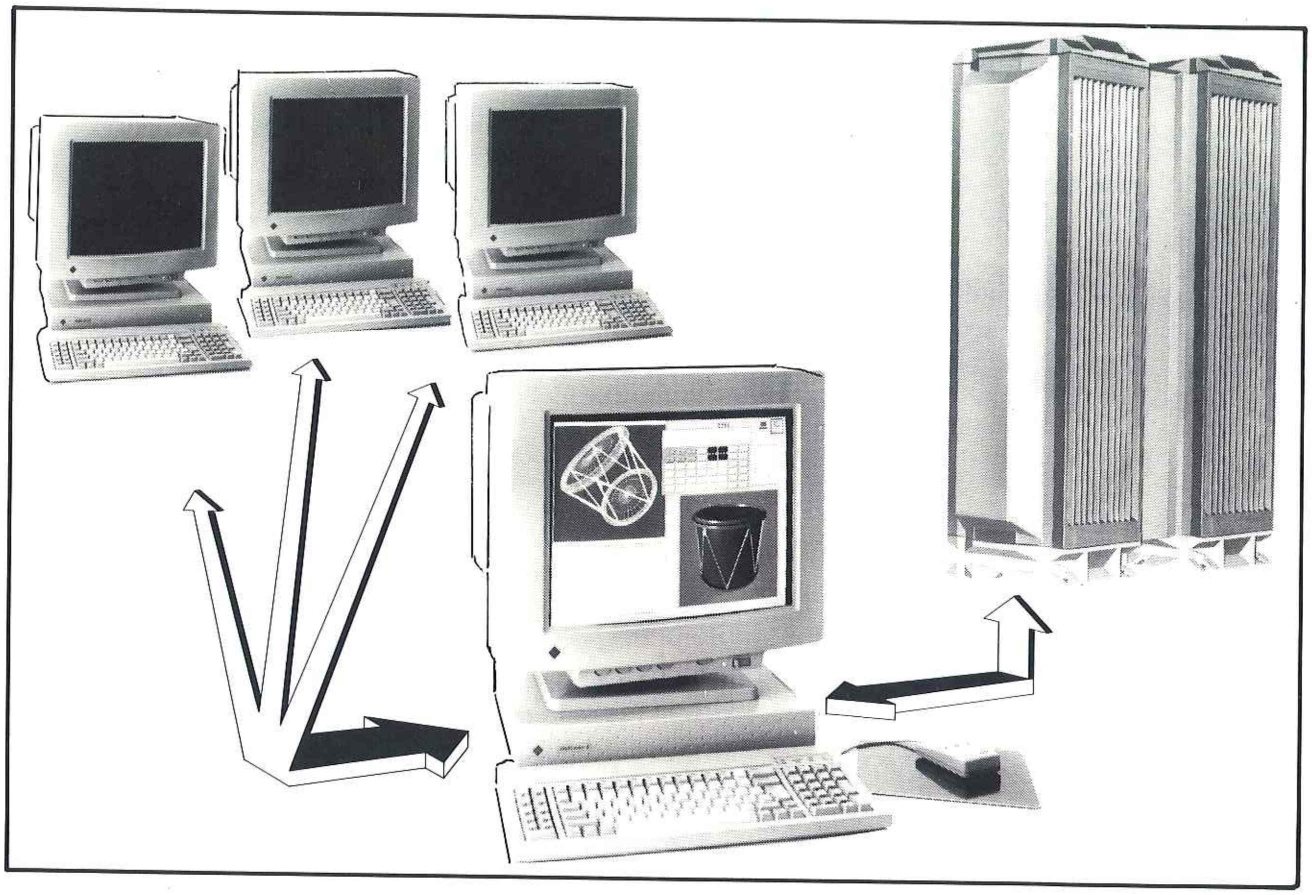
Product Specifications

PSS 21S-4P3 B3

I/A Series®
Open Information Server

Cancelled Per
Trans 021-03
Jan 1994



The I/A Series Open Information Server is a combination of hardware and software. This combination provides a high performance application platform and an engineering network information server to the I/A Series system user. Application programs running on this platform and throughout the supervisory network system have access to all I/A Series system process control data and files.

Hardware

I/A Series Open Information Server is based on the Sun Microsystems® SPARCsystem™ computing platform. This hardware consists of a high performance processor with SPARC® technology integer and floating point units. The SPARCsystem environment delivers superior processing power for process modeling, analysis, and supervisory control tasks.

Used in conjunction with I/A Series Information Network Interface modules, all real-time process values in a connected I/A Series system are available to OIS based application programs. The combination of a high performance architecture with an intelligent interface forms an efficient interface between real time process control and process plant information systems.

Software

The software for I/A Series Open Information Server consists of two major parts:

- Operating system with network, programming, and user environments.
- An application program interface to connected I/A Series systems.

Sun Microsystems SunOS™ software is the operating system for the I/A Series Open Information Server. This operating system meets IEEE POSIX Standard 1003.1-1988 and System V Interface Definition Issue II. SunOS software offers an application program development environment that is source compatible with all UNIX SVR4 environments. In addition to the operating system, this graphical interface for user specific applications is based on X/Open - XPG3.

FOXBORO®
A SIEBE COMPANY

I/A Series Application Interface Software offers data and services access to application programs operating in the I/A Series Open Information Server. These services include:

- Real-time process variable read and write.
- Data file read and write.
- Virtual terminal access.
- Network client-server application support

APPLICATION INTERFACE SOFTWARE

Application Interface Software (AIS) provides an application program interface (API) resident in a host computer. Host computer applications use AIS to gain access to objects and/or files in connected I/A Series systems. This application program interface, following international standards, insulates the customer application programs from host computer specifics. Customer applications deal with functional requirements instead of communication implementation details.

Host computer programmers write application programs using AIS subroutines to perform such functions as process variable access, file transfer, and terminal interface. The AIS routines work in conjunction with one or more Information Network Interface (INI) modules configured in connected I/A Series systems. AIS works with Information Network Interface modules direct or network connected to the host computer system. The host computer vendor determines appropriate host system hardware and software to support the connection.

Application functions available with AIS:

- Non-connected named process variable access
- Connected named process variable access
- Exception based, named process variable access
- Named file upload and download
- Named file partial upload
- Programmatic virtual terminal interface
- Redirection of virtual terminal input and output to and from files
- Menu-driven virtual terminal access
- Menu-driven file transfer

AIS optionally provides circuit optimization and data object path redundancy on multiple INI systems. The AIS package, by configuration, apportions process variable lists among several Information Network Interface modules. AIS also uses configurable alternate INI paths for object access upon failure of a particular INI. This maintains operating object access services originally provided by the failed INI path.

AIS additionally offers application program development tools. This package includes utility programs for exercising the package functions, and for comparing results during application development. Several optimizations are configurable for tuning the performance of the host computer application under development.

APPLICATION PROGRAM INTERFACE SERVICES

The Application Interface Software (AIS) accesses files and data objects residing in the I/A Series stations for user-written application programs resident in Host Computers.

The application program interface supports redundant program access to I/A Series process data via INI(s). It includes C language callable subroutines used by application programs to read global data from I/A Series processors, to write data to I/A Series processors, and to access files in I/A Series Application Processors.

The methods of object access are:

- Unbuffered read/write of data objects.
- Buffered read/write of data objects.
- Buffered read/write with continuous updates.
- Buffered read/write with continuous updates and queueing all changes.

Unbuffered Read/Write of Data Objects

This access method is appropriate when the data objects are accessed infrequently. Data objects of all value types except string can be read or written. Data objects of the string type can only be read. Both connectable and non-connectable data objects may be accessed. Related subroutines are:

- uread
- uwrite
- sread
- swrite

UNBUFFERED READ OF NON-STRING DATA OBJECTS
The caller of the uread subroutine specifies the gateway paths to be used and a set of data objects by name. The subroutine returns the value, status, and error code for each non-string data object.

UNBUFFERED WRITE OF NON-STRING DATA OBJECTS
The caller of the uwrite subroutine specifies the gateway paths to be used, a set of data objects by name, and the new values for the data objects. All data objects that are accessible are updated. An error code is returned for each data object.

UNBUFFERED READ OF STRING DATA OBJECTS
The caller of the sread subroutine specifies the gateway path to be used and the name of a data object. The subroutine returns the string, the status, and an error code for the data object. Only one data object may be read at a time.

UNBUFFERED WRITE OF STRING DATA OBJECTS
The caller of the write subroutine specifies the gateway path to be used and the name of a data object. The subroutine returns an error code for the data object. Only one data object may be written at a time.

Buffered Read/Write of Data Objects

This access method is appropriate when data objects are being accessed frequently. Only connectable data objects can be accessed. Related subroutines are:

- bopen
- sbopen
- bread
- bwrite
- clsset

SPECIFICATION OF DATA OBJECTS FOR BUFFERED READ/WRITE
The bopen subroutine establishes the change-driven connections for a set of data objects. The data objects are then accessible by using the bread and bwrite subroutines. The caller optionally specifies the gateway paths. The caller specifies a set of data objects by name, change value (delta), assumed value type, an indicator if the set is to be closed when the program exits, and an access mode (read-only or read/write). The subroutine makes the connections to the data objects and returns a data set number to be used in other calls to reference the set of data objects, a status, and an error code per data object informing the caller if and why a connection cannot be made.

BUFFERED READ OF DATA OBJECTS
The caller of the bread subroutine specifies a set of data objects by data set number. The subroutine returns the value and status for each data object in the data set.

BUFFERED WRITE OF DATA OBJECTS
The caller of the bwrite subroutine specifies a set of data objects by data set number and the new values. All data objects that are accessible are written. An error code is returned for each data object in the data set.

Buffered Read/Write with Continuous Update

This access method is appropriate when you want to have values updated automatically in memory or files without doing any bread calls. Related calls are:

- copen
- scopen
- bread
- bwrite
- clsset

The copen call establishes the change-driven connections and starts the updating process. The caller optionally specifies the gateway path(s) to be used. The caller specifies a set of data objects by name, change value (delta), assumed value type, an indicator if the set is to be closed when the application program exits, access mode (read-only or read/write), and an array to return the indexes into system common arrays where the value and status are stored. The subroutine returns a data set number to be used in the bread, bwrite, and clsset calls.

In using the scopen, it is also possible to specify the frequency of checking the delta. By updating the write value table in system common, it is possible to invoke a change-driven write of the value to the I/A Series object.

Note that bread calls are unnecessary, because value and status information is automatically updated to system common arrays in memory or files. Also note, however, breads are possible.

Buffered Read/Write with Continuous Update and Queueing of Changes

This access method is appropriate when you want the application program to be made aware of every change in value greater or equal to the delta and/or every change in status of object in the set. Related subroutines are:

- qopen
- sqopen
- qread
- bread
- bwrite
- clsset

The qopen call establishes the change-driven process variable object connections and starts the updating and queueing process. The caller optionally specifies the gateway path(s) to be used. The caller specifies a set of data objects by name, change value (delta), assumed value type, an indicator if the set is to be closed when the program exits, and an array to return the indexes into the system common array where the value and status are stored. The subroutine returns a data set number to be used in the qread, bread, bwrite, and clsset calls.

In using the sqopen, it is also possible to specify the frequency of checking the delta. By updating the write value table in system common, it is possible to invoke a change-driven write of the value to the I/A Series process variable object.

EXTRACTING CHANGES FROM A CHANGE QUEUE
The caller of the qread subroutine specifies a set of data objects by data set number. The subroutine returns the changes (value and status) from its queue for that data set. Each change is accompanied by a data set entry number. Some data objects may have no changes associated with them; others may have several changes. The maximum number of changes to be returned is specified by the caller. The actual number of changes is returned to the caller. A return code indicates the state of the queue.

CLOSING A SET OF DATA OBJECTS
Data sets previously opened with the bopen, sbopen, copen, scopen, qopen, or sqopen call can be closed with the clsset call.

Redundancy

The goal of the redundancy is to keep as many objects accessible as possible when INI(s) fail.

To supply redundancy it is necessary to have two or more INIs in the same group with some free lists. The redundancy algorithms keep as many lists open on the preferred INI as possible, or, as a second choice within a group. The result is that lists are moved from a failing preferred INI to a functional INI, then back to their preferred INI as soon as the preferred INI returns to a functional state ("OK").

The state of the INI is constantly monitored by the application program interface software. If one of the INIs changes state (from OK to FAILED or FAILED to OK), then the following actions are performed:

- All lists which are currently not opened on any INI are opened on their preferred INI. Interface software opens these lists if the INI is OK and has available resources. Unopened lists occur when other INI modules have no available resources to back up a failed INI module.
- All lists currently opened on a backup INI module are moved to their preferred INI module. Interface software opens these lists if the preferred INI module is OK and has available resources.
- All lists not currently opened by interface software on any INI module are opened on any INI module in the same group. Interface software opens these lists if the target INI module is OK and has available resources.

ACCESSING I/A Series FILES

There are three methods of file access:

- Transferring a file between I/A Series Application Processor and the Host.
- Transferring a file between I/A Series Application Processor and the Host, supported in FILUTL only.
- Reading part of a file from I/A Series Application Processors.

The file access allows the reading and writing of files. The application program access can be done through C language interface calls. Related subroutines are:

- iarfil
- iawfil
- pfread

Reading a File from I/A Series Application Processors

The iarfil routine transfers the entire contents of an I/A Series Application Processor file to an I/A Series Open Information Server file. The caller specifies the I/A Series Application Processor file name, I/A Series Open Information Server file name, gateway path, user id, group id, and I/A Series Application Processor logical name (as configured in the INI module). An error code is returned.

Writing a File to I/A Series Application Processors

The iawfil routine transfers the entire contents of an I/A Series Open Information Server file to an I/A Series Application Processor file. The caller specifies the I/A Series Application Processor file name, I/A Series Open Information Server file name, gateway path, user id, group id, and I/A Series Application Processor logical name (as configured in the INI module). An error code is returned.

Reading Part of a File from I/A Series Application Processors

The pfread routine transfers a specified portion of an I/A Series Application Processor file into the memory space of the application program. The caller specifies the I/A Series Application Processor file name, gateway path, userid, group id, logical name, the number of bytes to read, and the array in which to store the information. An error code is returned.

ACCESSING VIRTUAL TERMINAL SESSIONS

The virtual terminal access allows the use of VENIX features from I/A Series Open Information Server application programs using C language subroutine calls. Related subroutine calls are:

- vtinit
- vtprog
- vtend

Establishing a Session

The vtinit subroutine establishes a session by logging into an I/A Series Application Processor. The caller specifies a gateway path, the letterbug of the chosen I/A Series Application Processor, the target account login name (set up by the site system administrator prior to use), account password, and the desired prompt string for the VENIX operating system shell. An error code is returned.

Processing VENIX Command Lines

The vtprog subroutine issues VENIX command lines from a specified program array and returns responses to another specified array. The caller specifies the array containing the VENIX command lines, the number of characters to read from the command line array, and the response storage array. The number of characters stored in the response array and an error code are returned.

Terminating a Session

The vtend subroutine terminates a session with an I/A Series Application Processor. An error code is returned.

UTILITIES

FILUTL Utility Program

The utility program, FILUTL, allows the user to copy:

- a Host file to an I/A Series file.
- an I/A Series file to a Host file.
- a part of an I/A Series file to Host memory.

VTUTIL Utility Program

The utility program, VTUTIL, allows the user to:

- enter VENIX command lines from a Host terminal keyboard and receive prompts and responses on the Host screen.
- Enter VENIX command lines from a file and optionally receive VENIX responses back in a file and/or to the Host screen.
- Enter VENIX command lines form a Host keyboard and optionally receive VENIX responses back in file and/or to the Host screen.

Obtaining Information about a Data Set

AIS provides routines that can optionally be used to aid in developing an application. These routines return information that is available when the data set is opened. Providing this information relieves the application from housekeeping. Related subroutines are gsinfo, getnam, and getscn.

Optimizing the Use of Virtual Circuits

By default, AIS subroutines which communicate to INIs establish and release the X.25 virtual circuit to the INIs each time they are called. This approach allows more applications to make use of the INI services.

If you have application which makes frequent calls to such subroutines, you can optimize the application by specifying that virtual circuits are to be established once, kept for as long as needed, then released. Related subroutines are getcir and relcir.

AISTST Utility Program

AISTST exercises and inspects the AIS package. The following is provided:

- The capability to exercise all AIS subroutines except file access and virtual terminal access.
- The capability to report on the status of the package.
- The capability to turn traces on or off.

GWSTAT Utility Program

The gwstat utility program retrieves the status of INI modules.

AISDIS Utility Program

The aisdis program initially displays the status of configured INI modules and the total number of change messages per INI module. Secondary displays give the name, value, and status for the data objects of a specified data set. Aisdis also displays the number of changes per object either read from or written to a connected I/A Series system.

Aisdis is useful for tuning change and write deltas for data objects. Aisdis is also used to monitor change and write message load on INI modules.

IA2REM Utility Program

The ia2rem program transfers a file from an I/A Series Application Processor into a file on the I/A Series Open Information Server, personal computer, or UNIX workstation.

REM2IA Utility Program

The rem2ia program transfers a file from the I/A Series Open Information Server, personal computer, or UNIX workstation to an I/A Series Application Processor.

SET2LOTU Utility Program

The set2lotu program produces a Lotus 1-2-3 (WK1) format worksheet file from a specified AIS data set. The first column contains the data object name, the second column contains the data object value, and the third column contains the data object status. Each worksheet row contains information for one data object. The resulting worksheet file has as many rows as data objects in the source data set.

NAM2LOTU Utility Program

The nam2lotu program produces a Lotus 1-2-3 (WK1) format worksheet file from a specified list of I/A Series data object names. The first column contains the data object name, the second column contains the data object value, and the third column contains the data object status. Each worksheet row contains information for one data object.

IDX2LOTU Utility Program

The idx2lotu program produces a Lotus 1-2-3 (WK1) format worksheet file from a specified list of AIS data object indexes. The first column contains the data object name, the second column contains the data object value, and the third column contains the data object status. Each worksheet row contains information for one data object.

REMOTE OPEN APPLICATION SERVER FUNCTIONS

The I/A Series Open Information Server serves the information needs of network-connected workstations and MS-DOS personal computers running Sun Microsystems PC-NFS. The server makes it possible to have programmatic access to real-time process variable object and data files from the workstation or personal computer. Subroutine argument specifications of the remote programmatic interface are identical with those of the I/A Series Application Interface Software described previously.

Remote access of I/A Series process control data builds on Sun Microsystems Network File System (NFS) and Remote Procedure Calls (RPC). This access method uses the client/server model to allow client processes in workstations and/or personal computers to request AIS functions.

Workstation and/or Personal Computer Support

The following calls are supported in the remote workstations and/or personal computers: uread, uwrite, sread, (s)bopen, (s)copen, (s)qopen, bread, bwrite, gsinfo, getnam, getscn, cnvrt, wrtval, readcnt, readsta, readval, iarfil, and iawfil.

The following calls are NOT supported by workstations or personal computers but are supported within the I/A Series Open Information Server: qread, pfread, vtinit, vtprog, vtend, getcir, relcir, connec, inists, and mvbak. Calls not supported by remote access output a warning message on the workstation or personal computer standard error device (stderr) and return to the caller.

I/A Series Open Information Server supports aistst and the file utility in remote access.

I/A Series Open Information Server does not support the aisdis and gwstat utilities in remote access but does support the utilities directly on the server.

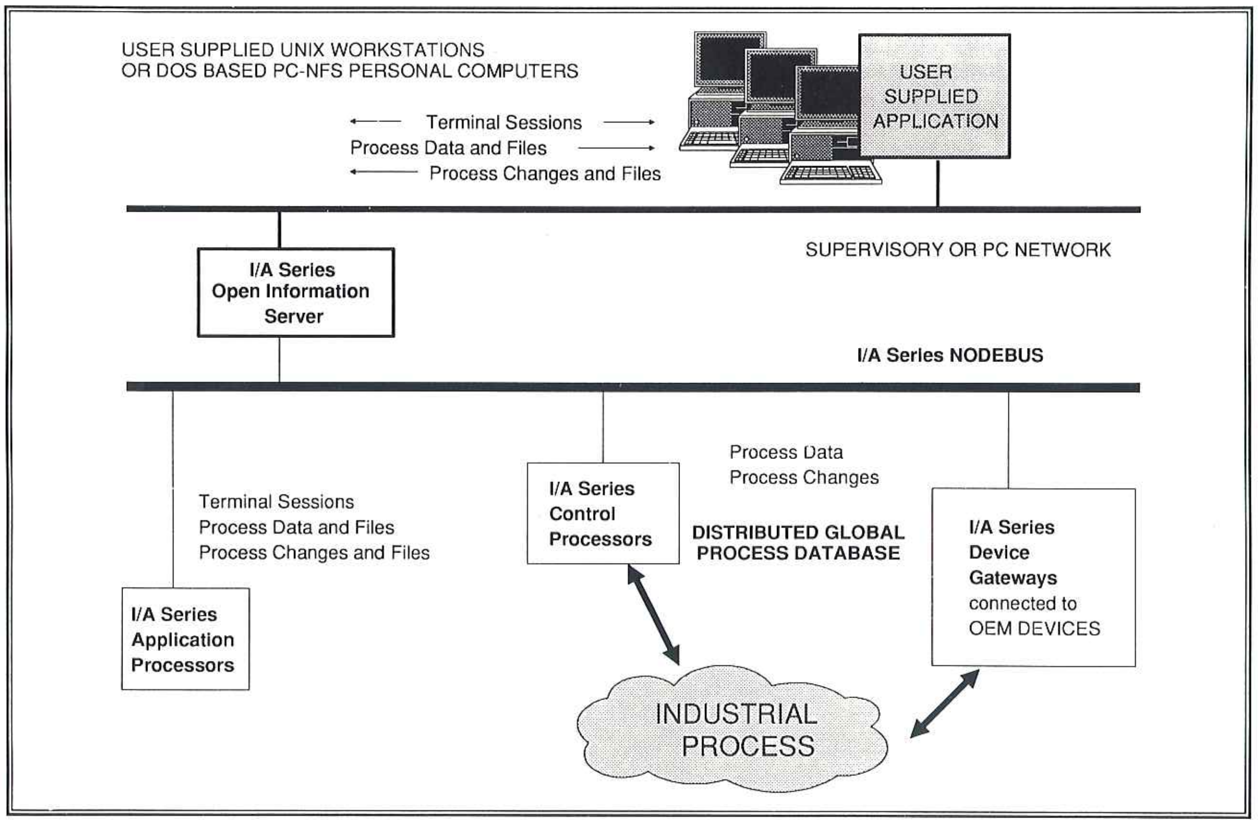

USER SUPPLIED UNIX WORKSTATIONS
OR DOS BASED PC-NFS PERSONAL COMPUTERS
USER SUPPLIED APPLICATION
Terminal Sessions
Process Data and Files
Process Changes and Files
SUPERVISORY OR PC NETWORK
I/A Series Open Information Server
I/A Series NODEBUS
Terminal Sessions
Process Data and Files
Process Changes and Files
I/A Series Control Processors
Process Data
Process Changes
DISTRIBUTED GLOBAL PROCESS DATABASE
I/A Series Device Gateways connected to OEM DEVICES
I/A Series Application Processors
INDUSTRIAL PROCESS

Figure 1. I/A Series Open Information Server

SPECIFICATIONS

Processor	
PERFORMANCE	21 SPE Cmarks (28.5 MIPS and 4.2 MFLOPS)
FLOATING-POINT UNIT	SPARC IEEE standard 76
CACHE	64KB
MAIN MEMORY	Standard RAM/32MB
Ethernet Interface	
MEDIA TYPE	Coaxial cable
DATA RATE	10MB
CONNECTOR	15 pin
SCSI Interface	
CONNECTOR	SCSI-2
Serial Input/Output	
PORTS	Two RS-423 (RS-232C compatible used to connect INI modules)
AUDIO	8KHz, 8-bit low pulse code modulation, internal speaker
System Bus	
TYPE	SBus
ADDRESS BUS	32-bit
DATA BUS	32-bit
Internal Storage	
FLOPPY DISK	MS-DOS compatible, 3.5 inch, 1.44MB/720 KB
HARD DISK DRIVE	
Format	3.5 inch disk
Formatted Capacity	424MB
Average Seek Time	16ms
Burst Transfer Rate (synchronous)	4.0 MB/s
Raw Disk Data Rate	1.6MB/s
External Storage	
TAPE DRIVE	
Format	5.25 inch tape, QIC-150
Formatted Capacity	150MB
Monitor	
COLOR AND MONOCHROMATIC	
Resolution	1152 X 900 pixels
Dots Per Inch	100
Pixel Aspect Ratio	1:1
Antiglare Treatment	Fine silica
Connector	13W3
Power Supply	100-120/200-240 V ac, autoranging
16-INCH COLOR	
Refresh Rate	76 or 66 Hz, non-interlaced
Dot Rate	106 or 93 MHz
19-INCH MONOCHROMATIC	
Refresh Rate	76 Hz, non-interlaced
Dot Rate	106 MHz
Keyboard	107 keys, low profile
Mouse	Optical, 3-button
Environment	
OPERATING TEMPERATURE	10°C to 40°C (50°F to 104°F)
NON-OPERATING TEMPERATURE	–20°C to 60°C (–4°F to 140°F)
OPERATING HUMIDITY	20% to 80%, noncondensing at 40°C
NON-OPERATING HUMIDITY	95%, non-condensing at 40°C

Regulations	
MEETS OR EXCEEDS THE FOLLOWING REQUIREMENTS:	
Safety	UL 478, CSA C22.2 No. 154-M1983, TUV (VDE 0806, IEC380)
RFI/EMI	FCC Part 15, DOC, VCCI, VDE 0871
X-ray Emissions	DHHS Rule 21, Subchapter J, PTB German X-ray Decree
Static Discharge	15 KV, no hard errors
Electrical	
AC VOLTAGE	90-132 V ac or 180-264 V ac
AC FREQUENCY	47-63 Hz
POWER	0.2 KVA
Dimensions and Weights	
SPARCstation 2 CHASSIS	
Height	7.1 cm (2.8 in)
Width	40.9 cm (16.0 in)
Depth	40.9 cm (16.0 in)
Net weight	10.4 kg (22.8 lb)
COLOR MONITOR	
Height	41.6 cm (16.4 in)
Width	40.6 cm (16.0 in)
Depth	45.3 cm (17.8 in)
Shipping Weight	27.3 kg (60.0 lb)
MONOCHROMATIC	
Height	45.0 cm (17.7 in)
Width	46.0 cm (18.1 in)
Depth	41.0 cm (16.1 in)
Shipping Weight	27.7 kg (61.0 lb)
Software	
OPERATING SYSTEM	SunOS 4.1.1 or later
LANGUAGE	C
NETWORKING	NFS, TCP/IP, SunNet™ X.25
GRAPHICS	Xlib
WINDOW SYSTEMS	Open Windows™
PROCESS CONTROL APPLICATIONS TOOLS	Foxboro Application Interface Software

MB 021
0192