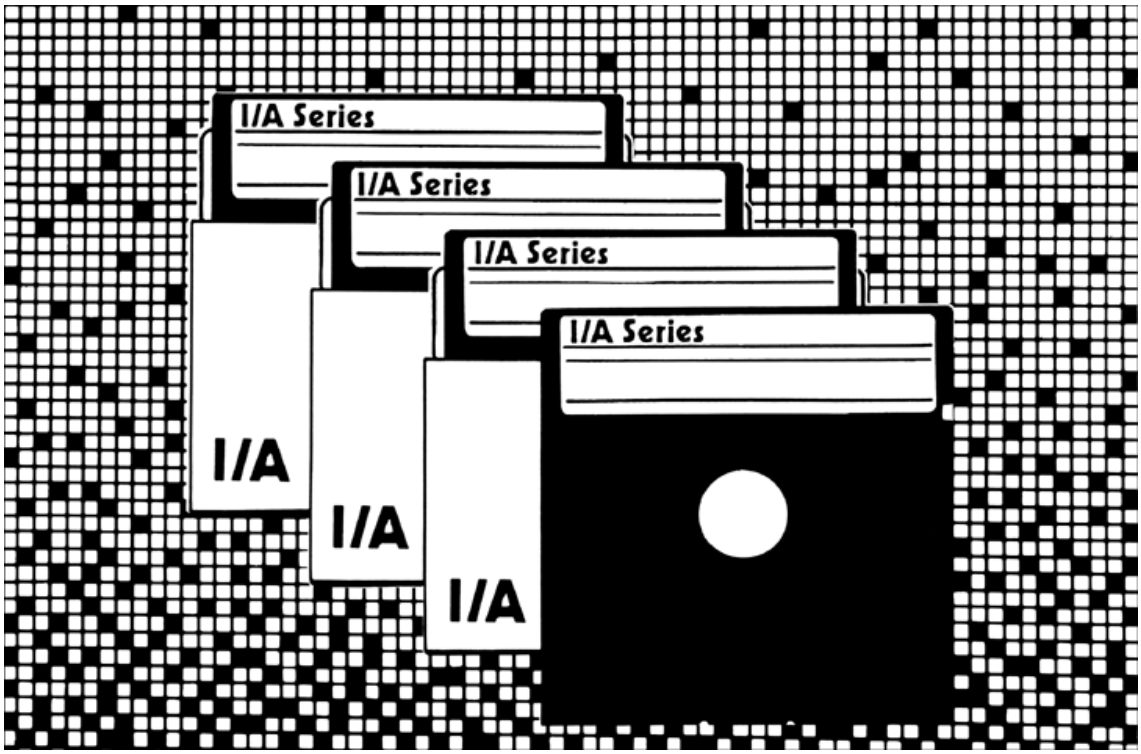
FOXBORO®

PSS 21S-1B1 B3

I/A Series® Software
Operating Software



The operating software is a set of programs that control and organize the activity of the I/A Series system. It directs the activities of the system modules, manages multi-users and multi-tasking, and manages the system's files without user participation or supervision. The operating software includes the operating systems and other subsystems, such as Inter-Process Communications and the Object Manager.

All processors use the same Real-Time Executive operating system; however, some processors also use an open Real-Time Application system.

- The Real-Time Executive System – Runs in every processor
- The Real-Time Application System – Runs as a task under the Real-Time Executive System on Application Processors (APs), Workstation Processors (WPs), and PC ATs.

REAL-TIME EXECUTIVE SYSTEM

The Real-Time Executive System is based on the VRTX operating system. Its mission is to direct processor activities on a prioritized, pre-emptive basis. Pre-emptive means that higher-priority interrupts are handled immediately. This gives the processor real-time operation.

Since the Real-Time Executive System is memory resident and uses pre-emptive scheduling, it can perform real-time operations at a very fast response rate.

REAL-TIME APPLICATION SYSTEM

The Real-Time Application System is based on VENIX, a variation of UNIX system V, with which it is compatible. Application programs that adhere to the UNIX system V Interface Definition (SVID) run under this system, once recompiled with the VENIX tool set.

The Real-Time Application System uses VENIX Distributed System Architecture (VENIX DSA), which allows access to resources (files, message queues, etc.) that are distributed throughout the network. Since this system is merged with the network, a number of communications problems are simplified:

- The Real-Time Application System does not need to transfer an entire file to a local processor just to read part of it.
- Only one copy of a file need exist on a network.
- The resources available to every processor are the sum of the resources on all processors.
- Tasks on different processors can synchronize and exchange messages.
- Since utilities do not need to be changed to use the network, the network is transparent to the utilities and to programs that use them.

Memory Management

Memory management includes the strategy for loading programs into memory for execution and any requests for additional memory that a program makes while executing (Dynamic Memory Allocation – DMA).

Program Loading

The Real-Time Application System has a task-loading function that allows pre-emptive swapping. That is, if a higher-priority task needs to run but there is not enough room, the first task's space is pre-empted, the task swapped out, and the higher-priority task loaded in its place.

Dynamic Memory Allocation

The Real-Time Application System allows a task to request additional memory while it is executing. A task can request a memory block of a specific size or return a block of memory to the free pool. If there is not enough memory available, the requesting task is notified.

Code Mapping

Code mapping means the system automatically moves unused parts of user tasks (or the kernel) to memory outside the task's address space until those parts are needed. Code mapping allows the system to handle programs larger than 64kb (a limit imposed by the processor), such as the Real-Time Data Base Manager.

File Management System

Real-Time Application System file management uses a distributed file system; that is, stations with bulk storage devices act as file servers for other stations. These file servers are called Application Processors (AP-10 and AP-20).

In this way, a task running on any Real-Time Application System station can make file access calls both locally and remotely (over the network). The file system is hierarchical; that is, it consists of directories that can contain files or subdirectories.

Each physical input/output (I/O) device, from the display and keyboard to main memory, is treated as a file, allowing consistent file and device I/O.

The file management system allows both synchronous and asynchronous access to the files. Synchronous operation implies that the task that makes the request suspends while the operating system performs the access. The task does not resume until the access completes (successfully or in error).

In asynchronous operation, the requesting task can make the access request, then continue to execute. The task can either check on the status of the access from time to time, execute until the access finishes, or suspend until the access finishes.

The file structure is based on the UNIX file system, which uses 24-bit logical disk-block pointers and 1024-byte disk blocks. A single file can grow dynamically up to a maximum of two gigabytes.

The file access calls are based on the standard VENIX file access calls. Foxboro has included an enhancement that checks file writing by reading the file back afterwards.

Error Handling Subsystem

The error handling subsystem insures that the system can recover from and report an error that occurs in the system. It does this either by taking the necessary action itself, or by passing information to other subsystems so they can handle the error.

Errors handled by the System Management subsystem are failures in hardware detected by a combination of error detection hardware and diagnostic software.

Errors handled by the operating system are software errors that can be due to design errors in user-written software, unavailable resources, and other runtime contention errors. The operating system works in conjunction with the System Management subsystem to report these errors.

Error Detection and Reporting

Software runtime errors are detected and handled in several ways. Calls to the operating system are checked to see that all the call parameters are within limits. Whenever the operating system encounters an error, it sends an interrupt to the calling task.

When a task receives the interrupt signal, it can invoke a system subroutine to take the appropriate action (for some classes of errors) or terminate.

The error-reporting function can be invoked to report errors in a consistent format to the specified output device.

Error Analysis

Each station can save the relevant text and data areas and the contents of the processor's registers when a runtime error occurs. You can also trigger this "dump" by code within the task (for example, for debugging). The dump is automatically written to a file and a message to that effect is printed.

There is also software for Foxboro to analyze the dump file for any processor in the system. This software can interpret the state of the various data structures within the operating system, do runtime stack backtracking, and print a listing of various sections of memory for interpretation.

Real-Time Application System Priority Scheduling

The task is the basic unit that is scheduled. A task is a program image and its execution environment (register values, status of open files, current directory, etc.). The Real-Time Application System allows a task to create or delete itself and child tasks.

The scheduling system schedules tasks according to their priority. Priority for background (not real-time) tasks is a value calculated using the task state, processor use time, and "nice" value. Priority for real-time tasks is based on the real-time nice value. Low numbers (high priorities) run first.

Background tasks in the running state change priority every second, when the priority is recalculated. The calculation lowers the priority (increases the priority number) for processor use time and higher nice values. It raises the priority (decreases the priority number) for time spent waiting to run.

Tasks in the sleeping state assume the priority of the event they are waiting for, which is always a higher priority than running tasks. Giving a higher priority to tasks that use a lot of input/output resources helps keep resources available.

The scheduler always switches to the task with the highest priority. Background tasks are selected only if there are no real-time tasks that want to run. Ties go to the task that has been waiting the longest (first in, first out).

Networking Capabilities

The Real-Time Application System DSA supports networking. Built into the system are the capabilities of keeping track of which processors are attached to which node, and how to (and who can) access them. The Machine Access Table and the User Access Table determine the configuration of the network.

User Access

User access permission to remote processors is handled by user and group id mapping tables. Once a connection is made to a processor on a remote machine, a processor task checks the id mapping tables to make sure that the user and group id's are allowed access. If no id mapping table exists for the connecting processor, access is denied to all users of that processor.

Utilities, Libraries, and the Command Language

The Real-Time Application System provides two command shells: the Bourne shell and the C shell. Both are command interpreters with high-level programming-language constructs (IF-THEN, DO-WHILE, etc.). You can create command files to perform tedious or repetitive command sequences.

The Real-Time Application System has commands associated with or supporting each of the following:

- Text editing and formatting
- Data sorting
- Spelling checker
- System administration
- File management
- Distributed System Architecture (DSA)
- Inter-Process Communications
- Display Manager
- Object Manager
- Virtual Terminal Emulation

TIMING FUNCTIONS

The timing functions include a Real-Time Application System clock interrupt handler, a station clock for time and date, and a user interface for scheduling tasks.

Each station provides user tasks with facilities for dealing with time of day and for software timers. These timing facilities are driven by interrupts or “ticks” coming in from a real-time clock. The Real-Time Application System uses these real-time clock interrupts for time-of-day scheduling functions, timeout functions, and notification functions such as wake-up signals.

Clock Interrupts

The real-time clock “ticks” every 10 milliseconds. The minimum periodic scheduling frequency is one second for the Real-Time Application System, and 100 milliseconds for the Real-Time Executive System. The network time-of-day enables all stations to synchronize their clocks to within one-tenth of a second. Any user task can read the time of day, and the System Management subsystem provides a facility to change it.

Station Clock

The Application Processor and SPECTRUM Slave Gateway stations can be configured to act as master timekeepers. Workstation Processors and Communication Processors also contain calendar clocks, which maintain system time during power failures, provided the enclosure has battery backup. Station processor real-time clock chips maintain station times between updates from the master timekeeper.

Time-of-Day Scheduling

You can schedule or deschedule a task and list currently scheduled tasks. Tasks can be scheduled to:

- Run once at a specified time of day
- Run periodically at a specified interval (regardless of changes in station time)
- Run periodically at a specific time of day (Real-Time Application System only)

Clock Timers

The Real-Time Application System provides software timers that user tasks can use to:

- Acquire a timer for its use
- Specify the time-out period of the timer
- Specify a handler or an action that should occur if the timer expires
- Enable the timer
- Disable the timer
- Deallocate the timer

OBJECT MANAGER

The Object Manager is a subsystem that controls access to data units called objects. There are two classes of objects:

- Control and I/O Objects (compound:block.parameters)
- Shared Objects
  - Tasks
  - Devices
  - Aliases (character strings)
  - Variables (integer, character, long integer, floating point, and string)

The Object Manager acts as an interface between applications and the data they require. It keeps track of the locations of objects so you can write applications that access data by name only, without knowing the system configuration.

Because of the Object Manager, object databases can be created, modified, or moved to another station without having to modify any of the applications that access the data. You can write an application before creating the database it uses. The Object Manager allows applications to:

- Create, locate, and delete objects
- Get or set single object values (such as process variables)
- Read or write sets of shared objects (variables)
- Receive notification when an object value changes by a specified amount

INTER-PROCESS COMMUNICATIONS

"Process," in this usage, refers to the UNIX term for tasks and their executing environment, rather than the manufacturing or production process. In addition to the standard Inter-Process Communications, The Foxboro Company has added extensions which provide for distributed communications between stations in the system. These extensions allow for messages that can:

- Pass data or control information between tasks
- Notify another task of an event
- Synchronize the use of common network resources
- Communicate with applications distributed throughout the system, without necessarily knowing where they are

Inter-Process Communications supports not only connection-oriented communications, but also connectionless and broadcast communications. The user interface to Inter-Process Communications is through a series of C language subroutines.

LANGUAGE PROCESSING

The Real-Time Application System supports the C and FORTRAN programming languages, including compilers, linkers, debuggers, and semantic checkers. All system libraries are fully supported for C language. Object manager libraries are supported for FORTRAN and C language.

SYSTEM MANAGEMENT

The System Management software extracts information about the system and displays it for the user. In addition, it allows the operator to intervene in the system operation.

There are three types of information that can be monitored in this way:

- The topology of the communications network
- Statistics concerning the performance of the network
- Any station faults that occur

System Management software automatically tracks the equipment configuration. At initialization, System Management establishes domains, each consisting of a System Monitor and its stations, and begins monitoring network resources.

System Management software consists of a System Monitor, Station Managers, a Display Manager, Error Protocol software, Network Booting software, and Software Management software. These work together to present displays to the network maintenance engineer and to support user requests for network maintenance-control actions or network resource information.

System Monitor

The System Monitor software, in conjunction with the exception handler, provides the capability for application programs and operating systems to display error messages at workstations and to print them. There can be more than one System Monitor, each in control of a number of different stations.

Station Manager

Each System Monitor has a group of stations in its domain and communicates with them through their Station Manager software. A Station Manager sends statistical and event information to the System Monitor. This information is presented to the user upon request and automatically updated.

System Management Displays

You can see the System Management displays through a series of menus and submenus and by selecting items on the displays themselves.

The System Management menu provides access to a hierarchy of displays that show:

- All configured System Monitors
- Stations in each System Monitor domain
- Status of each System Monitor domain
- Status of each station in a System Monitor domain
- Information about all the devices (e.g., CRT monitors or printers) attached to the station
- Either current or historical System Management performance parameters (some of which can be reset or changed), including network communications information
- Either current or historical station peripheral performance parameters
- A history of systems exceptions and operator-initiated events

Each station status is automatically updated, is color coded based on status, and blinks until it is acknowledged.

In addition, there are certain actions that you can take at designated workstations, as provided for in the configuration management menu:

- Change a station state (e.g., by reloading it)
- Change network time
- Change station configuration options:
  - Enable/disable reports to the System Monitor
  - Reboot the station
  - Update the EEPROM
  - Issue a checkpoint command
  - Enable/disable historical recording

These displays also allow you to deal with faults in various ways:

- In each domain, examine the stations and attached devices
- Invoke on-line cable test
- Load off-line station test
- Fail the station
- Restart the station
- Attach/detach devices

Station displays are updated whenever the fault status changes.

The System Management displays have a help function so that users can get additional information while using the displays.

The Foxboro Company
33 Commercial Street
Foxboro, Massachusetts 02035-2099
United States of America
http://www.foxboro.com
Inside U.S.: 1-508-543-8750 or 1-888-FOXBORO (1-888-369-2676)
Outside U.S.: Contact your local Foxboro Representative.

MB 021 Printed in U.S.A. 0689