System Programmer's Guide

Convergent Technologies

SYSTEM PROGRAMMER'S GUIDE

Specifications Subject to Change.

Fourth Edition (February 1982)
This edition (numbered A-09-00014-02-C) replaces the previous editions (numbered A-09-00014-02-B, A-09-00014-02-A, and A-09-00014-01-A) and makes them obsolete.

Contents

PREFACE

It is assumed that the reader of this guide is very familiar with computer concepts and terms. Further, it is assumed that the reader has already read and has available for reference:

Operator's Guide
Installation Guide
CTOS Operating System Manual
Executive Manual
Utilities Manual
Workstation Hardware Manual
Peripherals Hardware Manual

It may be of interest that this document was prepared using the Convergent Word Processor.

SUMMARY OF CHANGES

This edition (09-00014-02-C) of the System Programmer's Guide differs from the preceding one (09-00014-02-B) in the ways summarized below.

AWS WORKSTATIONS

Several sections have been added, and all sections have been updated to reflect the introduction of the AWS 220, AWS 230, and AWS 240 workstations.

SYSINIT and SIGNON

Notes have been added to Section 5 (System Programming) which describe these new features.

EXECUTIVE

The Executive's handling of the font and the command table has changed. See "Notes on the Executive" in Section 5 (System Programming).

CUSTOMIZING SAM

The description of OpenByteStream has been updated to reflect the use of the Queue Manager by Spooler Byte Streams.

APPENDIX

The four appendicies have been removed. These were source listings of the assembly language modules Sysgen.Asm, Request.Asm, RqLabl.Asm, and Samgen.Asm. These source files are contained in the Standard Software release diskettes.

GUIDE TO TECHNICAL DOCUMENTATION

This Manual is one of a set that documents the Convergent™ Family of Information Processing Systems. The set can be grouped as follows:

Introductory
- Installation Guide
- Operator's Guide
- Executive Manual

Hardware
- Workstation Hardware Manual
- AWS-210 Hardware Manual
- Peripherals Hardware Manual

Operating System
- CTOS™ Operating System Manual
- System Programmer's Guide
- System Utilities Manual
- Batch Manual

Programming Languages
- COBOL Manual
- FORTRAN Manual
- BASIC Manual
- Pascal Manual
- Assembly Language Manual

Program Development Tools
- Editor Manual
- Debugger Manual
- Linker/Librarian Manual

Data Management Facilities
- ISAM Manual
- Forms Manual
- Sort/Merge Manual

Text Management Facilities
- Word Processing Manual
- Font Designer Manual

Communications
- Asynchronous Terminal Emulator Manual
- 3270 Terminal Emulator Manual
- 2780/3780 RJE Terminal Emulator Manual

This section outlines the contents of these manuals.

Introductory

The Installation Guide describes the procedure for unpacking, cabling, and powering up a system.

The Operator's Guide addresses the needs of the average user for operating instructions. It describes the workstation switches and controls, keyboard function, and floppy disk handling.

The Executive Manual describes the command interpreter, the program that first interacts with the user when the system is turned on. It specifies commands for managing files and invoking other programs such as the Editor and the programming language compilers.

Hardware

The Workstation Hardware Manual describes the mainframe, keyboard, and video display. It specifies system architecture, printed circuit boards (Motherboard, Processor, I/O-Memory, Video Control, ROM Expansion, and RAM Expansion), keyboard, video monitor, Multibus interface, communications interfaces, power supply, and environmental characteristics of the workstation.

The AWS-210 Hardware Manual describes the mainframe, keyboard, and video display of the AWS-210 workstation. It specifies architecture, theory of operation of the printed circuit boards, (Motherboard, Deflection, and CPU), keyboard, video monitor, expansion interface, communications interface, power supply, and environmental characteristics of the workstation.

The Peripherals Hardware Manual describes the disk subsystems. It specifies the disk controller Motherboard, controller boards for the floppy disk and the Winchester disks, power supplies, disk drives, and environmental characteristics.

Operating System

The CTOS™ Operating System Manual describes the Operating System. It specifies services for managing processes, messages, memory, exchanges, tasks, video, disk, keyboard, printer, timer, communications, and files. In particular, it

specifies the standard file access methods: SAM, the Sequential Access Method; RSAM, the Record Sequential Access Method; and DAM, the Direct Access Method.

The System Programmer's Guide addresses the needs of the system programmer or system manager for detailed information on Operating System structure and system operation. It describes (1) cluster architecture and operation, (2) procedures for building a customized Operating System, and (3) diagnostics.

The System Utilities Manual describes utilities such as Backup Volume, IVolume, Restore, Change Volume Name, PLog, Maintain File, Dump, etc.

The Batch Manual describes the batch manager, which executes batch jobs under control of job control language (JCL) files.

Programming Languages

The COBOL, FORTRAN, BASIC, Pascal, and Assembly Language Manuals describe the system's programming languages. Each manual specifies both the language itself and also operating instructions for that language.

The Pascal Manual is supplemented by a popular text, Pascal User Manual and Report.

The Assembly Language Manual is supplemented by a text, the Central Processing Unit, which describes the main processor, the 8086. It specifies the machine architecture, instruction set, and programming at the symbolic instruction level.

Program Development Tools

The Editor Manual describes the text editor.

The Debugger Manual describes the Debugger, which is designed for use at the symbolic instruction level. Together with appropriate interlistings, it can be used for debugging FORTRAN, Pascal, and assembly language programs. (COBOL and BASIC, in contrast, are more conveniently debugged using special facilities described in their respective manuals.)

The Linker/Librarian Manual describes the Linker, which links together separately compiled object files, and the Librarian, which builds and manages libraries of object modules.

Data Management Facilities

The ISAM Manual describes the multikey Indexed Sequential Access Method. It specifies the procedural interfaces and shows how these interfaces are called from the various languages.

The Forms Manual describes the Forms facility that includes (1) the Forms Editor, which is used to interactively design and edit forms, and (2) the Forms run time, which is called from an application program to display forms and accept user input.

The Sort/Merge Manual describes (1) the Sort and Merge utilities that run as a subsystem invoked at the Executive command level, and (2) the Sort/Merge object modules that can be called from an application program.

Text Management Facilities

The Word Processing Manual describes the word processor. It specifies the interactive word processor and the list processor that merges text from records into the blanks of a form document.

The Font Designer Manual describes the interactive utility for designing new fonts (character sets) for the video display.

Communications

The Asynchronous Terminal Emulator Manual describes the asynchronous terminal emulator.

The 3270 Terminal Emulator Manual describes the 3270 emulator package.

The 2780/3780 RJE Terminal Emulator Manual describes the 2780/3780 emulator package.

specifies the standard file access methods: SAM, the Sequential Access Method; RSAM, the Record Sequential Access Method; and DAM, the Direct Access Method.

The System Programmer's Guide addresses the needs of the system programmer or system manager for detailed information on Operating System structure and system operation. It describes (1) cluster architecture and operation, (2) procedures for building a customized Operating System, and (3) diagnostics.

The System Utilities Manual describes utilities such as Backup Volume, IVolume, Restore, Change Volume Name, PLog, Maintain File, Dump, etc.

The Batch Manual describes the batch manager, which executes batch jobs under control of job control language (JCL) files.

Programming Languages

The COBOL, FORTRAN, BASIC, Pascal, and Assembly Language Manuals describe the system's programming languages. Each manual specifies both the language itself and also operating instructions for that language.

The Pascal Manual is supplemented by a popular text, Pascal User Manual and Report.

The Assembly Language Manual is supplemented by a text, the Central Processing Unit, which describes the main processor, the 8086. It specifies the machine architecture, instruction set, and programming at the symbolic instruction level.

Program Development Tools

The Editor Manual describes the text editor.

The Debugger Manual describes the Debugger, which is designed for use at the symbolic instruction level. Together with appropriate interlistings, it can be used for debugging FORTRAN, Pascal, and assembly language programs. (COBOL and BASIC, in contrast, are more conveniently debugged using special facilities described in their respective manuals.)

The Linker/Librarian Manual describes the Linker, which links together separately compiled object files, and the Librarian, which builds and manages libraries of object modules.

Data Management Facilities

The ISAM Manual describes the multikey Indexed Sequential Access Method. It specifies the procedural interfaces and shows how these interfaces are called from the various languages.

The Forms Manual describes the Forms facility that includes (1) the Forms Editor, which is used to interactively design and edit forms, and (2) the Forms run time, which is called from an application program to display forms and accept user input.

The Sort/Merge Manual describes (1) the Sort and Merge utilities that run as a subsystem invoked at the Executive command level, and (2) the Sort/Merge object modules that can be called from an application program.

Text Management Facilities

The Word Processing Manual describes the word processor. It specifies the interactive word processor and the list processor that merges text from records into the blanks of a form document.

The Font Designer Manual describes the interactive utility for designing new fonts (character sets) for the video display.

Communications

The Asynchronous Terminal Emulator Manual describes the asynchronous terminal emulator.

The 3270 Terminal Emulator Manual describes the 3270 emulator package.

The 2780/3780 RJE Terminal Emulator Manual describes the 2780/3780 emulator package.

1. GETTING STARTED

Convergent software is distributed on floppy diskettes. The CTOS operating system is contained on several consecutively numbered floppy diskettes. These diskettes are configured so that it is possible to boot a workstation from one of the diskettes and run such utilities as IVolume, Backup Volume, and Restore.

The exact contents of each distribution diskette is listed in the Release Notice.

Software Installation

The Release Notice for Standard Software contains detailed instructions for software installation. After the standard software is installed, optional software should be installed as described in the Release Notice for the particular software. The contents of Convergent distribution diskettes is also listed in the Release Notice for the standard or optional software.

Using the OS in a hard disk-based environment

When you have finished the installation procedures described in the Release Notices, your hard disk system will be ready for use. These procedures place the Convergent software in directory <Sys>, and create the directory <Spl> for the use of the Spooler. You will wish to create more directories before you begin to create files. Directories allow you to group related files and to protect those files with a common password. The "Create Directory" command is described in the Executive Manual.

Using The OS in a Cluster or Mini-Cluster Environment

The use of the OS in a Cluster or Mini-Cluster environment is the same as that in a single user environment, except that the disks connected to the Master Workstation are shared by all workstations in the cluster. As in the hard disk-based environment, directories should be created on the hard disk for grouping your files.

Part of the installation procedure installs a special directory ("$") for each user. These directories are useful for programs which need to use a temporary file. Placing the file in the $ directory allows the program to execute at multiple workstations without collision

in file names. These directories are described in the "File Management" section of the CTOS Operating System Manual.

Using the OS in an IWS floppy disk-based environment

The CTOS distribution floppy disks that you received are your master copy of Convergent software. They are shipped without write enable tabs and contain little or no unused file space. Do NOT put write enable tabs on these floppies and/or attempt to use them as your working copy of the operating system.

You will also have received a "Starter Kit" set of diskettes with your floppy disk-based system. The contents of these disks is described in the Release Notice.

It is recommended that you don't actually use these diskettes, but that you copy them and use the copies. You may easily copy a diskette using tests 5 (format) and 12 (copy) of the Floppy Diagnostic (see "Diagnostics" section of this manual).

One of these diskettes will be for bootstrapping CTOS, and running certain system utilities, and the others will be for running various programs supplied by Convergent. The minimum set of files for a bootable diskette is

```
SysImage.Sys
Signon.Run
Sys.Font
.User
```

A diskette for using the Executive will need to contain the files

```
Exec.run              (or CommExec.Run)
Files.run             (optional)
VolumeStatus.run      (optional)
Sys.Cmds
```

You may want to copy the files from the distribution floppy disks onto a larger number of floppy disks in order to retain file space on each floppy disk for the creation of work files (for the assembler, editor, etc.). While the grouping of files on floppy disks is a matter of preference, you may want to consider this organization:

1. a system floppy that contains:

```
a CTOS system image
Signon.Run
```

.User
Exec.Run
Sys.Font
Sys.Cmds
Copy.Run
Command.Run
Files.Run
Format.Run

2. an Editor floppy that contains:

Exec.Run
Sys.Cmds
Editor.Run

3. an Assembler floppy that contains:

Exec.Run
Sys.Cmds
Assembler.Run

4. a Linker/Librarian floppy that contains

Exec.Run
Sys.Cmds
Librarian.Run
Linker.Run
CTOS.Lib

5. a floppy for each optional language (BASIC, FORTRAN, COBOL, Pascal)

Using The OS in an AWS-220 or AWS-230 standalone environment

Using the OS in AWS standalone environments requires a fairly well planned strategy that organizes the application into functional units partitioned between mini-floppy diskettes. This is necessary in order to minimize the amount of diskette switching that the user must perform due to the limited capacity of the mini-floppies. The suggested strategy is similar to that used for the user copies of the system software mentioned in the previous section.

As an example, we have a 580-line BASIC application that utilizes Convergent Forms and ISAM, and includes a 400-line Pascal module. This application uses 5 forms and 3 ISAM data sets (each of which is composed of 2 files).

We have configured this application to run on an AWS-220, using one diskette to boot from and a second to contain the data sets. The user places the boot diskette in the floppy drive (f0) and boots. After CTOS initialization and system Signon the application is loaded. It reads the 5 forms from the boot diskette, then requests that the user replace the boot diskette with his data diskette. It then opens the 3 data sets and proceeds with the application.

The boot diskette contains the following files:

```
SysImage.sys
Signon.Run
Signon.txt
.User
Basic.Run
Application.basic    (The application program)
Menu.form
AddStock.form
AddNewItem.form
EnterOrder.form
FillOrderLine.form
```

And the data diskette contains:

```
Item.isam        Item.ind
Order.isam       Order.ind
OrderLine.isam   OrderLine.ind
```

Thus we have a "natural" partitioning into the system and user environments. Diskette 1 contains the operating system (sysImage.sys), the system support utilities and data (Signon.Run, Signon.txt, and .User), the BASIC interpreter (Basic.Run), the application BASIC program (Application.basic), and the forms used

by the application. Diskette 2 contains the data sets used by the application. Once the system is booted the user has signed on, the boot diskette is no longer needed. The data diskette is used for all subsequent interactions.

On a 22Ø we take maximum advantage of the one diskette drive. On a 23Ø we could either leave both diskettes mounted or allow for two data diskettes.

Using The Debugger

The use of the Convergent Debugger is straightforward. The debugger is automatically installed on hard disk and Cluster systems by the procedures described in the Release Notice for Standard Software.

In floppy disk-based systems, use of the debugger is more complicated. The debugger requires two files on the system diskette for swapping memory back and forth to disk. These files are opened for the debugger during CTOS initialization. Since the removal of a floppy diskette from the floppy drive causes all files on that diskette to be closed by CTOS, you must not remove the system floppy from drive Ø if you plan to use the debugger. Also, this diskette must not be write protected, as the debugger needs to be able to write to it.

The minimum set of files which you must place on this diskette are as follows:

```
Exec.run
Sys.Cmds
Sysimage.sys
Sys.Font
Debugger.sys
Signon.Run
.User
```

You must leave at least 96 sectors free on the floppy, so that the second debugger file, DebuggerSwap.sys, may be created by CTOS during its initialization.

2. CTOS FAILURE ANALYSIS

CTOS Initialization Error Status Description and Analysis

The CTOS operating system tests the following hardware components in its initialization process:

Memory parity error detection circuitry
Memory above 128K
Keyboard
Interrupt Circuitry
Programmable interval timer (PIT)

In addition, on IWS workstations the following hardware components are tested during CTOS initialization:

Bus timeout circuitry
Video
Real time clock (RTC)

If the video test succeeds but any of the other tests fail, it will display the following message on the screen:

INITIALIZATION ERROR STATUS xxxxh

Each bit set in the error status word corresponds to an error condition detected during the test. The meaning of each bit is described below. The operating system will continue to load the Executive if any error other than a video error is detected. If the video test fails, the operating system will halt and beep 10 times. It also displays the error code in the LED's on the keyboard (and on the Memory-I/O board of IWS workstations). In order to distinguish from the error codes which are displayed by the bootstrap ROM, the operating system turns on the LED's on the OVERTYPE key and on the LOCK key.

The LED's on the Memory-I/O board of IWS workstations are numbered according to the following convention: if you are facing the LED's on the Memory-I/O board, the right most LED is LED 0 and the left most one is LED 5.

Memory- I/O	Keyboard LED	LED Error	Description
Ø		F1Ø	IWS - Video hardware does not respond. Possible cause: (a) There is no video board (b) The video board not seated
1		F9	IWS - Dma failure in "load font"
2		F8	IWS - Dma failure in "read font".
3		F3	IWS - The font read back from the Font RAM fails to compare with the font written to it.

The errors generated by other tests are recorded in the bits of the initialization error status word that is displayed on the video screen. The meaning of each bit is:

Bit 4	Memory test failure (128k and up).
Bit 5	IWS - Bus timeout interrupt is not generated when a non-existent memory location is referenced.
Bit 6	Bad memory parity is not detected.
Bit 7	Keyboard hardware does not respond. Possible causes are: (a) Keyboard not connected

(b) 8Ø48 in the keyboard

(c) 8251A USART

Bit 8 Keyboard does not return good status after the reset command. Possible cause are:

(a) keys are pressed during the initialization.

(b) 8Ø48 in the keyboard

Bit 9 Keyboard ROM checksum failure. Possible cause: Bad 8Ø48 in the keyboard.

Bit A Keyboard loopback test failure.

Bit B Keyboard interrupt test failure - no interrupt is generated during loopback, or TRANSMIT READY status in 8251A does not generate interrupt.

Bit C IWS RTC test failure - no RTC interrupt, or time interval between two RTC interrupts is inconsistent with the time interval measured by PIT.

Bit D PIT test failure - no PIT interrupt.

Bit E Continuous PIT interrupts.

Bit F (Mini-Cluster master workstation CTOS only) Communications hardware test failure.

CTOS Crash Status Description and Analysis

When the CTOS operating system detects a fatal error condition, it reports the error, dumps memory to a crash file (if the CrashDump.Sys file exists) and reboots itself.

If the Debugger is configured into the operating system and is loaded in memory when the fatal error occurs, the operating system will enter the debugger before it does a memory dump and reboot. You can use the debugger to investigate the cause of the fatal error. When the GO key is pressed, the debugger will exit and the system will proceed with the memory dump and reboot sequence.

The error message is displayed on the video screen in the system crash and reboot sequence. It is displayed when the error condition is detected, and also when the debugger is entered. During system reboot, the video screen is blanked but the error messages reappear after the OS is reloaded. The same information is again displayed by SysInit and Signon when they reinitialize the screen. The information is also placed in the system log file, [Sys]<sys>Log.Sys (use the PLog command to display the log file).

The error messages contain an error code and eight status words. They are displayed in the following format:

```
CRASH STATUS (ERC xxx.)
     xxxxh xxxxh xxxxh xxxxh xxxxh xxxxh xxxxh xxxxh
```

The decimal value of the error code is displayed in the parentheses. The eight status words are displayed in hexdecimal. See Appendix A of the CTOS Operating System Manual for a description of the error codes.

The information placed in the system log file also contains a ninth word which is the workstation type of the station which logged the error. The types are defined as follows:

Ø ... Standalone IWS
1 ... Cluster workstation
2 ... Mini-Cluster master workstation
3 ... CommIOP cluster master workstation

4 ... Application cluster workstation
5 ... Application standalone workstation

The first status word contains the hexdecimal error code. The second word is the process number of the process that was running when the fatal condition occurred. The seventh and the eighth word contain the CS and IP of the instruction following the procedure call to the CTOS fatal error handler, unless specified otherwise. The other four words either are unused or have information unique to each error condition. The error conditions which use those four words are described below.

Error Code	Description Of Error Status
21	Memory protection fault. The third word contains the value of port 56h. The seventh and eighth words are CS and IP when the memory protection fault interrupt was detected.
22	Bus Timeout. The seventh and eighth words are the CS and IP of the instruction following the one that caused the bus timeout (usually by doing I/O to a non-existent port or referencing a non-existent memory location). (On an IWS, the third word contains the value of port 56h.)
23	Memory Parity. The seventh and eighth words are CS and IP when the parity error interrupt was detected. The fifth and the sixth words indicate the memory location where the parity error was detected. The fifth word contains the 16 least significant bits of the 20 bits physical memory address and the 4 low order bits of the sixth word contain the 4 most significant bits. The

fourth word contains the current contents of the memory location where the memory parity error was detected. (On an IWS, the third word contains the value of port 56.)

24 Power failure (IWS only). The third word contains the value of port 56. The seventh and eighth words are CS and IP when the power failure interrupt was detected.

25 Unknown non-maskable interrupt. The third word contains the value of port 56. The seventh and eighth words are CS and IP when the non-maskable interrupt was detected.

26 Stray interrupt. The third word contains "interrupt type" multiplied by 6 The seventh and eighth words are the CS and IP when the interrupt was detected.

On an IWS workstation, the fourth and fifth words contain the values of the ISR and the IRR register of 8259A respectively. The sixth word contains the value of the mask register of the 8259A.

27 Divide overflow. The seventh and eighth words are the CS and IP of the instruction following the one that caused the overflow. The other status words contain the same information as they would for an error 26.

3. BUILDING A CUSTOMIZED CTOS SYSTEM IMAGE

The versions of the CTOS Operating System that are contained on the set of distribution floppy disks are configured to support the maximum set of peripherals and features. The parameters are set to accomodate relatively high (but not maximum) system throughput. After you have become familiar with the operating system, you may wish to build a customized version that: changes parameters, excludes features that you do not need, incorporates your own system service processes and device handlers.

This section of the System Programmer's Guide is in four parts, covering the different aspects of building a customized OS. Additional information is also available in the "CTOS Configuration" section of the current Release Notice for Standard Software.

System Build

On the Standard Software release disks (refer to the release notice for directories of the diskettes) are the file "Sysgen.Asm" plus "prefix" files for each version of CTOS that are assembled with "sysgen.Asm" ("Swp.Asm", "Res.Asm", "Mws.Asm", etc.). These files contain definitions in which various CTOS parameters are defined. This section describes the various CTOS structures that the sysgen parameters affect.

In the "Sysgen.Asm" module (and also in "Request.Asm"), there are various conditionals based on the type of CTOS being built. These conditionals are documented in the file "Sysgen.Asm" mentioned above.

As the conditionals defined in Sysgen.Asm vary slightly from release to release, whenever you get a new release of the Standard Software, you should read the comments contained in the beginning of Sysgen.Asm before making any changes to that file or to the related "prefix" files that are assembled with it.

When making changes in "sysgen.asm" or "request.asm", make sure that the change is made for the appropriate type of CTOS you are creating.

Sysgen %SET Macros

Near the beginning of "Sysgen.Asm" are several calls to The Assembly Language macro "%Set". Care should be taken when changing the values in the macros, as different values will cause CTOS's size to change. The form of the macro calls is "%Set(parameter, value)", where "parameter" is a counter or flag used in building CTOS structures, and "value" is a non negative integer that is assigned to "parameter". Each of these is described in detail in comments in "Sysgen.Asm", along with instructions on how to change them.

These declarations may be made in the "prefix" file which is assembled with "Sysgen.Asm", in which case the declarations in "Sysgen.Asm" are ignored.

Keyboard Translation Table

In "sysgen.asm" is an array called "rgInfoIKey". This is used by the keyboard process to map keystrokes to characters. Each entry in the array has three fields, the value returned when unshifted, when shifted, and any attributes that may apply. In front of the table, is a section that defines the masks which may be OR'd together to define the attributes assigned to any character.

Device Declarations

Near the end of Sysgen.Asm are the device declarations. A device is a floppy disk or a hard disk. For each device, CTOS allocates a structure called a Device Control Block ("dcb") which has a size of 76 bytes. The default definitions for IWS operating systems are for 3 hard disks and 2 floppy disks. There is an Assembly Language macro used to define these devices, "dcbDisk". The fields of each macro are defined in Sysgen.Asm. Each device is assigned a name and a password in its declaration. The default names are "fØ", "f1" (floppy disks), "dØ", "d1", "d2" (hard disks). For each hard disk device the password assigned is the same as its name. The floppies are not assigned passwords in the standard release.

Adding CTOS System Service Processes

You can add custom services to CTOS. These may be in two forms, initialization routines that are executed when CTOS is first booted, and resident CTOS system service processes.

Initialization Routines -- There are three kinds of CTOS initialization routines, "hardware", "intDisable", and "intEnable". The "hardware" routines are specified in "sysgen.asm" with the macro "InitProcHardware()". The "intDisable" routines are called before all of the CTOS processes are initialized, and before interupts are enabled. The "intEnable" routines are called after the processes have been started, with interupts enabled. These three types of routines are described in some detail in "sysgen.asm". In all three cases, object modules which contain the subroutines must be added to the appropriate link list ("objLinkSwp", "objLinkRes", "objLinkCws", etc.). The new modules should be placed after "sysgen.obj" and before "osEnd.obj" in the list of object modules.

Resident CTOS services: process declarations -- A user service may be added to CTOS in the form of a resident process. This is added by declaring the process interface in "sysgen.asm", and by including the object modules related to the process in the correct "objLinkxxx" file. Any intitialization routines are linked after "sysgen.obj" in the list, and the resident code before "sysgen.obj". The process is declared using the macro "OsProcDesc", with four parameters, entry point, stack size, priority, and default exchange. The entry point is the address (CS:IP) that will be be used when the process is first started (placed on the run queue). The stack size is the maximum number of bytes that the process will need to execute, including space for local (stack) variables, procedure calls, CTOS request interface routines (which use 64 bytes to build a request on the caller's stack), and various interrupts (Convergent requires at least 128 bytes dedicated for this purpose in order to save process context). The process priority must be picked carefully. In general, a process which is a "user" of services supplied by another process should have lower priority, but the writer of any new process must decide the priorities. The last parameter in a process declaration is the default exchange. If this is not required (i.e. the process does not use any of the CTOS services which require a default exchange) then Ø may be used. If the process does require a default exchange (or any other exchanges) then these exchanges should be allocated in "sysgen.asm" by setting the parameter "nSysExchange" to a higher number (Convergent processes require 24 exchanges).

Resident CTOS services: request interface(s) -- A process which implements a new service must have some way of interacting with users. The mechanism is the "request". This may be accomplished by intercepting a request generated by the Convergent interface, or by adding a new request. To intercept a request, all that is required is to replace the exchange number in the "SysRequest" macro with an exchange which is dedicated to the new process. This new exchange should NOT be the default exchange of the process. The new process would then "wait" at it's dedicated exchange for messages. The addition of a new request means using the "UsrRequest" macro in "request.asm" in the same fashion as the "SysRequest" macros define requests for services provided by Convergent CTOS processes. The fields are the exchange where the request is to be sent, the size of the control information in the request, the number of request and response "PbCb"'s, the local service code, the request number, and a description of the arguments put on the stack when the CTOS subroutine interface is used. The "SysRequest" and "UsrRequest" macros generate ten tables which may be used by a process in servicing a request. The tables and their sizes are public and are in segments defined as follows:

name:	segment name:	segment class:	segment group:	type:
RgRqExchgSys	RqSegØ	Const	DGroup	word
RgSCntlInfoSys	RqSeg1	Const	DGroup	word
RgNReqPbCbSys	RqSeg2	Const	DGroup	byte
RgNRespPbCbSys	RqSeg3	Const	DGroup	byte
RgLocalServiceCodeSys	RqSeg4	Const	DGroup	word
RcLookUpSys	OCode	Code	RqGroup	word
nSysRequest	Data	Data	DGroup	word
RgRqExchgUsr	RqSegØ	Const	DGroup	word
RgSCntlInfoUsr	RqSeg1	Const	DGroup	word
RgNReqPbCbUsr	RqSeg2	Const	DGroup	byte
RgNRespPbCbUsr	RqSeg3	Const	DGroup	byte
RgLocalServiceCodeUsr	RqSeg4	Const	DGroup	word
RcLookUpUsr	OCode	Code	RqGroup	word
nUsrRequest	Data	Data	DGroup	word

CTOS Procedural Interface

The tables are indexed into by the request number (times 2 for word tables, complemented for "UsrRequest"'s). The primary function of these tables is to allow CTOS to format a request block for the

user, based on the request number of the interface that the user has called. For example, suppose a user invokes the "LoadTask" routine as follows:

```
push ax         ; fh of the open file (will be placed
                ; in the request block at offset 12)

push bx         ; priority (will be placed in the
                ; request block at offset 14)

xor  ax,ax

push ax         ; fDebug (will be placed at offset
                ; 16)

call LoadTask   ; load the task

cmp  ax, 0      ; see if any error

jne  Error
```

The routine "LoadTask" is declared in "CTOS.lib" ("rqLabl" module), and is an entry into the procedural interface, with the only information being passed is the request code for "LoadTask", or 29. From the request number the procedural interface indexes into the "RcLookUpSys(Usr)" table generated by the invocation of the macro:

```
%SysRequest(29, "LoadTask", exchTask, 0A00h,
   6, 0, 0, %( %fh, %w(14) %w(16) ))
```

The value in the table is an "offset" pointer to an (unlabeled) table describing how many arguments to expect on the stack, and the location in the request block that each is to be copied. Using this information, the interface takes arguments off of the stack, builds a request block (on the user's stack), and makes a "request" in behalf of the user. The procedural interface routine then waits at the user's default response exchange for the CTOS service to "respond". When the service routine is finished and does a "respond" with the request block, the Procedural Interface routine gets a message which is a pointer to the original request block, and the routine then takes out the "ercRet" (offset 8 in the request block), cleans up the user's stack, and returns to the user with "ercRet" in 8086 register AX.

CTOS Request Dispatcher

The Request Dispatcher uses the Rq field of the request block to get the information it needs to process the request. It will send a message pointing to the request block to the appropriate exchange as defined in the "SysRequest" or "UsrRequest" macro. The dispatcher uses "rgRqExchgSys(Usr)" to find out where to the request goes. The remaining tables are available as a convienence to the service process in deciding what to do with a request. Convergent CTOS processes handle many requests each, and to make it easy to distinguish between then the local service code is used. The local service code for a user process is defined in whatever manner the process wishes. In particular, "RgLocalServiceCodeUsr" may be accessed in an Assembly Language program by declaring it as follows:

```
RqSeg4 segment public 'const'
extrn RgLocalServiceCodeUsr: word
RqSeg4 ends
```

To index into a user request table, take the request number from the request block, complement it, double it if you are indexing into a word table, and index into the array:

```
mov  si, word ptr requestBlock+10  ; get rq code
not  si                            ; negate code
shl  si, 1                         ; double index
mov  ax, rgLocalServiceCodeUsr[si] ; get local code
```

Any errors are reported to the caller by placing the error code in the "ercRet" field of the request block. When the service is complete, the caller may be requeued on the ready queue by issueing a "respond" to the response exchange in the request block. The calling process will be placed on the ready queue, and when it is run the dispatcher will continue from the "wait" issued earlier, remove the error code from the request block, clean up the user stack, and return with the error code in register ax.

Program Termination

If a user process needs to know when an application program has been terminated, the macro "%TerminationRequest" in "sysgen.asm" may be used.

When an application has been terminated (with an "ErrorExit" or "Chain") a request will be generated for each number given. The requests will have one argument, the termination code. Also, any CTOS process may terminate the currently running applicatio program by doing a call to ErrorExit with the termination code as an argument. Note that since the termination process has lower priority than other system processes, that this call will return (unlike the same call issued by an application program).

The module "RqLabl.asm" is used for defining the names of the various CTOS routines. The twenty bit addressing of the 8086 processor is taken advantage of to construct various unique combinations of segment address and offsets which all enter CTOS at the same place. The formula used is:

```
cs = -(rq * 0FFF8h)
ip = 210h + ((rq * 2) AND 0Fh) + (80h * (rq / 8))
```

From this address CTOS determines the request number of the routine using the inverse formula:

```
rq = ((ip AND 0Fh) / 2) + NEG(cs)
```

These formulas are included in macro definitions in "RqLabl.asm", which are invoked as:

```
%RqName(29, "LoadTask")
```

If this module is changed the new object module resulting should be added to the libraries "CTOS.lib" and "OS.lib" using the Convergent Librarian.

For each of the three modules "sysgen", "request" and "rqLabl" there is a macro definition file which contains the defintions of macros used in the module. You should NOT make changes to these files. They are:

```
Sysgen.mdf
Request.mdf
RqLabl.mdf
```

Removing Optional CTOS Services

There are several standard services which are included in CTOS that may be removed at the user's discretion.

In general, removing a CTOS service requires two steps, removing references in "sysgen.asm" of that service, and replacing the object module(s) which implement the service with (Convergent supplied) dummy modules. The optional CTOS services are described below:

Debugger

The Convergent debugger is optional. To remove the debugger from CTOS, remove the debugger initialization routine declaration from "sysgen.asm" ("%InitProcIntEnable(InitDebugNub)"), assemble the new (copy of) "sysgen.asm", and link CTOS with the following modules replaced or deleted in the link files (i.e. objLinkSwp", "objLinkRes", etc.):

replace the first occurence of DBG.LIB(...) with "DBG.LIB(dbgDum)"
delete the second occurence of DBG.LIB(...) from the list

Hard Disk

If your configuration does not include a hard disk, you may wish to remove the hard disk handlers in CTOS. This involves reducing the number of "vhb"'s to be the number of floppy disk devices, remove the hard disk initialization routine "CheckDisks" (Delete "%InitProcIntDisable(CheckDisks)" in "sysgen.asm"), remove the hard disk device declarations, and to link the new CTOS with the following modules replaced or deleted in the link files:

replace "hDisk" with "hDkdum"
delete "fsIn3" from list

Programmable Interval Timer

If you do not require the timer (all Convergent Comm programs require this timer) you may remove the handler from CTOS. Simply link CTOS with "timdum" instead of "timer" in "objLinkSwp" or "objLinkRes".

Building A CTOS System Image

In order to build a new CTOS, you must first assemble Sysgen.Asm with its prefix file, and then link all of the CTOS object modules together. The CTOS release disks contains the libraries which have all of the

required object modules, as well as the source files for Sysgen, Request, and RqLabl. In addition, you will find a submit files that links CTOS (link.sub), and one that assembles the Assembly Language programs (assemble.sub). Also, there are files (used by the link.sub submit file) that contain the list of object modules required for the CTOS standard versions.

If the only changes you have made are to the parameters in "sysgen.asm", then you will not need to modify any other files. Simply submit "Assemble.sub" with arguments "Sysgen Res" (or "Swp" or "Mws", etc.), and (if there are no errors) then submit "link.sub" with the appropriate argument ("Res" for a resident CTOS, "Swp" for a swapping CTOS, "Mws" for a mini-cluster master, etc.). If you have made changes involving CTOS services, then you may have to modify the files containing the list of object modules for the CTOS versions you wish to build (ObjLinkSwp, ObjLinkRes, etc.). The result of the Linker may then be copied into [sys]<sys>SysImage.sys to complete the building of a custom CTOS.

EXAMPLE -- Configure the a CommIOP-Cluster System for 1 CommIOP and 3 cluster workstations per channel. This is done by changing Sysgen.Asm (a Copy of the original) as follows (the meaning of the %Set macros and the "rgIopChan" statement is described in comments in the file Sysgen.Asm) for the parameters in the condition WsType 3:

```
Change  %Set(nIop, 2)      to  %Set (nIop, 1)
        %Set(nWsLine1,4)       %Set(nWsLine1,3)
        %Set(nWsLine2,4)       %Set(nWsLine1,3)
        %Set(nWsLine3,4)       %Set(nWsLine1,Ø)
        %Set(nWsLine4,4)       %Set(nWsLine1,Ø)
```

This allows 7 users (6 cluster workstations plus the master). Next, assemble Sysgen by typing into the Executive command form "Submit" and filling out the fields as follows:

```
Submit
  File name                 Assemble.Sub
  [Arguments]               Sysgen  Iop
  [Force Expansion?]
  [Show Expansion?]
```

After you press go, the Assembler is invoked with the form filled in to assemble the sysgen modules. When assembly is complete, submit the file "Link.sub" with argument "Iop":

```
Submit
  File name                    Link.Sub
  [Arguments]                  Iop
  [Force Expansion?]
  [Show Expansion?]
```

The Linker form is filled in, leaving the cursor at the version number field. The default is "Usr-Iop" but you may enter anything you wish. If the version has any spaces, it must be surrounded by single quotes. Press GO again, to invoke the Linker program. A new version of CTOS named "CtIop.run" is created, and may be copied into [sys]<sys>Sysimage.sys as described in the Release Notice for Standard Software.

For additional examples of customized operating systems, look at the system build files for CtRes.Run, CtSwp.Run, CtInit.Run, and CtFd.Run (files Sysgen.Asm, Res.Asm, Swp.Asm, Init.Asm, Fd.Asm, ObjLinkRes, ObjLinkSwp, ObjLinkInit, and ObjLinkFd). These IWS standalone operating systems have the following characteristics:

CTOS version:	Res	Swp	Init	Fd
debugger ?	yes	yes	no	no
max number of floppies	2	2	1	2
max number of hard disks	3	3	1	Ø
memory resident ?	yes	no	yes	yes

4. BUILDING A CUSTOMIZED SAM

The Sequential Access Method (SAM) implementation contained in CTOS.Lib may be customized by generating a tailored SAMGEN module. First, the SAMGEN.ASM source file must be edited to reflect the desired device support. After editing, the SAMGEN.ASM file must be assembled and then the resulting object file, SAMGEN.OBJ, must be included in the list of object modules at link time. Alternately, if the new SAMGEN.OBJ is to become the default SAMGEN.OBJ, the Librarian utility may be used to overwrite the SAMGEN.OBJ module contained in CTOS.Lib. The default SAMGEN.OBJ contained in CTOS.Lib is configured to include disk, keyboard, video, parallel printer, null, and spooler byte streams. The default SAMGEN.OBJ excludes communication and serial printer byte streams.

Excluding Byte Streams

A particular byte stream that is not used by an application system may be excluded from the customized SAMGEN.OBJ in order to reduce the applicaton system's memory requirements. To exclude a byte stream, the %DeviceOpen macro and all %tagProc macros (also any %DevDepProc macros in the case of disk or video byte streams) associated with the byte stream to be omitted should be deleted from the SAMGEN.ASM file. For example, to exclude disk and spooler byte streams, the following source lines would be deleted from SAMGEN.ASM:

```
%DeviceOpen([Disk],  OpenByteStreamAD)

%DeviceOpen([Spl],  OpenByteStreamSpl)

%TagProc(tagDiskRead,  FillBufferAD,
          FlushBufIllegal, CheckPointBsAD,
          ReleaseByteStreamAD)

%tagProc(tagDiskWrite, FillBufIllegal,
          FlushBufferAD, CheckPointBsAD,
          ReleaseByteStreamAD)

%tagProc(tagSplWrite, FillBufIllegal,
          FlushBufferAD, CheckpointBsAD,
          ReleaseByteStreamSpl)

%DevDepProc(GetBsLfa, GetBsLfaAsync)

%DevDepProc(SetBsLfa, SetBsLfaAsync)
```

Note that in the above example, two device dependent procedures, GetBsLfa and SetBsLfa, are associated with disk byte streams and must also be deleted. After editing, the SAMGEN.ASM source file must be assembled and the resulting object module included in the list of object modules in the Linker form.

Including a Convergent Byte Stream

A user may also include Convergent supplied byte streams that are not included in the default SAMGEN.OBJ module. Implementations of serial printer and communication byte streams are included in CTOS.Lib but are not specified in the default SAMGEN.OBJ module. To include either of these byte streams in a customized SAMGEN.OBJ, appropriate entries must be made in SAMGEN.ASM. For example, to add both serial printer and communication byte streams, the following entries must be made in SAMGEN.ASM:

```
%DeviceOpen([Comm], OpenByteStreamC)

%DeviceOpen([SLpt], OpenByteStreamSLp)

%tagProc(tagCommRead, FillBufferC, FlushBufIllegal,
          CheckPointBsC, ReleaseByteStreamC)

%tagProc(tagCommWrite, FillBufIllegal,
          FlushBufferC, CheckPointBsC,
          ReleaseByteStreamC)

%tagProc(tagCommModify, FillBufferC, FlushBufferC,
          CheckPointBsC, ReleaseByteStreamC)

%tagProc(tagSLptWrite, FillBufIllegal,
          FlushBufferC, CheckPointBsC,
          ReleaseByteStreamC)
```

After editing, the SAMGEN.ASM source file must be assembled and the resulting object module included in the list of object modules in the Linker form.Adding a User Byte StreamA user may also add his own byte stream to the SAM configuration contained in CTOS.Lib. After coding the five device dependent routines which are needed (OpenProc,FillProc,FlushProc,CheckPointProc, and ReleaseProc), entries for the new byte stream must be added to the SAMGEN.ASM source file. For example, to add a read only byte stream and a write only byte stream for a device Foo, the following entries must be made in SAMGEN.ASM:

```
%DeviceOpen([Foo], OpenByteStreamFoo)
```

```
%tagProc(tagFooRead, FillBufferFoo,
          FlushBufIllegal, CheckPointBsFoo,
          ReleaseByteStreamFoo)

%tagProc(tagFooWrite, FillBufIllegal,
          FlushBufferFoo, CheckPointBsFoo,
          ReleaseByteStreamFoo)
```

After editing, the SAMGEN.ASM source file must be assembled and the resulting object module included in the list of object modules in the Linker form. The object modules for the five device dependent routines may be either added to CTOS.Lib or included in the list of object modules in the Linker form.

Substituting an Alternate Byte Stream

A user may wish to substitute an alternate version of a byte stream for the version contained in CTOS.Lib. This new version may be either a Convergent supplied byte stream or a user written byte stream. For example, to substitute Convergent supplied synchronous disk byte streams for asynchronous disk byte streams, the first group of source lines shown below should be replaced with the second group of source lines:

```
%DeviceOpen([Disk], OpenByteStreamAD)

%tagProc(tagDiskRead, FillBufferAD,
          FlushBufIllegal, CheckPointBsAD,
          ReleaseByteStreamAD)

%tagProc(tagDiskWrite, FillBufIllegal,
          FlushBufferAD, CheckPointBsAD,
          ReleaseByteStreamAD)

%tagProc(tagSplWrite, FillBufIllegal,
          FlushBufferAD, CheckPointBsAD,
          ReleaseByteStreamSpl)

%DevDepProc(GetBsLfa, GetBsLfaAsync)

%DevDepProc(SetBsLfa, SetBsLfaAsync)

%DeviceOpen([Disk], OpenByteStreamSD)

%tagProc(tagDiskRead, FillBufferSD,
          FlushBufIllegal, CheckPointBsSD,
          ReleaseByteStreamSD)

%tagProc(tagDiskWrite, FillBufIllegal,
          FlushBufferSD, CheckPointBsSD,
          ReleaseByteStreamSD)
```

```
%tagProc(tagSplWrite, FillBufIllegal,
         FlushBufferSD, CheckPointBsSD,
         ReleaseByteStreamSpl)

%DevDepProc(GetBsLfa, GetBsLfaSync)

%DevDepProc(SetBsLfa, SetBsLfaSync)
```

Note that in the above example, since spooler byte streams share several routines with disk byte streams, the %tagProc macro for spooler byte streams must also be replaced. After editing, the SAMGEN.ASM source file must be assembled and the resulting object module included in the list of object modules in the Linker form.

Byte Stream Work Area

The Byte Stream Work Area (BSWA) is a 130 byte memory work area for use by the various byte streams. The first 14 bytes of the BSWA are common among all the byte streams and are used by the device independent part of the Sequential Access Method. The other 116 bytes are free to be used by the particular byte streams. The 14 byte common area should be initialized by the OpenProc routine called by OpenByteStream. The format of the common area and the initial values for each field are shown below:

Offset	Field	Size	Initial Value
0	pBuffer	4	0
4	sBuffer	2	buffer size (in bytes)
6	ibRead	2	(Note 1)
8	ibWrite	2	(Note 2)
10	fOkToPutBack	1	TRUE or FALSE
11	fPutBack	1	FALSE
12	bPutBack	1	0
13	tag	1	Appropriate tag value

Notes:

1. ibRead is the read position in the buffer. If the byte stream is write only, then ibRead should be set to 0FFFFH. Otherwise, ibRead should set to 0.

2. ibWrite is the write position in the buffer. If the byte stream is read only, ibWrite should be set to 0FFFFH. Otherwise, ibWrite should set to 0.

3. TRUE = 0FFH FALSE = 0.

Byte Stream Buffer Area

The Byte Stream Buffer Area is a user supplied memory area to be used by the various byte streams to buffer I/O. The pBuffer and sBuffer fields of the BSWA are used by the device dependent part of byte streams to describe a buffer to be used by the device independent part of byte streams. The device independent part of byte streams will transfer user data to/from the buffer described by the pBuffer and sBuffer fields. When this buffer is fully exhausted, a device dependent routine will be called to write/read a new buffer. Examples of three different buffering schemes follow.

1) No Buffering. Some byte streams (such as keyboard and video) do not buffer their I/O and therefore ignore the Byte Stream Buffer Area. These byte streams set the sBuffer field in the BSWA to zero. Consequently, every read or write operation will cause a device dependent routine to be called.

2) Single I/O Buffer. Some byte streams (such as synchronous disk byte streams) use a single I/O buffer. These byte streams use the pBuffer and sBuffer fields in the BSWA to describe this buffer.

3) Pool of Asynchronous I/O Buffers. Some byte streams (such as asynchronous disk byte streams) divide the Byte Stream Buffer Area into a pool of I/O buffers to be used with asynchronous I/O. The pBuffer and sBuffer fields in the BSWA describe a single buffer to be used by the device independent part of byte streams while other buffers from the pool may be involved in asynchronous I/O.

For example, asynchronous disk byte streams divide the Byte Stream Buffer Area into two buffers. For an output byte stream (modeWrite), one of these two buffers may be used for an asynchronous write operation while the other is being filled with user data by the device independent part of byte streams. When the buffer being filled by the device independent part of byte streams becomes full, the device dependent routine, FlushBufferProc, is called to write the full buffer. FlushBufferProc will make sure any asynchronous write operations involving the other buffer have finished and then start an asynchronous write operation with the buffer just filled. Finally, the pBuffer and sBuffer fields of the BSWA are set to point at the empty buffer which was either idle or just finished an asynchronous write operation.

The %DeviceOpen Macro

The macro %DeviceOpen(deviceName,OpenProc) declares a procedure OpenProc which will be called when OpenByteStream is called with a device spec that matches the deviceName declared in %DeviceOpen. OpenByteStream will attempt to match the passed device spec with each of the deviceNames declared with the %DeviceOpen macro. If a match occurs, the appropriate OpenProc proc- ure will be called. If no match occurs, the device spec is not in brackets ([...]), and disk byte streams are included in the SAM configuration, the device spec is assumed to be a filespec and the OpenProc for disk byte streams is called.

If no match occures and the device name is in brackets ([...]), and if the spooler byte streams are included, the device spec is assumed to be a spooler queue name.

If no match occurs and disk byte streams are not included in the SAM configuration, the error code "Not Implemented" (7) is returned from OpenByteStream. The device specification passed to OpenByteStream and the device name declared with the %DeviceOpen macro are compared up to the number of characters in the device name. Thus, the device spec "[Comm]B" used to call OpenByteStream will match the device name "[Comm]" because the first six characters (the length of the device name) match.

The OpenProc procedure called by OpenByteStream is responsible for initializing the common part of the Byte Stream Work Area (BSWA) and assigning a value to the tag byte of the BSWA. This tag value will be used to route the device independent calls (WriteByte, WriteBsRecord, ReadByte, etc.) to the appropriate device dependent routines declared by the %tagProc macro. The procedural interface for the device dependent OpenProc is the same as the device independent OpenByteStream call:

```
OpenProc(pBSWA, pbDevSpec, cbDevSpec, pbPassword,
         cbPassword, mode, pBufferArea,
         sBufferArea) : ErcType
```

The %tagProc Macro

The macro %tagProc(tagName, FillBufferProc, FlushBufferProc, CheckPointBsProc, ReleaseProc) declares four device dependent procedures to be called when the various device independent procedures (ReadByte and ReadBsRecord, Writebyte and WriteBsRecord, CheckPointBs, ReleaseByteStream) are called. The tagName parameter is declared as a PUBLIC BYTE and is given a value. This value is used to route

the device independent calls to the apppropriate device dependent routines. The apprpriate tag should be decalred as an EXTERNAL BYTE within the OpenProc procedure and the tag byte of the BSWA should be assigned this tag value (see above section). Note that both CheckPointProc and then ReleaseProc are called when the device independent routine CloseByteStream is called. The procedural interfaces for FillBufferProc, FlushBufferProc, CheckPointProc, and ReleaseProc are shown below.

FillBufferProc

DESCRIPTION

The FillBufferProc routine is called whenever a read operation is attempted with an empty buffer. The ibRead field of the BSWA is the index pointing to the next byte of the buffer described by the pBuffer and sBuffer fields. An empty buffer condition is detected when ibRead >= sBuffer. The FillBufferProc may either (or both):

1) Fill the buffer and set ibRead to a value less than sBuffer. The user's following read byte stream operations will be serviced from this buffer until it is exhausted. If no additional string is to be passed back to the user (see below) set cbRet to zero.

2) Pass a string of bytes (up to cbMax long) back to the user. The pbRet and cbRet parameters should be set to point at the string. If this string is the only data to be passed back to the user (no buffer), the pBuffer and sBuffer fields in the BSWA should remain unchanged.

Byte streams using buffered I/O should pass a buffer of data back by setting cbRet to zero and updating pBuffer, sBuffer, and ibRead (case 1 above).

Byte streams using unbuffered I/O (e.g. keyboard) should keep ibRead set to ØFFFFH so that FillBufferProc will be called for any read operation. FillBufferProc should then return bytes via cbRet and pbRet (case 2 above).

The procedure FillBufIllegal is a fill buffer routine that returns ercNotImplemented (7) whenever it is called. Byte streams for which read operations are illegal should set ibRead to ØFFFFH and use FillBufIllegal so that FillBufIllegal is called for any read operation.

PROCEDURAL INTERFACE

FillBufferProc (pBSWA,cbMax,pPbRet,pCbRet):ErcType

where

pBSWA — is the memory address of the same Byte Stream Work Area that was supplied to OpenByteStream.

cbMax	is the maximum count of bytes of data that the calling process will accept. pPbRet is the memory address of 4 bytes into which the memory address of the data is returned.
pCbRet	is the memory address of a word into which the actual count of data bytes made available is returned.

FlushBufferProc

DESCRIPTION

The FlushBufferProc routine is called whenever a write operation is attempted with an full buffer. The ibWrite field of the BSWA is the index pointing to the next byte of the buffer described by the pBuffer and sBuffer fields. An full buffer condition is detected when ibWrite >= sBuffer. The FlushBufferProc may either (or both)

1) Write the buffer and set ibWrite to a value less than sBuffer. The user's following write byte stream operations will be serviced with this buffer until it is once again full. If no additional string is to be written (see below) set cbRet to zero.

2) Write a string of bytes described by the pb and cb parameters. The cbRet word should be set to the count of bytes written from this string. If this string is the only data to be written (no buffer), the pBuffer and sBuffer fields in the BSWA should remain unchanged.

Byte streams using buffered I/O should write a buffer of data, set cbRet to zero, and update pBuffer, sBuffer, and ibWrite (case 1 above).

Byte streams using unbuffered I/O (e.g. video) should keep ibWrite set to ØFFFFH so that FlushBufferProc will be called for any write operation. FlushBufferProc should then write the string described by cb and pb and return the number of bytes written in cbRet (case 2 above).

The procedure FlushBufIllegal is a flush buffer routine that returns ercNotImplemented (7) whenever it is called. Byte streams for which write operations are illegal should set ibWrite to ØFFFFH and use FlushBufIllegal so that FlushBufIllegal is called for any write operation.

PROCEDURAL INTERFACE

```
FlushBufferProc (pBSWA,pb,cb,pCbRet):ErcType
```

where

pBSWA — is the memory address of the same Byte Stream Work Area that was supplied to OpenByteStream.

pb	is the memory address of the data to be written.
cb	is the count of bytes to write.
pCbRet	is the memory address of the word into which the count of data bytes successfully written is returned.

CheckPointProc

DESCRIPTION

The CheckPointProc procedure checkpoints the open output byte stream identified by the memory address of the Byte Stream Work Area. CheckPointProc writes any partially full buffers and waits for all write operations to complete successfully before return- ing. fIsPartOfClose is a flag that is set to TRUE is this call to CheckPointProc is part of a device independent call to CloseByteStream.

PROCEDURAL INTERFACE

CheckPointProc (pBSWA,fIsPartOfClose):ErcType

where

pBSWA — is the memory address of the same Byte Stream Work Area that was supplied to OpenByteStream.

fIsPartOfClose — is set to TRUE if this call to CheckPointProc is part of a call to the device independent procedure CloseByteStream.

ReleaseProc

DESCRIPTION

The ReleaseProc procedure closes the device/file associated with the open output byte stream identified by the memory address of the Byte Stream Work Area. ReleaseProc does not properly write remaining parially full buffers before closing the device/file.

PROCEDURAL INTERFACE

ReleaseProc (pBSWA):ErcType

where

pBSWA — is the memory address of the same Byte Stream Work Area that was supplied to OpenByteStream.

The %DevDepProc Macro

The macro %DevDepProc(devDepCall,devDepProc) declares a routine devDepProc to be called when the device dependent routine devDepCall is called. This macros allows for substitution of alternate routines to handle the any device dependent byte stream routines defined (such as QueryVidBs, GetBsLfa, SetBsLfa, or SetImageMode). The user may also omit any device dependent routines by deleting the appropriate %DevDepProc macros.

Supplied Routines

Several routines contained in CTOS.Lib may be useful to a user who is writing his own byte stream. FlushBufIllegal and FillBufIllegal both return the error code "Not Implemented" (7) when called and can be used with read only and write only byte streams. ChkptNop returns the status code "OK" (Ø) when called and can be used for which no CheckPoint actions are neccessary (e.g. read only byte streams. ReleaseEasy returns status code "OK" (Ø) when called and can be used with devices that require no disconnect logic.

Error Checking

A certain amount of error checking is recommended for all byte streams. Each of the device dependent routines FillBufferProc, FlushBufferProc, CheckPointProc, and ReleaseProc should check the tag byte in the passed BSWA to make sure it is appropriate for the device dependent routine (i.e. ReleaseByteStreamD would check that the BSWA tag byte was either tagDiskRead or tagDiskWrite). If the BSWA tag is not appropriate, the device dependent routine should return the error code "Invalid BSWA" (2325). The device dependent routine OpenProc should check that the mode parameter is appropriate to the byte stream (i.e. OpenByteStreamK would check that mode was equal to modeRead). If the mode is not appropriate, the device dependent routine OpenProc should return the error code "Invalid Mode" (2315).

5. SYSTEM PROGRAMMING

Communications (SIO) Programming

There are several idiosyncrasies of the Serial Input/Output (SIO) and Multi-Protocol Serial Controller (MPSC) ICs which must be considered when writing a Communications Handler for the Convergent System. These are especially important in an IWS Cluster configuration, in which a programming error in one workstation might conceivably bring down the entire cluster. While there is considerably less danger in an AWS configuration, where separate Controller ICs are used for Cluster and RS-232 communications, it is still good programming practice to adhere to the following restrictions, so that the Handler may be used throughout the Convergent Family.

Since the SIO and MPSC controllers contain 2 "channels" which support 2 independent communication lines, extreme care is necessary, when programming one channel, to ensure that commmunications which may be going on in the other channel are not disturbed.

Since all Convergent communications software uses the "Status Affects Vector" mode of interrupt handling, all user-written software which is to coexits with any Convergent handlers must also use this mode, as supported by the CTOS Communications Interrupt Handler.

This implies that any user-written handlers which change Write Register 1 of Channel B must always set the "Status Affects Vector" bit (bit 2); since the Channel Reset operation leaves all internal SIO/MPSC registers in an undefined state, this register in particular must be re-written immediately following a Channel Reset. Interrupts must be disabled during the interval surrounding a Channel Reset and the re-writing of register 2. In general, it is recommended that interrupts be disabled during any code sequence which issues I/O instructions to the SIO or MPSC controller, or, for that matter, any I/O device.

Another programming restriction, that pertains whenever Communications DMA is in progress, is that any I/O instruction to the SIO Controller must be made using the LockIn and LockOut routines supplied by Convergent in CTOS.Lib. These routines use a combination of LOCK prefixes, Segment over-ride prefixes, and JMP instructions to ensure that the I/O is done correctly. This restriction is critical in that failure to obey it can have unpredictable results on both the I/O operation being attempted, and the DMA operation in progress on the other Channel. In a cluster

environment, this will almost certainly result in a system crash at the workstation in question, and possibly severe performance degradation throughout the cluster. While this restriction is not necessary on an AWS workstation, it is recommended that all general-purpose software adhere to it, so as to work throughout both the AWS and IWS families. The LockIn and LockOut procedures are described in the "Communications Management" section of the CTOS Operating System Manual.

Debugging hints for system implementors

A majority of of the subtle debugging problems you will encounter will be caused by one of these mistakes:

Providing insufficient stack space - and thus overwriting data or code. You need to provide enough stack for the parameters, local variables and return addresses of all subroutines which might be active at any one time, plus 64 bytes for CTOS to build requests blocks if you use any CTOS services, plus 64 bytes for CTOS to save your process state when an interrupt occurs in the system.

Removing more or fewer items from the stack upon procedure exit than you pushed before calling the procedure.

Modifying the contents of a Request Block while it is still being operated upon by an operating system service.

Attempting to utilize the default response exchange for requests of an asynchronous nature. The use of any procedural interface to an operating system service, or the use of any object module procedure (such as the sequential access method), REQUIRES that the default response exchange be dedicated to use in a purely synchronous manner (Each Request that specifies the default response exchange must be followed by a Wait that specifies the same exchange before another Request may specify it.)

Whenever you find that the software you are debugging is failing in a way which appears to be random or to resemble hardware or operating system failure, we suggest that you carefully desk check your code for the mistakes listed above.

Notes on the Debugger

The Debugger requires two files to be present on the [Sys] volume, the "Debugger File" and the "Debugger Swap File."

If the Debugger File is not found during CTOS initialization, the Debugger will not be activated. If it is found, it will be opened and left open. (If the Debugger is activated, it may be deactivated and the Debugger File closed using the Debugger's CODE-K command; see the Debugger Manual for details.)

If the Debugger Swap File is not found during CTOS initialization, it will be created.

At IWS workstations the Debugger File is [Sys]<Sys>Debugger.Sys. At AWS workstations the Debugger File is [Sys]<Sys>DebuggerAws.Sys.

At IWS and AWS workstations which were booted locally, i.e. at master, stand-alone, or locally booted local file system workstations, the Debugger Swap File is [Sys]<Sys>DebuggerSwap.Sys.

At IWS and AWS cluster and local file system workstations which were booted from the master, the Debugger Swap File is [Sys]<$nnn>DebuggerSwap.Sys, where "nnn" is the user number assigned to the workstation (refer to "Cluster Architecture" section of this manual).

The Operating System reads the Debugger from the Debugger File into the common pool of unallocated memory just above the current limit of long-lived storage. If there is not enough unallocated memory, the Operating System makes room by saving memory on the Debugger Swap File. Then, when the Debugger yields control, the saved memory is restored. Therefore, if you are using the Debugger, you cannot count on the contents of unallocated memory. In addition, beware that if you are debugging a program that contains an interrupt handler, that interrupt handler will not be aware that the Debugger has been swapped in and hence malfunction. For example, if you have issued the request OpenRTClock, and if the storage of the request block is overlaid by the swapped-in Debugger, the program will fail. Similarly, a pseudo-interrupt handler used in conjunction with the Programmable Interval Timer will continue running after the Debugger has been swapped in and therefore function improperly. Beware of using the Swapping Debugger in such circumstances.

If the Debugger is not activated, ACTION-A and ACTION-B will sound the audio alarm but have no other effect. In addition, if an application system generates an Interrupt 3 (in order to enter the Debugger), the application system will Error Exit with the status code "Debugger crash" (1005). (See the Debugger Manual for details.)

Notes on System Initialization

After CTOS has finished its initialization, it will automatically chain to a user-specified run file. This file is specified as the "Chain File" parameter in the system build configuration file (SysGen.Asm), and defaults to [sys]<sys>SysInit.Run if not specified.

SysInit is used to run a Batch stream (see the Batch Manual) after a workstation has booted but before the user signs on. This batch stream can be used to initialize system services, e.g. the Queue Manager, the Spooler, or ISAM.

On a workstation which booted from a local disk, the file [Sys]<Sys>SysInit.Jcl is used as a batch stream. If this file does not exist, then SysInit exits immediatly.

On a workstation which booted from communication lines, the file [Sys]<Sys>WSxxx>SysInit.Jcl is used as a batch stream, where "xxx" is the workstation type (see Section 6, "Mini-Cluster Architecture", below). If this file does not exist, then the file [Sys]<Sys>WS>SysInit.Jcl is used. If neither file exists, then SysInit exits immediatly.

If there is a batch stream to be run, SysInit will initialize the video and, if applicable, display the most recent crash status. Furthermore, it will display a text file if so desired. SysInit then chains to [Sys]<Sys>Batch.Run to run the batch stream. Detailed discussions of these functions follow:

1) Video. If the workstation is an IWS, the font file [sys]<sys>Sys.Font is loaded and the video is initialized with 8Ø columns and 34 lines. Otherwise the workstation is an AWS; the video is initialized with 8Ø columns and 28 lines. Errors which occur during this phase of the initialization cause the system to crash.

2) Crash Status. If the system was rebooted following a crash, SysInit will display the crash status on the video display.

3) Text File. SysInit will look for a text file with the name [sys]<sys>SysInit.Txt. If it exists, it will be displayed on the video display. Note that this is a convenient mechanism by which messages can be broadcast to several workstations. As an example, consider a configuration in which all cluster workstations boot from communication lines. Whenever these workstations boot, any messages contained in SysInit.Txt will be displayed.

4) Batch Stream. SysInit then chains to [Sys]<Sys>Batch.Run to run the batch stream for the workstation. If the chain operation fails (e.g. because [Sys]<Sys>Batch.Run does not exist), SysInit error exits with an appropriate error status.

When the batch stream is completed, Batch exits.

Example 1:

A typical SysInit batch stream file for a master workstation installs system services such as the Queue Manager and the Spooler. To do this, you would create a file <sys>SysInit.Jcl on the [SYS] volume of the master workstation. The following batch stream would install the Queue Manager and the Spooler using default parameters.

```
$Job SysInit
$Run [sys]<sys>InstallQmgr.Run
$Run [sys]<sys>InstallSpl.Run
$End
```

(For a discussion of how to pass parameters to the subsystems, see the Batch Manual.)

Example 2:

Suppose a cluster has several IWS workstations without local file systems, two of which are used to run the Spooler. The two special workstations are set up as type 8 workstations (by setting an internal switch - see Section 6 below). The following batch stream is stored in [Sys]<Sys>WS00 8>SysInit.Jcl on the master, and is run by the special workstations whenever they boot. It installs the Spooler (with default parameters).

```
$Job SysInit8
$Run [sys]<sys>InstallSpl.Run
$End
```

Notes on System Signon

During the initialization of CTOS, a user-specified file is set up as the initial Exit Run File. This file is specified as the "Exit File" parameter in the system build configuration file (SysGen.Asm), and defaults to [sys]<sys>Signon.Run if not specified.

Signon is used to check attempted access to the system by users when they begin operations, and to specify the system configuration each user uses.

When the workstation is booted using an Operating System with default "Chain File" and "Exit File" parameters, CTOS initializes then chains to SysInit with Signon as the Exit Run File. (If SysInit is not present, then CTOS chains to Signon directly.) Signon either runs a Batch stream then exits to Signon, or exits to Signon directly.

Signon re-initializes the video, displays a crash status if applicable, types out an optional text file and finally displays a form into which the user enters a name and password. The video, crash status and text file operations of Signon are almost identical to those discussed in the previous section for SysInit.Run; the only difference is that the text file [Sys]<sys>Signon.Txt is displayed on the video instead of [Sys]<sys>SysInit.Txt.

A system administrator grants access to the system for a user creating the file [sys]<sys>xxxx.User where "xxxx" is the user's name. If he wishes to force the user to use a password, he password-protects the file (see the Set Protection command described in the Executive Manual). After the user enters his name and password and presses GO, Signon attempts to open this file for reading. If the file is not there or if an improper password is supplied, Signon will return an error message and request a valid user name and password. Otherwise, the file is parsed for a series of arguments which characterize that particular user. The format for these arguments is ":KeyWord:Data" and are as follows:

A) Path (required). Three fields must be present in all user files --- ":SignonVolume:", ":SignonDirectory:" and ":SignonFilePrefix:". Signon uses these arguments to do a SetPath request on behalf of the user immediately after Login.

B) Password (optional). The argument supplied to ":SignonPassword:" is used as the password

for all subsequent file operations. If not supplied, the password entered by the user is used.

C) Exit File (required). Whenever this particular user exits from an application subsystem (either normally or abnormally), CTOS will automatically load and execute the run file specified as the ":SignonExitFile" argument.

D) Text File (optional). If the entry for ":SignonTextFile:" exists, that file is displayed on the video screen immediately after the user signs on.

E) Chain File (optional). If the entry for ":SignonTextFile:" exists, Signon will chain to it after completing. If this argument does not exist, Signon will exit to the argument supplied to ":SignonExitFile:". If an error occurs, an appropriate mesage is displayed and the user must attempt to signon again. Parameters are passed to the chain file in the same manner as they are in the Executive parameter interface.

F) Executive Command File (optional). If the entry ":ExecCmdFile:" exists, Signon stores it in the Application System Control Block (See the CTOS Operating System Manual). The Executive uses this field to determine which file to read commands from. (See "Notes on the Executive," below.)

In order for the password mechanism to work, the user configuration file must be read-access protected, and both the [Sys]<Sys> directory and the Sys volume must have passwords. If the user does not supply a valid file, directory, or volume password, the attempt to open his .User file will fail, and Signon will display an error message and force the user to try again.

Example 1:

Joseph uses the Executive to issue commands and invoke subsystems. His user configuration file ([sys]<sys>Joseph.User) is as follows:

```
:SignonExitFile:[sys]<sys>CommExec.Run
:SignonVolume:Win
```

```
:SignonDirectory:Joseph
:SignonFilePrefix:
:SignonPassword:
:SignonTextFile:[sys]<sys>System.Txt
```

This file exhibits several characteristics of interest. First, as desired, any termination from a subsystem causes an exit to the run file [sys]<sys>CommExec.Run (the Executive). Since there is no Chain file argument, the Executive will also be invoked as soon as Signon finishes. The path will be set to [Win]<Joseph> and the text file [sys]<sys>System.Txt will be displayed using video bytestreams. The password Joseph supplies in the Signon form will be used for all subsequent file system operations.

Example 2:

Consider an data entry operator. You may wish to limit the capability of this user to running your order entry application program written in BASIC.

```
:SignonExitFile:[sys]<sys>Signon.Run
:SignonChainFile:[sys]<sys>Basic.Run
Basic
[sys]<sys>OrderEntry.Bas
10
:SignonVolume:Sys
:SignonDirectory:Orders
:SignonFilePrefix:
:SignonPassword:
```

As soon as Signon is complete, this user will enter the Basic Interpreter with his path set to [Sys]<Orders>. The sequence of parameters to the Chain File are identical to those appearing in the Executive command form: "Basic" is the command name, "[sys]<sys>OrderEntry.Bas" is the initial program to be loaded and run, and "10" is the maximum number of files that Basic will allow to be open at once. When the BASIC interpreter exits, the user will be returned to Signon rather than the Executive.

Notes on the Executive

The command interpretation functions of the Executive are table driven. The table is contained in a file, which defaults to [sys]<sys>Sys.Cmds, unless otherwise specified in the user configuration file. If the effective command file is malformed or cannot be read, the Executive uses [sys]<sys>Sys.Cmds. If [sys]<sys>Sys.Cmds is unusable, the Executive will not function. You should boot from a floppy disk known to contain a good Sys.Cmds and, while this Sys.Cmds is effective, repair the malformed Sys.Cmds, for example by replacing it with a good version.

The effective command file may be modified using the Executive commands New Command and Remove Command, as described in the Executive Manual. To aid you in configuring this file to suit your specialized requirements, the Release Disks contain the following files: Sys.Cmds, Sys.Cmds-Initial, and Cmds.Sub. Sys.Cmds is the standard Convergent command table -- it defines all the commands listed in the Command Summary of the Executive Manual. Sys.Cmds-Initial is a minimal set of commands, containing Path, Logout, New Command, Copy, Screen Setup, and Submit. If the current command file contains these commands, then the submit file Cmds.Sub can be run to add all of the other commands. You can, in addition, edit a copy of this submit file and submit this edited copy to configure a customized command file.

The effective command file may change during the course of a session. The file will be closed each time you execute a Copy, Print, Rename, Delete, Set Protection, or any command that invokes an application system. When the Executive subsequently reopens the command file, it will again determine the effective file as described above. Note that the automatic volume recognition performed by the Operating System does not extend to automatic command file recognition: if your user configuration file specifies a command file on a volume which is not mounted, so that the effective command file is [sys]<sys>Sys.Cmds, mounting the missing volume will not automatically change the effective command file.

Note that the Executive uses several files in addition to its run file. The first of these, [sys]<sys>Sys.Font, contains the standard Convergent font. It is loaded whenever the Executive is loaded at an IWS and the field fExecFont of the Application System Control Block is Ø. (This field is set to Ø by the LoadFont operation. Thus if an application system loads an alternate font and does not set fExecFont to

255 (ØFFh), the Executive will reload the standard font when the subsystem exits.) The Executive will not function on an IWS unless [sys]<sys>Sys.Font exists. (If this file contains an alternate font, the Executive will use the alternate font instead of the standard font.)

Another file used by the Executive is Copy.run, in directory [sys]<sys>. The Copy command is doubly implemented: both in Copy.run and within the Executive. If Copy.run is not present, the Executive implements its Copy function using the built-in Copy. If Copy.run is present it may or may not be used: the choice is based on whether, with the full Executive in memory, there is enough memory available for file buffers that are large enough to permit an efficient copy.

6. MINI-CLUSTER ARCHITECTURE

Hardware Configuration

A Convergent Mini-Cluster System consists of one master workstation, up to three cluster workstations, up to three interconnect cables, and two terminators. There are no restrictions on the position of the master workstation within the cluster. However, the total length of cable must not exceed 8ØØ feet. The RS-422 communications channel is used for cluster operation, allowing interconnect via inexpensive twisted pair cable, rather than coaxial cable. For IWS masters, Communications channel A must be configured for RS-422 operation; this channel is reserved by the CTOS Operating System for use by the Cluster, and may not be used by applications subsystems. For AWS-24Ø masters, the dedicated RS-422 channel is used, without interfering with the two dedicated RS-232 channels. The standard speed for cluster communications is 3Ø7 Kilobaud.

Software Components

The difference between the Mini-Cluster and Standalone configurations of the Operating System is the presence of an Agent Process. The Mini-Cluster System contains two versions of the Operating System (one for the master workstation, and the other for the cluster workstations). The master workstation version of CTOS contains a process known as the 'SRP Agent', and cluster workstation version of CTOS contains the 'Workstation Agent' process.

The SRP Agent process communicates with the cluster workstations and issues requests on their behalf to the processes on the master workstation. The Workstation Agent process routes requests to the master workstation.

The master workstation is bootstrapped exactly as a standalone. A cluster workstation without local mass storage is loaded from the master workstation. The Cluster Workstation Operating System run file is obtained from a different file for each possible type of cluster workstation. If the file [Sys]<Sys>WSnnn>SysImage.Sys exists, where "nnn" is the workstation type, then it is loaded. Otherwise the file [Sys]<Sys>WS>SysImage.Sys is loaded. The workstation type is determined as follows:

Workstation	Workstation Type
IWS	ØØØ (Switch settable - see below)
AWS-21Ø	255
AWS-22Ø	254
AWS-23Ø	254
AWS-24Ø	253

The type of an IWS workstation can be set by changing the setting of an internal switch. Bits 5 - 8 of switch S2 on the I/O-Memory board encode the workstation type, with bit 5 the most significant and "On" denoting 1. For example if switch 5 is On and switches 6, 7, and 8 are all Off, the workstation type is 1ØØØ binary (8 decimal).

Cluster Workstations with local mass storage may be bootstrapped exactly as a standalone, if a diskette containing a valid system image has been inserted in drive Ø, or one is present on the hard disk, if any.

A Crash Dump capability is also included for cluster workstations. If a file of the form [Sys]<Sys>WSnnn>CrashDump.Sys is present, a crashdump is performed by workstation nnn after a system crash. Additional files are also required if it is desired to use the debugger from a cluster workstation. This is described in detail in the Debugger Manual.

Communications Protocol

The protocol used for cluster communications is a subset of the American National Standard for Advanced Data Communications Control Procedures, ANSI X3.66-1979 (also known as ADCCP). The UN (Unbalanced, normal response mode, modulo 8) Class of Procedure is used, with Two-Way-Alternate Transmission. Optional functions 4, 5, and 6 (respectively, Unnumbered Information, Initialization, and Unnumbered Polling) are also included within the protocol, and used for Cluster Workstation Boot and CrashDump. This protocol belongs to a group referred to as the 'bit-oriented' protocols, which also include IBM's Synchronous Data Link Control (SDLC) and the International Standard for High-level Data Link Control (HDLC), and is almost totally compatible with the appropriate subset of the latter two protocols.

Theory of Operation

The master workstation assumes the role of the primary station in cluster communications, and as such, controls all operation over the communications link. This is one of the two main functions of the Srp Agent Process, the other being the presentation of user requests to the Master Workstation Operating System and returning their responses to the proper cluster workstation.

The Agent maintains two lists of cluster workstations; those from which it is receiving proper responses to its polling are referred to as **Responsive** workstations; those which have not been responding to polls are referred to as **Unresponsive.**

Every 1ØØ milliseconds, the Agent begins a polling cycle, during which it polls all responsive workstations. If the polling cycle has not expired, an unresponsive workstation is polled. There is at least one unresponsive workstation polled every half second. Responses that are to be returned to cluster workstations have priority over polling messages and are sent as soon as possible.

The mechanism for polling is as follows: an appropriate control frame is transmitted over the link, and the line is then monitored for a reply from the cluster workstation. At the cluster workstation, the internal tables of the CTOS nucleus have been set up to route file system requests (and others to be processed by the master workstation) to the exchange of the Agent Process. Thus, when such a request is made, the Workstation Agent is activated. It notes the occurence of the request in an internal array (rgRcb), and when a poll is received from the master workstation, the request is moved to an internal buffer and transmitted to the Master.

When the request is received by the Master SRP Agent, a series of protocol and higher level checks is performed. Assuming these are successful, the request is moved to one of a group of buffers within the master workstation known as Transmission Buffers. Fields within the request are modified to ensure that the request is returned to the SRP Agent's exchange , and that the response fields of the request are returned to buffers allocated within the Transmission Buffer.

The request is then issued by the SRP Agent. When the response is returned, the SRP Agent, at its first opportunity, adds protocol information to the response and transmits it to the cluster workstation. The

Workstation Agent searches its tables for the matching request, and moves the response data to the requestor's buffers. It then issues a Respond call for the request, which is returned to the requestor.

In general, the error recovery procedures of the ADCCP protocol are capable of recovering from the vast majority of errors which may occur during normal operation. The protocol does, however, depend quite heavily on strict adherance to certain configuration rules. In particular, each workstation within a cluster must be assigned a unique workstation id number when the system is first configured. An attempt to bring up two workstations with the same id will probably result in both crashing. A similar fate will befall an attempt to bring up two master workstations on the same cluster.

Within these constraints, the cluster software is designed to be extremely resilient; the master workstation's communication with one cluster workstation is unaffected by any normal events at any other cluster workstation, such as powering down or pressing the Reset button. Except for the highly unlikely set of failure modes which leave the RS-422 transmitter constantly enabled, it is also similarly unaffected by hardware or software failures at another workstation.

A crash at the master workstation results in the error code "master workstation not running" being returned to any requests from cluster workstations. Thus, a properly written application system may respond in a fail-soft manner, and resume full operation when the master workstation comes back up. Note that any files which had been opened have to be re-opened, since all files are closed when the master workstation restarts operation after failure.

Performance Considerations

The performance of a Convergent Mini-Cluster System is heavily dependent on the following parameters which may be configured at System Build time:

- Number of Sectors per Transmission Buffer
- Number of Transmission Buffers and Request Control Blocks

The first parameter has a significant impact on both the system's performance, and the amount of memory available for application systems at the master

workstation. Since the Workstation Agent splits mass storage requests larger than a single buffer into multiple requests, this parameter limits the maximum size request that can be performed (from a cluster workstation) in a single revolution of the disk. The Convergent-supplied Operating System uses a value of five (5) sectors per transmission buffer. Applications doing significant numbers of large disk transfers and desiring better performance should use a higher value; those doing mostly small disk transfers or needing more memory should use a lower value.

For the cluster workstation, this parameter affects only the size of the DMA buffer and the maximum request size for that particular workstation. (e.g. other cluster workstations can have a CTOS with different Sysgen parameter size for the number of sectors in the Transmission Buffer. The master workstation must have a Transmission Buffer that is at least as large as that of any of the cluster workstations').

The second parameter limits the total number of requests which may be in progress at the master workstation at one time. For applications doing only synchronous I/O, one buffer per workstation is sufficient. For maximum performance in applications doing asynchronous I/O operations, this parameter should be set to the maximum number of outstanding requests times the number of workstations; however, a tradeoff must be performed in order to determine the optimum compromise for the application's memory requirements.

The Convergent-supplied Operating System use a value of six transmission buffers (two per workstation). Formulae for computing the memory requirements for transmission buffers are given in the System Build section. In the cluster workstation this parameter specifies the number of Request Control Blocks.

7. CLUSTER ARCHITECTURE

Hardware Configuration

A Convergent Cluster System consists of:

- o 1 Master Workstation,
- o 1 or 2 CommIOPs,
- o up to 16 Cluster Workstations,
- o up to 16 interconnect cables,
- o up to 4 terminators.

A Communications I/O Processor (CommIOP) is a programmable controller that occupies one Multibus slot. Each CommIOP supports two RS-422 communication channels operating at 307 Kilobaud. The total length of the twisted pair cable attached to each line must not exceed 800 feet. Up to four cluster workstations may be attached to each line.

Both SIO communication channels on the master workstation are available to the user.

Software Components

The difference between the Cluster and Standalone configurations of the Operating System is the presence of an Agent Process. The Cluster System contains two versions of the Operating System (one for the master workstation, and the other for the cluster workstations). The master workstation version of CTOS contains a process known as the 'Srp Agent', and the cluster workstation contains the 'Workstation Agent' process.

The difference between a Cluster and Mini-Cluster configurations is that the CommIOPs handle communication with the cluster workstations for the Cluster configuration. In Mini-Cluster configurations, the master workstation directly handles this communication. In a Cluster configuration, both SIO channels on the master workstation are available to the user. The Operating System for the cluster workstation is identical in both configurations. It is named [Sys]<Sys>WS>SysImage.Sys.

The Master Workstation Operating System is bootstrapped exactly as a standalone except for the fact that the CommIOPs are also loaded at initialization time. The CommIOP code file is [sys]<sys>CommIOP>sysImage.sys. A memory dump of the CommIOP can optionally be performed at initialization time to a file named [sys]<sys>CommIOPn>sysimage.sys where "n" is the CommIOP number.

Communications Protocol

The protocol used for cluster communications is a subset of the American National Standard for Advanced Data Communications Control Procedures, ANSI X3.66-1979 (also known as ADCCP). The UN (Unbalanced, normal response mode, modulo 8) Class of Procedure is used, with Two-Way-Alternate Transmission. Optional functions 4, 5, and 6 (respectively, Unnumbered Information, Initialization, and Unnumbered Polling) are also included within the protocol, and used for Cluster Workstation Boot and CrashDump. This protocol belongs to a group referred to as the 'bit-oriented' protocols, which also include IBM's Synchronous Data Link Control (SDLC) and the International Standard for High-level Data Link Control (HDLC), and is almost totally compatible with the appropriate subset of the latter two protocols.

Theory of Operation

In a Mini-Cluster configuration, the Srp Agent process controls communications with the cluster workstations in addition to submitting requests on behalf of the cluster workstation to the master workstation processes. In a Cluster configuration, the CommIOP controls communications with the cluster workstations.

The CommIOPs and the master workstation communicate by means of a set of queues located in the master's memory. These queues contain addresses of buffers which are used by the CommIOP to copy requests received by the cluster workstation.

The CommIOP contains its own indepedent processor. It handles communications with the cluster workstations and transfers incoming requests and outgoing responses between the CommIOP's internal buffers and the master workstation's memory. Each CommIOP has two channels or lines.

The CommIOP maintains two lists of cluster workstations for each of its lines: those from which it is receiving responses to polling are referred to as **Responsive** workstations, those which have not been responding to polls are referred to as **Unresponsive** workstations.

Every 5Ø milliseconds, the CommIOP begins a polling cycle for one of its lines (therefore, each line has a polling cycle every 1ØØ milliseconds). During the polling cycle, the CommIOP polls all responsive workstations. If the polling cycle has not expired, an unresponsive workstation is polled. There is at least one responsive workstation polled every half second.

The remaining time in the cycle is used for polling any cluster workstations which provided requests or were given responses during the initial polls. Responses that are to be returned to cluster workstations have priority over polling messages and are sent as soon as possible.

The master workstation's Agent process contains a module called the **IOP Handler.** This handler takes incoming requests received from the CommIOP and issues them on behalf of the cluster workstations. Responses are placed on the CommIOP's outgoing queue to be subsequently transmitted to the cluster workstation.

With the addition of the CommIOP, a cluster has more memory available for user functions on the master workstation than a comparable mini-cluster configuration since it does not have a DMA buffer, has fewer transmission buffers, and has less code.

Performance Considerations

The performance of a Convergent Cluster System is heavily dependent on the following parameters which may be configured at System Build time:

- Number of sectors per Transmission Buffer
- Number of Transmission Buffers and Request Control Blocks
- Cluster Line Balancing

The first parameter (Number of sectors per Transmission Buffer) has a significant impact on both the system's performance, and the amount of memory available for applications systems at the master workstation. Since the Workstation Agent splits mass storage requests larger than a single transmission buffer into multiple requests, this parameter limits the maximum size request that can be performed on a cluster workstation in a single revolution of the disk.

The Convergent-supplied Operating System uses a value of five (5) sectors per transmission buffer. Applications doing significant numbers of large disk transfers and desiring better performance should use a higher value; those doing mostly small disk transfers or needing more memory should use a lower value.

For the cluster workstation, this parameter affects only the size of the DMA buffer (twice the size of the transmission buffer) and the maximum request size for that particular workstation. (e.g. other cluster workstations could have a CTOS with different Sysgen parameter size for the number of sectors in the

Transmission Buffer. The master workstation must have a Transmission Buffer that is at least as large as that of any of the Cluster Workstations).

The second parameter limits the total number of requests which may be in progress at the master workstation at one time. For applications doing only synchronous I/O, one buffer per workstation is sufficient. For maximum performance in applications doing asynchronous I/O operations, this parameter should be set to the maximum number of outstanding requests times the number of workstations; however, a tradeoff must be performed in order to determine the optimum compromise for the application's memory requirements.

The number of cluster workstations for each communications line is configurable. The distribution among the lines can determine performance at the cluster workstation. For example, if one line has one cluster workstation attached, and the other has 3, the performance of the single cluster workstation is significantly better than the other three.

It is not necessary for each line to be filled (i.e. a line may be configured for 4 workstations and have only 1 cluster workstation attached to it). However, for each cluster workstation configured, system control structures are allocated in the master workstation, which imposes a memory penalty.

8. IWS DIAGNOSTICS

There are three kinds of IWS diagnostics: Bootstrap ROM, CTOS initialization, and stand-alone.

The function of the diagnostic in the Boot ROM is to ensure that enough of the Information Processing System (IPS) is functional to permit the Boot ROM to load the OS or a diagnostic from disk or from Comm (RS-422 Communications line). The Boot ROM checksums the ROM, tests the first 128kb of memory, and attempts to read from the floppy, Winchester, or Comm using DMA but not interrupts. If an error is detected, it displays a definitive error code in the six LED's of the Memory-I/O board and beeps the audio output. The error code is as precise as possible and differentiates between "missing address mark in data field" and "data error in data field" for example.

By design, the Boot ROM does NOT test any functions which are not required to load a program from disk/floppy/comm. This is to keep the code in the Boot ROM as small as possible. It does NOT test I/O interrupts, non-maskable interrupts (parity error, etc.), memory above 128kb, video, keyboard, etc. The CTOS initialization code tests all of these functions before loading the Convergent Executive.

CTOS initialization verifies the operation of I/O interrupts, non-maskable interrupts (parity error, etc.), memory above 128kb, video, keyboard (reset, checksum, loop-back), real-time clock, interval timer, SIO (status during and after initialization only), etc. CTOS initialization failure detection is described in the section "CTOS Failure Analysis" of the System Programmer's Guide.

The stand-alone diagnostics are intended to determine whether each major element of an IPS is functional. When possible, they isolate to the board level. Each stand-alone diagnostic is supplied on its own floppy disk in a format suitable for loading by the Bootstrap ROM.

Stand-alone diagnostics may be run by an untrained person using all default parameters to determine whether the function is error-free. They can also be run by trained personel to identify the exact manner of failure.

Summary of Diagnostics

Video

The Video Stand-alone Diagnostic loads the font, cursor, and style RAMs. It then displays a sequence of moving patterns which exercise all modes, characters, and attributes. The video output on the screen provides visual verification of correct operation. If an error is detected, this diagnostic displays an error message on the video screen (if possible) and displays an error code in the internal LED's and keyboard LED's.

Keyboard

The Keyboard Stand-alone Diagnostic presumes that the video is operational. It resets the keyboard microprocessor, checksums the keyboard ROM, performs a loop-back test, and prompts the user to type characters on the keyboard. It continuously displays on the video an indication of all keys currently depressed. Each time a key with an LED is depressed, the state of the LED is toggled. Error messages are displayed on the video.

Floppy

Using default parameters, the Floppy Disk Stand-alone Diagnostic requests a scratch floppy in drive zero and tests seek, format, and single and multi- sector write/read. Entry of specific parameters allows a wide variety of tests to be run on drives Ø, 1 or both.

Winchester

Similar to Floppy.

System (memory, timers, printer)

Using default parameters, the System Diagnostic verifies the operation of I/O interrupts, non-maskable interrupts (parity error, etc.), memory, video, keyboard (reset, checksum, loop-back), real-time clock, and programmable interval timer. Entry of specific parameters allows a wide variety of tests to be run. These include several tests of the line printer.

Communications (SIO)

Using the default parameters, the Communication Stand-alone Diagnostic requires that a loop-back cable be installed and tests both channel A and B in Asynchronous, Character-Synchronous, and Bit-Synchronous (BOP) modes with and without interrupts and with and without DMA. The diagnostic asks whether channel A is switch selected for RS-232c or RS-422 and performs its test accordingly. Entry of specific parameters allows a wide variety of tests to be run.

Communications Input/Output Processor

Using default parameters, the CommIOP Stand-alone Diagnostic loads diagnostic code into the CommIOP memory, requests that a loop-back cable be installed and tests both channel A and B in Asynchronous and Bit-Synchronous (BOP) modes with and without interrupts and with and without DMA. It also tests the interval timer, interrupts to/from the 8Ø85, and the ability of the CommIOP to access IPS memory. Entry of specific parameters allows these tests to be varied.

When and How To Run Diagnostics

Except for the video diagnostic, all stand-alone diagnostics presume that the video subsystem is operational and use the video screen to display error messages. Except for the video and keyboard diagnostics, all stand-alone diagnostics presume that the keyboard is operational and accept input from the keyboard. Because of these dependencies, the preferred sequence of execution is: video first, keyboard second, then others as desired.

Special Keys Used in the Stand-Alone Diagnostics

key:	use:
BACK SPACE	deletes last character typed
DELETE	deletes all characters typed
RETURN	indicates field entered
GO	begins execution
FINISH	terminates the test being executed (Hold the FINISH key depressed until the diagnostic acknowkedges)
ACTION	(hold ACTION and press "A" to enter Debugger; press GO to exit Debugger)

Running Diagnostics from a Cluster Workstation

Bootstrapping from the master

In order to run diagnostic from a cluster workstation which doesn't have any disk drives, it is neccessary to bootstrap the diagnostic from the master workstation. This is done using the Bootstrap command of the Executive. Insert the diagnostic floppy into drive fØ at the master workstation. Next, type in at the cluster workstation the BootStrap command as follows:

```
Command   BootStrap
BootStrap
  File name  [fØ]<Sys>Sysimage.sys
```

The diagnostic will be loaded into memory and will start execution. As soon as this happens, the station is no longer a part of the cluster.

CAUTION

When running the Communication Diagnostic, disconnect the cluster cables from the workstation after the diagnostic is loaded and before any tests are run.

Bootstrapping from a floppy

If you are unable to bootstrap from the master, a last resort would be to disconnect the Mass Storage SubSystem from the master workstation and connect it to the cluster workstation to be tested. You should NOT move the Mass Storage SubSystem, as that might permanently damage the Winchester disk.

When the mass storage subsystem is attached (read the Installation Manual for instructions), power up the Mass Storage Subsystem and the workstation, and insert the diagnostic floppy into floppy drive Ø. You should now be able to bootstrap the diagnostic from the floppy. Note that the cluster workstation is NOT capable of bootstrapping from a Winchester disk.

Video Diagnostic

The video diagnostic tests the operation of the Convergent video subsystem. To run this diagnostic, insert the floppy disk labelled "Video Diagnostic" into floppy drive zero and press the RESET button. After being loaded from the floppy, the diagnostic presents a series of pictures on the screen. Each picture includes a rectangular area in the upper center of the screen that contains descriptive text . If the picture does not match the description, the video hardware is not functioning correctly. If a Convergent advanced-video board is installed, the diagnostic will recognize it and display some additional pictures to test its features. The video diagnostic does not use the keyboard. Thus, no interaction with the diagnostic is necessary or possible.

The status of the screen attributes is displayed prominently at the top of the screen. The screen attributes are:

- Screen width (80/132 columns)
- Character attributes (yes/no)
- Brightness (half/full)
- Background (dark/light, ie. normal or reverse video)

The sequence of pictures is as follows:

Banner, which runs in two phases. The first introduces the diagnostic. The second clears the text frame, identifies the video as either standard or advanced, and tests the font by reading it back from the video board. This is the only picture that does not use character attributes.

Font display, which shows the Convergent font in light-background cells against a dark background.

Cursored string, which fills the screen with text, and displays a cursor on each line except where overlaid by the descriptive text. This runs in 132-column mode with dark background, then again in 80-column mode with light background.

Checkerboard, which tests for pincushion distortion and edge sharpness.

Mosquito Net, which displays a field of one-pixel-thick lines to test for pincushion distortion and line clarity.

Character Attributes, which shows fields of text with all possible character attributes: blinking, half-bright, underlined, and reverse video. The picture is shown twice, once with dark background with screen full-bright, and then in light background with screen half-bright.

Advanced Video Style, which is only run on advanced video hardware. This test loads the style ram and shows the advanced video features: double height and width, superscripts and subscripts, bold, double-underline, offset subscripts, and cursor-attribute.

The sequence of pictures repeats until the Reset button is pushed.

Keyboard Diagnostic

The keyboard diagnostic tests the proper operation of the Convergent keyboard. Insert the floppy disk labelled "Keyboard Diagnostic" into floppy drive Ø and press the RESET button. The diagnostic identifies itself and displays a picture of the keyboard on the screen. You should not press any keys until the test specifically asks you to do so. If there is no Keyboard display, first make sure the brightness control is properly adjusted.

First, the RESET function of the 8Ø48 keyboard microprocessor is tested. This assures that the keyboard is properly connected to the system, and is correctly responding. Proper functioning of this test displays "RESET OK" on the bottom of the screen.

Next, the keyboard ROM checksum is compared to the proper value. The test displays "Checksum ROM OK" and continues to the next test. An error during this test probably indicates improper functioning of the 8Ø48.

The loopback test sends all possible character combinations to the keyboard which are then echoed back. If this test fails, the value sent and the value echoed are displayed on the screen.

The final test requires operator interaction. Press any key on the keyboard and the corresponding key on the screen should be displayed in reverse video. Pressing a key which has an LED associated with it toggles it on / off.

Floppy Diagnostic

Preface

This description of the floppy disk diagnostic is not, in itself, sufficient to enable troubleshooting of a malfunctioning floppy disk controller or drive. Refer to the Peripherals Hardware Manual for information on the Floppy Disk Controller. This information (especially the instruction set of the uPD765 and the format of the main status register and other status registers) is vital to understanding the status information included in error messages.

This description is sufficiently self-contained, however, to permit the reader to perform an extensive battery of tests and to determine whether the floppy disk subsystem is performing correctly or requires remedial attention.

Overview

The objective of the Floppy Disk Diagnostic (FDD) is to exercise all functions of the floppy disk controller and drives and (in the event of malfunction) to identify the failing function and mode of failure as completely as possible.

In addition to testing the floppy disk controller and drives, this diagnostic pre-tests the area of memory which it is going to use as a buffer and is prepared to detect and report non-maskable (type 2) interrupts and all varieties of extraneous interrupts. The floppy disk diagnostic depends on the correct operation of the processor, memory, and the 8237-2 DMA controller. Unless explicitly requested otherwise, it also depends on the 8259A interrupt controller.

The "functions" ("tests" 1Ø - 17) allow you to read/modify the data of a specified sector, repetitively execute format/read/write operations in a manner suitable for troubleshooting with test equipment, and perform other miscellaneous functions.

Operation

When first loaded, FDD announces itself:

```
F L O P P Y   D I A G N O S T I C  x.yy
Insert scratch floppy in each drive to be tested.

Change Parameters (Y/N)?
```

Answer N <RETURN> to run the standard test sequence using drive Ø and default settings for all parameters. Answer Y <RETURN> to select the drive(s) to be tested, the test sequence, or non-standard parameter settings.

If you specify that you want to change parameters, you are asked:

```
Enter each drive # to test followed by <RETURN>
Type <RETURN> when all drive #'s have been entered.
drive # to test:
```

Type in each drive number to test followed by <RETURN>. Type <RETURN> only when all drives to test have been specified. You will hear a beep if you type <RETURN> without listing any drives to test.

You are asked:

```
Run standard test sequence?
```

Answer Y <RETURN> to select the standard test sequence (1, 2, 3, 5, 6, 8, 6, and 8) to be run on the drive(s) which you have selected. The tests are itemized below. If you answer N <RETURN>, the list of available tests and functions are listed on the screen. You are prompted:

```
test # to run:
```

Type in each test number that you wish to run followed by <RETURN>. You will hear a beep if no such test number exists. You can specify up to eight tests to run. When you have specified the test sequence you wish to run, type an additional <RETURN>.

Suggestion

The standard test sequence is the best test of the floppy controller, the electronics of the floppy drive, and the floppy disk. Test 7 is the best test of the floppy disk head load mechanism and pressure pad. Test 9 tests the interrupt when the access door is opened or closed. Function 12 is occasionally useful as a test because the data patterns are more varied.

Test Descriptions

Test 1 - Selection

The selection test verifies the basic functioning of the processor/floppy interface by reading the selected drive's status. If no floppy is inserted in the drive, the message 'Drive not ready' is displayed. If the floppy being tested does not have a write tab, the message 'Write protect set for drive n' is displayed.

Test 2 - Recalibrate

This test checks out the basic control functions to the stepper motor. The test first issues a "recalibrate" command followed by a "seek" command to cylinder 77. Then another recalibrate command is issued, which is aborted by turning off the stepper motor. If this test is successful, the test performs a full recalibrate and checks for proper operation.

Test 3 - Seek Sequential

This test seeks from track Ø to track 77 a track at a time and reports any error conditions.

Test 4 - Seek Random

This test seeks to tracks 38, 37, 39, 36, ... 77, Ø.

Test 5 - Format

The floppy disk in the selected drive is formatted. The floppy disk is then re-read to check if the data compares to the data written on the floppy. This data (called the filler data) is a changeable parameter (see below).

Test 6 - Sequential Write/Read Single Sectors

Each sector on the disk is written one at a time. For each track, sectors 1, 3, 5, ... are first written, and then sectors 2, 4, 6 ... The data value written in sectors 1, 3, 5... is specified by the filler

parameter. Sectors 2, 4, 6 ... are written with the 1's complement of the filler parameter. After each track is written, the data in each sector is verified with the data written.

Test 7 - Random Write/Read Single Sectors

This tests is similar to test 6, except the data is written to tracks 38, 37, 39, 36 ...77, Ø. Test 7 is the best test of the floppy drive head load mechanism. If test 7 shows errors when test 6 does not, the problem is almost certainly in the floppy drive rather than the floppy controller or the floppy disk. Check the pressure pad and the head load mechanism.

Test 8 - Sequential Write/Read Multiple Sectors

This test writes to the floppy one-half track at a time. The first half will be written with the filler word, the second half is written with the 1's complement of the filler. The sectors is then read and compared to the data written.

Test 9 - Ready/Not-Ready Interrupt

This test verifies that the opening/closing of the door causes an interrupt with the proper drive status. The test prompts you to open the door (in order to make the drive not ready). The test expects you to do so within 3Ø seconds. The test then prompts you to close the door to check the "ready" interrupt.

Function 1Ø - Read a Track

This function reads an entire track into a buffer and print it 256 bytes at a time. You are prompted with:

```
Cylinder number:
```

Type in a number from Ø to 77 followed by <RETURN>. The 256 bytes of sector 1 are printed, you are prompted with:

```
More?
```

Type in Y <RETURN> to see the next buffer, etc. Type N <RETURN> to be again prompted for a new cylinder. Type <RETURN> to the cylinder number prompt to terminate the function.

Function 11 - Display/Modify

This function allows you to read and optionally modify a selected sector. You are prompted with:

```
cylinder :
```

Type a decimal number from Ø to 77 followed by <RETURN>. (Type just <RETURN> if you wish to exit this function.) You are prompted with:

```
sector  :
```

Type a decimal number from 1 to the number of sectors on a cylinder (this is 15 for a standard double density floppy disk) followed by <RETURN>. The first 256 bytes of the sector will be displayed. You are prompted with:

```
more?
```

Type Y <RETURN> if you wish to see the next 256 bytes of the sector, type N <RETURN> if you do not. You are then asked if you wish to modify the sector:

```
modify?
```

Type N <RETURN> to inspect another sector or exit the function. Type Y <RETURN> to modify selected bytes in the sector. You are prompted with:

```
byte :
```

Type in a decimal byte number from Ø to 511. The current value of the byte is displayed in hex:

```
n: xx
```

Type in the new hexadecimal value for the byte followed by <RETURN>. Just type <RETURN> if you want do not want to modify the byte. The prompt "byte :" will continue until you type <RETURN> only. You are then be asked:

```
Write sector?
```

Type Y <RETURN> if you wish to write the sector to disk or N <RETURN> if you decide not to write the sector. You are now prompted with "cylinder". Type another cylinder number, or just <RETURN> to terminate the function.

Function 12 - Copy

This function copies the data from the first drive specified in the initial prompt to the second drive. Note that the target floppy disk must have been formatted (using test 5) before function 12 can copy onto it. The data is copied one track at a time. After each track is copied, the data on the target disk is verified.

Function 13 - Read ID

You are prompted with:

cylinder:

Type in the cylinder number that you wish to read (from Ø to 77) followed by <RETURN>. Type just <RETURN> to terminate the function. This function displays the ID of the next sector to pass under the read/write head.

NOTE: Functions 14, 15, and 16 are designed to support troubleshooting using test equiptment.

Function 14 - Format Loop

All the sectors of cylinder Ø are formatted continuously until <FINISH> is pressed. This test runs solely on the first drive selected.

Function 15 - Read Loop

The sectors on cylinder Ø are read continuously until <FINISH> is pressed. This test (like tests 14 and 16) runs solely on the first drive specified.

Function 16 - Write Loop

The sectors on cylinder Ø are written continuously until <FINISH> is pressed. This test (like tests 14 and 15) runs solely on the first drive specified.

Function 17 - Compare

The data written on the first two drives selected is compared against each other.

Function 18 - Set Double Density

This function pre-sets parameters to double density (512 byte sectors). These are:

double density - YES
sector size code - 2
gap length (read/write) - 1Bh
gap length (format) - 54h
data length - ØFFh
bytes per sector - 512
sectors per track - 15

No actual test is performed. This function is useful to include in a loop to interchange single/double density testing.

Function 19 - Set Single Density

This function pre-sets parameters to single density (512 byte sectors). These are:

double density - NO
sector size code - 2
gap length (read/write) - 1Bh
gap length (format) - 3Ah
data length - ØFFh
bytes per sector - 512
sectors per track - 8

Parameter Prompts

After the test sequence to be performed is selected, you are able to vary test parameters. Certain ones should be modified with great care since they can either cause the diagnostic to incorrectly report failure, or to crash. You can leave the default setting of a parameter by entering <RETURN>, or enter a new value followed by <RETURN> to change it. You will

hear a beep if the parameter value you specified is incorrect (e.g. typing in "maybe" to a yes/no question. You are prompted with:

<RETURN> to leave the parameter unchanged,
<GO> to begin tests.

When you are ready to start the test, make sure that scratch disks are placed in the appropriate drive(s). Press <GO>. The tests will begin.

To terminate the test sequence, press and hold <FINISH>. To suspend a test, press and hold the space bar; to resume the test press <GO>. The parameters are:

times to run

The default value is 1. The test sequence which you entered us run for this number of times unless an error occurs or you press <FINISH> to interrupt the test.

output to line printer:

The default is NO. Specify Y <RETURN> if you wish to monitor the test output to the line printer.

page breaks

The default is NO. Specify Y <RETURN> if you wish to be prompted with "Press NEXT PAGE to continue" each time there is a full page of text on the video.

halt on error:

The default is YES. Any data error which is not recoverable after the specified number of retries (see below) will cause the test sequencing to the be aborted. Specify N <RETURN> to continue the test sequence even on hard (non-recoverable) data errors.

suppress error printouts:

The default is NO. Type Y <RETURN> if you do not wish to see the floppy status messages on each error.

use interrupts:

The default is YES. Type N <RETURN> if you wish all floppy interaction to proceed using status checking strategies instead of interrupts.

retry count:

The default is 4. Specify a new decimal number followed by <RETURN> to indicate the number of retries to be performed on any data read/write error.

filler data:

The default is 55 <hex>. This parameter specifies the data to be written in each byte of each sector for any formatting, or writing operation.

buffer address high byte:
second byte:
third byte:
buffer address low byte:

The above four parameters specify the buffer address to be used for all read/write/format operations. This parameter allows testing of DMA transfers to/from high memory addresses. The first 2 bytes specify a segment base address; the last 2 bytes specify an offset address. Do NOT specify an odd address or an address lower than the default.

The remaining parameters are very interdependent and should be changed only by people who are have a thorough knowledge of the disk controller:

double density
single sided
sector size code
gap length (read/write)
gap length (format)
first byte of Specify parameters
second byte of Specify parameters
data length
bytes per sector
sectors per track
tracks per cylinder

```
cylinders per disk
maximum seek time
loop on error
halt while waiting
ignore end of cylinder status
bypass errors
software debug
```

Floppy Duplication

Floppy diskettes may be duplicated on dual floppy disk-based systems using tests 5, 12, and 17. In order to duplicate a diskette, the following procedure may be used.

Place the Floppy Diagnostic diskette in drive Ø and bootstrap from it. When the program is loaded, remove the diagnostic diskette from drive Ø and place the diskette to be copied in drive Ø (remove the write-protect tab first). Place a blank diskette in drive 1.

Format the blank diskette in drive 1 with test 5 by answering the diagnostics prompts as follows (not shown here is other information displayed by the diagnostic before each prompt is given):

```
Change Parameters (Y/N)?    y  <RETURN>
drive # to test:            1  <RETURN>
drive # to test:               <RETURN>
Run standard test sequence?    n <RETURN>
test # to run:              5  <RETURN>
test # to run:                 <RETURN>
times to run:                  <GO>
```

When the format is completed, copy and verify the diskettes by running tests 12 and 17 as follows:

```
Change Parameters (Y/N)?    y  <RETURN>
drive # to test:            Ø  <RETURN>
drive # to test:            1  <RETURN>
Run standard test sequence?    n <RETURN>
test # to run:             12  <RETURN>
test # to run:             17  <RETURN>
test # to run:                 <RETURN>
times to run:                  <GO>
```

Before the copy actually begins, you will be asked to verify that you actually want to do it. If so, then answer yes to the prompt:

```
Copying from drive: Ø to drive: 1
OK?                         y  <RETURN>
```

If you are duplicating a number of diskettes, you may optimize the above two steps by formatting all the blank diskettes first, and then copying all of them. After the first format is done, answer No to the prompt:

```
Change Parameters (Y/N)?    n  <RETURN>
```

The diagnostic remembers the parameters you had previously entered, and will go right into the format test. Similiarly, you only need change parameters for the first time you do the copy/verify tests.

Winchester Diagnostic

CAUTION

The Winchester diagnostic erases all files from the Winchester disk.

Preface

This description of the Winchester disk diagnostic is not, in itself, sufficient to enable troubleshooting of a malfunctioning Winchester disk controller or drive. Refer to the Peripherals Hardware Manual for information on the Winchester Disk Controller. This information (especially the format of the main status register and other status registers) is vital to understanding the status information included in error messages.

This description is sufficiently self-contained, however, to permit the reader to perform an extensive battery of tests and to determine whether the Winchester disk subsystem is performing correctly or requires remedial attention.

Overview

The objective of the Winchester Disk Diagnostic is to exercise all functions of the Winchester disk controller and drives and (in the event of malfunction) to identify the failing function and mode of failure as completely as possible.

In addition to testing the Winchester disk controller and drives, the diagnostic pre-tests the area of memory which it is going to use as a buffer and is prepared to detect and report non-maskable (type 2) interrupts and all varieties of extraneous interrupts. The Winchester disk diagnostic depends on the correct operation of the processor, memory, and the 8237-2 DMA controller. Unless explicitly requested otherwise, it also depends on the 8259A interrupt controller.

Operation

When first loaded, Winchester Disk Diagnostic announces itself:

W I N C H E S T E R D I A G N O S T I C Rev x.y

C A U T I O N

All files on Winchester disk will be erased.
Drive # to test :

Type in the drive number to be tested followed by <RETURN>. The drive number should be Ø, 1, 2, or 3. After the drive number is entered, Winchester Disk Diagnostic will display the parameters associated with the drive, that is, number of cylinders, number of heads, and number of sectors per track. Then you are prompted:

Change Parameters (Y/N)?

If you answer N <RETURN> the standard test sequence will be run on the drive specified. Answer Y <RETURN> to select the test sequence or non-standard parameter settings.You are prompted:

Run standard test sequence?

Answer Y <RETURN> to run the test sequence (1, 2, 3, 4, 5, 6, 7, 8, and 1Ø) on the drive you selected. The tests are itemized below. Answer N <RETURN> to select your own test sequence. You are prompted with:

Display test menu?

Enter Y <RETURN> if you want to see the list of available tests listed on the screen. You are prompted:

test # to run:

Type in each test number that you wish to run followed by <RETURN>. A beep is sounded if no such test number exists. When you have specified the test sequence to run, type <RETURN>.

Test Descriptions

Test 1 - Selection

The selection test asserts basic functioning of the processor / Winchester interface by reading the selected drive's status.

Test 2 - Recalibrate

This test checks out the recalibrate command. The test issues a "recalibrate" command followed by a "seek" command to the last cylinder of the disk.

Test 3 - Seek Sequential

This test seeks from cylinder Ø to the last cylinder a cylinder at a time and reports any error conditions.

Test 4 - Seek Random

This test seeks to cylinders Ø, n-1, 1, n-2, ... n/2, Ø.

Test 5 - Format Disk

The media on the selected drive is formatted. Any error conditions are reported.

Test 6 - Write/Read Single Sectors

This test selects three cylinders to test: the cylinder Ø, the last cylinder, and the cylinder in the middle. Each sector on these three cylinders is written one at a time on each cylinder. The order the sectors within a track is 1, 3, 5, ... 2, 4, 6 ... Random data is used. After each cylinder is written, the data in each sector is verified.

Test 7 - Random Write/Read Multiple Sectors

This test writes to the entire disk a track at a time. Random data is used. After each track is written, the data are verified. The order of cylinders written is Ø, n-1, 1, n-2, ...

Test 8 - Sequential Write/Read Multiple Sectors

This test is the same as Test 7, execept that the order of cylinders written is Ø, 1, 2, 3, 4, 5 ...

Test 9 - Get Drive Parameters

This test issues the 'Drive configuration command' and displays the 6 bytes of drive parameters.

Test 1Ø - Write / Read Beyond Head/Cylinder Boundary

This test is similar to test 8 except that each write/read goes over a track boundary.

Test 11 - Sequential Write / Read Single Sectors

Each sector on the disk is written one at a time on each cylinder. The order the sectors within a track is 1, 3, 5, ... 2, 4, 6 ... Random Data is used. After each cylinder is written, the data in each sector is verified.

Test 12 - Overlapped Seek

This test issues "seek commands" to every drive specified, waits the "maximum seek time", and checks the completion status of all the drives under test. This operation is done for each cylinder on the disk. Even numbered drives (Ø, 2..) start from cylinder Ø, odd numbered drives (1, 3..) start from the innermost cylinder.

Test 13 - Sequential Write/Read single sector, Overlapped Seeks

This tests first selects a drive and seeks to the target cylinder. It then issues seek commands to all other drives specified, and writes a sector on the first drive. After verifying the seek completion status of the other drives, it re-reads the sectors and verifies the data read.

Test 14 - Invalid head & cylinder

This test issues a seek to a cylinder with an illegal head number. It then issues a seek to an invalid cylinder number. The resulting status messages are displayed on the screen.

Test 15 - Display/Modify a Sector

This test allows you to read and optionally modify a selected sector. You are prompted with:

Action(read data- Ø OR <CR>, read ID- 1)? <hex>:

Type in the function you wish to perform.

cylinder? <hex>:

Type in a hexdecimal number from Ø to the number of cylinders on the disk followed by <RETURN>. Type <GO> if you wish to exit this test. You are prompted with:

head? <hex>:

Type Ø to 3 followed by <RETURN>.

sector? <hex>:

Type a hexdecimal number from 1 to the number of sectors on a cylinder followed by <RETURN>. The first 256 bytes of the sector will be displayed. You will be prompted with:

more?

Type Y <RETURN> if you wish to see the next 256 bytes of the sector, type N <RETURN> if you do not. You will then asked if you wish to modify the sector:

modify?

Type N <RETURN> to see another sector. If you type Y <RETURN>, you are again prompted with "cylinder:", "head", "sector". Type in <RETURN> for each if you wish to write the modified contents to the same sector. You can copy the contents to another sector by changing any or all the values. If you type N <RETURN>, you will be prompted once again with 'action(read data- Ø OR <CR>, read ID- 1)'. Type Y <RETURN> if you wish to modify selected bytes in the sector. You will be prompted with:

byte :

Type a decimal byte number from Ø to 511. The current value of the byte will be displayed in hex:

xx

Type the new hexadecimal value for the byte followed by <RETURN>. Type just <RETURN> if you want do not want

to modify the byte. The prompt "byte :" will continue until you type <RETURN> only. You will then be asked:

Write sector?

Type Y <RETURN> if you wish to write the sector to disk; type N <RETURN> otherwise. If Y <RETURN> is typed, you will again be prompted with "Action(read data- Ø OR <CR>, read ID- 1)". Type just GO if you wish to end this test. If N <RETURN> is typed, this test will exit.

Test 16 - Command looping

This test allows you to prepare a loop of individual commands to the controller. This test is intended to be used primarily with special test equipment.

Test 17 - Get Diagnostics

It is not currently supported.

Test 18 - Sequential Write/Read Single Sectors II

This test is similar to test 6 except each word in a sector is written with its own word index number in the sector. This test is useful to detect any cylinder/head addressing problems with the drive.

Parameter Prompts

After the test sequence is selected, you are able to vary certain test parameters. Certain ones should be modified with great care since they can either cause the diagnostic to incorrectly report failure, or cause the diagnostic to crash. You can leave the default setting of a parameter by entering just <RETURN>, or change the value by entering a new value followed by <RETURN>. You will hear a beep if the parameter value you specified is incorrect (e.g. typing in "maybe" to a yes/no question. You now have the opportunity to change various parameters. You are prompted with:

> THE FOLLOWING PARAMETERS ARE AVAILABLE FOR CHANGE
> <RETURN> to leave the parameter unchanged,
> <GO> to begin tests.

Press <GO> when you are ready to start the tests. The parameters are:

times to run:

The default value is 1. The test sequence which you entered will run for this number of times unless an error occurs or you press <FINISH> to interrupt the test.

output to line printer:

The default is NO. Specify Y <RETURN> if you wish to monitor the test output to the line printer.

page breaks:

The default is NO. Specify Y <RETURN> if you wish to be prompted with "Press NEXT PAGE to continue" each time there is a full page of text on the video.

halt on error:

The default is NO. Any data error which is not recoverable after the specified amount of retries (see below) will not cause the test sequencing to be aborted. However the test sequence will stop if some disk controller errors are detected. Specify Y <RETURN>

to stop the test sequence even on hard (non-recoverable) data errors.

suppress error printouts:

The default is NO. Type Y <RETURN> if you do not wish to see the Winchester status messages on each error.

use interrupts:

The default is YES. Type N <RETURN> if you wish for all Winchester interaction to proceed using status checking strategies instead of interrupts.

retry count:

The default is Ø. Specify a new decimal number followed by <RETURN> to indicate the number of retries to be performed on any data read/write error.

filler data -- format parameter:

The default is 39h. This parameter specifies the byte value written in each sector when the disk is formatted (test 5).

loop on error:

The default is NO. This default should be changed only by qualified hardware personnel.

buffer address high byte:
second byte:
third byte:
buffer address low byte:

The above four parameters specify the buffer address to be used for all read/write transactions. This parameter allows testing of DMA transfers to/from high memory addresses. Do not specify an odd address or an address lower than the default.

The remaining parameters are very interdependent and should be changed only by people who are have a thorough knowledge of the disk controller.

offset -- Format parameter
space -- Format parameter
bytes per sector
sectors per track
tracks per cylinder
cylinders per disk
maximum seek time
maximum I/O completion time
perform seek before read/write
halt while waiting
software debug
read diagnostics
change dma word count

Operation

Each test in the sequence will be performed on the drive specified. To abort the testing, press and hold any key on the keyboard while a test is in progress. The test stops and you are prompted with:

Press GO to continue, FINISH to terminate the test.

Press FINISH to restart the diagnostic and to select a new test sequence and/or parameters.

Incomplete Data Transfer

In tests 6, 7, 8, 10, 11, 13, 15, and 18, if a residual dma count is deteced after the data transfer, the message "Incomplete data transfer" is displayed and the test continues. This indicates a transient error which is automatically corrected by the Operating System and may therefore be safely ignored. (The value of the residual byte count can be displayed by setting the "Halt on error" parameter to "YES" and rerunning the test.)

System Diagnostic (memory, timers, printer)

The system diagnostic tests the clocks, memory, and the printer. Insert the floppy disk labelled "System diagnostic" into floppy drive Ø and press the RESET switch. The diagnostic identifies itself and asks for parameters. If you want to run the standard tests (memory, clock test) then answer N to the "Change parameters" question and press <RETURN>. Each test runs and displays any errors on the screen.

The test sequence can be aborted by pressing the <CANCEL> or <FINISH> keys.

Operation

When first loaded, the System diagnostic announces itself:

```
S Y S T E M  D I A G N O S T I C  REV x.y
Change parameters (Y/N)?
```

Type N and <RETURN> if you want to run the standard tests. No further questions are asked and the tests begin to run.

Type Y to select a new test sequence. The diagnostic displays a list of available tests.

```
Run standard tests?
```

Type Y and <RETURN> if you want to run the standard test sequence. You are then prompted with various test parameters which you can selectively change. IF you type N, the program will ask you to type in the test numbers. Type in a test number followed by <RETURN>. When you have entered all the desired tests, press <RETURN> only.

Test Descriptions

Test 1 -- Memory Test

This tests all memory except that which is used by the program. It initially finds the memory bounds by accessing words at 4K increments until a non-existent memory interrupt occurs. It then writes and reads back zeroes in those same locations until ONES are read (indicating non existent memory). If a discrepancy is found, the diagnostic will print out the possible

switch settings, the actual amount of memory available, and the test limits.

The memory test has six distinct passes. The passes are:

- o Write and read Ø's to all test memory.
- o Write and read 1's to all test memory.
- o Write and read address patterns (even addresses)
- o Reread even address patterns from test memory.
- o Write and read address patterns (odd addresses).
- o Reread odd address patterns from test memory.

Errors during the memory test usually come in one of two varieties:

- o Word read does not equal the word written.
- o Parity errors.

If an error occurs during the test, the relevant information will be displayed you will be prompted with "Continue ?" Type Y and <RETURN> if you want to continue the test, or N and <RETURN> if you want to restart the program.

Test 2 -- Memory Test (GALPAT)

This memory test runs an exhaustive test of the memory from 64K. This test takes several hours to run. A '*' will be printed for every 32K which it tests. Parity errors and invalid data will be reported.

Test 3 -- Parity Error Test

This test verifies the proper functioning of the parity generation/checking logic. A byte is written with odd parity. It is then re-read with even parity checking. If no parity interrupt is generated, the user is informed that the test failed.

Test 4 -- RAM Write Protect Test

This test verified the proper functioning of the write protect feature.

Test 5 -- Clock / Programmable Timer Test

This test verifies the proper functioning of the system clocks. It sets the Programmable Interval Timer (PIT) to interrupt after one second, and counts the number of ticks received from the Real Time Clock (RTC). After the PIT has interrupted, the number of ticks counted is compared to the RTC frequency entered. Variations will be reported. The test also reports an error if the PIT does not interrupt within 2 seconds.

Test 6 -- Real Time Clock

This test insures that the real time clock is functioning properly. The test displays COUNTING DOWN from XXX MIN YYY SEC. If the real time clock is functional, the minutes and seconds will count down to zero and the test will terminate.

The test indicates improper real time clock functioning if no interrupts have occurred within a half second. The test indicates whether the problem appears to be an interrupt problem, or the real time clock itself.

Test 7 -- Printer control function and interrupt test

This test requires interaction with the operator and issues form feeds, vertical tabs, line feeds, carriage returns, bells, and other assorted control codes to the printer. This test assures that the system to printer interface is functioning correctly.

Test 8 -- Printing test

This test sends a sliding print pattern to the line printer. If the continuous printing parameter is YES, this test terminates only when the operator presses <FINISH> or <CANCEL> or a printer error is discovered. The operator should assure that the pattern is regular.

Parameter Prompts

The diagnostic displays a selection of parameters which can be changed. These parameters may have no effect on the tests that you have selected. You can press <RETURN> if you do not wish to change a parameter. Press <GO> when you are ready to begin the test. You are prompted:

```
THE FOLLOWING PARAMETERS ARE AVAILABLE FOR CHANGE
<RETURN> to leave current value unchanged,
<GO> when all parameters are satisfactory.
Times to run?
```

The default is 1. Press <RETURN> if you want to run each selected test only once. Otherwise, type the number of times you wish the test to run.

Stop on memory error

The default is YES. This parameter applies to the memory test. Type N and <RETURN> to have the test print the memory error messages but continue with the test.

# of seconds to run real time clock test?

The default is 5. This parameter is applicable only for test 3 (real time clock test).

Frequency of real time clock?

The default is 6Ø Hz. Type in 5Ø for a 5ØHz clock.

Page Breaks?

The default is NO. Specify Y <RETURN> if you wish to be prompted with "Press NEXT PAGE to continue" each time there is a full page of text on the video.

Print status on line printer interrupt?

The default is NO. Typing in YES will cause the line printer status to be displayed every interrupt. This parameter is applicable only for the printer tests.

Continuous printing on slide test?

The default is NO. Specify YES if test 4 is to print continuously until <FINISH> is pressed. If NO is selected, test 5 prints 66 lines to the printer.

Test printer using interrupts?

The default is YES. Type NO if you want to test the printer using BUSY looping. This parameter is applicable only for the printer tests.

Communications Diagnostic

The Communications Diagnostic tests the Serial Input/Output (SIO) communications controller. Both SIO channels (A and B) are exercized using externally installed loopback plugs. Channel A is internally jumpered for RS-232c or RS-422 (cluster) operation. The loopback plug and parameter specification must agree with the internal jumpering. If either channel is jumpered for external clocking, an external clock signal must be provided for that channel.

The default parameters for this test assume that communications channel A is set to RS-422 and channel B is RS-232 (this is the setting set at the factory). If your system has both channels set to RS-232 then you must change the parameters and answer NO to the Parameter Prompt "Channel A RS-422?".

Loopback Requirements

For RS-232 Configurations, each channel must have the following signals looped back:

FROM	TO
Transmit Data (pin 2)	Receive Data (pin 3)
Request to Send (pin 4)	Clear To Send (pin 5)
	Carrier (pin 8)
Data Terminal Ready (pin 2Ø)	Data Set Ready (pin 6)
	Ring Indicator (pin 22)
Secondary Transmit (pin 14)	Secondary Receive Data (pin 16)

For RS-422 Cluster configurations, the required signal loopback for channel A is provided within the system; the workstation should not be connected to an operational cluster during testing. Workstations that are jumpered for RS-422 operation but that do not have the motherboard jumpered to use the cluster (dual 9-pin) connectors, require a 25 pin RS-422 loopback plug to be connected to channel A. Channel B requires a RS-232c loopback plug in all configurations.

Standard test Sequence

The following tests are executed unless otherwise specified: 1, 2, 3, 4, 5, 6, 7, 9, 10, 12, 13. If you specify a different test sequence, do not specify test 6 or test 10 as the first data test.

Parameter Prompts

The following parameters can be changed if desired. You are prompted with:

<RETURN> to leave the parameter unchanged,
<GO> to begin tests.

Times to run:

Number of times to execute sequence of tests: default is 5.

Bypass Errors?

Specify YES to allow test to continue if error is encountered.

Specify YES when running test 11.

Software Debug?

For use by Convergent software engineers.

Channel A RS-422?

Specify NO if Channel A is configured RS-232.

Test Descriptions

Test 1 - Static Status Test

This test verifies that all control signals are properly looped back, that a clock signal is present, that the internal jumpers are consistent, and that the RS-422 clock detector operates properly.

Test 2 - Asynchronous Mode Test

This test exercises both channels of the SIO in Asynchronous mode, at varying baud rates and character lengths. All possible parity settings are tested.

Test 3 - Character Sync Multi-Sync Test

This test verifies that the SIO is capable of recognizing all valid Sync characters, in character synchronous mode. Both channels are tested, at varying baud rates.

Test 4 - Character Sync Data Transfer Test

This test checks data transfer in character synchronous mode, at varying baud rates. Both channels are tested.

Test 5 - Character Sync CRC-16 Test

This test checks the operation of the SIO's CRC generation and checking circuitry, in character Synchronous mode, using CRC-16.

Test 6 - Character Sync DMA Test - DMA In

This test performs high speed data transfer over Channel A, in Character Synchronous mode, using program status loop for transmission and DMA for reception.

Test 7 - Bit Sync Data Transfer Test

This test checks data transfer in bit synchronous mode, at varying baud rates. Both channels are tested.

Test 8 - (not used)

Test 9 - Bit Sync Abort Test

This test checks the SIO's ability to recognize and generate an abort condition, in bit synchronous mode.

Test 1Ø - Bit Sync DMA Test - DMA In

This test performs high speed data transfer over Channel A, in Bit Synchronous mode, using program status loop for transmission and DMA for reception.

Test 11 - Inter-processor Test

This test performs full speed (615K baud) transfers between two systems, using DMA for both transmission and reception. For proper operation, both systems must be running this test, configured for RS-422 cluster operation, and connected by a standard cluster cable. They must not, however, be connected to an operational cluster at this time.

Test 12 - Character Sync Dma Test - DMA Out

This test performs high speed data transfer over Channel A, in Character Synchronous mode, using DMA for transmission and program status loop for reception.

Test 13 - Bit Sync Dma Test - DMA Out

This test performs high speed data transfer over Channel A, in Bit Synchronous mode, using DMA for transmission and program status loop for reception.

NOTE

Tests 1, 6, 10, 12, and 13 are the only tests that do not use interrupts.

Error Message Format

For errors detected during the test, a standard error message will be displayed, giving a description of the error category, followed by a message of the following form:

```
Channel n  I = i    Was: a b c    Shd Be: x y z
```

where "n" is the channel on which the error occurred (A or B).

For Data Transfer errors, "i" indicates the decimal byte position within the block at which the error occurred, aa is the hexadecimal value received, and "x" is the hexadecimal value expected. "b", "c", "y", and "z" are not meaningful.

For Status errors, "a" indicates the SIO status (Register 0) received, and "x" the status expected. In test 1, "b" "c" indicates the received value of the Extended Status Register (port 60h), and "y" "z" the value expected. "b", "c", "y", and "z" are all hexadecimal. "i" is not meaningful for status errors and "b", "c", "y", and "z" are not meaningful for tests other than test 1.

CommIOP Diagnostic

Overview

The CommIOP diagnostic verifies the proper operation of each CommIOP. The diagnostic is a system confidence test of the CommIOP processor, its RAM, the CT to multibus path, and the CommIOP communications hardware. Each CommIOP is tested individually.

The CommIOP has its own microprocessor which interprets commands provided by the workstation. The diagnostic gives various commands to the CommIOP and interprets the results.

To run the diagnostic you must insert a Cluster Communication Cable between the lines of the CommIOP to be tested. For CommIOP 1, connect the cable between "LINE 1" and "LINE 2" on the back of the workstation. For CommIOP 2, use "LINE 3" and "LINE 4".

The CommIOP LED

The CommIOP has an LED which is located at the top center of the board. When the workstation is powered up, or the RESET button is pressed, the microprocessor blinks this LED. If the LED does not blink, the CommIOP is not functioning properly.

The CommIOP uses the LED to blink its current status with a two digit code. The first digit is blinked, followed by a short pause (the LED is off), and then the second digit is blinked.

Under normal circumstances it is not necessary to interpret the code being displayed since error conditions are reported by the diagnostic. There are conditions, however, when the CommIOP cannot report the error to the diagnostic due to hardware malfunction on the CommIOP. In this case, it is necessary to interpret the code being displayed. A list of these status codes is provided below.

Operation

Bootstrap the CommIOP diagnostic, by inserting the CommIOP Installation Diskette in the floppy drive and pressing the RESET button in the back of the workstation.

When first loaded, the CommIOP diagnostic announces itself:

Change Parameters?

Answer N RETURN to run the standard test sequence on CommIOP 1 (configured for I/O address 8Ø4Ø). Answer Y RETURN to run only certain tests, or to change certain test parameters (such as to run the test on CommIOP 2). The list of parameters is given below.

If you answer yes to "Change Parameters?", you are prompted:

Run all tests?

Answer Y RETURN to run all the tests. Type N RETURN if you wish to select a list of individual tests to run. The tests available are displayed on the screen. Type in each test number you wish to run followed by RETURN. When all the desired test numbers have been entered, press RETURN only. If you select a nonexistent test or improperly type in a test number (for instance 1A RETRURN), an audible tone is signalled. Simply re-enter the test number correctly.

Test Descriptions

Test 1 - IOP Functional Test

This test verifies the basic functioning of the CommIOP. The following items must be functioning or correctly set up in order for this test to succeed:

- The proper address must be selected on CommIOP address selection switches (bits 1-4 on switch SW1 on the CommIOP board). See the CommIOP Installation Instructions for the correct switch settings.
- The CommIOP processor, ROM, and multibus interface must be functional.
- The Multibus-to-CT address mapping switch S2 (located on the CPU board of the workstation) must be correct. See the CommIOP Installation Instructions for the correct switch settings.

Any failure of the above items cause this test to fail. If the functional test fails, check the following to isolate the fault.

- Make sure the address selection switches (bits 1-4 on switch SW1) on the CommIOP(s) are properly set. This is the problem if the diagnostic displays the message **"No CommIOP is responding at port address : xxxx".**

- Make sure that the LED on the CommIOP blinks on and off when the RESET button of the workstation is pressed. Reboot the diagnostic, and check if the LED is blinking. If it is not, the CommIOP hardware is probably faulty.

- If the LED does indeed blink when RESET, but stays on or off, after Test 1 is run (and failed), the likely problem is the CommIOP-to-CT Multibus path. Make sure that bit 1 on S2 of the CPU board on the workstation is ON.

- If the test is still failing, and the LED is blinking before and after the test is run, then check the two-digit code which is being displayed on the LED. The code should be 12 when the workstation is first RESET, and 11 after the test has run. If another code is displayed, refer to the CommIOP Status Codes Section below for an interpretation of the error.

Test 2 - CommIOP Memory Test

This test checks the CommIOP's memory. The test writes Ø's, 1's, and an address pattern in each location of the CommIOP's memory. Any errors are reported by the diagnostic.

The diagnostic correlates any single bit errors to the actual chip location on the CommIOP board.

Test 3 - Multibus Test

This test verifies the CommIOP-to-CT Multibus path. One or more 16K segments are selected on the CT memory. The CommIOP performs memory tests on these segments. Any errors are reported by the diagnostic.

Test 4 - CommIOP to CT Interrupt Test

This test verifies that interrupts from the CommIOP to the workstation function correctly. A small test program is loaded in the CommIOP memory. This program interrupts the workstation.

Interrupts from all possible levels are monitored by the diagnostic. The diagnostic indicates whether no interrupt was received or an interrupt(s) occured on a level which is different from the one specified (see Parameters) in the parameter list.

If this test fails, verify that the interrupt level selection switches (bits 5-8 on switch SW1 of the CommIOP board) are properly set. See the CommIOP Installation Instructions for the proper switch settings.

Test 5 - CommIOP Comm Test

This test verifies the communications portion of the CommIOP hardware. A small test program is loaded into the CommIOP which loops messages between the two channels. In order to run this test, you must install a cluster communications cable between the channels of the CommIOP to be tested (lines 1 and 2 for CommIOP 1, lines 2 and 3 for CommIOP 2).

The most common error message displayed is:
"SIO Channels are not cross-connected."

If this error message is displayed, make sure that the cable is properly connected between the two lines. Also make sure that the ribbon cable connector is properly attached to the CommIOP board (see the CommIOP Installation Instructions).

Parameter Prompts

After the test sequence to be performed is selected, you are able to change certain test parameters. When the diagnostic is first run, certain default settings are chosen for the test. To leave the current setting of the parameter, press RETURN. To change a parameter, enter the new value followed by RETURN. A audible tone is sounded if an invalid parameter is entered.

To begin the tests, press GO.

To abort the tests, press and hold FINISH. To suspend a test, press any key other than GO or FINISH. To resume the test, press GO.

times to run

The default value is 1. The test sequence selected will run for this number of times unless an error occurs (and "Stop on Error?" was selected) or FINISH is pressed.

Stop on error?

The default is Yes. Specify N RETURN is you wish the tests to continue after errors. The number of errors encountered during the tests are displayed at the end of the diagnostic.

CommIOP address

The default is 8040. Enter 8040, 8041, 8050, 8051 to specify CommIOP 1, 2, 3, or 4 respectively. Any other value is rejected before the tests are run.

CommIOP Interrupt Level

The default is 2. Specify the interrupt level to be used by the CommIOP to communicate with the workstation. Levels 0, 2, 5 or 6 are available to the CommIOPs.

The operating software uses the same interrupt level for all of the CommIOPs. The standard interrupt level used is level 2.

CT memory address 16K segment for Multibus test

The default is 4 (i.e. 64 to 79K) on the workstation. This parameter is used by Test 3 (Multibus test). Specify any segment which you wish to use for the test. You cannot specify a segment less than 64K (4), since the CommIOP would overwrite the diagnostic. Also you cannot specify a segment which is outside CT memory bounds or greater that 512K (32).

Number of 16K segments to test

The default is 1. This parameter is used by Test 3 (Multibus test). Specify the number of 16K segments to be tested. All of the segments must be within CT memory bounds and less that 512K.

Operation

The diagnostic first verifies that all parameters are correct. The diagnostic displays an error message if an invalid CommIOP address or interrupt level was specified, or if a segment is outside the valid range. Change the parameter which was flagged as incorrct and run the test again.

Each test is performed by the CommIOP and monitored by the diagnostic. The code executed by the CommIOP microprocessor for tests 1 through 3 resides in the CommIOP ROM. The code for tests 4 and 5 is loaded by the CommIOP from CT memory.

CommIOP Status Codes

The following is the list of status codes which are blinked by the CommIOP but are not returned to the diagnostic due to hardware malfuntions.

11	Command executed successfully.
12	No command has been received from the master since the workstation was RESET. If this status is flashed after a test is run, the workstation to CommIOP connection is not functioning properly.
13	Checksum error in CommIOP Boot ROM.
14	The CommIOP Multibus register is not functioning correctly.

9. AWS DIAGNOSTICS

There are three kinds of diagnostics: Bootstrap ROM, CTOS initialization, and stand-alone. Unless otherwise stated, the following applies to the entire AWS family of machines. (I.e. AWS-210, 220, 230, 240).

The function of the diagnostic in the Boot ROM is to ensure that enough of the Application Work Station (AWS) is functional to permit the Boot ROM to load the OS or a diagnostic from disk or from Comm (RS-422 Communications line). The Boot ROM checksums the ROM, tests the memory, and attempts to read from the floppy, Winchester, or Comm using DMA but not interrupts. If an error is detected, it displays a definitive error code on the screen and beeps the audio output.

The AWS Boot ROM will initialize the video, comm, and keyboard before attempting to load a program from disk/floppy/comm. It does NOT test I/O interrupts or non-maskable interrupts (parity error, etc.). The CTOS initialization code tests all of these functions before loading the Convergent Executive.

CTOS initialization verifies the operation of I/O interrupts, non-maskable interrupts (parity error, etc.), system memory, video, keyboard (reset, checksum, loop-back), real-time clock, interval timer, SIO (status during and after initialization only), etc. CTOS initialization failure detection is described in the section "CTOS Failure Analysis" of the System Programmer's Guide.

The stand-alone diagnostics are intended to determine whether each major element of an AWS is functional. When possible, they isolate to the board level. Each stand-alone diagnostic may be loaded by the Boot ROM from individual floppy diak or from the master work station over the comm line.

Stand-alone diagnostics may be run by an untrained person using all default parameters to determine whether the function is error-free. They can also be run by trained personel to identify the exact manner of failure.

Summary of Diagnostics

System (video, timer, keyboard)

The System Diagnostic verifies the operation of I/O interrupts, non-maskable interrupts (parity error, etc.), video, keyboard (reset, checksum, loop-back), real-time clock, and programmable interval timer. Using default parameters, the video, timer, and keyboard tests are executed consecutively in that order. However, each test within the system diagnostic may be exercised individually and specific parameters changed to allow a wide variety of tests to be run.

Video

The Video test displays a sequence of moving patterns which exercise all modes, characters, and attributes. The video output on the screen provides visual verification of correct operation. If an error is detected, this diagnostic displays an error message on the video screen (if possible) and displays an error code in the internal LED's and keyboard LED's.

Keyboard

The Keyboard test presumes that the video is operational. It resets the keyboard micro-processor, checksums the keyboard ROM, performs a loop-back test, and prompts the user to type characters on the keyboard. It continuously displays on the video an indication of all keys currently depressed. Each time a key with an LE. is depressed, the state of the LED is toggled. Error messages are displayed on the video.

Timer

The timer test verifies the correct operation of the 8253 programmable interval timer and its associated interrupt logic.

Memory

The Memory Diagnostic tests are different from the Memory Test of the AWS Boot ROM in that the Boot ROM test does not test with parity enabled, and uses a simpler pattern for its test. The boot ROM test tests all of memory, while this test tests all of the memory except that which the program uses.

Floppy

Using default parameters, the Floppy Disk Stand-alone Diagnostic requests a scratch floppy in drive zero and tests seek, format, and single and multi- sector

write/read. Entry of specific parameters allows a wide variety of tests to be run on drives Ø, 1 or both.

Winchester

Similar to Floppy.

Communications (SIO)

Using the default parameters, the Communication Stand-alone Diagnostic requires that a loop-back cable be installed and tests both channel A and B in Asynchronous, Character-Synchronous, and Bit-Synchronous (BOP) modes with and without interrupts and with and without DMA. The diagnostic asks whether channel A is switch selected for RS-232c or RS-422 and performs its test accordingly. Entry of specific parameters allows a wide variety of tests to be run.

When and How To Run Diagnostics

Except for the video diagnostic, all stand-alone diagnostics presume that the video subsystem is operational and use the video screen to display error messages. Except for the video and keyboard diagnostics, all stand-alone diagnostics presume that the keyboard is operational and accept input from the keyboard. In addition, the keyboard diagnostic assumes that the 8253 interval timer is operating properly. Because of these dependencies, the System Diagnostics should be executed before other stand-alone diagnostics are performed.

Special Keys Used in the Stand-Alone Diagnostics

key:	use:
BACK SPACE	deletes last character typed
DELETE	deletes all characters typed
RETURN	indicates field entered
GO	begins execution
FINISH	terminates the test being executed (Hold the FINISH key depressed until the diagnostic acknowkedges)
ACTION	(hold ACTION and press "A" to enter Debugger; press GO to exit Debugger)

Running Diagnostics from an AWS

Diagnostic programs may be loaded from the master work station or, if local storage is present, from a floppy disk.

Bootstrapping from the master

In order to run diagnostic from a cluster workstation which doesn't have any disk drives, it is neccessary to bootstrap the diagnostic from the master workstation. This may be done in one of two ways.

(1) Using the Bootstrap command of the Executive to load diagnostics from the master work station.

Insert the floppy containing the diagnostic into drive fØ at the master workstation. Next, type in at the cluster workstation the Bootstram command as follows:

```
Command   Bootstrap
Bootstrap
  File name  [fØ]<Ct>xxx
```

where "xxx" is the name of the run file containing the diagnostic to be run, e.g. AwsSystem.diag. (See the Release Notice for the precise names of the diagnostic run files and for the floppy on which each is contained.)

(2) Loading diagnostics with Boot ROM.

First, copy the run file for the diagnsotic to be run from the distribution diskette to a file [Sys]<Sys>WSnnn>SysImage.Sys on the master work station, where "nnn" is any workstation type not otherwise used by the system. "nnn" must be less than 256. (See the release notice for the locations and names of the diagnostic files.)

One possible numbering scheme for diagnostic files residing permanently at the master workstation is as follows:

nnn	Description
1ØØ	System Diagnostic
1Ø1	Memory Diagnostic
1Ø2	Printer Diagnostic
1Ø3	Floppy Diagnostic
1Ø4	Communication Diagnostic
1Ø5	Winchester Diagnostic

After copying the diagnostic to be run onto system volume, push the reset button of the AWS while holding down the space bar. The boot ROM menu and the version number should be dispalyed as follows:

```
V x.x
B,C,D,L,M,P,T:
```

Next type a 'T' after the colon. The boot ROM should respond with

```
V x.x
B,C,D,L,M,P,T:T
OS:
```

Next, enter the three digit workstation type for the diagnostic to be loaded, then press RETURN. The boot ROM menu will be redisplayed:

```
V x.x
B,C,D,L,M,P,T:T
OS:nnn

B,C,D,L,M,P,T:
```

Type the letter 'B' to start loading the diagnostic.

CAUTION

When running the Communication Diagnostic, disconnect the cluster cables from the workstation after the diagnostic is loaded and before any tests are run.

Bootstrapping from a floppy

Insert the diskette containing the diagnostic to be executed into the floppy drive. If the power is already on, simply press the reset button. Otherwise, turn the power on and the diagnostic will be loaded without further user intervention.

System Diagnostic

The AWS workstation System Diagnostic tests the video, timer, and keyboard. When the diagnostic is loaded, you will be asked:

```
Change Parameters  (Y/N) ?
```

If you type in "no" followed by a RETURN then the three tests will automaticaly start executing. If you answer "yes" then the diagnostic will display its menu and ask:

```
Tests are:

  1 - Video test
  2 - Timer test
  3 - Keyboard test

Standard test sequence is 1, 2, and 3.
Run Standard Test Sequence (Y/N) ?
```

If you say "yes", then the tests that will be run are the , video, timer and keyboard in that order. If you say "no", then you will be asked:

```
Test to run ?
```

Type in the number of each test followed by a <cr> (RETURN) character. After all test numbers are entered, type in a final RETURN.

After the test selection is made, you will be prompted to enter any optional parameters to the tests:

```
THE FOLLOWING PARAMETERS ARE AVAILABLE FOR CHANGE

<cr> to leave current value unchanged
<GO> when all parameters are satisfactory

Times to Run ? [1]
```

The number of times to run is the only parameter for this diagnostic. Type in a GO character to start the test sequence.

```
Test 1 -- Video Test
```

The video test is the same as that of the IWS video diagnostic (see System Programmer's Guide, section 8 for description of IWS diagnostics), except that it runs through all of its patterns only once, and then starts the next test.

Test 2 -- Timer Test

The timer test verifies the correct operation of the 8253 programmable interval timer and its associated interrupt logic.

Test 3 -- Keyboard Test

The AWS keyboard test is the same as the IWS test, except that if you type ACTION-FINISH (hold down both keys at once), the test terminates and goes on to the next test in the sequence.

In the Video and Timer tests, if you type in a CANCEL character while the test is running, the tests will pause until a GO or FINISH character is typed in. If a GO character is typed in, the test will continue, and if a FINISH is typed in the test sequence will be terminated.

Memory Diagnostic

The AWS workstation Memory Diagnostic tests the workstation memory. When the diagnostic is loaded, you will be asked:

```
Change Parameters  (Y/N) ?
```

If you type in "no" followed by a RETURN then the first (shorter) test will automaticaly start executing. If you answer "yes" then the diagnostic will display its menu and ask:

```
Tests are:

  1 - Memory test
  2 - Memory test - Galpat

Standard test sequence is 1.
Run Standard Test Sequence (Y/N) ?
```

If you say "yes", then the only test that will be run is the first memory test. If you say "no", then you will be asked:

```
Test to run ?
```

Type in the number of each test followed by a <cr> (RETURN) character. After all test numbers are entered, type in a final RETURN.

After the test selection is made, you will be prompted to enter any optional parameters to the tests:

```
THE FOLLOWING PARAMETERS ARE AVAILABLE FOR CHANGE

<cr> to leave current value unchanged
<GO> when all parameters are satisfactory
```

The first parameter you may change is:

```
Times to Run ? [1]
```

The default will be in brackets. To change this, type in the number of times you wish to execute the test,followed by a RETURN.

The next parameters is:

```
Stop on memory error ? [yes]
```

Type in a "y" or a "n" followed by a RETURN to change this.

The last parameter is:

```
Start segment for Galpat (1 = 64K, 2 = 128k, etc.)
[1]
```

If you are using the Galpat memory test, you may wish to change this parameter if you suspect a memory problem in a specific memory range.

Type in a GO character to start the test sequence.

Test 1 -- Memory test

The Memory Diagnostic tests are different from the Memory Test of the AWS Boot ROM in that the Boot ROM test does not test with parity enabled, and uses a simpler pattern for its test. The boot ROM test tests all of memory, while this test tests all of the memory except that which the program uses. This test has six distinct passes. The passes are:

- Write and read Ø's to all test memory.
- Write and read 1's to all test memory.
- Write and read address patterns (even addresses)
- Reread even address patterns from test memory.
- Write and read address patterns (odd addresses).
- Reread odd address patterns from test memory.

If an error occurs during the test, the relevant information will be displayed and you will be prompted with:

```
Continue ?
```

Type "y" and RETURN if you want to continue the test, or "n" and RETURN if you want to restart the program.

Test 2 -- Memory Test (Galpat)

This memory test runs an exhaustive test of the memory from 64K. This test takes several hours to run. A sequence of "*" characters will be displayed during the test so that you may know it is running. Parity and invalid data will be reported.

If you type in a CANCEL character while the test is running, the tests will pause until a GO or FINISH character is typed in. If a GO character is typed in, the test will continue, and if a FINISH is typed in the test sequence will be terminated.

Printer Diagnostic

The objective of the AWS Printer Diagnostic is to test the parallel printer port and its supporting logic, and (in the event of a malfunction) locate and identify the cause of the malfunction as completely as possible.

The printer port may be exercised both with and without using interrupts. In addition, there are several parameters controlling the operation of the diagnostic program that may be modified by the operator.

The diagnostic requires that a printer with a Centronics interface be connected to the printer port on the AWS.

Operation

When first loaded, the AWS Printer Diagnostic announces itself with:

```
AWS Printer Diagnostic Rev x.y
Copyright 1981 by Convergent Technologies Inc.

Change Parameters (Y/N) ?
```

If you answer N <RETURN> when the diagnostic has just been loaded, the standard test sequence will be run with a default set of control parameters. Answering N <RETURN> on subsequent runs of the diagnostic will use the previously selected test sequence and control parameters. If the test sequence and control parameters are to remain unchanged, the tests will begin executing immediately. Answering Y <RETURN> to this prompt at any time will allow the test sequence and control parameters to be changed before the tests are executed.

If the test sequence and control parameters are to be changed, the diagnostic will prompt with:

```
Run standard test sequence (Y/N) ?
```

If you answer Y <RETURN>, the standard test sequence will be selected. The standard test sequence consists of tests 1 & 2. (See the Test Descriptions section of this document for a detailed description of each test.) If you answer N <RETURN>, the diagnostic will allow the test sequence to be changed before the tests are executed.

If the test sequence is to be changed, the diagnostic will display a menu of the available tests and prompt with:

```
Enter test # followed by RETURN,
or just RETURN to end selections.

Test # to run
```

Type in the number of the test that you wish to run, followed by <RETURN>. A beep is sounded if no such test number exists. The tests may be entered in any sequence desired, and may be repeated within the sequence. A maximum of 2 tests is allowed in the sequence. The test entering mode may be exited either by entering 2 tests, or by entering <RETURN> with no test number.

Once the test sequence has been defined, the diagnostic will afford the opportunity to change certain control parameters. (See the Control Parameters section of this document for a detailed description of each parameter.) The diagnostic will prompt with:

```
Enter new parameter value, followed by <RETURN>,
just <RETURN> to leave the current value unchanged,
or <GO> to begin tests.
```

The diagnostic will then list a parameter and its current value, and will prompt for a new value. Answering <RETURN> will keep the current value of the parameter unchanged. Answering [value]<RETURN> will make [value] the new value for the parameter. If [value] is inappropriate for the given parameter, a beep will be sounded, and you will be prompted to reenter the parameter value. After keeping or changing the value for a parameter, the diagnostic will list the next parameter for examination or modification. After the last parameter has been examined or modified, the diagnostic will start over with the first parameter. Answering <GO> whenever the diagnostic prompts for a parameter value will finish the parameter modification and start execution of the tests. It is not necessary to examine or modify all of the parameters prior to test execution. The diagnostic will use the most recent set of parameters.

As the tests are executed, the diagnostic will display and update the test being run and the current test sequence pass.

At any time during the execution of the tests, pressing <FINISH> will abort the execution of the test in progress and restart the diagnostic. Pressing any other key (except <GO>) will suspend the execution of the test in progress. After suspending the test execution, the diagnostic will prompt with:

```
Testing suspended...
Press <GO> to resume, <FINISH> to terminate
```

Pressing <GO> will resume testing at the point where it was suspended. Pressing <FINISH> will terminate the test in progress and retart the diagnostic.

Should a malfunction occur, the error detected and the current printer port status will be displayed. Depending on the operating parameters selected, the testing will either be continued or aborted at this time.

Test Descriptions

Test 1 --- Barber pole - No interrupts

This test outputs one page (66 lines of 132 columns) of a test pattern consisting of all 96 printable ASCII characters. Each successive line is shifted one character to the left, resulting in a 'barber pole' pattern. Before the test pattern is transmitted, the status of the printer is checked. If the printer is busy, not selected, or the printer buffer is busy, an error will be flagged, and the test suspended. If the printer status is ok, the diagnostic will then transmit the test pattern. After each character is transmitted to the printer, the diagnostic polls the printer status. If the printer buffer does not become free to accept another character before the maximum wait time (see the Parameter Descriptions section of this document for details), an error is flagged and the test is suspended.

Test 2 --- Barber pole - Interrupts

This test is identical to test 1, except that after a character is transmitted to the printer, instead of polling the printer status, the diagnostic waits for an interrupt to be generated by the printer acknowledge, signifying that the printer is ready to receive another character. If the interrupt is not received before the maximum wait time (see the Parameter Descriptions section of this document for details), or the printer status shows the printer is still busy after the interrupt, an error is flagged, and the test is suspended.

Test 3 --- Printer function tests

This test allows any sequence of hex character or control codes to be output to the printer. The diagnostic prompts to input a hex control code. The code may be entered from the keyboard as 1 or 2 hex digits, followed by <RETURN>. An invalid hex entry will cause a beep to be sounded and the prompt re-issued. The code will be transmitted to the printer immediately, and the diagnostic will then wait for a printer acknowledge interrupt, signifying that the printer is ready to receive another character. If the interrupt is received before the maximum wait time (see the Parameter Descriptions section of this document for details) and the printer status shows the printer not

busy after the interrupt, the diagnostic will prompt to input another control code. The test may be exited at any time by pressing <FINISH> when prompted to enter a control code. If a printer acknowledge interrupt is not received before the maximum wait time, or the printer status shows the printer busy after the acknowledge interrupt, an error is flagged and the test is suspended.

Parameter Descriptions

Times to run — Default value [1]

Determines how many times the selected test sequence will run (how many passes). The test sequence will be run this many times, or until an error occurs. Parameter value may be any decimal number from 1 to 9999.

Max printer wait time (ms) — Default value [6ØØØ]

Determines the maximum time (in milliseconds) the diagnostic should wait for the printer to become ready to accept another character before declaring that an error has occurred. In test 1, this is the maximum time that the diagnostic should spend polling the printer status. In tests 2 & 3, this is the maximum time that the diagnostic should spend waiting for an acknowledge interrupt from the printer. Parameter value may be any decimal number from Ø to 9999, however, the minimum must be at least as long as the time required for the printer to execute its slowest function (usually a full page form feed), or false timeout errors will be flagged.

Software debug — Default value [NO]

Determines if the debugger may be entered after an error is detected. After an error is detected and displayed, if this option is selected, the diagnostic will prompt with:

Debug ?

Answer with Y <RETURN> if you wish to enter the debugger at this time. Answer with N <RETURN> if not. Parameter value may be YES or NO.

Bypass errors & continue — Default value [NO]

Determines if the testing may be continued after the detection of an error. After an error is detected and displayed, if this option is selected, the diagnostic will prompt with:

Ignore error & continue ?

Answer with Y <RETURN> if you wish the testing to continue where it left off when the error was detected. Answer with N <RETURN> if you wish the testing to be aborted and the diagnostic restarted. Parameter value may be YES or NO.

Error Message Format

If a malfunction should occur, and the diagnostic detects an error during testing, an error message will be displayed and testing will be suspended. The error message displayed contains two sections: the error message, and the printer status.

The error message section is a brief explanation of the type of error that was detected by the diagnostic (e.g. >>> Timeout waiting for printer interrupt <<<).

The printer status section displays the status of the printer port at the time of the error. Each status port bit is displayed (bit function and actual state).

AWS-220/230 Floppy Diagnostic

Preface

This description of the floppy disk diagnostic is not, in itself, sufficient to enable troubleshooting of a malfunctioning floppy disk controller or drive. Refer to the AWS-220/230 Hardware Manuals for information on the Floppy Disk Controller. This information (especially the instruction set of the uPD765 and the format of the main status register and other status registers) is vital to understanding the status information included in error messages.

This description is sufficiently self-contained, however, to permit the reader to perform an extensive battery of tests and to determine whether the floppy disk subsystem is performing correctly or requires remedial attention.

Overview

The objective of the AWS Floppy Disk Diagnostic is to test the floppy disk controller and drives, and (in the event of a malfunction) locate and identify the cause of the malfunction as completely as possible.

Any or all floppy drives on the system may be tested by the diagnostic. The diagnostic also pre-tests the area of memory that will be used as the disk I/O buffer & reports any errors found. The diagnostic assumes that the processor, memory (other than that in the buffer), 8237-2 DMA Controller, and the 8259A Interrupt Controller are all functioning properly.

There are several tests and functions that may be run in the diagnostic. The sequence of tests to be run may be selected by the operator. In addition, there are several parameters controlling the operation of the diagnostic program that may be modified by the operator.

Operation

When first loaded, the AWS Floppy Disk Diagnostic announces itself with:

```
AWS Floppy Disk Diagnostic Rev x.y
Copyright 1981 by Convergent Technologies Inc.
```

```
Change Parameters (Y/N) ?
```

If you answer N <RETURN> when the diagnostic has just been loaded, the standard test sequence will be run with a default set of control parameters. Answering N <RETURN> on subsequent runs of the diagnostic will use the previously selected test sequence and control parameters. If the test sequence and control parameters are to remain unchanged, the tests will begin executing immediately. Answering Y <RETURN> to this prompt at any time will allow the test sequence and control parameters to be changed before the tests are executed.

If the test sequence and control parameters are to be changed, the diagnostic will prompt with:

```
Run standard test sequence (Y/N) ?
```

If you answer Y <RETURN>, the standard test sequence will be selected. The standard test sequence consists of tests 1, 2, 3, 4, 5, 6, 7 & 8. (See the Test Descriptions section of this document for a detailed description of each test.) If you answer N <RETURN>, the diagnostic will allow the test sequence to be changed before the tests are executed.

If the test sequence is to be changed, the diagnostic will display a menu of the available tests and functions and prompt with:

```
Enter test # followed by RETURN,
or just RETURN to end selections.

Test # to run
```

Type in the number of the test that you wish to run, followed by <RETURN>. A beep is sounded if no such test number exists. The tests may be entered in any sequence desired, and may be repeated within the sequence. A maximum of 8 tests is allowed in the sequence. The test entering mode may be exited either by entering 8 tests, or by entering <RETURN> with no test number.

Once the test sequence has been defined, the diagnostic will afford the opportunity to change certain control parameters. (See the Control Parameters section of this document for a detailed description of each parameter.) The diagnostic will prompt with:

```
Enter new parameter value, followed by <RETURN>,
just <RETURN> to leave the current value unchanged,
or <GO> to begin tests.
```

The diagnostic will then list a parameter and its current value, and will prompt for a new value. Answering <RETURN> will keep the current value of the parameter unchanged. Answering [value]<RETURN> will make [value] the new value for the parameter. If [value] is inappropriate for the given parameter, a beep will be sounded, and you will be prompted to reenter the parameter value. After keeping or changing the value for a parameter, the diagnostic will list the next parameter for examination or modification. After the last parameter has been examined or modified, the diagnostic will start over with the first parameter. Answering <GO> whenever the diagnostic prompts for a parameter value will finish the parameter modification and start execution of the tests. It is not necessary to examine or modify all of the parameters prior to test execution. The diagnostic will use the most recent set of parameters.

As the tests are executed, the diagnostic will display and update the test being run and the current test sequence pass.
At any time during the execution of the tests, pressing <FINISH> will abort the execution of the test in progress and restart the diagnostic. Pressing any other key (except <GO>) will suspend the execution of the test in progress. After suspending the test execution, the diagnostic will prompt with:

```
Testing suspended...
Press <GO> to resume, <FINISH> to terminate
```

Pressing <GO> will resume testing at the point where it was suspended. Pressing <FINISH> will terminate the test in progress and retart the diagnostic.

Should a malfunction occur, the error detected and the current floppy controller status will be displayed. Depending on the operating parameters selected, the testing will either be continued or aborted at this time.

Test and Function Descriptions

Test 1 - Selection

The selection test verifies the basic functioning of the processor / floppy interface by reading the status of the selected drive. If the selected drive does not have a floppy inserted and the door closed, then message 'Drive not ready' will be displayed. If the floppy being tested has a write protect tab applied, the message 'Write protect set for drive n' is displayed. Any other errors detected are also displayed.

Test 2 - Recalibrate

This test check the basic function of the head positioning mechanism and track zero sensor. The test first initializes the floppy interface, issuing a recalibrate command to the floppy controller in the process. The test then issues a command to step the head inward to the center of the disk. The test finally issues another recalibrate command, stepping the head to track Ø. Any errors detected are displayed.

Test 3 - Seek sequential

This test checks the function of the head positioning mechanism by stepping the head from track Ø to the innermost track on the disk, one track at a time. Any errors detected are displayed.

Test 4 - Seek random

This test checks the function of the head positioning mechanism by stepping the head from track TPD/2 to track TPD/2-1 to track TPD/2+1 to track TPD/2-2 to track TPD/2+2 etc... Where TPD is the number of tracks per disk. Any errors detected are displayed.

Test 5 - Format

This test formats the diskette(s) in the selected drive(s). The diskettes are formatted a track at a time. Data is first written to the disk, and then read back, to verify it. The format used is spiraled, that

is, logical sector 1 on each track is located 3 physical sectors farther from the index hole than logical sector 1 on the previous track. If the disk is double sided, both sides of each track are formatted before the head is moved to the mext track.

Test 6 - Sequential write/read Single sectors

This test writes data to each sector on a track, one sector at a time. First, the odd numbered logical sectors are written, using the filler data (See the Operating Parameters section of this document for details). Then the even numbered logical sectors are written, using the one's compliment of the filler data. After all the sectors on the track are written, the data in each sector is verified against the data written. The test is repeated on three tracks, track Ø, track TPD/2 and track TPD-1, where TPD is the number of tracks per disk. If the disk is double sided, both sides of each track are tested before the head is moved to the next track. Any errors detected are displayed.

Test 7 - Random write/read Single sectors

This test is the same as test 6, except that all tracks are tested, in the same order as in test 4.

Test 8 - Sequential write/read Multiple sectors

This test writes data to each sector on a track, one half track at a time. First, sectors 1 to SPT/2 are written, using the filler data (See the Operating Parameters section of this document for details). Then sectors SPT/2+1 to SPT are written, using the one's compliment of the filler data. SPT is the number of sectors per track on the disk. The data in each sector is then verified against the data written. If the disk is double sided, both sides of each track are tested before the head is moved to the next track. Any errors detected are displayed.

Test 9 - Ready/Not-ready interrupt

This test verifies that the opening and closing of the floppy drive door causes an interrupt with the proper drive status. The test first prompts to make the drive

not ready, by opening the door. The test expects this to be accomplished within 30 seconds. If the interrupt is not received within the time allotted, or the drive status is incorrect after the interrupt is received, an error will be displayed. If all goes well opening the door, the test then prompts to make the drive ready again, by opening the door. The test also expects this to be accomplished within 30 seconds. Likewise, if this interrupt is not received within the allotted time, or the drive status is incorrect after the interrupt, an error will be displayed.

Function 10 - Read a track

This function reads an entire track into the DMA buffer, and displays it 256 bytes at a time. The test prompts for a cylinder number and, if the disk is double sided, a head number. Any invalid entries will be flagged. Enter just '<RETURN>' to the cylinder number prompt to exit the function. The track specified will then be read into the buffer, and the first 256 bytes will be displayed. The test will prompt with 'More ?'. Answer with 'Y <RETURN>' to display the next 256 bytes in the buffer. Answer with 'N <RETURN>' to read another track.

Function 11 - Display/Modify

This function allows you to display, and optionaly, modify any sector on the disk. The test prompts for a cylinder number, a sector number, and, if the disk is double sided, a head number. Any invalid entries will be flagged. Enter just '<RETURN>' to the cylinder number prompt to exit the function. The sector specified will be read into the DMA buffer, and the first 256 bytes will be displayed. The test will prompt with 'More ?'. Answer with 'Y <RETURN>' to display the next 256 bytes in the buffer. Answer with 'N <RETURN>' to continue the test. The test then prompts with 'Modify ?'. Answer 'N <RETURN>' to inspect another sector or exit the test. Answer 'Y <RETURN>' if you wish to modify the sector. The test will prompt with 'Byte ?'. Enter the decimal number of the byte you wish to modify, or just return to quit making changes. The test will then display the byte number and its hex value. Enter a new hex value for the byte followed by '<RETURN>' if you wish to change the byte, or just '<RETURN>' if you do not wish to change the byte. After you have completed modifying the sector, the test will prompt with 'Write sector ?'. Answer with 'Y <RETURN>'

if you wish the sector to be re-written with the modified data, and with 'N <RETURN>' if you do not wish to re-write the sector.

Function 12 - Copy

This function copies the entire contents of one diskette onto another. The source and destination drives are determined by the initial 'Drives to test ?' prompt when the diagnostic was first started. The source drive is the first drive specified, and the destination drive is the second drive specified. The destination diskette must be formatted (using test 5 or some other means...) prior to attempting to copy to it. After eack track is written to the destination disk, it is verified against the data on the source disk. If the disk is double sided, both sides of the diskette is copied and verified before the head is moved to the next track.

Function 13 - Read ID

This function reads the ID field of the next sector to pass under the read/write head. The test prompts for the cylinder number, and if the disk is double sided, for the head number.

Note: Functions 14, 15 & 16 are designed to support troubleshooting using test equipment.

Function 14 - Format loop

This function continuously formats one track on the disk. The test prompts for the cylinder number, and if the disk is double sided, the head number. This test runs continuously until '<FINISH>' is pressed.

Function 15 - Read loop

This function continuously reads one track on the disk. The test prompts for the cylinder number, and if the disk is double sided, the head number. This test runs continuously until '<FINISH>' is pressed.

Function 16 - Read loop

This function continuously writes one track on the disk. The test prompts for the cylinder number, and if the disk is double sided, the head number. This test runs continuously until '<FINISH>' is pressed.

Function 17 - Compare

This function compares the data on two diskettes. It is the same as the verify portion of the copy function (Function 12).

Parameter Descriptions

Times to run — Default value [1]

Determines how many times the selected test sequence will run (how many passes). The test sequence will be run this many times, or until an error occurs. Parameter value may be any decimal number from 1 to 9999.

Output to line printer — Default value [No]

Determines if the information displayed on the screen during the course of running the diagnostic will also be echoed on the line printer port. If no printer is plugged in, or the printer plugged in fails to become on-line and ready in 10 seconds, an error message is displayed on the screen, and the operator is prompted to either try the printer again, or stop echoing.

Page breaks — Default value [No]

Determines if the diagnostic will pause and wait for a keyboard input each time the screen becomes filled with information.

Halt on error — Default value [Yes]

Determines if testing is to be halted after a non-recoverable error has occurred. If set to 'No', testing will be continued even after a non recoverable error has occurred.

Suppress error printouts — Default value [No]

Determines if the floppy controller status is to be printed each time an error occurs.

Use interrupts — Default value [Yes]

Determines if all floppy status checking should be done by using interrupts from the floppy controller, or by polling the floppy controller.

Retry count — Default value [4]

Determines the number of times a read or write operation is to be retried in the event of an error.

Filler data — Default value [55h]

Specifies the data to be written in each byte of a sector in any format or write operation.

Buffer segment address lo / hi
Buffer offset address lo / hi — Default value [varies]

The four parameters together determine the starting address of the block of memory to be used as the DMA buffer for all disk I/O. The buffer size is computed as follows :

BufferSize = BytesPerSector * SectorsPerTrack

The buffer must be word aligned, and must not cross a 64k memory boundary, due to DMA hardware limitations. If the buffer parameters entered violate these conditions, an error will be flagged, and the diagnostics will be restarted, allowing the parameters to be changed.

Max seek wait time (ms) — Default value [4000]

Determines the maximum time (in milliseconds) the diagnostic should wait for the floppy disk drive to finish any operation before declaring that an error has occurred. Parameter value may be any decimal number from 0 to 9999, however, the minimum must be at least as long as the time required for the floppy disk drive to execute a full disk seek, or false timeout errors will be flagged.

Bypass errors & continue — Default value [NO]

Determines if the testing may be continued after the detection of an error. After an error is detected and displayed, if this option is selected, the diagnostic will prompt with:

Ignore error & continue ?

Answer with Y <RETURN> if you wish the testing to continue where it left off when the error was detected. Answer with N <RETURN> if you wish the testing to be aborted and the diagnostic restarted. Parameter value may be YES or NO.

Software debug Default value [NO]

Determines if the debugger may be entered after an error is detected. After an error is detected and displayed, if this option is selected, the diagnostic will prompt with:

Debug ?

Answer with Y <RETURN> if you wish to enter the debugger at this time. Answer with N <RETURN> if not. Parameter value may be YES or NO.

The remaining parameters are very interdependent, and should be changed only by those who have a thorough knowledge of the disk controller.

Sector size code
Gap length (read/write)
Gap length (format)
First byte of Specify parameters
Second byte of Specify parameters
Data length
Bytes per sector
Sectors per track
Tracks per cylinder
Cylinders per disk
Loop on error
Halt while waiting
Ignore end of cylinder status

Error Message Format

When an error is detected by one of the diagnostic tests or functions, a brief description of the error detected is displayed, followed by the status of the floppy disk interface. The status information includes any or all of the following that are currently in effect or are abnormal:

1) The floppy controller command most recently executed. The command is displayed as a string of hex bytes.

2) The contents of the working main status register of the floppy controller, as of the most recent command.

3) The status bytes returned by the floppy controller from the most recent command.

4) The interrupt main status byte from the floppy cotroller, from the most recent interrupt.

5) The Interrupt working main status byte from the floppy controller, from the most recent interrupt.

6) The Interrupt status bytes returned by the floppy controller from the most recent interrupt.

7) The residual byte count from the last DMA transfer.

8) The number of soft errors that have occurred during the execution of the current command.

9) The number of hard errors that have occurred during the execution of the current command.

10) The total number of interrupts received during the execution of the current pass of the current test or function.

The interpretation of the bytes displayed in items 1-6 above may be found in the product data sheets for the NEC uPD765 floppy disk controller.

AWS-240 Disk Diagnostic

CAUTION

The Disk diagnostic erases all files from the disk.

Preface

This description of the disk diagnostic is not, in itself, sufficient to enable troubleshooting of a malfunctioning disk controller or drive. Refer to the Peripherals Hardware Manual for information on the AWS-240 Disk Controller. This information (especially the format of the main status register and other status registers) is vital to understanding the status information included in error messages.

This description is sufficiently self-contained, however, to permit the reader to perform an extensive battery of tests and to determine whether the Winchester disk subsystem is performing correctly or requires remedial attention.

Overview

The objective of the Disk Diagnostic is to exercise all functions of the disk controller and both winchester and floppy disk drives. In the event of a malfunction, it will identify the failing function and mode of failure as completely as possible.

In addition to testing the disk controller and drives, the diagnostic pre-tests the area of memory which it is going to use as a buffer and is prepared to detect and report non-maskable (type 2) interrupts and all varieties of extraneous interrupts. The disk diagnostic depends on the correct operation of the processor, memory, and the 8237-2 DMA controller. Unless explicitly requested otherwise, it also depends on the 8259A interrupt controller.

Operation

When first loaded, the Disk Diagnostic announces itself:

D I S K D I A G N O S T I C Rev x.y

```
C A U T I O N
All files on disk will be erased.
Drive # to test :
```

Type in the drive number to be tested followed by <RETURN>. The drive number should be Ø, 1, 2, or 3. After the drive number is entered, Winchester Disk Diagnostic will display the parameters associated with the drive, that is, number of cylinders, number of heads, and number of sectors per track. Then you are prompted:

```
Change Parameters (Y/N)?
```

If you answer N <RETURN> the standard test sequence will be run on the drive specified. Answer Y <RETURN> to select the test sequence or non-standard parameter settings.You are prompted:

```
Run standard test sequence?
```

Answer Y <RETURN> to run the test sequence (1, 2, 3, 4, 5, 6, 7, 8, and 1Ø) on the drive you selected. The tests are itemized below. Answer N <RETURN> to select your own test sequence. You are prompted with:

```
Display test menu?
```

Enter Y <RETURN> if you want to see the list of available tests listed on the screen. You are prompted:

```
test # to run:
```

Type in each test number that you wish to run followed by <RETURN>. A beep is sounded if no such test number exists. When you have specified the test sequence to run, type <RETURN>.

Test Descriptions

Test 1 - Selection

The selection test asserts basic functioning of the processor / disk interface by reading the selected drive's status.

Test 2 - Recalibrate

This test checks out the recalibrate command. The test issues a "recalibrate" command followed by a "seek" command to the last cylinder of the disk.

Test 3 - Seek Sequential

This test seeks from cylinder Ø to the last cylinder a cylinder at a time and reports any error conditions.

Test 4 - Seek Random

This test seeks to cylinders Ø, n-1, 1, n-2, ... n/2, Ø.

Test 5 - Format Disk

The media on the selected drive is formatted. Any error conditions are reported.

Test 6 - Write/Read Single Sectors

This test selects three cylinders to test: the cylinder Ø, the last cylinder, and the cylinder in the middle. Each sector on these three cylinders is written one at a time on each cylinder. The order the sectors within a track is 1, 3, 5, ... 2, 4, 6 ... Random data is used. After each cylinder is written, the data in each sector is verified.

Test 7 - Random Write/Read Multiple Sectors

This test writes to the entire disk a track at a time. Random data is used. After each track is written, the data are verified. The order of cylinders written is Ø, n-1, 1, n-2, ...

Test 8 - Sequential Write/Read Multiple Sectors

This test is the same as Test 7, execept that the order of cylinders written is Ø, 1, 2, 3, 4, 5 ...

Test 9 - Get Drive Parameters

This test issues the 'Drive configuration command' and displays the 6 bytes of drive parameters.

Test 1Ø - Write / Read Beyond Head/Cylinder Boundary

This test is similar to test 8 except that each write/read goes over a track boundary.

Test 11 - Sequential Write / Read Single Sectors

Each sector on the disk is written one at a time on each cylinder. The order the sectors within a track is 1, 3, 5, ... 2, 4, 6 ... Random Data is used. After each cylinder is written, the data in each sector is verified.

Test 12 - Invalid head & cylinder

This test issues a seek to a cylinder with an illegal head number. It then issues a seek to an invalid cylinder number. The resulting status messages are displayed on the screen.

Test 13 - Display/Modify a Sector

This test allows you to read and optionally modify a selected sector. You are prompted with:

Action(read data- Ø OR <CR>, read ID- 1)? <hex>:

Type in the function you wish to perform.

cylinder? <hex>:

Type in a hexdecimal number from Ø to the number of cylinders on the disk followed by <RETURN>. Type <GO> if you wish to exit this test. You are prompted with:

head? <hex>:

Type Ø to 3 followed by <RETURN>.

sector? <hex>:

Type a hexdecimal number from 1 to the number of sectors on a cylinder followed by <RETURN>. The first 256 bytes of the sector will be displayed. You will be prompted with:

more?

Type Y <RETURN> if you wish to see the next 256 bytes of the sector, type N <RETURN> if you do not. You will then asked if you wish to modify the sector:

modify?

Type N <RETURN> to see another sector. If you type Y <RETURN>, you are again prompted with "cylinder:", "head", "sector". Type in <RETURN> for each if you wish to write the modified contents to the same sector. You can copy the contents to another sector by changing any or all the values. If you type N <RETURN>, you will be prompted once again with 'action(read data- Ø OR <CR>, read ID- 1)'. Type Y <RETURN> if you wish to modify selected bytes in the sector. You will be prompted with:

byte :

Type a decimal byte number from Ø to 511. The current value of the byte will be displayed in hex:

xx

Type the new hexadecimal value for the byte followed by <RETURN>. Type just <RETURN> if you want do not want to modify the byte. The prompt "byte :" will continue until you type <RETURN> only. You will then be asked:

Write sector?

Type Y <RETURN> if you wish to write the sector to disk; type N <RETURN> otherwise. If Y <RETURN> is typed, you will again be prompted with "Action(read data- Ø OR <CR>, read ID- 1)". Type just GO if you

wish to end this test. If N <RETURN> is typed, this test will exit.

Test 14 - Command looping

This test allows you to prepare a loop of individual commands to the controller. This test is intended to be used primarily with special test equipment.

Test 15 - Get Diagnostics

It is not currently supported.

Test 16 - Sequential Write/Read Single Sectors II

This test is similar to test 6 except each word in a sector is written with its own word index number in the sector. This test is useful to detect any cylinder/head addressing problems with the drive.

Parameter Prompts

After the test sequence is selected, you are able to vary certain test parameters. Certain ones should be modified with great care since they can either cause the diagnostic to incorrectly report failure, or cause the diagnostic to crash. You can leave the default setting of a parameter by entering just <RETURN>, or change the value by entering a new value followed by <RETURN>. You will hear a beep if the parameter value you specified is incorrect (e.g. typing in "maybe" to a yes/no question. You now have the opportunity to change various parameters. You are prompted with:

> THE FOLLOWING PARAMETERS ARE AVAILABLE FOR CHANGE
> <RETURN> to leave the parameter unchanged,
> <GO> to begin tests.

Press <GO> when you are ready to start the tests. The parameters are:

times to run:

The default value is 1. The test sequence which you entered will run for this number of times unless an error occurs or you press <FINISH> to interrupt the test.

output to line printer:

The default is NO. Specify Y <RETURN> if you wish to monitor the test output to the line printer.

page breaks:

The default is NO. Specify Y <RETURN> if you wish to be prompted with "Press NEXT PAGE to continue" each time there is a full page of text on the video.

halt on error:

The default is NO. Any data error which is not recoverable after the specified amount of retries (see below) will not cause the test sequencing to be aborted. However the test sequence will stop if some disk controller errors are detected. Specify Y <RETURN>

to stop the test sequence even on hard (non-recoverable) data errors.

suppress error printouts:

The default is NO. Type Y <RETURN> if you do not wish to see the disk status messages on each error.

use interrupts:

The default is YES. Type N <RETURN> if you wish for all disk interaction to proceed using status checking strategies instead of interrupts.

retry count:

The default is 4. Specify a new decimal number followed by <RETURN> to indicate the number of retries to be performed on any data read/write error.

filler data -- format parameter:

The default is 39h. This parameter specifies the byte value written in each sector when the disk is formatted (test 5).

loop on error:

The default is NO. This default should be changed only by qualified hardware personnel.

buffer address high byte:
second byte:
third byte:
buffer address low byte:

The above four parameters specify the buffer address to be used for all read/write transactions. This parameter allows testing of DMA transfers to/from high memory addresses. Do not specify an odd address or an address lower than the default.

The remaining parameters are very interdependent and should be changed only by people who are have a thorough knowledge of the disk controller.

offset -- Format parameter
space -- Format parameter
bytes per sector
sectors per track
tracks per cylinder
cylinders per disk
maximum seek time
maximum I/O completion time
perform seek before read/write
halt while waiting
software debug
read diagnostics
change dma word count

Operation

Each test in the sequence will be performed on the drive specified. To abort the testing, press and hold any key on the keyboard while a test is in progress. The test stops and you are prompted with:

Press GO to continue, FINISH to terminate the test.

Press FINISH to restart the diagnostic and to select a new test sequence and/or parameters.

Incomplete Data Transfer

In tests 6, 7, 8, 10, 11, 13, 15, and 18, if a residual dma count is deteced after the data transfer, the message "Incomplete data transfer" is displayed and the test continues. This indicates a transient error which is automatically corrected by the Operating System and may therefore be safely ignored. (The value of the residual byte count can be displayed by setting the "Halt on error" parameter to "YES" and rerunning the test.)

Communications Diagnostic

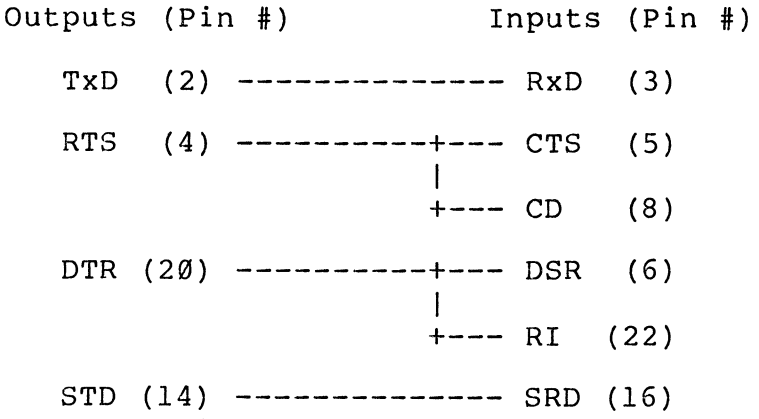
The objective of the AWS Communications Diagnostic is to test the Multi-Protocall Serial Communications Controller (MPSC) and its supporting logic, and (in the event of a malfunction) locate and identify the cause of the malfunction as completely as possible.

Both channels of the MPSC may be exercised in several different modes of operation. All combinations of operating parameters that are appropriate for each mode of operation are automatically tested. In addition, there are several parameters controlling the operation of the diagnostic program that may be modified by the operator.

The diagnostic requires that each channel tested be looped back externally. The loopback requirements are described below.

Loopback Requirements

In order for the diagnostic to operate properly, each channel to be tested must be looped back to itself externally. The following pins on the external RS-232 connectors must be connected to each other as follows:



Operation

When first loaded, the AWS Communications Diagnostic announces itself with:

```
AWS Communications Diagnostic Rev x.y
Copyright 1981 by Convergent Technologies Inc.

Change Parameters (Y/N) ?
```

If you answer N <RETURN> when the diagnostic has just been loaded, the standard test sequence will be run on both communications channels with a default set of control parameters. Answering N <RETURN> on subsequent runs of the diagnostic will use the previously selected test sequence and control parameters. If the test sequence and control parameters are to remain unchanged, the tests will begin executing immediately. Answering Y <RETURN> to this prompt at any time will allow the test sequence and control parameters to be changed before the tests are executed.

If the tests and control parameters are to be changed, the diagnostic will prompt with:

```
Run standard test sequence (Y/N) ?
```

If you answer Y <RETURN>, the standard test sequence will be selected. The standard test sequence consists of tests 1, 2, 3, 4 & 5. (See the Test Descriptions section of this document for a detailed description of each test.) If you answer N <RETURN>, the diagnostic will allow the test sequence to be changed before the tests are executed.

If the test sequence is to be changed, the diagnostic will display a menu of the available tests and prompt with:

```
Enter test # followed by RETURN,
or just RETURN to end selections.

Test # to run
```

Type in the number of the test that you wish to run, followed by <RETURN>. A beep is sounded if no such test number exists. The tests may be entered in any sequence desired, and may be repeated within the sequence. A maximum of 5 tests is allowed in the sequence. The test entering mode may be exited either by entering 5 tests, or by entering <RETURN> with no test number.

Once the test sequence has been defined, the diagnostic will afford the opportunity to change certain control parameters. (See the Control Parameters section of this document for a detailed description of each parameter.) The diagnostic will prompt with:

```
Enter new parameter value, followed by <RETURN>,
just <RETURN> to leave the current value unchanged,
or <GO> to begin tests.
```

The diagnostic will then list a parameter and its current value, and will prompt for a new value. Answering <RETURN> will keep the current value of the parameter unchanged. Answering [value]<RETURN> will make [value] the new value for the parameter. If [value] is inappropriate for the given parameter, a beep will be sounded, and you will be prompted to reenter the parameter value. After keeping or changing the value for a parameter, the diagnostic will list the next parameter for examination or modification. After the last parameter has been examined or modified, the diagnostic will start over with the first parameter. Answering <GO> whenever the diagnostic prompts for a parameter value will finish the parameter modification and start execution of the tests. It is not necessary to examine or modify all of the parameters prior to test execution. The diagnostic will use the most recent set of parameters.

As the tests are executed, the diagnostic will display and update the test being run, the current test sequence pass being run, the channel currently being tested, and the current set of test parameters being used (if appropriate).

At any time during the execution of the tests, pressing <FINISH> will abort the execution of the test in progress and restart the diagnostic. Pressing any other key (except <GO>) will suspend the execution of the test in progress. After suspending the test execution, the diagnostic will prompt with:

```
Testing suspended...
Press <GO> to resume, <FINISH> to terminate
```

Pressing <GO> will resume testing at the point where it was suspended. Pressing <FINISH> will terminate the test in progress and retart the diagnostic.

Should a malfunction occur, the error detected will be displayed, a list of probable causes will be given, and the current MPSC status will be displayed. Depending on the operating parameters selected, the testing will either be continued or aborted at this time.

Test Descriptions

Test 1 --- Static status tests

This test actually runs several sub-tests of increasing complexity to test the basic Processor/MPSC/RS-232 interfaces. The first sub-test that is run checks the interface between the Processor and the MPSC by writing to and reading from control registers within the MPSC. This sub-test determines if the Processor and the MPSC can successfully communicate to each other. The second sub-test that is run checks the interface between the MPSC and the outside world (a la RS-232). Several control lines are tested for proper loopback and function. The signals tested are:

Data Terminal Ready
Request to Send
Clear to Send
Carrier Detect
Data Set Ready
Ring Indicator
Secondary Transmit Data
Secondary Receive Data

The third and final sub-test that is run checks the same control signals as the second sub-test, but does so using interrupts between the MPSC and the Processor.

Test 2 --- Asynchronous mode tests

This test transmits and receives data from the MPSC, through the RS-232 transmitters, through the loopback, through the RS-232 receivers and back into the MPSC. The MPSC is initialized to transmit and receive data in asynchronous format. The transmit data is obtained from a transmit buffer, under interrupts, and the received data is placed into a receive buffer, also under interrupts. After all data in the transmit buffer has been transmitted & received, the buffers are compared for errors, and any found are displayed. Any other errors detected during the execution of the test are displayed when detected. All combinations of baud rate (2400, 4800, 9600 and 19200 baud), data length (5, 6, 7, and 8 bits), parity (none, odd and even parity), and stop bits (1, 1 1/2 and 2) are tested.

Test 3 --- Character sync CRC-16 tests

This test is the same as test 2 except that the MPSC is initialized to transmit and receive data in Bi-Sync format with the CRC-16 CRC polynomial. The test is run at each of several baud rates (2400, 4800, 9600 and 19200 baud). Data length is fixed at 8 bits and parity is fixed at none during the tests.

Test 4 --- Bit sync data transfer tests

This test is the same as test 2 except that the MPSC is initialized to transmit and receive data in SDLC bit sync format with the SDLC CRC polynomial. The test is run at each of several baud rates (300, 600, 1200 and 2400 baud). Data length is fixed at 8 bits and parity is fixed at none during the tests.

Test 5 --- Bit sync abort / idle line tests

This test is the same as test 2 except that the MPSC is initialized to transmit and receive data in SDLC bit sync format with the SDLC CRC polynomial. The test is run at each of several baud rates (300, 600, 1200 and 2400 baud). Data length is fixed at 8 bits and parity is fixed at none during the tests. Normal data transmission and reception is tested, as in test 4. In addition, generation and recognition of an abort transmission sequence and restarting of the transmission is tested.

Parameter Descriptions

Times to run — Default value [1]

Determines how many times the selected test sequence will run (how many passes). The test sequence will be run this many times on each channel to be tested, or until an error occurs. Parameter value may be any decimal number from 1 to 9999.

Test Channel A ... Test Channel B — Default value [YES]

Determines which channels to test. There is an individual YES/NO parameter for each of channels A & B. Parameter value may be YES or NO.

Max interrupt wait time (ms) — Default value [200]

Determines the maximum time (in milliseconds) the diagnostic should wait for an I/O interrupt before declaring that an error has occurred. Parameter value may be any decimal number from 0 to 9999, however, the minimum must be at least as long as the transmission time for a single character or false timeout errors will be detected.

Echo on line printer — Default value [NO]

Determines if all diagnostic dialogue should be copied to the parallel line printer port as well as the screen. If the dialogue is to be echoed to the printer, and if at any time the printer is not plugged in, not ready, or not on line for more than 10 seconds, the diagnostic will ask for permission to continue echoing to the printer. The echo feature may be continued or disabled at that time. Parameter value may be YES or NO.

Software debug — Default value [NO]

Determines if the debugger may be entered after an error is detected. After an error is detected and displayed, if this option is selected, the diagnostic will prompt with:

```
Debug ?
```

Answer with Y <RETURN> if you wish to enter the debugger at this time. Answer with N <RETURN> if not. Parameter value may be YES or NO.

Bypass errors & continue Default value [NO]

Determines if the testing may be continued after the detection of an error. After an error is detected and displayed, if this option is selected, the diagnostic will prompt with:

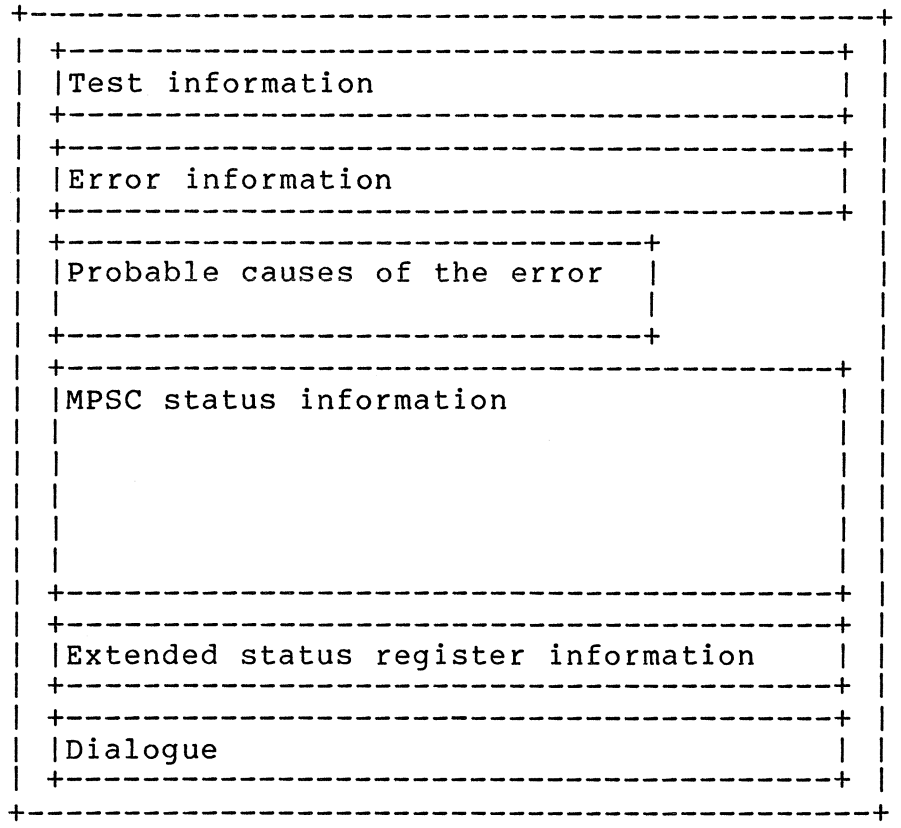
Ignore error & continue ?

Answer with Y <RETURN> if you wish the testing to continue where it left off when the error was detected. Answer with N <RETURN> if you wish the testing to be aborted and the diagnostic restarted. Parameter value may be YES or NO.

Error Screen Format

If a malfunction should occur, and the diagnostic detects an error during testing, an error information screen will be displayed. The screen contains much information pertaining to the error that was detected.

The information given on the error information screen is divided into several areas as outlined below:



Test information section

The test information section contains details of the test that was running when the error occurred. The test number and name, the channel being tested, the test sequence pass, and all test parameters at the time of the error are displayed here.

Error information section

The error information section contains details of the error condition that was detected. The actual error that was detected (e.g. >>> Receive error - Data <<<), as well as the actual and expected test results (if applicable for this error condition) are displayed (e.g. Was - 35 Should be - 34).

Probable causes section

The probable causes section lists from 1 to 3 conditions that may have caused the error to occur. The causes given are by no means the only possible things that could or would cause the error detected. They do however provide a starting point from which the cause of the error may be located.

MPSC Status section

The MPSC status information section displays the status of the MPSC channel being tested when the error occurred. In addition to the actual state of each bit in the MPSC status registers at the time of the error, the expected state of each status register bit is given (Ø, 1 or X (don't care)). A complete description of each bit in the two MPSC status registers (RRØ and RR1) may be found in the MPSC data sheets (Intel 8274 or NEC 77Ø1).

Extended status register information section

The extended status register information section displays the information contained in the Extended Communications Status Register at the time of the error. The register contains information for both communications channels, however, only the information pertaining to the channel being tested is displayed. As in the MPSC Status section of the error screen, both the actual state of the each bit in the register at the time of the error, and the expected state of each bit is displayed.

Dialogue section

The dialogue section is where any dialogue with the diagnostic program takes place. All requests for debugging, continuing or aborting the testing (see the

Operating Parameters section of this document for details) take place here.

10. TROUBLESHOOTING THE COMMIOP

The Convergent Cluster System contains 1 or 2 Communications I/O Processors (CommIOPs). The CommIOP is an independent processor which communicates with the master workstation by means of shared memory.

Any problems encountered by the CommIOP in its communications lines or its own hardware are reported to the master workstation. These are entered in the System Error Log ([sys]<sys>Log.sys). The contents of the log can be printed with the PLog Utility (see Utilities Manual).

The following is a list of messages which are logged and suggested solutions:

CommIOPs not successfully loaded.
Error reading file [sys]<sys>CommIop>sysimage.sys
The CommIOP load file cannot be found or cannot be read by the CTOS initialization. Make sure that the file exists (if not, copy it from the CommIOP Installation Diskette), and that it is un-protected (remove any password on the file).

CommIOP n not successfully started.
Initialization error xxxx.
The CommIOP could not be loaded correctly. A list of CommIOP error codes are listed below. Run the CommIOP diagnostic to verify that it is correctly functioning.

CommIOP n crashed with error xxxx.
The CommIOP crashed during operation. A list of CommIOP error codes are listed below. Run the CommIOP diagnostic to verify that it is correctly functioning.

CommIOP carrier problem for line y.
The CommIOP detected line communication problems on the specified line. Remove the cluster workstations one at a time from the line until the message **Commiop channel restart after carrier problem for line y** appears in the log.

CommIOP Error Codes

The following is a list of error codes which are returned by the CommIOP when any error conditions are detected. Run the CommIOP diagnostic to isolate the malfunction. Some error codes are not included in this list since they are only returned by CommIOP test functions and are of no other interest.

8601 2199 CommIOP time out.
The CommIOP did not return completion information within an allotted period of time.
If you get this error at initialization, the CommIOP is most probably not functioning. If you get this error after the CommIOP was started, make sure that the CommIOP code file [sys]<sys>CommIOP>sysimage.sys was not corrupted.

8602 219A Line not configured.
The communications line number if not currently configured in the system.

8603 219B Missing system image for CommIOP.
Missing or un-readable load image for CommIOP
([sys]<sys>CommIOP>SysImage.Sys).

8604 219C CommIOP loading error.
The CommIOP could not be loaded successfully.

8605 219D Invalid CommIOP data structure.
The CommIOPs queues unexpectedly overflowed. This is most probably a software problem. Analyze the CommIOP crash dump file.

8606 219E CommIOP channel restart.
The CommIOP line was restarted following a carrier problem.

8607 219F CommIOP channel hold.
A problem on the CommIOP line was discovered. All communications with the cluster workstations are discontinued until the problem is solved. This problem usually requires human intervention.

8610 21A2 CommIOP command failure.
The CommIOP returned erroneous control information to the master.

8615 21A7 Bad Master Workstation to CommIOP command.
The CommIOP did not recognize the command from the master workstation.

8616 21A8 CommIOP Boot Checksum failure.
The CommIOP checksum test failed while

loading its code file from the master workstation.

8617 21A9 CommIOP Stacker/destacker failure.
The Multibus interface hardware (stacker/destacker) on the CommIOP is not functional.

8618 21AA Bad CommIOP interrupt.
The CommIOP received an interrupt from an unknown source.

8621 21AD CommIOP RAM failure in write / read test.

8622 21AE CommIOP RAM failure. Illegal bit set.

8623 21AF CommIOP RAM failure. Illegal bit cleared.

8624 21BØ CommIOP RAM failure in addressing test.

8631 21B7 CommIOP handler time out.
The CommIOP did not get proper status information from the master workstation. The most probable cause is a software problem in the master workstation which caused the Agent Process to be permanently suspended.

8632 21B8 CommIOP invalid check word.
The 'CHECK' word in the CommIOP queues was invalid.

8633 21B9 CommIOP RAM checksum error.
The CommIOP periodically checksums its code area. This test failed during operation. Run the memory test in the CommIOP diagnostic.

8634 21BA Invalid CommIOP message.
The CommIOP received a message from the master workstation which was invalid.

8635 21BB Invalid CommIOP buffer pointer.
The CommIOP received a memory address of a buffer which is invalid.

8636 21BC CommIOP carrier problem.

8637 21BD CommIOP software inconsistency.

8641 21C1 CommIOP timer failure.
The timer hardware on the CommIOP failed the initialization tests.

8642 21C2	CommIOP DMA failure. The DMA hardware on the CommIOP failed the initialization tests.
8643 21C3	CommIOP SIO static test failure. The Communication hardware on the CommIOP failed the static initialization test.
8644 21C4	CommIOP SIO functional test failure. The Communication hardware on the CommIOP failed the functional test.

USER'S COMMENT SHEET

System Programmer's Guide
DP-100 A-09-00014-02-C

We welcome your comments and suggestions. They help us improve our manuals. Please give specific page and paragraph references whenever possible.

Does this manual provide the information you need? Is it at the right level? What other types of manuals are needed?

Is this manual written clearly? What is unclear?

Is the format of this manual convenient in arrangement, in size?

Is this manual accurate? What is inaccurate?

Name_____ Date_____
Title_____ Phone_____
Company Name/Department_____
Address_____
City_____ State_____ Zip Code_____

☐ Please check here if you'd like a reply.

Thank you.
All comments become the property of Convergent Technologies, Inc.

Seal or tape for mailing - do not use staples

fold

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1309 SANTA CLARA, CA.

POSTAGE WILL BE PAID BY -

Convergent Technologies™
2500 Augustine Drive
Santa Clara, Ca. 95051

ATTN: TECHNICAL PUBLICATIONS

fold

cut along here

Convergent Technologies
2500 Augustine Drive, Santa Clara, CA 95051 • (408) 727- 8830
Printed in U.S.A.