AWS/IWS

CONVERGENT TECHNOLOGIES

CTOS™ Operating System

- **Real-time, message-based multitasking operating system**
- **Event-driven priority scheduling scheme**
- **Interprocess communication facility**
- **Support for guest operating systems: DISTRIX™, MS™-DOS, and CP/M-86®**
- **Optional multi-partition configuration permits concurrent applications**
- **High level video management supports windowing**
- **Comprehensive file management system takes full advantage of the performance characteristics of the mass storage subsystems**
- **Assured integrity of file information via multi-level password protection and duplication and verification of control structures**
- **Complete software compatibility and transportability between standalone, cluster, and network configurations**
- **Provides machine independent software interfaces**

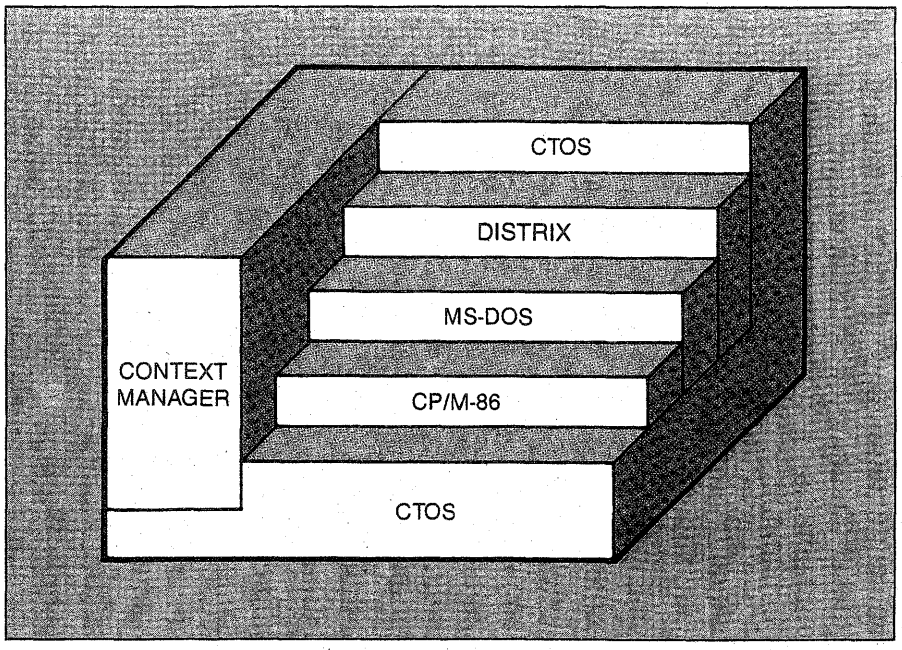
The CTOS Operating System is a real-time, message-based, multitasking operating system for the Convergent™ Family of Workstations. It provides a reliable, high-performance base upon which to build real-time interactive applications.

The CTOS Operating System combines disk- and memory-resident modules and structures to provide a high level of responsiveness with minimum system requirements. It uses an event-driven priority based scheduling algorithm to ensure that processor resources are always assigned to the active processing task with the highest priority. Interprocess communication is performed via messages to pass data and control information to other processes and synchronize their execution. Requests for operating system services are performed through this message scheme.

The CTOS Operating System is designed to take full advantage of the performance characteristics of its hardware environment. Its modular structure supports not only CTOS-based applications but also supports three *guest* operating systems: DISTRIX, MS-DOS, and CP/M-86.

CTOS can optionally be configured in a multi-partitioned environment. The muti-partition version of CTOS separates memory into several *partitions* in which applications may be executing simultaneously. With the Context Manager, each partition may be executing a different application with a different operating system environment (DISTRIX, MS-DOS, CP/M-86, or CTOS).

A unique aspect of the CTOS Operating System is its message-based architecture. The CTOS Operating System provides support for local resource-sharing networks or *clusters*. In cluster configurations, the operating system provides transparent access to system service processes that execute at a master workstation. Inter-workstation communication is handled via a high-speed communications channel.



The CTOS Operating System can be further expanded by means of Convergent's CT-NET which extends the workstation's communication capabilities for operation in Local Area Networks (LANs) and Wide Area Networks (X.25 and both Switched and Leased Point-to-Point connections).

CTOS File Management Services provide efficient, high-speed, reliable access to mass storage files. Speed and efficiency are attained through thoughtful placement of control data structures to minimize disk arm movement and hashing techniques to reduce access time to these structures. Frequently used control structures are maintained in system memory.

The integrity of the file information is protected against hardware malfunction through duplication of key control structures, and against unauthorized access through multimode file access control and multilevel file password protection.

The CTOS Operating System's modular structure combined with a carefully planned model for extension provide an adaptable environment ideal for implementing highly specialized applications.

COMPONENTS OF THE OPERATING SYSTEM

The CTOS Operating System consists of five basic components which include:

- The Kernel
- System Service Processes
- System Common Procedures
- Objective Module Procedures
- Device Handlers and Interrupt Routines

THE CTOS KERNEL

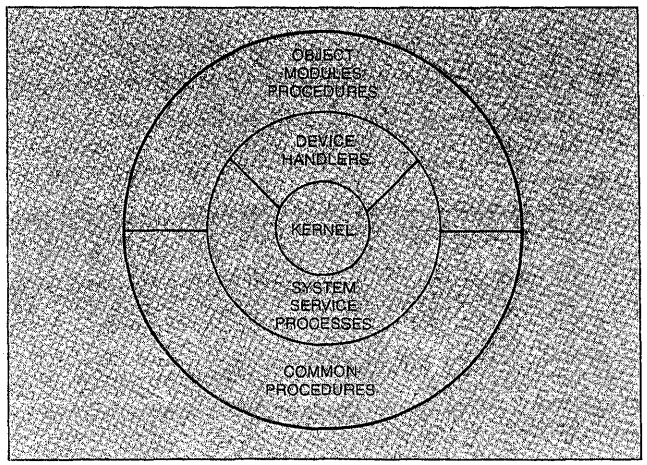
The *Kernel,* the most primitive, yet most powerful CTOS element, provides process, memory, and message management facilities. It is responsible for scheduling the execution of processes, including the saving and restoring of the process state.

A process is the element of computation which competes for access to the processor. Each process is assigned a priority to allow the operating system to schedule its execution appropriately. The CTOS Operating System itself consists of several processes, each assigned a specific set of functions.

The Kernel provides interprocess communication primitives (IPC), which are the building blocks for synchronization and information transmission between processes.

Process Scheduling

Each process has associated with it a priority that indicates its importance relative to the other processes in the system. The priority of a process is assigned when it is created. The CTOS Operating System schedules a process for execution based on its priority. The scheduler maintains a queue of the processes that are eligible for execution. Priority determines which process among those eligible is executed. At any one time, the processor resource is allocated by the operating system to the highest priority executable process. Rescheduling occurs when a system event allows a process with a higher priority to execute.



Operating System Components

Interprocess Communication

The CTOS Kernel provides IPC primitives to facilitate the consistent yet flexible exchange of information between processes. Both system and application processes utilize these primitives. The objects used in the IPC facility are messages and exchanges.

A *message* conveys information and provides synchronization between processes. Although only 32 bits of information are communicated between processes, these 32 bits are typically the address of a larger data structure.

An *exchange* is the path over which messages are communicated from process to process (or from interrupt handler to process). An exchange consists of two first-in-first-out queues; one for processes waiting for messages, the other for messages which have not yet been collected. Each process is assigned an exchange when it is first created.

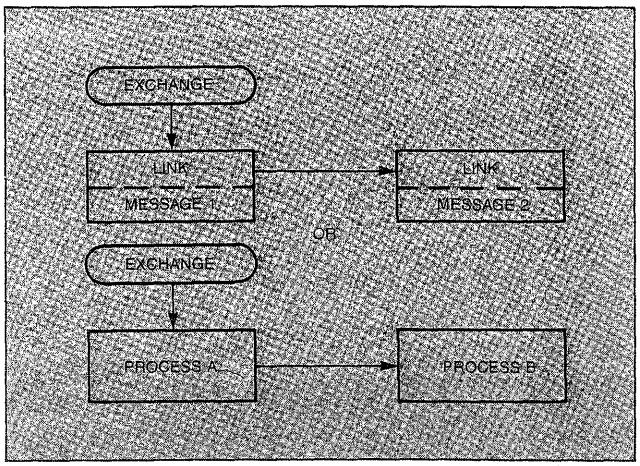
Processes or messages may be queued at an exchange at any given instant. If a process waits on an exchange where message(s) are queued, the oldest message is removed and given to the process. The process then continues execution. Similarly, if a message is sent to an exchange at which processes are queued, the process which waited first is removed from the queue, given the message, and made ready.

All system events are coordinated via messages to exchanges. Interrupts from device controllers or timers communicate with system or application processes by queueing messages on exchanges.

Interprocess Communication Primitives

The CTOS Kernel provides several primitives to support message passing between processes: Send, Wait, Check, Request, and Respond.

- The *Send Primitive* is used to transmit a message from one process to another via an exchange.
- The *Wait Primitive* is issued by a process to wait for a message at a specified exchange.
- The *Check Primitive* is used by a process to inquire if any message is enqueued at the specified exchange.
- The *Request Primitive* is used by a process to access System Services.
- The *Respond Primitive* is used by a system service to notify a client process that the requested service has been completed.



Relationship of Messages, Exchanges and Processes.

SYSTEM SERVICE PROCESSES

The CTOS Operating System consists of several processes which manage system resources. System service processes are scheduled for execution in the same manner as application processes. All communication among processes is accomplished using the IPC primitives. Since there is no assumption as to the physical location of a process, processes can be located on several workstations to achieve a distributed environment.

User Access to System Services

Requests for system services are accomplished by sending a formatted message or *request* to the exchange serviced by the appropriate system service. Each request is assigned a unique identifier or *request code.*

The operating system maps each request code to the exchange serviced by the appropriate system service process. This mapping function allows user applications to correctly function without change in multiple operating system configurations. It also allows operating system functions to be dynamically installed on an as-needed basis.

CTOS system services can be accessed via high-level language procedure calls. Procedures located in the CTOS kernel automatically create the appropriate request and route it to the correct exchange for servicing.

In addition, system services can be accessed via the IPC facility which allows an increased degree of concurrency between multiple input/output operations and computation.

Installable System and User Services

The CTOS Operating System design is highly modular. Only a small subset of the system services must be loaded when the Operating System is first initialized. Other system services may be dynamically installed on an as-needed basis. This feature allows the CTOS Operating System to be tailored according to the requirements of each new installation.

New functions can be added without modifying the operating system. A user can add new operating system functions by installing a new system process which registers its request codes in the System Service Exchange Table.

ADDITIONAL CTOS FACILITIES

System common procedures are CTOS subroutines that the programmer can invoke to perform some common system functions such as accessing the current date and time. System common procedures execute in the same context as the invoking process and at the same priority.

Object module procedures are subroutines that are not part of the CTOS image itself. Most applications systems do not require a full set of these procedures. At the time the application system is built, the desired subset of these object module procedures is bound with the application. Examples of object module procedures are the Sequential Access Method procedures.

Device Handlers and *Interrupt Routines* of the CTOS Operating System are insulated from all but the most sophisticated application system developer by convenient interfaces to the system service processes.

VIDEO MANAGEMENT

The CTOS Operating System provides several operations for video management functions to control the video display in a device independent manner. The screen consists of a number of separate, rectangular areas called *frames.* Each frame can be scrolled horizontally or vertically independent of the other frames.

Applications are provided with three levels of video access: Video Display Management (VDM), Video Access Method (VAM), and the Sequential Access Method (SAM).

The *Video Display Management* functions provide functions to determine the level of video capability, load a new character font into the font RAM, and set up the video frames.

The *Video Access Method* provides direct access to the characters and character attributes of each frame. Character addressing and attribute selection are speci-

fied in a hardware-independent fashion to allow applications to function without change on several different types of hardware.

The *Sequential Access Method* provides device-independent access to devices such as printers, files, keyboard, as well as the video display. Video-specific extensions to the SAM provide direct cursor addressing, control of cursor attributes, and so on.

CONTEXT MANAGER SUPPORT

The CTOS Operating System provides support for multiple virtual screens. With the Context Manager, each partition under its control is allocated a virtual screen. VAM and SAM operations are directed to the virtual screen whenever the partition is not the 'active context'.

For more information on the Context Manager, please refer to the Context Manager Data Sheet.

APPLICATION ENVIRONMENT

Under the CTOS Operating System, an application system is implemented in logical software segments named tasks. These tasks can either be loosely- or tightly-coupled, but all perform related portions of the same application. These tasks execute asynchronously.

A *task* is an executable program. It is created by translating source programs into object modules, and linking them together. The result is stored in a run file on disk. The operating system reads the run file from disk into memory, relocates the inter-segment references, and creates a process which begins execution at the program's entry point.

A process can create new processes or load additional tasks. Each process operates independently and is not controlled by the parent task or process. Processes communicate among themselves and system service processes through the interprocess communication facility.

MULTI-PARTITION CONFIGURATION

The CTOS Operating System may be configured in a multi-partition environment. The multi-partition version of CTOS separates memory into several partitions each capable of handling user applications concurrently.

In the multi-partition version, the CTOS Kernel provides Inter-Partition Communication primitives which allow processes to communicate with processes located in other locations. Process scheduling is modified such that processes of equal priority share the processor resource at regular intervals.

In conjunction with the Context Manager, all of the partitions can control the keyboard and video. Users select the current application by means of a few keystrokes.

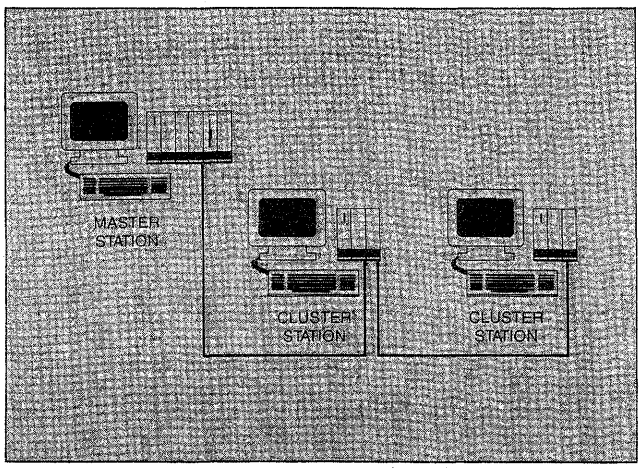
THE CONVERGENT CLUSTER

The modular structure of the CTOS Operating System provides the ideal environment for a distributed architecture. By separating the operating system into several processes which communicate via messages, some system services may be relocated to other physical locations without change to the rest of the system.

In a cluster environment, the Interstation Communication (ISC) Facility is an upward-compatible extension of the Interprocess Communication (IPC) facility of the CTOS Operating System. The Interstation Communication Facility consists of two processes: the Master Workstation Agent process which resides at the Master Workstation; and the Cluster Workstation Agent process at each Cluster Workstation. These two processes exchange information utilizing the high speed communications line operating at speeds up to 1.8 million bits per second.

The Cluster Workstation Agent has the function of converting interprocess requests into interstation messages for transmission to the Master Workstation. The Master Workstation Agent reconverts the interstation message to an interprocess request, and queues it onto the exchange of the service process which performs the desired function.

Applications require no change for operation in standalone, master workstation, or cluster workstation environments. The Interprocess Communication (IPC) and Interstation Communication (ISC) Facilities automatically route requests to the appropriate service for processing.



Typical Cluster Configuration

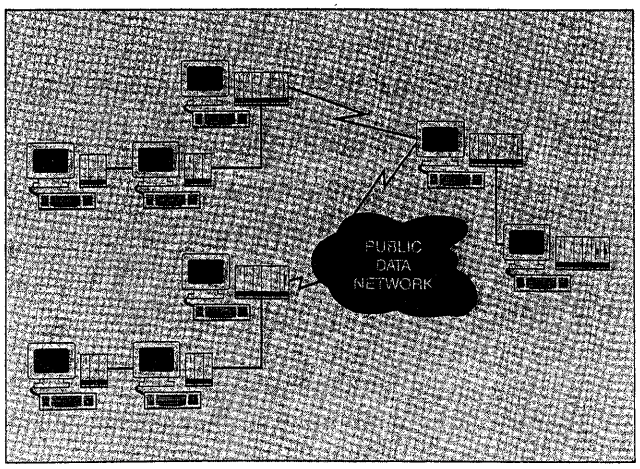
CT-NET SUPPORT

CT-NET is a logical extension to the workstation cluster. Access to Local and Wide Area Networks is achieved through additional system services which extend the Interstation Communication Facility across several workstations and clusters referred to as *nodes*.

Unlike other operating system implementations, the standard CTOS utilities can be used for access to files at other network nodes. User applications may access system services located at any node in the network.

FILE MANAGEMENT

The CTOS File Management Services are efficient, reliable, and convenient to use.



Typical CT-NET Configuration

Efficiency is provided through careful data placement to minimize disk arm movement, input/output directly into the application area without intervening operating system access, and retention of the most recently used directory information in memory.

Reliability is provided through multi-level password protection to prevent unauthorized access, and by duplication of volume control structures to ensure that damage to one copy does not result in loss of data.

Convenience is provided through a hierarchical file organization, long file names, dynamic file length, automatic recognition of volumes when placed online, and device independence.

File System Structure

The CTOS File Management System has a hierarchical organization by node, volume, directory, and file.

Node

A *node* selects a workstation or cluster in a CT-NET network.

Volume

A *volume* is the media of a disk drive that was formatted and initialized. A floppy disk and the media sealed inside a Winchester disk drive are examples of volumes.

Duplicate volume control structures are maintained to ensure reliability. The primary and secondary copies are located on different cylinders and at different rotational positions and, when possible, are accessed by different read/write heads. These duplicates ensure that damage to one copy does not cause loss of data.

Directory

The files of a volume are divided into one or more directories. A *directory* is a collection of related files on one volume. The maximum number of directories that can be created on a volume is specified when the volume is initialized.

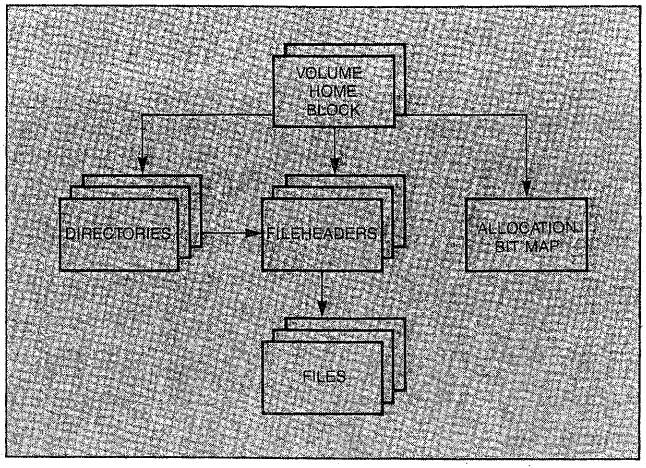
The maximum number of files that can be created in each directory is specified when the directory is first created. Each directory can be protected with a password.

File

A *file* is a set of related disk records which are treated as a unit. The files of a volume consist of integral numbers of 512-byte sectors and must be completely contained on that volume. There are no other restrictions on file size.

Information about each file is maintained in a *file header block* which, like other volume control information, is duplicated to ensure against loss of data. The CTOS file organization provides for very efficient retrieval of sectors. A file is organized as one or several *disk extents*, or contiguous areas on the disk. The list of available sectors is maintained in a freespace *bit map*. File space is allocated in a manner which optimizes for contiguous areas on the disk.

When a file is opened, the location of the disk extents is read into memory. Retrieval of data sectors is highly efficient since all of the information concerning its location is already memory-resident. No directory information must be retrieved from disk after the file is opened.



Volume Structures

File Protection Levels

A file is assigned a file protection level which specifies the access allowed to a file. CTOS provides nine levels of file protection.

Passwords can be assigned at the volume, directory, and file level. Volume passwords allow unrestricted access to all files within the volume.

Automatic Volume Recognition

The CTOS System automatically recognizes volumes placed online. For example, when a floppy disk is inserted into a disk drive, the system reads the disk to determine whether it contains a volume and, if it does, that no other volume of the same name is already online.

Volume references are first searched at the workstation's local storage and then, in cluster workstations, at the master workstation.

FILE MANAGEMENT OPERATIONS

The CTOS Operating System contains several File Management Operations to control the operations of the CTOS File System.

All of the operations for the creation and maintenance of volumes, directories, and files are directly available to the application.

The file management system provides 512-byte sector access to files for the most efficient data access. Applications which require fast data access can read or write up to 65,000 bytes of data in one operation. Operations are provided to asynchronously transfer data such that applications can overlap input/output operations with computation.

CTOS provides two modes of file access: shared and exclusive. *Shared-access files* may be accessed simultaneously by several applications located within the workstation or on multiple workstations in cluster configurations. Shared files are opened read-only. Files opened for *exclusive access* are available solely to the application for access and modification.

Users requiring shared-access to files for modification may utilize the Indexed Sequential Access Method (ISAM) described below.

FILE ACCESS METHODS

Several access methods augment the file management system. These access methods are:

Sequential Access Method (SAM): SAM allows the sequential reading and writing of byte strings of arbitrary length. The Sequential Access Method is device independent and is compatible with access to the video, keyboard, communications ports, printers, and spooler.

Record Sequential Access Method (RSAM): RSAM allows the sequential reading and writing of variable length records.

Direct Access Method (DAM): DAM allows reading, writing, and deleting by record number. The fixed length of the records in a DAM file is specified when the file is created.

Indexed Sequential Access Method (ISAM): ISAM allows fixed-length records to be stored, updated, retrieved, and deleted by any of multiple record keys. The implementation uses a balanced tree (sometimes called "B-tree" or "block splitting"). ISAM allows multiuser access to a file in which individual records may be locked for retrieval or update.

CT-DBMS is a relational data base management system which provides data independent access to fixed-length records. CT-DBMS provides the capablities offered by ISAM and, in addition, field-level security, transaction rollback, and recovery of transactions by roll forward from the archive.

DEVICE MANAGEMENT

Device Management provides access to physical devices at a more primitive level than File Management. Access at this level is intended for formatting new media or when reading or writing disks in non-CTOS formats for exchange with other types of computers.

COMPATIBILITY

The CTOS Operating System executes on the IWS™, AWS™, and NGEN™ Families of Workstations and the Convergent MegaFrame™.

Convergent Technologies™ 2500 Augustine Drive, Santa Clara, CA 95051
(408) 727-8830 TWX 910-338-2149

10K-0584-BA 11-00004-C