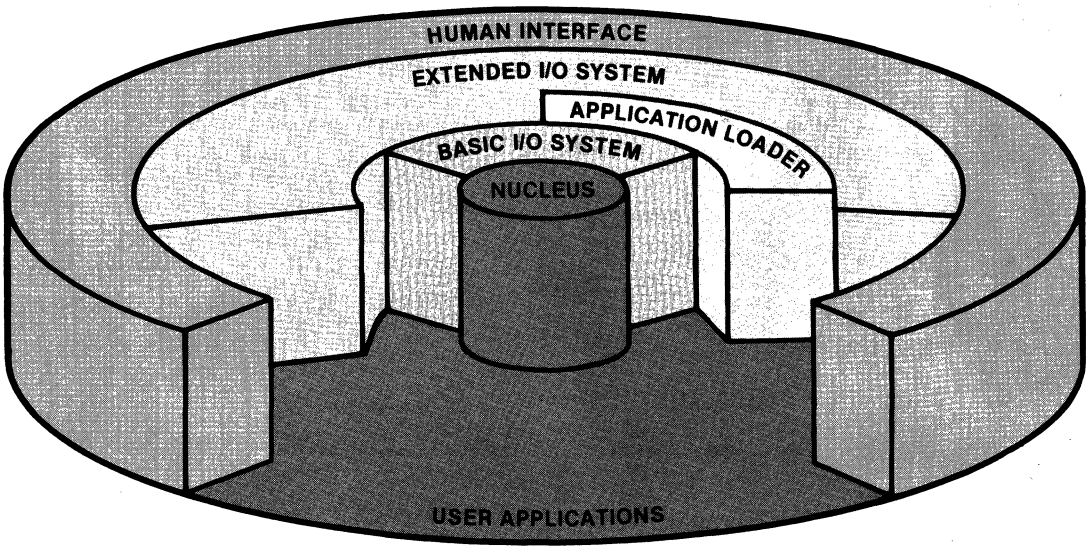
iRMX 86™ OPERATING SYSTEM

- **Structured multiple application environment**
- **Object-oriented architecture**
- **(P)ROM or RAM based**
- **User configurable and extensible**
- **Real-Time priority-oriented scheduler**
- **Complete bootstrap and application loaders**
- **Powerful error management**
- **Comprehensive I/O system**
- **Extensive human interface**
- **Interactive system debugger**

The Intel iRMX 86 Operating System is an easy to use, comprehensive multiprogramming software system for Intel iSBC 86 and 88 Single Board Computers and other iAPX 86 and iAPX 88-based microcomputers. The iRMX 86 Operating System extends the architecture of the underlying processor by providing a collection of new operations that act on the Operating System objects provided by the system or user created extensions. The multiprogramming environment of the iRMX 86 Operating System, based on a real time, event-driven scheduler, provides an efficient foundation for applications including process control, intelligent terminals, office systems, data communications and medical electronics.

Each layer of the operating system simplifies user access to the underlying hardware by taking advantage of the mechanisms provided by the layers below. Each layer of the iRMX 86 Operating System is configurable to allow applications to customize the system to particular needs.



iRMX 86™ Operating System Layers

April 1981
142988.01

FUNCTIONAL DESCRIPTION

Services provided by the iRMX 86 Operating System include facilities for executing programs concurrently, sharing resources and information, servicing asynchronous events, and interactively controlling system resources and utilities. In addition, the iRMX 86 Operating System provides all major real time facilities including priority-based system resource allocation, means for concurrently monitoring and controlling multiple external events, real time clock control, interrupt management, and task dispatching. The iRMX 86 Operating System contains the following modules: An object-oriented Nucleus; Device Independent Basic and Extended I/O Systems; Terminal Handler; Bootstrap and Application Loaders; Human Interface with complete command line interpreter; and an interactive, object-oriented Debugger.

Because the modules and services provided by the operating system are user selectable, application specific operating systems can be created by iRMX 86 users. The iRMX 86 Operating System therefore eliminates the need for custom operating system design, thereby reducing development time, cost, effort, and risk.

FEATURE OVERVIEW

The iRMX 86 Operating System provides users with simple, easy-to-use, quality software tools for creating a wide range of application systems. Some important features are:

Structured Application Environment

The iRMX 86 Operating System provides a consistent structure from application to application, and CPU to CPU, thus allowing experience gained on one system to be easily transferred to others. Often entire programs can be ported from one application to another.

Object-Oriented Architecture

The iRMX 86 Operating System extends the capability of the underlying CPU by adding a number of new data structures (objects) and a number of functions to operate on these objects. This architecture provides a simple, symmetric, and easy to learn interface to a comprehensive system. The Nucleus provides the means to create, manipulate, and delete the basic objects necessary for any application. It also provides a mechanism for users to create their own object definitions and use them as part of the basic operating system. Each of the outer layers of the system add to the list of available objects by using this same extension mechanism.

(P)ROM or RAM Based

The iRMX 86 Operating System can be made resident in (P)ROM or can be loaded into RAM from a secondary storage device using one of the supplied Loaders. Being able to place all system software in (P)ROM offers three benefits: 1) Systems may be moved to harsh environments that preclude the use of disks; 2)The overhead expense of providing mass storage devices can be eliminated; and 3) System performance can be increased by eliminating the wait states required for most RAM's, and disk accesses required for most disk-based operating systems.

User Configurable

Users of the iRMX 86 Operating System are able to use a wide range of features or select only those which meet the specific requirements of a particular application. Each system call provided by the operating system may be removed from the system if it is not used. Each task can specify the use of the 8087 Numeric Data Processor. Jobs can be configured with specific running environments. Individual I/O port addresses are also configurable, making the system ideal for component level applications.

This complete modularity along with many other user options allows users to configure systems in a cost-effective manner regardless of the application environment. Each layer of the system may be configured in this manner, or (with the exception of the Nucleus) left out of the system altogether.

Nucleus

The Nucleus of the iRMX 86 Operating System provides the foundation upon which a variety of applications systems can be built. It includes the facilities to manage the basic objects of the system necessary to perform multiprogramming, multitasking, critical section management, and extensive task-to-task communication and control.

Embedded in the Nucleus are the facilities to support concurrent program execution and handling

of simultaneous asynchronous events. These facilities allow interrupts coming from specialized peripheral devices to be serviced in an efficient manner. The iRMX 86 Operating System allows the CPU hardware to be used by multiple applications, thus reducing the overall system size, complexity, and cost. These facilities are built from four key concepts:

OBJECT MANAGEMENT — Just as floating point numbers are data structures using operators such as multiply and subtract to operate on them, iRMX 86 objects are data structures with system calls to manipulate them. Because of the uniform structure of the system, users have a foundation on which to tailor the Nucleus to the application by removing system calls not necessary for the application and by adding objects and system calls customized for the application. The basic objects of the Nucleus are:

- SEGMENTS — Store data in dynamically created RAM buffers with a specified length.
- MAILBOXES — Provide a mechanism for intertask and interprogram object and data transfer. Mailboxes are locations for objects to be sent and received. For example, using a mailbox for intertask communication permits a time-critical task to forward data to a non-time-critical task for processing. Mailboxes are generally used to pass data segments from task to task, although any object (user or system defined) may be transferred.
- SEMAPHORES — Manage mutual exclusion and synchronization. A semaphore is used to signal another task when processing has been completed or when resources are available. A semaphore provides a low-overhead signalling mechanism.
- REGIONS — Control access to critical sections by allowing only one task at any given time to access a portion of code. Examples are a nonreentrant procedure or code for controlling a peripheral device that can only service one request at a time. In addition, regions can be used to protect data structures from being manipulated by more than one procedure at a time.
- TASKS — Perform the actual work of the application by executing software modules. Each task in the system has the characteristics of a unique processor. It has its own code, priority, stack, data area, and status. If the task is designated as using the 8087 Numeric Data Processor, it also has its own copy of the NDP registers, stack, and status. Task execution is based on an event-driven, priority-based scheduling algorithm.
- JOBS — Permit isolation of application tasks, objects, and memory to provide a multiprogramming environment. Jobs encapsulate an application and limit the degree of interaction between sets of tasks.
- COMPOSITE OBJECTS — Permit users to create objects not found in the set of Nucleus objects. These new objects appear to other facilities in the iRMX 86 system as if they are part of the original system. This means that Composite objects can be manipulated using the Mailboxes and Object Directories in exactly the same manner as other objects.

SCHEDULING — The iRMX 86 Nucleus offers a priority-oriented, event-driven scheduling mechanism that supports up to 255 different priority levels. The scheduler uses the task priority to determine which task receives control of the CPU, and to ensure that the highest priority task ready to execute is given control of the system. That task will continue to run until a higher priority interrupt occurs, or until the running task requests resources that are not available. This priority scheduling allows the system to be responsive to the external environment while allocating resources among the application tasks.

INTERRUPT MANAGEMENT — The iRMX 86 Operating System provides two levels of interrupt management: Interrupt Handlers and Interrupt Tasks. The first optimizes response time, the other optimizes response capabilities. Interrupt tasks allow use of all iRMX 86 system calls and mask only lower priority interrupts. Interrupt handlers permit direct control over the CPU's interrupt logic and only allow the use of interrupt system calls. This structure allows users to easily perform buffering of data while leaving complex processing of the data to interrupt tasks.

For systems requiring more than the 8 interrupt levels of the master 8259A interrupt controller, the iRMX 86 Operating System allows applications to configure up to 7 slave 8259A's. Using the slave devices, systems can respond in real time to as many as 57 interrupt sources.

ERROR MANAGEMENT — When a task issues an iRMX 86 system call, the results may not always

be what the task expects to achieve. For example, the task may request memory that is not available, or it may use an invalid parameter. The iRMX 86 Operating System may be configured to provide two levels of comprehensive error management: hierarchical error handling and selective error processing.

Hierarchical error handling permits a task to handle various errors at different levels of the system. Errors common to a number of tasks can be addressed by system-wide error handlers. Application-specific errors can be routed to job-level handlers. In addition, if a task has a need for a unique error handler, an error handler can be specified for that task. This flexibility means global error handlers can be created for the majority of the errors, reducing the amount of error handling software to be written.

In addition, each application can select the type of error to be processed by the error handlers. The errors are divided into two categories: programmer errors such as invalid parameters; and environmental condition errors such as detection of insufficient memory to meet requirements.

Terminal Handler

The Terminal Handler supports real time, asynchronous I/O between the operator's terminal and application tasks. It provides a line buffer which stores ASCII characters as they are input from the console. Special editing characters are used to control the terminal and the buffer contents, and are not entered into the data. The Handler may be configured as an output-only version to support those applications not requiring terminal input.

I/O System

The iRMX 86 I/O System provides an extensive facility for device independent I/O through a series of supplied device drivers, or any number of user supplied device drivers that can be configured to operate at any I/O port address. The Basic I/O System (BIOS) implements an asynchronous interface to the device drivers allowing users to explicitly overlap I/O functions with other operations. The Extended I/O System (EIOS) performs all of the synchronization necessary to do read-ahead and write-behind buffering automatically, and to reference files with logical names. By configuring the appropriate interface, applications can develop an I/O subsystem with the optimum degree of device control while requiring a minimum of design time and effort. Furthermore, the device-independent nature of the system allows use of different devices without redesign.

The I/O System provides access to three types of files:

- NAMED FILES — allow applications to refer to collections of bytes (files) by using a name. These names are cataloged in directories to allow file access by different tasks and jobs. Directories are special named files that store directory and access information about other named files and directory files.
- PHYSICAL FILES — provide a mechanism to make actual physical connections to storage devices. This type of file is typically used to communicate with simple devices such as printers and terminals.
- STREAM FILES — are mechanisms for communicating between tasks and jobs as if the data were written and then read from a FIFO file.

The named files may be organized in a hierarchical structure as shown in Figure 1, where the triangular files are named data files, and the rectangular files represent directories. This hierarchy allows data to be grouped logically and accessed with a minimum of overhead.

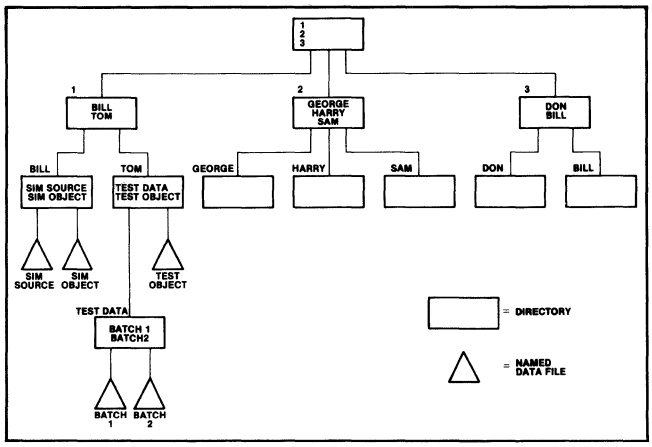


Figure 1. Example of Named-File Tree

Loaders

The iRMX 86 Operating System contains two loaders: A Bootstrap Loader capable of loading a file from mass storage into system RAM; and an Application Loader available to tasks as I/O system calls.

The Bootstrap Loader can be configured to load from a specific device, or to use the first device that becomes ready after the system has been started. It can also be configured to load a file specified by the operator at the system console.

The Application Loader provides a simple mechanism for loading application code and data files into the system. It can be used to load absolute code into a fixed location, or to load relocatable code into dynamically allocated memory locations. It can also be used to support code overlay functions.

Human Interface

The Human Interface is the uppermost layer of the iRMX 86 Operating System. It supports the user by providing a number of utilities useful in typical applications. It also provides the application programmer with a number of tools to generate custom utilities using the basic system utilities or by interfacing directly to the Command Line Interpreter.

Human Interface commands supplied with the iRMX 86 Operating System include commands to perform: creating a directory file; creating, copying, deleting, and renaming files; loading and starting application programs; formatting a device volume; and submitting a command file in a batch mode. The Human Interface also provides some utilities useful in debugging applications: a debug command enabling users to start commands via the system debugger, and a file copying facility used in conjunction with the iSBC 957A Monitor to convert MDS files to and from the iRMX 86 file format.

Interactive System Debugger

The iRMX 86 Operating System provides a comprehensive tool for interactive software debugging. The Debugger has two capabilities that greatly simplify the process of debugging a multitasking system. First, the Debugger allows users to debug several tasks while the balance of the application system continues to run in real-time. Second, the Debugger allows programmers to interactively view and modify system constructs as well as the system RAM and CPU registers. The debugger is structured to enable system designers to track system-wide problems easily. It can also remain in the final application as a continuous maintenance tool.

SPECIFICATIONS

iSBC™ Supported Hardware

SINGLE BOARD COMPUTERS

iSBC 88/40
iSBC 86/05
iSBC 86/12A

MASS STORAGE

iSBC 204	Flexible disk controller
iSBC 206	Hard disk controller
iSBC 215A	Winchester disk controller
iSBC 215B	Winchester disk controller
iSBC 220	SMD disk controller
iSBC 254	Bubble Memory board

MULTIMODULE™ BOARDS

iSBX 218	Flexible disk controller (when used with the iSBC 215)
iSBC 337	Numeric data processor
iSBX 351	Serial I/O channel

User iAPX 86 and iAPX 88 Based Systems

The iRMX 86 system runs on user designed boards with the following components:

8253	Programmable Interval Timer
8259A	Programmable Interrupt Controller
8251A	USART (When the Terminal Handler is configured into the system)
8087	Numeric Data Processor (when NDP tasks are configured into the system)

ORDERING INFORMATION

The ordering options for the iRMX 86 Operating System are listed below. All options include a full year of update service. All the options including the word "KIT" are shipped with a complete set of manuals, the iSBC 957A system monitor for the iSBC 86/12A Single Board Computer, and an iRMX 86 Customer Training Course credit voucher that is valid for 6 months after the date of purchase.

Part Number	Description
RMX 86 KIT ARO, and RMX 86 KIT BRO:	Single and double density OEM license requiring Royalties for each derivative work
RMX 86 KIT AST, and RMX 86 KIT BST:	Single and double density license for one additional development site
RMX 86 KIT ABY, and RMX 86 KIT BBY:	Single and double density OEM Buy-out requiring no further Royalties
RMX 86 AWX, and RMX 86 BWX:	Single and double density update service for an additional year.

intel®

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, CA 95051 • (408) 734-8102 x598

Printed in U.S.A./B-032/1081/20K/BA/UC
AFN-01723B