment)
22px ux fx cx nx aabb expr	UNS (px,ux,fx,cx,nx,attr)=expr (px=portexpr 0-15) (cx=cardexpr 0-255) (ux=unitexpr 0-65535) (nx=chanexpr 0-65535) (fx=ftypeexpr 0-255) (aabb=attribute)

Table K-1 SPL Pseudocodes for Statement Encoding

PCODE(S)	SOURCE FORM AND COMMENTS
23pp..pp00	Activate program name "pp..pp"
24pp..pp00	Deactivate program name "pp..pp"
25pp..pp00	Restart program name "pp..pp"
26pp..pp00	Stop program name "pp..pp"
27expr pp..pp00	Spool portexpr, pathname "pp..pp"
28expr pp..pp00	Spool portexpr, pathname "pp..pp", DELETE
29nn classexpr ff..ff00 expr1, expr2...exprN	Alarm classexpr, "formatstring",x,x,x,x... (nn=# of expressions, 8 max)
2Ann portexpr classexpr ff..ff00 expr1...exprN	Print portexpt classexpr, "formatstring", x,x,x,x... (nn=# of expressions, 8 max)
2Bnn ll..ll00 ff..ff00 expr1, expr2...exprN	Log logfilemane "ll..ll",,"formatstring", x,x,x,x... (nn=# of expressions, 8 max)
2Cnn classexpr ff..ff00 expr1, expr2...exprN	Job classexpr, "formatstring", x,x,x,x... (nn=# of expressions, 8 max)
2D	No operation (NOP)
2Exxxx	Gosub xxxx
2Fxxxx	Oneror xxxx
30	Errorwait
31	Errorabort
32xxaabbcc	Save aa, bb, cc (xx=# of attributes, if xx=0 save all)
33tttttt00	Starttrend trend name "ttttttt"
34tttttt00	Stoptrend trend name "ttttttt"
35-FF	reserved

Table K-1 SPL Psuedocodes for Statement Encoding

PCODE(S)	MEANING/FORMAT	VALUE/RANGE	TYPE
00	reserved		
01	one	1	FF integer
02	minus one	-1	FF integer
03	day of month	1..31	FE integer
04	month of year	1..12	FE integer
05	current year	e.g., 1996	FE integer
06	current day of week	0..6 (0=Monday)	FE integer
07	current Julian day	1..366	FE integer
08	current time	00:00:00..23:59:59	E7 time
09ttbb	typed byte	bb	tt
0Attwww	typed word	www	tt
0Btddddddd	typed constant	ddddddd	tt
0C	value of PI	3.141593	E0 float
0D	zero	0	FF integer
0Eaabb	program attr "aabb"		??
0Fpp..pp00	object attr path "pp..PP"		??
1x	register A-P	x=0-F	??
20expr	REF (expr)	expr value=0-255	??
21tt..tt00expr	&table (expr)	table path "tt..tt"	??
22px ux fx cx nx aabb	UNS (px,ux,fx,cx,nx,attr) px=portexpr 0-15 ux=unitexpr 0-65535 fx=ftypeexpr 0-255 cx=cardexpr 0-255 nx=chanexpr 0-65535 aabb=attribute	0-15 0-65535 0-255 0-255 0-65535	??
23expr	(expr)		??
24expr	-expr	unary negate	??
25iexpr	NOT iexpr	unary 1's complement	FE integer
26fexpr	INT (fx)	convert float to integer	FF integer
27fexpr iexpr	FIX (fx,ix)	convert float to fixed pt.	?? integer
28iexpr	FLOAT (ix)	convert integer to float	E0 float
29expr	ABS (x)	absolute value	??

Table K-2 SPL Pseudocodes for Term Encoding

PCODE(S)	MEANING/FORMAT	VALUE/RANGE	TYPE
2Aexpr	ROUND (fx)	round off	E0 float
2Bexpr	SIN (fx)	Sine value	E0 float
2Cexpr	COS (fx)	Cosine value	E0 float
2Dexpr	TAN (fx)	Tangent value	E0 float
2Eexpr	ARCTAN (fx)	Arctangent value	E0 float
2Fexpr	LOG (fx)	Logarithm	E0 float
30expr	LN (fx)	Natural log	E0 float
31expr	SQRT (x)	Square root	??
32texpr1 texpr2	Between two times	0=not between,1=between	FE integer
33xx..xx00	MEAN (x,x...x,x)	Mean value from expr list	??
34xx..xx00	MAX (x,x...x,x)	Max value from expr list	??
35xx..xx00	MIN (x,x...x,x)	Min value from expr list	??
36iexpr	TODAY (ix)	Is today this (one of these) day(s) of week? 0=no, 1=yes	FE integer
37	MON	0	FE integer
38	TUE	1	FE integer
39	WED	2	FE integer
3A	THU	3	FE integer
3B	FRI	4	FE integer
3C	SAT	5	FE integer
3D	SUN	6	FE integer
3Eexpr iexpr	RETYPE (x,ix)	convert to a data type	??
3F	DATATYPE (x)	return data type of expr	FE integer
40	DATE	current date	E3 hex date
41-FF	reserved		

Table K-2 SPL Pseudocodes for Term Encoding

PCODE(S)	DESCRIPTION	EXAMPLE	NOTES
00	End of Expression		
01	Exponential	A**B	A to the B power
02	Multiplication	A*B	
03	Division	A/B	
04	Remainder after Division	A MOD B	7 MOD 3=1
05	Addition	A+B	
06	Subtraction	A-B	
07	Equality	A==B	0 if not, 1 if equal
08	Inequality	A<>B	1 if not, 0 if equal
09	Greater than	A>B	1 if A>B, then 0
0A	Greater than or Equal	A>=B	1 if A.B or A=B, else 0
0B	Less than	A<B	1 if A<B, else 0
0C	Less than or Equal	A<=B	1 if A,B or A=B, else 0
0D	Bitwise AND	A AND B	32-bit integer
0E	Bitwise OR	A OR B	32-bit integer
0F	Bitwise Exclusive OR	A XOR B	32-bit integer
10	Bitwise Shift left	A SHL B	1 SHL 3=8
11	Bitwise Shift Right	A SHR B	256 SHR 7=2
12-FF	reserved		

Table K-3 SPL Pseudocodes for Expression Encoding

APPENDIX L - GROUP FILES

This Appendix explains the format of group files used by the SAGE^{MAX}. Group files are text files that contain editable lines of ASCII text. There are three types of line in a group file:

- comment lines
- subgroup path lines
- object reference lines

Although group files may contain an unlimited number of editable lines, SAGE^{MAX} will only use the first 17 lines of the file.

Comment lines begin with a semicolon (;) which must be the first character of the line. The remainder of the line is ignored.

If the first line of the file is a comment, the text following the semicolon is assumed to be the group description.

Subgroup path lines must have a backslash (\) character as the first character in the line. The backslash is followed by a pathname fragment (up to 26 characters in length) which specifies up to two subdirectories and a file name. This pathname is always implicitly preceded by **\GROUPS** and any extension (if present) is stripped off. The extension of group files is **.GRP**.

The pathname may optionally be followed by a **TAB** and description text. If present, this text appears when the group is displayed as a menu. The description should serve as an explanation about the intended use of the subgroup.

Object reference lines contain a reference to an object, one of its attributes and an optional description of the reference. The reference may be specified in any one of four forms:

- object name
- object name;attribute
- type\object name
- type\object name;attribute

If present, the object type precedes the object name and is followed by a backslash (\). Valid object type codes are:

- PT point
- PG program
- VR variable
- GL global

The object name can be up to 24 characters long. The object name may be optionally followed by a semicolon (;) and a two-character attribute. If the attribute name is not present, the default attribute of the specified object is assumed. If the type is not specified, the SAGE^{MAX} searches all object types to find the named object.

When a group is displayed by the operator interface, each group member is handled in a particular way. If the group member is a subgroup, then its name is displayed followed by any description text which follows the group name in the group file.

If the group member is a comment line, it is not displayed by the operator interface. The comment line is used only for documenting the group file.

If a group member is an object reference, then it is displayed in one of three forms depending on the presence or absence of a description message. If the reference has a description after it in the group file, up to 33 characters of that description are displayed preceding the value of the point attribute. If there is no description in the group file for a particular member and the object referenced was a point, then the point message file is examined to see if the point has a corresponding point description message. If so, up to 33 characters of the point description message are displayed. In every other case, only the object reference (type, name and attribute) is displayed. **Figure L-1** illustrates a sample group file.

```

;Floor 1 Zone Temperatures
PT\F1_ROOM101;CV      Zone Temp Floor 1 Room 101
PT\F1_ROOM102;CV      Zone Temp Floor 1 Room 102
PT\F1_ROOM105;CV      Zone Temp Floor 1 Room 105
PT\F1_ROOM106;CV      Zone Temp Floor 1 Room 106
PT\F1_ROOM108;CV      Zone Temp Floor 1 Room 108
PT\F1_ROOM109;CV      Zone Temp Floor 1 Room 109
PT\F1_ROOM110;CV      Zone Temp Floor 1 Room 110
PT\F1_ROOM112;CV      Zone Temp Floor 1 Room 112
\SETPTS\FLR1\F1SETPTS      Control Setpoints Floor 1
\SCHEDS\FLR1\F1SCHEDS      Schedules for Floor 1
\LIGHTS\FLR1\F1LIGHTS      Lighting Control for Floor 1
\OVRIDS\FLR1\F1OVRIDS      Control Overrides for Floor 1
\PROGS\FLR1\PROGRAMS      SPL Programs for Floor 1

\MAIN                    Return To MAIN Group
;
;
;*****
;  Group : \F1TEMPSFloor 1 Zone Temperatures
;
;  This group file is a subgroup of group MAIN, and shows
;  the zone temps of rooms on the first floor.
;*****

```

Figure L-1 Sample Group File

APPENDIX M - DCS FILES

This Appendix explains the structure of the Points Definition Files (**.PDFs**) used by the data capture/data stuff (**DCS**) background job of the SAGE^{MAX}.

The Points Definition File (**.PDF**) contains lines of text which adhere to one of several possible formats. The lines may be up to 128 characters long before the carriage return and linefeed (CRLF). Extra characters will be ignored. The seven possible line formats for **.PDF** files are:

- ;comment line
- >pathname
- >>pathname
- objectname,description
- !pathname
- !!pathname
- objectname=value

Lines which begin with a semicolon (;) character are treated as comment lines and are ignored by **DCS**.

The >*pathname* format causes further data capture list output to be directed to *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain *wildcards* as described later in this section.

The >>*pathname* format cause further data capture list output to be appended to *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain *wildcards* as described later in this section.

The *objectname,description* format will read (capture) the named object/attribute and output its value to the list output file. If the /S switch is present in the job string, then the output is formatted in a form that is suitable for subsequent use as a data stuff **.PDF** file. The *description* is any arbitrary text, presumably used to describe the named object in more detail.

The !*pathname* format causes further data stuff audit output to be directed to *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain wildcards as described in the following sections.

The !!*pathname* causes further data stuff audit output to be appended to *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain wild cards as described later in this section.

The *object=value* format is used to write (stuff) the named object attribute with the new *value*. The acceptable formats for the new value are described in **Table M-1**.

Alternately, if the /S option in the job string is selected, information is formatted for data stuffing. For example:

SOMEPOINT;AT=75.3

The .PDF file can contain as many lines as desired. There is no limit to the number of times that the list output may be redirected. The pathname for list/audit output is verified before a new list output file is opened or appended. This means that **DCS** checks for the existence of each subdirectory in the pathname. If a subdirectory does not exist, then it is created automatically by **DCS**.

CAUTION

You must be careful to use the >, >>, ! and !! formats correctly, since they may overwrite existing files!

The >, >>, ! and !! formats allow the percent (%) character to appear in the pathname. If it is used, the % character must be followed by a single letter code which indicates one of the time and date values shown in **Table M-2**. You can specify any combination of these time and date codes in conjunction with normal pathname characters to form unique time/date-based pathnames for **DCS** files. This allows the generation of time-ordered files by **DCS**. Since **DCS** automatically creates subdirectories, you may freely use these % codes in subdirectory names, if desired.

% CODE	TIME UNIT	DIGITS GENERATED
N	month	01-12
D	day of month	01-31
C	year of century	00-99
Y	year	e.g., 1991
H	hour	00-23
M	minute	00-59
W	week of year	01-52
K	day of week	1-7, 1=SUN
J	day of year	001-366

Table M-2 Wildcards Used in >, >>, ! and !! Pathnames

For example, the pathname

>LOG\%W%K%H%M.PRN

might be used as the name of a data capture list file which was run every day, perhaps several times a day. The resulting files would show the week of the year (%W), day of the week (%K), and time (%H%M) as the file name, e.g., **14021335.PRN**.

APPENDIX N - SAGE^{MAX} MOUNTING DIMENSIONS

This Appendix contains a drawing that can be used to locate the positions of the three mounting holes necessary to secure the SAGE^{MAX} to a wall.

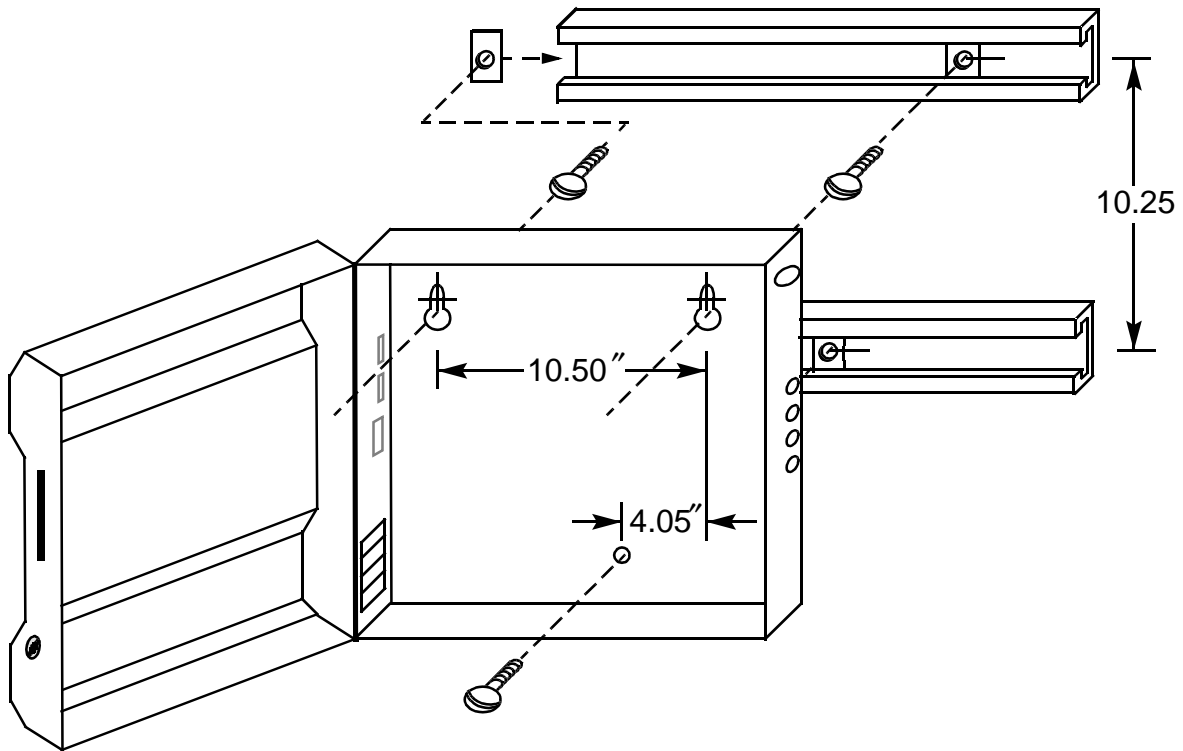


Figure N-1 Typical Mounting for the SAGE^{MAX}

APPENDIX O - FCC STANDARDS

The SAGE^{MAX} generates radio frequency energy, and if it is not installed and used properly--in strict accordance with the instructions in this manual--it may interfere with radio and television reception.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communication. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- re-orient or relocate the receiving antenna
- increase the separation between the equipment and receiver
- connect the equipment into an outlet on a circuit different from that to which the receiver is connected
- consult the dealer or an experienced radio/TV technician for help

The manufacturer is not responsible for any radio or TV interference caused by unauthorized modifications to this equipment. It is the responsibility of the user to correct such interference. If, after taking the above measures, you still have interference, consult your dealer or an experienced radio/television technician for additional suggestions. Also, the following booklet prepared by the FCC may be helpful.

“How to Identify and Resolve Radio/TV Interference Problems”

This booklet is available from the U.S. Government Printing Office, Washington, DC 20402. Request Stock No. 004-000-00345-4.

APPENDIX P - DATABASE CREATION SHEETS

This Appendix contains blank templates that may be copied and used when planning the database for the SAGE^{MAX}.

Included are database creation sheets for users, points, and groups.

Name (24) Description (64) Alarm Message Return Message	Net # ID #	Chan SChan	FType Card	EU File Privileges	Log Changes Famous Override Class
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO

APPENDIX Q - ASCII CODES

This Appendix lists and defines ASCII codes with their associated decimal and hexadecimal values.

ASCII CODE	DEC	HEX
NUL (null) (CTRL-@)	0	00h
SOH (start of heading) (CTRL-A)	1	01h
STX (start of text) (CTRL-B)	2	02h
ETX (end of text) (CTRL-C)	3	03h
EOT (end of transmission) (CTRL-D)	4	04h
ENQ (enquiry) (CTRL-E)	5	05h
ACK (acknowledge) (CTRL-F)	6	06h
BEL (bell) (CTRL-G)	7	07h
BS (backspace) (CTRL-H)	8	08h
HT (horizontal tab) (CTRL-I)	9	09h
LF (line feed) (CTRL-J)	10	0Ah
VT (vertical tab) (CTRL-K)	11	0Bh
FF (form feed) (CTRL-L)	12	0Ch
CR (carriage return) (CTRL-M)	13	0Dh
SO (shift out) (CTRL-N)	14	0Eh
SI (shift in) (CTRL-O)	15	0Fh
DLE (data link escape) (CTRL-P)	16	10h
DC1 (XON, resume) (CTRL-Q)	17	11h
DC2 (CTRL-R)	18	12h
DC3 (XOFF, stop) (CTRL-S)	19	13h
DC4 (CTRL-T)	20	14h
NAK (negative ack) (CTRL-U)	21	15h
SYN (synchronous idle) (CTRL-V)	22	16h
ETB (end of transmission block) (CTRL-W)	23	17h
CAN (cancel) (CTRL-X)	24	18h
EM (end of medium) (CTRL-Y)	25	19h
SUB (substitute character) (CTRL-Z)	26	1Ah
ESC (escape) (CTRL-[])	27	1Bh
FS (file separator) (CTRL-[])	28	1Ch
GS (group separator) (CTRL-[])	29	1Dh
RS (record separator) (CTRL-^)	30	1Eh

Table Q-1 ASCII Codes

ASCII CODE	DEC	HEX
US (unit separator) (CTRL-_)	31	1Fh
SP (space)	32	20h
! (exclamation point)	33	21h
" (quotation mark)	34	22h
# (number character)	35	23h
\$ (dollar sign)	36	24h
% (percent sign)	37	25h
& (ampersand)	38	26h
' (apostrophe)	39	27h
((left parenthesis)	40	28h
) (right parenthesis)	41	29h
* (asterisk)	42	2A
+ (plus sign)	43	2Bh
, (comma)	44	2Ch
- (hyphen)	45	2Dh
. (period)	46	2Eh
/ (slash)	47	2Fh
0	48	30h
1	49	31h
2	50	32h
3	51	33h
4	52	34h
5	53	35h
6	54	36h
7	55	37h
8	56	38h
9	57	39h
:	58	3Ah
;	59	3Bh
<	60	3Ch
=	61	3Dh
>	62	3Eh
?	63	3Fh
@	64	40h
A	65	41h

Table Q-1 ASCII Codes

ASCII CODE	DEC	HEX
B	66	42h
C	67	43h
D	68	44h
E	69	45h
F	70	46h
G	71	47h
H	72	48h
I	73	49h
J	74	4Ah
K	75	4Bh
L	76	4Ch
M	77	4Dh
N	78	4Eh
O	79	4Fh
P	80	50h
Q	81	51h
R	82	52h
S	83	53h
T	84	54h
U	85	55h
V	86	56h
W	87	57h
X	88	58h
Y	89	59h
Z	90	5Ah
[(left bracket)	91	5Bh
\ (back slash)	92	5Ch
] (right bracket)	93	5Dh
^ (carat)	94	5Eh
_ (underline)	95	5Fh
' (grave accent)	96	60h
a	97	61h
b	98	62h
c	99	63h
d	100	64h

Table Q-1 ASCII Codes

ASCII CODE	DEC	HEX
e	101	65h
f	102	66h
g	103	67h
h	104	68h
i	105	69h
j	106	6Ah
k	107	6Bh
l	108	6Ch
m	109	6Dh
n	110	6Eh
o	111	6Fh
p	112	70h
q	113	71h
r	114	72h
s	115	73h
t	116	74h
u	117	75h
v	118	76h
w	119	77h
x	120	78h
y	121	79h
z	122	7Ah
{ (left brace)	123	7Bh
(vertical line)	124	7Ch
} (right brace)	125	7Dh
~ (tilde)	126	7Eh
DEL (delete)	127	7Fh

Table Q-1 ASCII Codes

APPENDIX R - HEXADECIMAL EQUIVALENTS

This Appendix lists the decimal values of some commonly used hexadecimal numbers up to FFFFh.

Hex	Dec	Hex	Dec
0000h	00000	0020h	00032
0001h	00001	0021h	00033
0002h	00002	0022h	00034
0003h	00003	0023h	00035
0004h	00004	0024h	00036
0005h	00005	0025h	00037
0006h	00006	0026h	00038
0007h	00007	0027h	00039
0008h	00008	0028h	00040
0009h	00009	0029h	00041
000Ah	00010	002Ah	00042
000Bh	00011	002Bh	00043
000Ch	00012	002Ch	00044
000Dh	00013	002Dh	00045
000Eh	00014	002Eh	00046
000Fh	00015	002Fh	00047
0010h	00016	0030h	00048
0011h	00017	0031h	00049
0012h	00018	0032h	00050
0013h	00019	0033h	00051
0014h	00020	0034h	00052
0015h	00021	0035h	00053
0016h	00022	0036h	00054
0017h	00023	0037h	00055
0018h	00024	0038h	00056
0019h	00025	0039h	00057
001Ah	00026	003Ah	00058
001Bh	00027	003Bh	00059
001Ch	00028	003Ch	00060
001Dh	00029	003Dh	00061
001Eh	00030	003Eh	00062
001Fh	00031	003Fh	00063

Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
0040h	00064	0070h	00112	0100h	00256	3000h	12288
0041h	00065	0071h	00113	0200h	00512	3100h	12544
0042h	00066	0072h	00114	0300h	00768	3200h	12800
0043h	00067	0073h	00115	0400h	01024	3300h	13056
0044h	00068	0074h	00116	0500h	01280	3400h	13312
0045h	00069	0075h	00117	0600h	01536	3500h	13568
0046h	00070	0076h	00118	0700h	01792	3600h	13824
0047h	00071	0077h	00119	0800h	02048	3700h	14080
0048h	00072	0078h	00120	0900h	02304	3800h	14336
0049h	00073	0079h	00121	0A00h	02560	3900h	14592
004Ah	00074	007Ah	00122	0B00h	02816	3A00h	14848
004Bh	00075	007Bh	00123	0C00h	03072	3B00h	15104
004Ch	00076	007Ch	00124	0D00h	03328	3C00h	15360
004Dh	00077	007Dh	00125	0E00h	03584	3D00h	15616
004Eh	00078	007Eh	00126	0F00h	03840	3E00h	15872
004Fh	00079	007Fh	00127			3F00h	16128
0050h	00080	0080h	00128	1000h	04096	4000h	16384
0051h	00081	0081h	00129	1100h	04352	4100h	16640
0052h	00082	0082h	00130	1200h	04608	4200h	16896
0053h	00083	0083h	00131	1300h	04864	4300h	17152
0054h	00084	0084h	00132	1400h	05120	4400h	17408
0055h	00085	0085h	00133	1500h	05376	4500h	17664
0056h	00086	0086h	00134	1600h	05632	4600h	17920
0057h	00087	0087h	00135	1700h	05888	4700h	18176
0058h	00088	0088h	00136	1800h	06144	4800h	18432
0059h	00089	0089h	00137	1900h	06400	4900h	18688
005Ah	00090	008Ah	00138	1A00h	06656	4A00h	18944
005Bh	00091	008Bh	00139	1B00h	06912	4B00h	19200
005Ch	00092	008Ch	00140	1C00h	07168	4C00h	19456
005Dh	00093	008Dh	00141	1D00h	07424	4D00h	19712
005Eh	00094	008Eh	00142	1E00h	07680	4E00h	19968
005Fh	00095	008Fh	00143	1F00h	07936	4F00h	20224
0060h	00096	0090h	00144	2000h	08192	5000h	20480
0061h	00097	0091h	00145	2100h	08448	5100h	20736
0062h	00098	0092h	00146	2200h	08704	5200h	20992
0063h	00099	0093h	00147	2300h	08960	5300h	21248
0064h	00100	0094h	00148	2400h	09216	5400h	21504
0065h	00101	0095h	00149	2500h	09472	5500h	21760
0066h	00102	0096h	00150	2600h	09728	5600h	22016
0067h	00103	0097h	00151	2700h	09984	5700h	22272
0068h	00104	0098h	00152	2800h	10240	5800h	22528
0069h	00105	0099h	00153	2900h	10496	5900h	22784
006Ah	00106	009Ah	00154	2A00h	10752	5A00h	23040
006Bh	00107	009Bh	00155	2B00h	11008	5B00h	23296
006Ch	00108	009Ch	00156	2C00h	11264	5C00h	23552
006Dh	00109	009Dh	00157	2D00h	11520	5D00h	23808
006Eh	00110	009Eh	00158	2E00h	11776	5E00h	24064
006Fh	00111	009Fh	00159	2F00h	12032	5F00h	24320

Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
6000h	24576	9000h	36864	C000h	49152	F000h	61440
6100h	24832	9100h	37120	C100h	49408	F100h	61696
6200h	25088	9200h	37376	C200h	49664	F200h	61952
6300h	25344	9300h	37632	C300h	49920	F300h	62208
6400h	25600	9400h	37888	C400h	50176	F400h	62464
6500h	25856	9500h	38144	C500h	50432	F500h	62720
6600h	26112	9600h	38400	C600h	50688	F600h	62976
6700h	26368	9700h	38656	C700h	50944	F700h	63232
6800h	26624	9800h	38912	C800h	51200	F800h	63488
6900h	26880	9900h	39168	C900h	51456	F900h	63744
6A00h	27136	9A00h	39424	CA00h	51712	FA00h	64000
6B00h	27392	9B00h	39680	CB00h	51968	FB00h	64256
6C00h	27648	9C00h	39936	CC00h	52224	FC00h	64512
6D00h	27904	9D00h	40192	CD00h	52480	FD00h	64768
6E00h	28160	9E00h	40448	CE00h	52736	FE00h	65024
6F00h	28416	9F00h	40704	CF00h	52992	FF00h	65280
7000h	28672	A000h	40960	D000h	53248	FF00h	65280
7100h	28928	A100h	41216	D100h	53504	FF01h	65281
7200h	29184	A200h	41472	D200h	53760	FF02h	65282
7300h	29440	A300h	41728	D300h	54016	FF03h	65283
7400h	29696	A400h	41984	D400h	54272	FF04h	65284
7500h	29952	A500h	42240	D500h	54528	FF05h	65285
7600h	30208	A600h	42496	D600h	54784	FF06h	65286
7700h	30464	A700h	42752	D700h	55040	FF07h	65287
7800h	30720	A800h	43008	D800h	55296	FF08h	65288
7900h	30976	A900h	43264	D900h	55552	FF09h	65289
7A00h	31232	AA00h	43520	DA00h	55808	FF0Ah	65290
7B00h	31488	AB00h	43776	DB00h	56064	FF0Bh	65291
7C00h	31744	AC00h	44032	DC00h	56320	FF0Ch	65292
7D00h	32000	AD00h	44288	DD00h	56576	FF0Dh	65293
7E00h	32256	AE00h	44544	DE00h	56832	FF0Eh	65294
7F00h	32512	AF00h	44800	DF00h	57088	FF0Fh	65295
8000h	32768	B000h	45056	E000h	57344	FFF0h	65320
8100h	33024	B100h	45312	E100h	57600	FFF1h	65321
8200h	33280	B200h	45568	E200h	57856	FFF2h	65322
8300h	33536	B300h	45824	E300h	58112	FFF3h	65323
8400h	33792	B400h	46080	E400h	58368	FFF4h	65324
8500h	34048	B500h	46336	E500h	58624	FFF5h	65325
8600h	34304	B600h	46592	E600h	58880	FFF6h	65326
8700h	34560	B700h	46848	E700h	59136	FFF7h	65327
8800h	34816	B800h	47104	E800h	59392	FFF8h	65328
8900h	35072	B900h	47360	E900h	59648	FFF9h	65329
8A00h	35328	BA00h	47616	EA00h	59904	FFFAh	65330
8B00h	35584	BB00h	47872	EB00h	60160	FFFBh	65331
8C00h	35840	BC00h	48128	EC00h	60416	FFFCCh	65332
8D00h	36096	BD00h	48384	ED00h	60672	FFFDh	65333
8E00h	36352	BE00h	48640	EE00h	60928	FFFEh	65334
8F00h	36608	BF00h	48896	EF00h	61184	FFFFh	65535

APPENDIX S - SPECIAL SYSTEM VARIABLES

This Appendix lists special system variables that are used by the SAGE^{MAX}. These variables can be monitored from operator interfaces or used as terms in SPL program statements.

VARIABLE	MEANING
\$ALARMS	shows whether or not the SAGE ^{MAX} has any unacknowledged alarms. 0=no, 1=yes
\$MODE	can have a value from 0-255 and reflects the current mode used by the SAGE ^{MAX} global calendar
\$HOLIDAY	reflects the current holiday status as determined by the SAGE ^{MAX} global calendar. 0=normal, 80h (128)=holiday

Table S-1 Special System Variables

APPENDIX B - SUMMARY OF ERROR MESSAGES

This Appendix lists the error codes that are used by the SAGE^{MAX}. A brief description follows error codes whose meanings may be unclear.

Error numbers followed by an asterisk are *trappable errors*. When trappable errors occur (due to a read or write attribute request, for example) from within a SAGE^{MAX} program, you can use certain SAGE^{MAX} Programming Language (SPL) statements to handle the error condition in special ways. The uses of these programming statements are explained in detail in **Chapter 11: Programming**.

#	CODE	MEANING
0	#Success	successful (no errors)
1	#RequestInvalid	cannot do this request
2	#Argument Error	an argument is bad
3	#InvalidResponse	bad return from a query
4*	#CRCErrror	CRC Error
5*	Timeout	timed out waiting
6	#UnknownDatatype	not a standard data type
7*	#NAKResponse	some special error occurred
8	#InvalidChannel	bad channel number specified
9	#NoSuchAttribute	specified attribute does not exist
10	#NC@RecordDeleted	attempt to read a deleted record
11	#NC@NoSuchName	name was not found in search
12	#NC@TempFlushFailed	wild search file flush failed
13	#NC@TempCreateFailed	wild search file could not be created
14	#NC@LseekCacheFailed	wild search file seek failed
15	#NC@ReadCacheFailed	wild search file read failed
16	#NC@ObjectFileSeekFailed	database file rewind failed
17	#NC@ObjectFileOpenFailed	database file open failed
18	#NC@ObjectFileReadFailed	database file read failed
19	#NC@RecordLocked	could not delete record
20	#NC@NoSuchRecord	bad record number
21	#NoSuchObjectType	bad database type
22	#NC@NoFreeLocks	could not lock record

Table B-1 SAGE^{MAX} Error Messages

#	CODE	MEANING
23	#NCNC@ObjectFileWriteFailed	could not write record
24*	#TemporarilyBlocked	
25	#InvalidObjectName	
26	#InvalidTask	invalid task parameter specified
27	#OB@InvalidTimer	invalid timer handle specified
28	#OB@NoMoreTimers	no more timers available
29	#ObjectTypeNotInitialized	object type is being initialized
30	#UnknownPeer	unknown peer name
31	#InvalidDriver	invalid driver type
32	#NoSuchClass	class does not exist
33	#DOSTempCreateFailed	temporary file could not be created
34	#InvalidPort	invalid port number specified
35	#InvalidUnit	invalid unit number specified
36	#InvalidSession	invalid virtual terminal session
37	#InvalidService	protocol service unknown
38	#AlreadyAcknowledged	transaction already acknowledged
39	#NoSuchCard	card does not exist
40	#BadFtypeForCard	ftype invalid for card
41	#BadDatumSize	data size was rejected
42	#ConversionError	data conversion error
43*	#DataRejected	data was rejected
44	#BadUserOrCodeword	invalid user name or codeword
45	#NoFreeSessions	no free VT or file sessions
46	#PrivilegeViolation	attempted to perform a privileged operation
47	#VTAlreadyOpen	virtual terminal already open
48	#PortBusy	port is busy
49	#BadDrive	invalid drive number
50	#EndOfFile	end of file was reached
51	#NoSuchFile	file name specified does not exist
52	#BadFileHandle	invalid file handle number
53	#BadFileName	invalid file name format
54	#BadRecordNumber	invalid record number

Table B-1 SAGE^{MAX} Error Messages

#	CODE	MEANING
55	#FileHandleNotOpen	a specified file handle is not open
56	#FileInUse	specified file is currently being used
57	#FileHandleInUse	file number already assigned
58	#TransientFileTooBig	Job or OBL file was > 64K
59	#QueueFull	
60	#NoMatchFound	
61	#NoSuchName	
62	#MathConversionError	
63	#Underflow	
64	#Overflow	
65	#NotANumber	
66	#InvalidFormat	
67	#InvalidDataType	
68	#NoCompare	
69	#BadSampleInterval	
70	#InvalidTrendSignature	
71	#SampleLimitExceeded	
72	#TrendAlreadyEnabled	
73	#TrendNotFound	
74	#InvalidTrendFormat	
75	#NoAvailTmPackets	
76	#SampleFailureAlarm	
77	#NotCollected	
78	#NameUnknownToPeer	
79	#ChecksumError	
80	#InvalidProgramRegister	
81	#ProgramUnloaded	
82	#StackError	
83	#InvalidPcode	
84	#InvalidTerm	
85	#InvalidOperator	
86	#InvalidState	

Table B-1 SAGE^{MAX} Error Messages

#	CODE	MEANING
87	#TermError	
88	#ExpressionError	
89	#UnsuccessfulUnload	
90	#InvalidStateCharge	
91	#ProgramNotFound	
92	#IndexTooLarge	
93	unused	
94	unused	
95	unused	
96	#XOPRInvalidPassword	
97	#NotAnAVLFile	
98	#InvalidMessageNumber	
99	#InvalidLanguage	
100*	#DialBusy	
101	#NoPhoneNumber	
102*	#FailedToContact	
103	#NoResponseFromSite	
104 thru 199	unused	
200	#21@Success	DOS success
201	#21@InvalidFunction	
202	#21@FileNotFound	
203	#21@PathNotFound	
204	#21@NoFreeHandles	
205	#21@AccessDenied	
206	#21@InvalidHandle	
207	#21@MCBsDestroyed	
208	#21@OutOfMemory	
209	#21@InvalidAddress	
210	#21@InvalidEnvironment	

Table B-1 SAGE^{MAX} Error Messages

211	#21@InvalidFormat	
212	#21@InvalidAccessCode	
213	#21@InvalidData	
214	#21@ReservedError14	
215	#21@InvalidDrive	
216	#21@AttemptToRemoveCD	
217	#21@NotSameDevice	
218	#21@NoMoreFiles	
219	#21@DiskWriteProtected	
220	#21@UnknownDiskUnit	
221	#21@DriveNotReady	
222	#21@UnknownCommand	
223	#21@DataCRCError	
224	#21@BadRequestStructureLength	
225	#21@SeekError	
226	#21@UnknownMediaType	
227	#21@SectorNotFound	
228	#21@PrinterOutOfPaper	
229	#21@WriteFault	
230	#21@ReadFault	
231	#21@GeneralFailure	
232	#21@CannotCopyFileToItself	
233	#21@InvalidFileName	
234 thru 255	unused	

Table B-1 SAGE^{MAX} Error Messages

APPENDIX C - PHP ERROR CODES

This Appendix lists the possible PHP error codes that may be returned by a PHP slave device such as SAGE^{MAX}.

#	CODE	MEANING
00xx	BadCommand	xx is the invalid command ncode
01xx	BadSubUnit	xx is the invalid subunit number
0202	CRCErrror	subunit transaction had CRC error
0203	NoResponse	no response from subunit
0204	NoCard	card was not present
0205	BadType	card does not have fundamental type
0206	BadChannel	card does not have specific channel
0207	AttrNotFound	attribute requested could not be found
0208	BadData	data was rejected
0209	InvalidType	channel returned an unsupported data type
0302	NameNotFound	could not find named datum
0408	EOF	end of file (record number too big)
0500	BadID	access denied to this operator
0700	NoCache	invalid command (not all targets can cache)
0800	NoVirtual	target has no virtual terminal
0801	VirtualInUse	target's virtual terminal is busy
0900	RequestInvalid	request not recognized by target
0901	NAKResponse	message not received (negative acknowledgment)
0902	InvalidResponse	target's response was invalid
0903	InvalidService	requested service is not available

Table C-1 PHP Error Codes

APPENDIX D - PUP ERROR CODES

This Appendix lists the possible PUP error codes that may be returned by PUP devices.

CODE	MEANING
FFFFh	generic negative acknowledgment (NAK)
FFFEh	PUP command received correctly, but not supported
FFFDh	no such channel
FFFC	no such attribute
FFFBh	value for attribute not accepted
FFFAh - 8000h	reserved for generic NAK codes
7FFFh - 0000h	reserved for custom NAK codes

Table D-1 PUP Error Codes

APPENDIX E - XANP ERROR CODES

This Appendix lists the possible XANP error codes that may be returned by XANP devices.

CODE	MEANING
00h	card specified was not in configuration
01h	type specified was not appropriate for specified card
02h	channel specified was not appropriate for specified type
03h	attribute was not found
04h	datum size provided was not appropriate for attribute
05h	data conversion type was unsupported
06h	ISR data block CRC error
07h	data rejected
08h	invalid exception channel
09h	Peernet access error
0Ah	no port available for virtual terminal
0Bh	not in virtual terminal mode
0Ch	virtual terminal mode prevented by system variable
0Dh	virtual terminal already open
0Eh	port is busy
0Fh	bad close virtual terminal
10h	drive not ready
11h	bad drive number
12h	media not formatted
13h	disk not initialized
14h	media is unusable
15h	end of file encountered
16h	device full
17h	no such file(s)
18h	file not open
19h	disk write protected
1Ah	no room in directory

Table E-1 XANP Error Codes

CODE	MEANING
1Bh	bad filename
1Ch	device not mounted
1Dh	disk write-protected
1Eh	data CRC error
1Fh	seek error
20h	ID CRC error
21h	privilege violation
22h	FDC reset
23h	controller busy
24h	record not found
25h	FDC RAM test failed
26h	block number too large
27h	illegal directory or extent
28h	bad file number
29h	file name in use
2Ah	file number already assigned

Table E-1 XANP Error Codes

APPENDIX F - PEERNET ERROR CODES

This Appendix lists Peernet errors that are returned from the SAGE^{MAX}.

CODE	MEANING
xx01h	illegal command (xx is command code)
xx02h	point errors:
0002h	no such card
0102h	no such fundamental type
0202h	no such channel
0302h	no such attribute
0702h	data rejected
xx03h	access denied errors:
0203h	no port available

Table F-1 Peernet Error Codes

APPENDIX G - FUNDAMENTAL TYPES

This Appendix lists the fundamental point type codes used by XANP and Peernet networks on the SAGE^{MAX}.

NUMERIC CODE	TYPE CODE	POINT TYPE
00	SY	System Point
01	BI	Binary Input
02	BO	Binary Output
03	SS	Start/Stop Scheduling
04	AI	Analog Input
05	AO	Analog Output
06	AC	Analog Control
07	BC	Binary Control
08	MC	Motor Control
09	PG	Program
10	TR	Trend
11	TT	Terminal Type
12	TF	Terminal Flow
13	TC	Terminal Control
14	TD	Terminal Default
15		reserved

Table G-1 Fundamental Point Type Codes

APPENDIX H - PUP DATA TYPES

This Appendix lists the hexadecimal and decimal PUP numeric data types codes that are used by the SAGE^{MAX}.

The hexadecimal codes are followed by **h** and the decimal codes are provided in parentheses.

CODE	DIGIT FORMAT	MEANING
FFh (255)	XXXXXXXXXXXX.	signed 10 digit
FEh (254)	XXXXXXXXXXXX	unsigned 10 digit
FDh (253)	XXXXXXXXXXX.X	signed 9.1 digit
FCh (252)	XXXXXXXXXXX.X	unsigned 9.1 digit
FBh (251)	XXXXXXXXXX.XX	signed 8.2digit
FAh (250)	XXXXXXXXXX.XX	unsigned 8.2 digit
F9h (249)	XXXXXXXXX.XXX	signed 7.3 digit
F8h (248)	XXXXXXXXX.XXX	unsigned 7.3 digit
F7h (247)	XXXXXX.XXXX	signed 6.4digit
F6h (246)	XXXXXX.XXXX	unsigned 6.4 digit
F5h (245)	XXXXX.XXXXX	signed 5.5 digit
F4h (244)	XXXXX.XXXXX	unsigned 5.5 digit
F3h (243)	XXXX.XXXXXX	signed 4.6 digit
F2h (242)	XXXX.XXXXXX	unsigned 4.6 digit
F1h (241)	XXX.XXXXXXX	signed 3.7 digit
F0h (240)	XXX.XXXXXXX	unsigned 3.7 digit
EFh (239)	XX.XXXXXXXX	signed 2.8 digit
EEh (238)	XX.XXXXXXXX	unsigned 2.8 digit
EDh (237)	X.XXXXXXXXXX	signed 1.9 digit
ECh (236)	X.XXXXXXXXXX	unsigned 1.9 digit
EBh (235)	.XXXXXXXXXXXX	signed .10 digit
EAh (234)	.XXXXXXXXXXXX	unsigned .10 digit

Table H-1 PUP Data Types

CODE	DIGIT FORMAT	MEANING
E9h (233)	channel map	one bit per channel
E8h (232)	bitmap or text	one bit per text field
E7h (231)	BCD (H/M/S)	hours is LSB
E6h (230)	BCD (H/M/S)	hours is LSB
E5h (229)	packed BCD	8 BCD digits as 4 bytes
E4 (228)	BCD date	MSW is year LSW/MSB is month LSW/LSB is day
E3h (227)	Binary date	MSW is year LSW/MSB is month LSW/LSB is day
E2h (226)	reserved	
E1h (225)	reserved	
E0h (224)	IEEE 784 32-bit floating point	
Dfh - 80h (233 - 128)	reserved	
7Fh (127)	bogus data sample	LSW is SAGE ^{MAX} error code MSW is reserved
7Eh - 09h (126 - 9)	reserved	
08h (8)	1991-2117 1-12 1-31 0-23 0-59 1-7 reserved	year (bits 25-31 in 32-bit Dword) month (bits 21-24) day (bits 16-20) hour (bits 11-15) minute (bits 5-10) day of week (bits 2-4, 1=Sun) (bits 0-1)
07h (7)	0 or 1	0=no, 1=yes (MSBs=0)
06h (6)	ASCIIZ string	value is seg:offset of string
05h (5)	Sunday, etc.	DOS Day-of-week
04h (4)	HH:MM:SS	DOS-style time
03h (3)	XXXX:XXXX	LSW is offset MSW is segment
02h (2)	XXXXXXXXXh	hexadecimal double word
01h (1)	XXXXh	hexadecimal word
00h (0)	XXh	hexadecimal byte

Table H-1 PUP Data Types

APPENDIX I - TREND OBJECT FILES

This Appendix explains the structure of trend object files used by the SAGE^{MAX}.

Trend objects that are created on the SAGE^{MAX} are binary data files which contain information about database objects which have been monitored at a known fixed interval. Up to eight points may be monitored together in one trend. Trend object file names may contain up to eight characters, reside implicitly on the **C:\TRENDS** directory and are given the extension **.TRN**.

A trend object file contains a *header record* which is 512 bytes long and describes the database objects to be monitored. The structure of the header record is shown in **Table I-1**.

The header record begins with the text **TREND** and is followed by a trend type code (see **Table I-2**).

After the header record is a series of *data sample records*, each of which contains the values of up to eight database objects at a particular sample interval. Each data sample record also contains a four-byte time/date stamp, the structure of which is explained in **Table I-3**.

HEADER RECORD (512 bytes)	"TREND"	(5 bytes)
	Trend Type (see Table I-2)	(1 byte)
	Sample interval (minutes)	(2 bytes)
	Sample Count	(4 bytes)
	Number of Bogus Samples (Failures)	(2 bytes)
	Bogus Sample Alarm List	(2 bytes)
	Current Sample Number	(4 bytes)
	Maximum Allowable Samples for Infinite Trends	(4 bytes)
	Reserved	(232 bytes)
	32-byte ASCIIZ Variable Definition #1	(32 bytes)
	32-byte ASCIIZ Variable Definition #2	(32 bytes)
	32-byte ASCIIZ Variable Definition #3	(32 bytes)
	32-byte ASCIIZ Variable Definition #4	(32 bytes)
	32-byte ASCIIZ Variable Definition #5	(32 bytes)
	32-byte ASCIIZ Variable Definition #6	(32 bytes)
	32-byte ASCIIZ Variable Definition #7	(32 bytes)
32-byte ASCIIZ Variable Definition #8	(32 bytes)	
DATA SAMPLE #1	Special Data Type (08h) of Time/Date Stamp	(1 byte)
	Time/Date Stamp (32 bits, see Table I-3)	(4 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #1	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #2	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #3	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #4	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #5	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #6	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #7	(5 bytes)
	Data Type (1 byte) = Sample Value (4 bytes) #8	(5 bytes)

Table I-1 Header Record and Sample Data Record for Trend Object Files

	TREND TYPE	TYPE CODE
	infinite = circular = averaged =	0 (00h) 1 (01h) 128 (80h)
DAILY TIME-ANCHORED	every 1 minute = every 15 minutes = every 30 minutes = every 60 minutes =	2 (02h) 3 (03h) 4 (04h) 5 (05h)
WEEKLY TIME-ANCHORED	every 1 minute = every 15 minutes = every 30 minutes = every 60 minutes = every day =	6 (06h) 7 (07h) 8 (07h) 9 (09h) 10 (0Ah)
MONTHLY TIME-ANCHORED	every 1 minute = every 15 minutes = every 30 minutes = every 60 minutes = every day =	11 (0Bh) 12 (0Ch) 13 (0Dh) 14 (0Eh) 15 (0Fh)
YEARLY TIME-ANCHORED	every 1 minute = every 15 minutes = every 30 minutes = every 60 minutes = every day = every week = every month =	16 (10h) 17 (11h) 18 (12h) 19 (13h) 20 (14h) 21 (15h) 22 (16h)

Table I-2 Trend Types

FIELD	RANGE	BIT NUMBERS IN 32-BIT DWORD
Year	1991-2117	31-25
Month	1-12	21-21
Day	1-31	20-16
Hour	0-23	15-11
Minute	0-59	10-5
Day of Week	1-7	4-2 (1=Sunday, 7=Saturday)
reserved		1-0

Table I-3 Structure of the Time/Date Stamp

APPENDIX J - NAME BINDING FILES

Name Binding Files (NBFs) are ASCII text files (with the extension **.NBF**) that you use for off-line database creation of points, programs, variables and globals. NBFs are translated into Binary Object (BOB) files using a SAGE^{MAX} file conversion utility.

There are three basic types of lines in a Named Binding File. *Comment lines* are lines that begin with a semicolon (;) and are ignored by the SAGE^{MAX} conversion program. *Binding Definition lines* are used to specify an object type and a series of parameters to define the object. *Continuation lines* begin with **TABs** and are available when it is impractical to fit the entire Name Binding definition on one line.

Binding Definition Lines can contain parameters that are optional. These optional parameters do not have to be included in Binding Definition Lines. The position of these parameters in the Binding Definition Line is very important in determining the proper syntax for the line.

For example, if only the first two parameters of a line are used, the remainder of the line can remain blank. If, however, the first three and last two parameters of a Binding Definition Line are used, you must use commas to indicate parameters that have been omitted.

The tables in this Appendix describe the required format of the Binding Definition lines for Name Binding Files (NBFs) and illustrate several examples of each type.

All numeric values shown in the following tables are assumed to be decimal unless followed by “H” or “h” which indicates hexadecimal.

Format: GL *name*=*peername*,*euname*

where

GL	is a two-letter mnemonic code used to identify global objects
<i>name</i>	is the name of the global object
<i>peername</i>	is the peername associated with the global object
<i>euname</i>	is the name of the optional Engineering Units file for the global object.

Examples: GL OutsideAirTemp=Main Building SAGE^{MAX},STDAI
GL OAT= Main Building SAGE^{MAX}

Table J-1 Binding Definition Format for Global Objects

Format:VR *name*=*type,value,famous,log,objprivs*

where

VR	is a two-letter mnemonic code used to identify variable objects
<i>name</i>	is the name of the variable object
<i>type</i>	is the numeric data type code (See Appendix H: PUP Data Types)
<i>value</i>	is the value of the variable object
<i>famous</i>	is the optional famous flag used to specify if the variable object is famous FAMOUSglobally announces changes to this variable (blank)no global announcement of changes
<i>log</i>	is the optional log flag used to specify whether or not changes made to this variable by operators should be logged as operator events LOGlog attribute changes (by operators) as operator events (blank)do not log changes
<i>objprivs</i>	is the optional object privilege bitmap associated with this variable.

Examples:VR hexbyte=00h,1Bh
 VR hexword=01h,FE00h
 VR hexdword=02h,12345678h
 VR DOStime=04h,12:34:56
 VR happyday=05h,friday
 VR occupied=07h,false
 VR holiday=07h,true
 VR Summer=07,Yes,Famous,Log,0001011b
 VR emergency=07,No
 VR pi=E0h,3.14159
 VR noon=E6h,12:00
 VR MidNight=E7h,00:00:00
 VR flags=E8h,00001011b
 VR HumanTemp=FEh,98.6,,01011111b

Table J-2 Binding Definition Format for Variable Objects

Format:PG *name=PLB,PRB,INI,lflag,rflag,preload,euname,famous,log,objprivs*

where

<i>PG</i>	is a two-letter mnemonic code used to identify program objects
<i>name</i>	is the name of the program object
<i>PLB</i>	is a path fragment that identifies the Program Logic Block
<i>PRB</i>	is a path fragment that identifies the Program Reference Block
<i>INI</i>	is a path fragment that identifies the program attribute Initial Values File
<i>lflag</i>	is a flag used to identify the nature of the PLB FILEprogram is file-resident STICKYprogram cannot be unloaded once it is loaded into RAM
<i>rflag</i>	is a flag used to identify the nature of the PRB FILEPRB is always file-resident RAMPRB is RAM-resident, but unloadable STICKYPRB cannot be unloaded once it is loaded into RAM
<i>preload</i>	is a flag used to specify when the program should be loaded PRELOADload program after SAGE reboot (blank)load program on demand
<i>euname</i>	is the name of the Engineering Units file for the program object
<i>famous</i>	is the famous flag used to specify if the program object is famous FAMOUSglobally announces changes to this program (blank)no global announcement of changes
<i>log</i>	is the log flag used to specify whether or not changes made to this program's attributes by operators should be logged as operator events LOGlog attribute changes (by operators) as operator events (blank)do not log changes
<i>objprivs</i>	is the optional object privilege bitmap associated with this program

Examples:

PG AHU39OSS=OSS,AHU39,File,File

PG PumpB=LeadLag,BPumps,,Sticky,RAM,Preload,LeadLag,Famous,Log

Table J-3 Binding Definition Format for Program Objects

Format:

PT *name*=*port,unit,chan,ftype,card,subchan,euname,famous,log,objprivs,ovrclass*
 (*tab*) *point description message*
 (*tab*) *alarm message*
 (*tab*) *return message*

where

PT is a two-letter mnemonic code used to identify point objects
name is the name of the point object
port is the port number of the network to be accessed
unit is the unit number of the network device containing the point
chan is the network channel number
ftype * is the fundamental type code (or numeric equivalent) of the point
 (see **Appendix G:Fundamental Types**)
card * is the card associated with the point
subchan * is the subchannel associated with the point
euname is the name of the Engineering Units file for the point object.
famous is the famous flag used to specify if the point object is famous
 FAMOUSglobally announces changes to this point
 (blank)no global announcement of changes
log is the log flag used to specify whether or not changes made to this
 point's attributes by operators should be logged as operator events
 LOGlog attribute changes (by operators) as operator events
 (blank)do not log changes
objprivs is the optional object privilege bitmap associated with this point
ovrclass specifies a SAGE^{MAX} class number to be used to override all alarm
 classes determined by the driver for this point. Zero means use the
 underlying alarm class.

* *ftype*, *card* and *subchan* parameters are only required for XANP and Peernet networks.

Table J-4 Binding Definition Format for Point Objects

APPENDIX K - SPL PSEUDOCODES

This Appendix lists and explains the pseudocodes (Pcodes) used by the SAGE^{MAX} Program Language (SPL) for statement, term and operator encoding. Refer to **Table K-1**, **Table K-2**, & **Table K-3**.

PCODE(S)	SOURCE FORM AND COMMENTS
00	Stop (this program)
01tt..tt00	tt..tt: (label "tt..tt")
02xxxx	Section xxxx
03iexpr	Swait iexpr
04iexpr	Mwait iexpr
05expr	Wait expr
06pp..pp00	Call plb "pp..pp"
07pp..pp00	Call plb "pp..pp" and stick
08	Return or Unload (this program)
09xxxx expr	If expr then xxxx (xxxx is offset to "jump to")
0Axxxxyyy expr	If expr xxxx else yyyy
0Bxxxx	Goto xxxx
0Cnn aaaa bbbb cccc ... expr	On expr Goto aaaa, bbbb, cccc... (nn=# of choices)
0D0rxxxx	Loop register r, label xxxx
0Eaabbexpr	program attribute "aabb"=expr
0Fpp..pp00expr	object attribute path "pp..pp"=expr
1xexpr	Register=expr (for register A, x=0) (for register B, x=1) (for register C, x=3) . . (for register O, x=E) (for register P, x=F)
20expr1 expr2	Ref (expr1)=expr2 (expr1=0-255)
21tt..tt00 expr1 expr2	&tab;e (expr1)=expr2 ("tt..tt" is table fragment)
22px ux fx cx nx aabb expr	UNS (px,ux,fx,cx,nx,attr)=expr (px=portexpr 0-15) (cx=cardexpr 0-255) (ux=unitexpr 0-65535) (nx=chanexpr 0-65535) (fx=ftypeexpr 0-255) (aabb=attribute)

Table K-1 SPL Pseudocodes for Statement Encoding

PCODE(S)	SOURCE FORM AND COMMENTS
23pp..pp00	Activate program name "pp..pp"
24pp..pp00	Deactivate program name "pp..pp"
25pp..pp00	Restart program name "pp..pp"
26pp..pp00	Stop program name "pp..pp"
27expr pp..pp00	Spool portexpr, pathname "pp..pp"
28expr pp..pp00	Spool portexpr, pathname "pp..pp", DELETE
29nn classexpr ff..ff00 expr1, expr2...exprN	Alarm classexpr, "formatstring",x,x,x,x... (nn=# of expressions, 8 max)
2Ann portexpr classexpr ff..ff00 expr1...exprN	Print portexpt classexpr, "formatstring", x,x,x,x... (nn=# of expressions, 8 max)
2Bnn ll..ll00 ff..ff00 expr1, expr2...exprN	Log logfilemane "ll..ll",,"formatstring", x,x,x,x... (nn=# of expressions, 8 max)
2Cnn classexpr ff..ff00 expr1, expr2...exprN	Job classexpr, "formatstring", x,x,x,x... (nn=# of expressions, 8 max)
2D	No operation (NOP)
2Exxxx	Gosub xxxx
2Fxxxx	Oneror xxxx
30	Errorwait
31	Errorabort
32xxaabbcc	Save aa, bb, cc (xx=# of attributes, if xx=0 save all)
33tttttt00	Starttrend trend name "ttttttt"
34tttttt00	Stoptrend trend name "ttttttt"
35-FF	reserved

Table K-1 SPL Psuedocodes for Statement Encoding

PCODE(S)	MEANING/FORMAT	VALUE/RANGE	TYPE
00	reserved		
01	one	1	FF integer
02	minus one	-1	FF integer
03	day of month	1..31	FE integer
04	month of year	1..12	FE integer
05	current year	e.g., 1996	FE integer
06	current day of week	0..6 (0=Monday)	FE integer
07	current Julian day	1..366	FE integer
08	current time	00:00:00..23:59:59	E7 time
09ttbb	typed byte	bb	tt
0Attwww	typed word	www	tt
0Bttddddddd	typed constant	ddddddd	tt
0C	value of PI	3.141593	E0 float
0D	zero	0	FF integer
0Eaabb	program attr "aabb"		??
0Fpp..pp00	object attr path "pp..PP"		??
1x	register A-P	x=0-F	??
20expr	REF (expr)	expr value=0-255	??
21tt..tt00expr	&table (expr)	table path "tt..tt"	??
22px ux fx cx nx aabb	UNS (px,ux,fx,cx,nx,attr) px=portexpr 0-15 ux=unitexpr 0-65535 fx=ftypeexpr 0-255 cx=cardexpr 0-255 nx=chanexpr 0-65535 aabb=attribute	0-15 0-65535 0-255 0-255 0-65535	??
23expr	(expr)		??
24expr	-expr	unary negate	??
25iexpr	NOT iexpr	unary 1's complement	FE integer
26fexpr	INT (fx)	convert float to integer	FF integer
27fexpr iexpr	FIX (fx,ix)	convert float to fixed pt.	?? integer
28iexpr	FLOAT (ix)	convert integer to float	E0 float
29expr	ABS (x)	absolute value	??

Table K-2 SPL Pseudocodes for Term Encoding

PCODE(S)	MEANING/FORMAT	VALUE/RANGE	TYPE
2Aexpr	ROUND (fx)	round off	E0 float
2Bexpr	SIN (fx)	Sine value	E0 float
2Cexpr	COS (fx)	Cosine value	E0 float
2Dexpr	TAN (fx)	Tangent value	E0 float
2Eexpr	ARCTAN (fx)	Arctangent value	E0 float
2Fexpr	LOG (fx)	Logarithm	E0 float
30expr	LN (fx)	Natural log	E0 float
31expr	SQRT (x)	Square root	??
32texpr1 texpr2	Between two times	0=not between,1=between	FE integer
33xx..xx00	MEAN (x,x...x,x)	Mean value from expr list	??
34xx..xx00	MAX (x,x...x,x)	Max value from expr list	??
35xx..xx00	MIN (x,x...x,x)	Min value from expr list	??
36iexpr	TODAY (ix)	Is today this (one of these) day(s) of week? 0=no, 1=yes	FE integer
37	MON	0	FE integer
38	TUE	1	FE integer
39	WED	2	FE integer
3A	THU	3	FE integer
3B	FRI	4	FE integer
3C	SAT	5	FE integer
3D	SUN	6	FE integer
3Eexpr iexpr	RETYPE (x,ix)	convert to a data type	??
3F	DATATYPE (x)	return data type of expr	FE integer
40	DATE	current date	E3 hex date
41-FF	reserved		

Table K-2 SPL Pseudocodes for Term Encoding

PCODE(S)	DESCRIPTION	EXAMPLE	NOTES
00	End of Expression		
01	Exponential	A**B	A to the B power
02	Multiplication	A*B	
03	Division	A/B	
04	Remainder after Division	A MOD B	7 MOD 3=1
05	Addition	A+B	
06	Subtraction	A-B	
07	Equality	A==B	0 if not, 1 if equal
08	Inequality	A<>B	1 if not, 0 if equal
09	Greater than	A>B	1 if A>B, then 0
0A	Greater than or Equal	A>=B	1 if A.B or A=B, else 0
0B	Less than	A<B	1 if A<B, else 0
0C	Less than or Equal	A<=B	1 if A,B or A=B, else 0
0D	Bitwise AND	A AND B	32-bit integer
0E	Bitwise OR	A OR B	32-bit integer
0F	Bitwise Exclusive OR	A XOR B	32-bit integer
10	Bitwise Shift left	A SHL B	1 SHL 3=8
11	Bitwise Shift Right	A SHR B	256 SHR 7=2
12-FF	reserved		

Table K-3 SPL Pseudocodes for Expression Encoding

APPENDIX L - GROUP FILES

This Appendix explains the format of group files used by the SAGE^{MAX}. Group files are text files that contain editable lines of ASCII text. There are three types of line in a group file:

- comment lines
- subgroup path lines
- object reference lines

Although group files may contain an unlimited number of editable lines, SAGE^{MAX} will only use the first 17 lines of the file.

Comment lines begin with a semicolon (;) which must be the first character of the line. The remainder of the line is ignored.

If the first line of the file is a comment, the text following the semicolon is assumed to be the group description.

Subgroup path lines must have a backslash (\) character as the first character in the line. The backslash is followed by a pathname fragment (up to 26 characters in length) which specifies up to two subdirectories and a file name. This pathname is always implicitly preceded by **\GROUPS** and any extension (if present) is stripped off. The extension of group files is **.GRP**.

The pathname may optionally be followed by a **TAB** and description text. If present, this text appears when the group is displayed as a menu. The description should serve as an explanation about the intended use of the subgroup.

Object reference lines contain a reference to an object, one of its attributes and an optional description of the reference. The reference may be specified in any one of four forms:

- object name
- object name;attribute
- type\object name
- type\object name;attribute

If present, the object type precedes the object name and is followed by a backslash (\). Valid object type codes are:

- PT point
- PG program
- VR variable
- GL global

The object name can be up to 24 characters long. The object name may be optionally followed by a semicolon (;) and a two-character attribute. If the attribute name is not present, the default attribute of the specified object is assumed. If the type is not specified, the SAGE^{MAX} searches all object types to find the named object.

When a group is displayed by the operator interface, each group member is handled in a particular way. If the group member is a subgroup, then its name is displayed followed by any description text which follows the group name in the group file.

If the group member is a comment line, it is not displayed by the operator interface. The comment line is used only for documenting the group file.

If a group member is an object reference, then it is displayed in one of three forms depending on the presence or absence of a description message. If the reference has a description after it in the group file, up to 33 characters of that description are displayed preceding the value of the point attribute. If there is no description in the group file for a particular member and the object referenced was a point, then the point message file is examined to see if the point has a corresponding point description message. If so, up to 33 characters of the point description message are displayed. In every other case, only the object reference (type, name and attribute) is displayed. **Figure L-1** illustrates a sample group file.

```

;Floor 1 Zone Temperatures
PT\F1_ROOM101;CV      Zone Temp Floor 1 Room 101
PT\F1_ROOM102;CV      Zone Temp Floor 1 Room 102
PT\F1_ROOM105;CV      Zone Temp Floor 1 Room 105
PT\F1_ROOM106;CV      Zone Temp Floor 1 Room 106
PT\F1_ROOM108;CV      Zone Temp Floor 1 Room 108
PT\F1_ROOM109;CV      Zone Temp Floor 1 Room 109
PT\F1_ROOM110;CV      Zone Temp Floor 1 Room 110
PT\F1_ROOM112;CV      Zone Temp Floor 1 Room 112
\SETPTS\FLR1\F1SETPTS      Control Setpoints Floor 1
\SCHEDS\FLR1\F1SCHEDS      Schedules for Floor 1
\LIGHTS\FLR1\F1LIGHTS      Lighting Control for Floor 1
\OVRIDS\FLR1\F1OVRIDS      Control Overrides for Floor 1
\PROGS\FLR1\PROGRAMS      SPL Programs for Floor 1

\MAIN      Return To MAIN Group
;
;
;*****
;  Group : \F1TEMPSFloor 1 Zone Temperatures
;
;  This group file is a subgroup of group MAIN, and shows
;  the zone temps of rooms on the first floor.
;*****

```

Figure L-1 Sample Group File

APPENDIX M - DCS FILES

This Appendix explains the structure of the Points Definition Files (**.PDFs**) used by the data capture/data stuff (**DCS**) background job of the SAGE^{MAX}.

The Points Definition File (**.PDF**) contains lines of text which adhere to one of several possible formats. The lines may be up to 128 characters long before the carriage return and linefeed (CRLF). Extra characters will be ignored. The seven possible line formats for **.PDF** files are:

- ;comment line
- >pathname
- >>pathname
- objectname,description
- !pathname
- !!pathname
- objectname=value

Lines which begin with a semicolon (;) character are treated as comment lines and are ignored by **DCS**.

The >*pathname* format causes further data capture list output to be directed to *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain *wildcards* as described later in this section.

The >>*pathname* format cause further data capture list output to be appended to *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain *wildcards* as described later in this section.

The *objectname,description* format will read (capture) the named object/attribute and output its value to the list output file. If the /S switch is present in the job string, then the output is formatted in a form that is suitable for subsequent use as a data stuff **.PDF** file. The *description* is any arbitrary text, presumably used to describe the named object in more detail.

The !*pathname* format causes further data stuff audit output to be directed to *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain wildcards as described in the following sections.

The !!*pathname* causes further data stuff audit output to be appended to *pathname*. If *pathname* does not exist, then it will be created. The *pathname* can contain wild cards as described later in this section.

The *object=value* format is used to write (stuff) the named object attribute with the new *value*. The acceptable formats for the new value are described in **Table M-1**.

Alternately, if the /S option in the job string is selected, information is formatted for data stuffing. For example:

SOMEPOINT;AT=75.3

The .PDF file can contain as many lines as desired. There is no limit to the number of times that the list output may be redirected. The pathname for list/audit output is verified before a new list output file is opened or appended. This means that **DCS** checks for the existence of each subdirectory in the pathname. If a subdirectory does not exist, then it is created automatically by **DCS**.

CAUTION

You must be careful to use the >, >>, ! and !! formats correctly, since they may overwrite existing files!

The >, >>, ! and !! formats allow the percent (%) character to appear in the pathname. If it is used, the % character must be followed by a single letter code which indicates one of the time and date values shown in **Table M-2**. You can specify any combination of these time and date codes in conjunction with normal pathname characters to form unique time/date-based pathnames for **DCS** files. This allows the generation of time-ordered files by **DCS**. Since **DCS** automatically creates subdirectories, you may freely use these % codes in subdirectory names, if desired.

% CODE	TIME UNIT	DIGITS GENERATED
N	month	01-12
D	day of month	01-31
C	year of century	00-99
Y	year	e.g., 1991
H	hour	00-23
M	minute	00-59
W	week of year	01-52
K	day of week	1-7, 1=SUN
J	day of year	001-366

Table M-2 Wildcards Used in >, >>, ! and !! Pathnames

For example, the pathname

>LOG\%W%K%H%M.PRN

might be used as the name of a data capture list file which was run every day, perhaps several times a day. The resulting files would show the week of the year (%W), day of the week (%K), and time (%H%M) as the file name, e.g., **14021335.PRN**.

APPENDIX N - SAGE^{MAX} MOUNTING DIMENSIONS

This Appendix contains a drawing that can be used to locate the positions of the three mounting holes necessary to secure the SAGE^{MAX} to a wall.

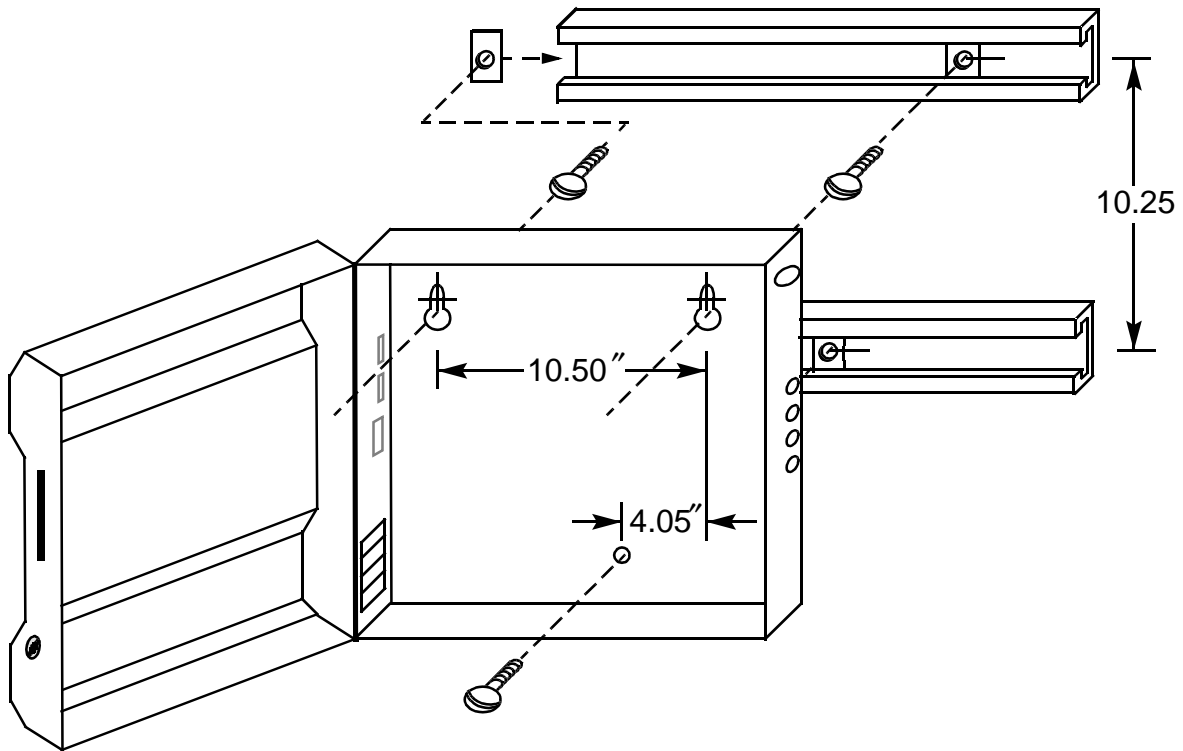


Figure N-1 Typical Mounting for the SAGE^{MAX}

APPENDIX O - FCC STANDARDS

The SAGE^{MAX} generates radio frequency energy, and if it is not installed and used properly--in strict accordance with the instructions in this manual--it may interfere with radio and television reception.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communication. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- re-orient or relocate the receiving antenna
- increase the separation between the equipment and receiver
- connect the equipment into an outlet on a circuit different from that to which the receiver is connected
- consult the dealer or an experienced radio/TV technician for help

The manufacturer is not responsible for any radio or TV interference caused by unauthorized modifications to this equipment. It is the responsibility of the user to correct such interference. If, after taking the above measures, you still have interference, consult your dealer or an experienced radio/television technician for additional suggestions. Also, the following booklet prepared by the FCC may be helpful.

“How to Identify and Resolve Radio/TV Interference Problems”

This booklet is available from the U.S. Government Printing Office, Washington, DC 20402. Request Stock No. 004-000-00345-4.

APPENDIX P - DATABASE CREATION SHEETS

This Appendix contains blank templates that may be copied and used when planning the database for the SAGE^{MAX}.

Included are database creation sheets for users, points, and groups.

Name (24) Description (64) Alarm Message Return Message	Net # ID #	Chan SChan	FType Card	EU File Privileges	Log Changes Famous Override Class
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO
					YES NO

Figure P-2 Database Creation Sheet for Points

APPENDIX Q - ASCII CODES

This Appendix lists and defines ASCII codes with their associated decimal and hexadecimal values.

ASCII CODE	DEC	HEX
NUL (null) (CTRL-@)	0	00h
SOH (start of heading) (CTRL-A)	1	01h
STX (start of text) (CTRL-B)	2	02h
ETX (end of text) (CTRL-C)	3	03h
EOT (end of transmission) (CTRL-D)	4	04h
ENQ (enquiry) (CTRL-E)	5	05h
ACK (acknowledge) (CTRL-F)	6	06h
BEL (bell) (CTRL-G)	7	07h
BS (backspace) (CTRL-H)	8	08h
HT (horizontal tab) (CTRL-I)	9	09h
LF (line feed) (CTRL-J)	10	0Ah
VT (vertical tab) (CTRL-K)	11	0Bh
FF (form feed) (CTRL-L)	12	0Ch
CR (carriage return) (CTRL-M)	13	0Dh
SO (shift out) (CTRL-N)	14	0Eh
SI (shift in) (CTRL-O)	15	0Fh
DLE (data link escape) (CTRL-P)	16	10h
DC1 (XON, resume) (CTRL-Q)	17	11h
DC2 (CTRL-R)	18	12h
DC3 (XOFF, stop) (CTRL-S)	19	13h
DC4 (CTRL-T)	20	14h
NAK (negative ack) (CTRL-U)	21	15h
SYN (synchronous idle) (CTRL-V)	22	16h
ETB (end of transmission block) (CTRL-W)	23	17h
CAN (cancel) (CTRL-X)	24	18h
EM (end of medium) (CTRL-Y)	25	19h
SUB (substitute character) (CTRL-Z)	26	1Ah
ESC (escape) (CTRL-[])	27	1Bh
FS (file separator) (CTRL-[])	28	1Ch
GS (group separator) (CTRL-[])	29	1Dh
RS (record separator) (CTRL-^)	30	1Eh

Table Q-1 ASCII Codes

ASCII CODE	DEC	HEX
US (unit separator) (CTRL-_)	31	1Fh
SP (space)	32	20h
! (exclamation point)	33	21h
" (quotation mark)	34	22h
# (number character)	35	23h
\$ (dollar sign)	36	24h
% (percent sign)	37	25h
& (ampersand)	38	26h
' (apostrophe)	39	27h
((left parenthesis)	40	28h
) (right parenthesis)	41	29h
* (asterisk)	42	2A
+ (plus sign)	43	2Bh
, (comma)	44	2Ch
- (hyphen)	45	2Dh
. (period)	46	2Eh
/ (slash)	47	2Fh
0	48	30h
1	49	31h
2	50	32h
3	51	33h
4	52	34h
5	53	35h
6	54	36h
7	55	37h
8	56	38h
9	57	39h
:	58	3Ah
;	59	3Bh
<	60	3Ch
=	61	3Dh
>	62	3Eh
?	63	3Fh
@	64	40h
A	65	41h

Table Q-1 ASCII Codes

ASCII CODE	DEC	HEX
B	66	42h
C	67	43h
D	68	44h
E	69	45h
F	70	46h
G	71	47h
H	72	48h
I	73	49h
J	74	4Ah
K	75	4Bh
L	76	4Ch
M	77	4Dh
N	78	4Eh
O	79	4Fh
P	80	50h
Q	81	51h
R	82	52h
S	83	53h
T	84	54h
U	85	55h
V	86	56h
W	87	57h
X	88	58h
Y	89	59h
Z	90	5Ah
[(left bracket)	91	5Bh
\ (back slash)	92	5Ch
] (right bracket)	93	5Dh
^ (carat)	94	5Eh
_ (underline)	95	5Fh
' (grave accent)	96	60h
a	97	61h
b	98	62h
c	99	63h
d	100	64h

Table Q-1 ASCII Codes

ASCII CODE	DEC	HEX
e	101	65h
f	102	66h
g	103	67h
h	104	68h
i	105	69h
j	106	6Ah
k	107	6Bh
l	108	6Ch
m	109	6Dh
n	110	6Eh
o	111	6Fh
p	112	70h
q	113	71h
r	114	72h
s	115	73h
t	116	74h
u	117	75h
v	118	76h
w	119	77h
x	120	78h
y	121	79h
z	122	7Ah
{ (left brace)	123	7Bh
(vertical line)	124	7Ch
} (right brace)	125	7Dh
~ (tilde)	126	7Eh
DEL (delete)	127	7Fh

Table Q-1 ASCII Codes

APPENDIX R - HEXADECIMAL EQUIVALENTS

This Appendix lists the decimal values of some commonly used hexadecimal numbers up to FFFFh.

Hex	Dec	Hex	Dec
0000h	00000	0020h	00032
0001h	00001	0021h	00033
0002h	00002	0022h	00034
0003h	00003	0023h	00035
0004h	00004	0024h	00036
0005h	00005	0025h	00037
0006h	00006	0026h	00038
0007h	00007	0027h	00039
0008h	00008	0028h	00040
0009h	00009	0029h	00041
000Ah	00010	002Ah	00042
000Bh	00011	002Bh	00043
000Ch	00012	002Ch	00044
000Dh	00013	002Dh	00045
000Eh	00014	002Eh	00046
000Fh	00015	002Fh	00047
0010h	00016	0030h	00048
0011h	00017	0031h	00049
0012h	00018	0032h	00050
0013h	00019	0033h	00051
0014h	00020	0034h	00052
0015h	00021	0035h	00053
0016h	00022	0036h	00054
0017h	00023	0037h	00055
0018h	00024	0038h	00056
0019h	00025	0039h	00057
001Ah	00026	003Ah	00058
001Bh	00027	003Bh	00059
001Ch	00028	003Ch	00060
001Dh	00029	003Dh	00061
001Eh	00030	003Eh	00062
001Fh	00031	003Fh	00063

Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
0040h	00064	0070h	00112	0100h	00256	3000h	12288
0041h	00065	0071h	00113	0200h	00512	3100h	12544
0042h	00066	0072h	00114	0300h	00768	3200h	12800
0043h	00067	0073h	00115	0400h	01024	3300h	13056
0044h	00068	0074h	00116	0500h	01280	3400h	13312
0045h	00069	0075h	00117	0600h	01536	3500h	13568
0046h	00070	0076h	00118	0700h	01792	3600h	13824
0047h	00071	0077h	00119	0800h	02048	3700h	14080
0048h	00072	0078h	00120	0900h	02304	3800h	14336
0049h	00073	0079h	00121	0A00h	02560	3900h	14592
004Ah	00074	007Ah	00122	0B00h	02816	3A00h	14848
004Bh	00075	007Bh	00123	0C00h	03072	3B00h	15104
004Ch	00076	007Ch	00124	0D00h	03328	3C00h	15360
004Dh	00077	007Dh	00125	0E00h	03584	3D00h	15616
004Eh	00078	007Eh	00126	0F00h	03840	3E00h	15872
004Fh	00079	007Fh	00127			3F00h	16128
0050h	00080	0080h	00128	1000h	04096	4000h	16384
0051h	00081	0081h	00129	1100h	04352	4100h	16640
0052h	00082	0082h	00130	1200h	04608	4200h	16896
0053h	00083	0083h	00131	1300h	04864	4300h	17152
0054h	00084	0084h	00132	1400h	05120	4400h	17408
0055h	00085	0085h	00133	1500h	05376	4500h	17664
0056h	00086	0086h	00134	1600h	05632	4600h	17920
0057h	00087	0087h	00135	1700h	05888	4700h	18176
0058h	00088	0088h	00136	1800h	06144	4800h	18432
0059h	00089	0089h	00137	1900h	06400	4900h	18688
005Ah	00090	008Ah	00138	1A00h	06656	4A00h	18944
005Bh	00091	008Bh	00139	1B00h	06912	4B00h	19200
005Ch	00092	008Ch	00140	1C00h	07168	4C00h	19456
005Dh	00093	008Dh	00141	1D00h	07424	4D00h	19712
005Eh	00094	008Eh	00142	1E00h	07680	4E00h	19968
005Fh	00095	008Fh	00143	1F00h	07936	4F00h	20224
0060h	00096	0090h	00144	2000h	08192	5000h	20480
0061h	00097	0091h	00145	2100h	08448	5100h	20736
0062h	00098	0092h	00146	2200h	08704	5200h	20992
0063h	00099	0093h	00147	2300h	08960	5300h	21248
0064h	00100	0094h	00148	2400h	09216	5400h	21504
0065h	00101	0095h	00149	2500h	09472	5500h	21760
0066h	00102	0096h	00150	2600h	09728	5600h	22016
0067h	00103	0097h	00151	2700h	09984	5700h	22272
0068h	00104	0098h	00152	2800h	10240	5800h	22528
0069h	00105	0099h	00153	2900h	10496	5900h	22784
006Ah	00106	009Ah	00154	2A00h	10752	5A00h	23040
006Bh	00107	009Bh	00155	2B00h	11008	5B00h	23296
006Ch	00108	009Ch	00156	2C00h	11264	5C00h	23552
006Dh	00109	009Dh	00157	2D00h	11520	5D00h	23808
006Eh	00110	009Eh	00158	2E00h	11776	5E00h	24064
006Fh	00111	009Fh	00159	2F00h	12032	5F00h	24320

Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
6000h	24576	9000h	36864	C000h	49152	F000h	61440
6100h	24832	9100h	37120	C100h	49408	F100h	61696
6200h	25088	9200h	37376	C200h	49664	F200h	61952
6300h	25344	9300h	37632	C300h	49920	F300h	62208
6400h	25600	9400h	37888	C400h	50176	F400h	62464
6500h	25856	9500h	38144	C500h	50432	F500h	62720
6600h	26112	9600h	38400	C600h	50688	F600h	62976
6700h	26368	9700h	38656	C700h	50944	F700h	63232
6800h	26624	9800h	38912	C800h	51200	F800h	63488
6900h	26880	9900h	39168	C900h	51456	F900h	63744
6A00h	27136	9A00h	39424	CA00h	51712	FA00h	64000
6B00h	27392	9B00h	39680	CB00h	51968	FB00h	64256
6C00h	27648	9C00h	39936	CC00h	52224	FC00h	64512
6D00h	27904	9D00h	40192	CD00h	52480	FD00h	64768
6E00h	28160	9E00h	40448	CE00h	52736	FE00h	65024
6F00h	28416	9F00h	40704	CF00h	52992	FF00h	65280
7000h	28672	A000h	40960	D000h	53248	FF00h	65280
7100h	28928	A100h	41216	D100h	53504	FF01h	65281
7200h	29184	A200h	41472	D200h	53760	FF02h	65282
7300h	29440	A300h	41728	D300h	54016	FF03h	65283
7400h	29696	A400h	41984	D400h	54272	FF04h	65284
7500h	29952	A500h	42240	D500h	54528	FF05h	65285
7600h	30208	A600h	42496	D600h	54784	FF06h	65286
7700h	30464	A700h	42752	D700h	55040	FF07h	65287
7800h	30720	A800h	43008	D800h	55296	FF08h	65288
7900h	30976	A900h	43264	D900h	55552	FF09h	65289
7A00h	31232	AA00h	43520	DA00h	55808	FF0Ah	65290
7B00h	31488	AB00h	43776	DB00h	56064	FF0Bh	65291
7C00h	31744	AC00h	44032	DC00h	56320	FF0Ch	65292
7D00h	32000	AD00h	44288	DD00h	56576	FF0Dh	65293
7E00h	32256	AE00h	44544	DE00h	56832	FF0Eh	65294
7F00h	32512	AF00h	44800	DF00h	57088	FF0Fh	65295
8000h	32768	B000h	45056	E000h	57344	FFF0h	65320
8100h	33024	B100h	45312	E100h	57600	FFF1h	65321
8200h	33280	B200h	45568	E200h	57856	FFF2h	65322
8300h	33536	B300h	45824	E300h	58112	FFF3h	65323
8400h	33792	B400h	46080	E400h	58368	FFF4h	65324
8500h	34048	B500h	46336	E500h	58624	FFF5h	65325
8600h	34304	B600h	46592	E600h	58880	FFF6h	65326
8700h	34560	B700h	46848	E700h	59136	FFF7h	65327
8800h	34816	B800h	47104	E800h	59392	FFF8h	65328
8900h	35072	B900h	47360	E900h	59648	FFF9h	65329
8A00h	35328	BA00h	47616	EA00h	59904	FFFAh	65330
8B00h	35584	BB00h	47872	EB00h	60160	FFFBh	65331
8C00h	35840	BC00h	48128	EC00h	60416	FFFCCh	65332
8D00h	36096	BD00h	48384	ED00h	60672	FFFDh	65333
8E00h	36352	BE00h	48640	EE00h	60928	FFFEh	65334
8F00h	36608	BF00h	48896	EF00h	61184	FFFFh	65535

APPENDIX S - SPECIAL SYSTEM VARIABLES

This Appendix lists special system variables that are used by the SAGE^{MAX}. These variables can be monitored from operator interfaces or used as terms in SPL program statements.

VARIABLE	MEANING
\$ALARMS	shows whether or not the SAGE ^{MAX} has any unacknowledged alarms. 0=no, 1=yes
\$MODE	can have a value from 0-255 and reflects the current mode used by the SAGE ^{MAX} global calendar
\$HOLIDAY	reflects the current holiday status as determined by the SAGE ^{MAX} global calendar. 0=normal, 80h (128)=holiday

Table S-1 Special System Variables

