intel®

INTELLEC® SERIES IV OPERATING AND PROGRAMMING GUIDE

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Order Number: 121753-004

INTELLEC® SERIES IV OPERATING AND PROGRAMMING GUIDE

Order Number: 121753-004

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

CAUTION

This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A Computing Device pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Additional copies of this manual may be obtained from:

Technical Publications, M/S DV2/292
Integrated Systems Operation—South
Intel Corporation
2402 W. Beardsley Road
Phoenix, Arizona 85027

Other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties to merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used only to describe Intel products:

AEDIT	iDBP	iOSP	MULTIMODULE
BITBUS	iDIS	iPDS	Plug-A-Bubble
BXP	iLBX	iRMX	PROMPT
COMMputer	i_m	iSBC	Promware
CREDIT	iMMX	iSBX	QUEX
Data Pipeline	Insite	iSDM	QUEST
GENIUS	int_el	iSXM	Ripplemode
i	int_elBOS	Library Manager	RMX/80
$\triangle_1$	Intelevision	MCS	RUPI
iATC	$int_eligent$ Identifier	Megachassis	Seamless
I^2ICE	$int_eligent$ Programming	MICROMAINFRAME	SOLO
ICE	Intellec	MULTIBUS	SYSTEM 2000
iCS	Intellink	MULTICHANNEL	UPI

and the combination of ICE, iCS, iRMX, iSBC, iSBX, or MCS and a numerical suffix.

REV.	REVISION HISTORY	DATE
-001	Original Release	1/83
-002	Revised to support Software Release 2.5	1/84
-003	General cleanup of the manual	3/84
-004	Revised for Software Release 2.8	6/84

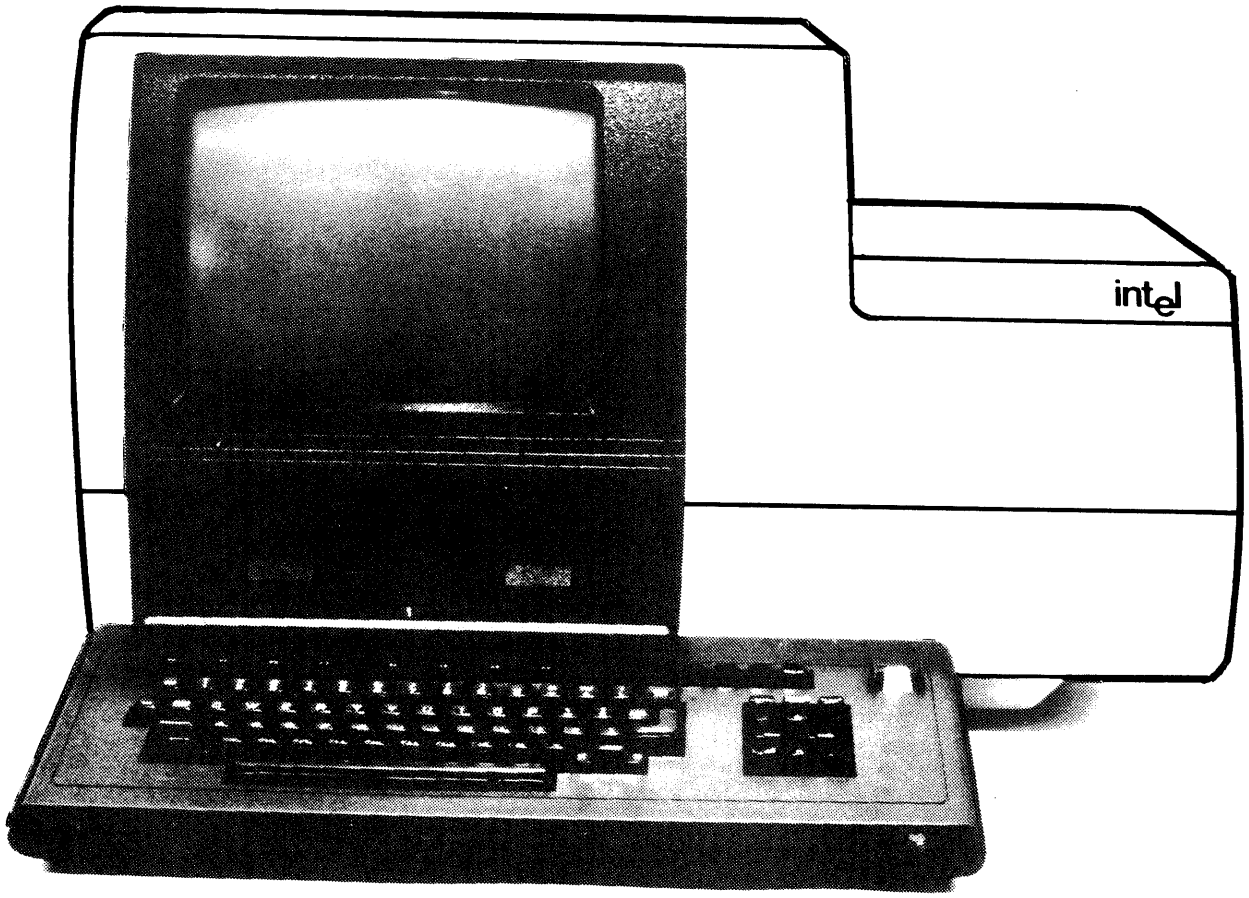
PREFACE

This manual provides operating and programming instructions for the Intellec Series IV Development System. The Series IV has two operating environments: one for 8086/8088-based software and one for 8080/8085-based software. Chapters 1-5 comprise the operations guide; 6 and 7 comprise the programmers reference manual.

This manual is designed to support new development system users as well as those who are already familiar with Intellec development systems. This manual assumes that you have read the *Intellec Series IV Microcomputer Development System Overview* and are familiar with the terms listed in its glossary.

This manual has seven chapters and six appendices:

- Chapter 1, "Introduction", introduces you to the Series IV Development System.
- Chapter 2, "Human Interface", describes the terminal, editing capabilities, device names, console operations, menu selection, the command line interpreter, command files, and job control functions.



Series IV Development System (Front View)

- Chapter 3, "Operation and Management", describes the iNDX hierarchical file structure, directory files, file access and ownership, and file system considerations.
- Chapter 4, "Command Descriptions", describes the Series IV commands and gives examples for storing, identifying, and manipulating your files and jobs.
- Chapter 5, "Network Operation", describes the NDS-II and remote job control, and operating your Series IV in the NDS-II Network.
- Chapter 6, "Programming Introduction", describes operating system considerations and target environments, and lists the built-in service routines.
- Chapter 7, "The 8086/8088-Based Environment", defines the conceptual considerations and external procedures for the Series IV service routines in an 8086/8088-based environment.
- Appendix A, "CLI Command Syntax", lists the syntax of the commands detailed in Chapter 4.
- Appendix B, "Parameters and System Service Routines", is a condensed version of the routines detailed in Chapter 7.
- Appendix C, "Error Messages and Exception Codes", lists the various error messages generated by the operating system and the UDI interface.
- Appendix D, "Object Module Relocation and Linkage", defines the possibilities for combining object modules.
- Appendix E, "Boot Device-Configuration Switch Assignments", shows the Configuration switch settings necessary to boot the operating system from various physical devices.
- Appendix F, "ASCII Codes", shows ASCII codes, their meanings, and their decimal, octal, and hexadecimal values.

Required Software

The Series IV is an Intellec Microcomputer Development System that provides support for both development and execution of programs using either the 8086/8088 chip or the 8080/8085 chip. The Series IV contains both hardware and software enhancements not available with earlier versions of Intellec development systems.

The system contains:

iNDX	—	the 8086/8088-based operating system.
ISIS-IV	—	the 8080/8085-based operating system.
AEDIT	—	an 8086/8088 screen-based text editor.
DEBUG-88	—	a low-level symbolic debugger.
MON85	—	a monitor for 8085 programs.

Related Publications

For more information on the Series IV Microcomputer Development System, see the following manuals:

- *AEDIT Text Editor User's Guide*, 121756;
- *Intellec Microcomputer Development System Overview*, 121752;
- *Series-IV ISIS-IV User's Guide*, 121880
- *DEBUG-88 User's Guide*, 121758

Notational Conventions

UPPERCASE — Characters shown in uppercase must be entered in the order shown. You may enter the characters in uppercase or lowercase.

italic — Italic indicates a meta symbol that may be replaced with an item that fulfills the rules for that symbol. The actual symbol may be any of the following:

directory-name	is that portion of a pathname, that acts as a file locator by identifying the device and/or directory containing the filename,
filename	is a valid name for the part of a pathname that names a file or is a full pathname.
pathname	is a valid designation for a file; in its entirety it consists of a volume name and/or a directory and a filename.
pathname1, *pathname2*, ...	are generic labels placed on sample listings where one or more user-specified pathnames would actually be printed.
system-id	is a generic label placed on sample listings where an operating system-dependent name would actually be printed.
V*x.y*	is a generic label placed on sample listings where the version number of the product that produced the listing would actually be printed.

[ ] — Brackets indicate optional arguments or parameters.

{ } — One and only one of the enclosed entries must be selected unless the field is also surrounded by brackets, in which case it is optional.

{ } ... — At least one of the enclosed items must be selected unless the field is also surrounded by brackets, in which case it is optional. The items may be used in any order unless otherwise noted.

| — The vertical bar separates options within brackets [ ] or braces { }.

... — Ellipses indicate that the preceding argument or parameter may be repeated.

[...] — The preceding item may be repeated, but each repetition must be separated by a comma.

punctuation — Punctuation other than ellipses, braces, and brackets must be entered as shown. For example, the punctuation shown in the following command must be entered as shown:

```
SUBMIT PLM86(PROGA,SRC,'9 SEPT 81')
```

input lines	In interactive examples, user input lines are printed in white on black to differentiate them from system output.
<cr>	Indicates a carriage return.
shading	Shading highlights the commands that can only be used if your development system is part of the NDS-II Network (see Chapter 5).

Contents

Contents (Continued)

Contents (Continued)

Tables

Tables (Continued)

Figures

CHAPTER 1
INTRODUCTION

1.1 Introduction

The Intellec Series IV Development System is an 8086/8088- and 8080/8085-based development system. The host execution environment is the iNDX operating system, which allows the Series IV to operate in the 8086/8088 mode. The ISIS-IV operating system, which runs under iNDX, provides the 8080/8085 environment. This manual provides operating and programming information on the iNDX operating system. Figure 1-1 illustrates the Series IV chassis.

1.2 The iNDX Operating System

The Intel Network Distributed Executive (iNDX) operating system resides on hard disk at the Series IV or at the Network Resource Manager (NRM), if part of the Network Development System II (NDS-II). The following paragraphs provide a general description of the operating system.

1.2.1 Human Interface

Interaction with the operating system on the Series IV is through the human interface. There are two features of the human interface immediately apparent to the user: the Command Line Interpreter (CLI), and the Syntax Builder. The Command Line Interpreter provides the keyboard and console interface for the actual entering of commands and command line console display. The Syntax Builder is a software feature that displays iNDX operating system commands,

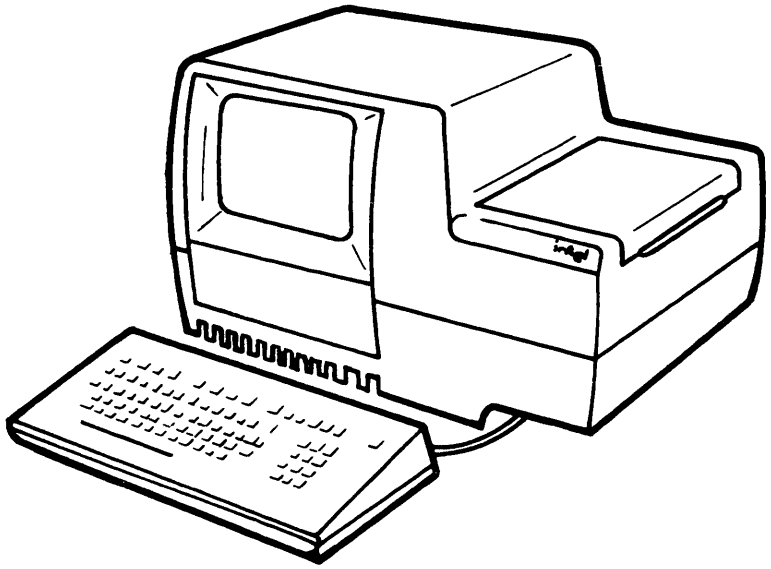


Figure 1-1. The Series IV Development System

command options, prompts, and help files designed to provide assistance in using the operating system. For a more detailed description of these two features of the human interface, refer to Chapter 2.

As seen in Figure 1-2, the human interface also consists of the distributed job control and loader. The command invoked in the CLI will have memory allocated by the Region Controller, then loaded onto system memory by the loader. The distributed job controller identifies and maintains job related information and acts as the queue manager for the Distributed Job Control (DJC) function. The Universal Development Interface (UDI) provides programmatic interface (system calls) between the user programs and the rest of the iNDX operating system.

1.2.2 Distributed File System

The Distributed File System (DFS) provides the interface between the human interface and the rudimentary parts of the operating system (Local File System, BIOS, etc.). Functionally the Distributed File System provides buffered and byte I/O, synchronous interface, file protection and device management. Included in the DFS module are:

- DFS Local Interface—maps UDI system calls and internal interface calls to DFS primitives. It also handles some services; i.e., file protection and user management.
- DFS Network Interface—maps Series IV service requests for remote resources to network protocol; i.e., network LOGON, network file I/O (NDS-II only).

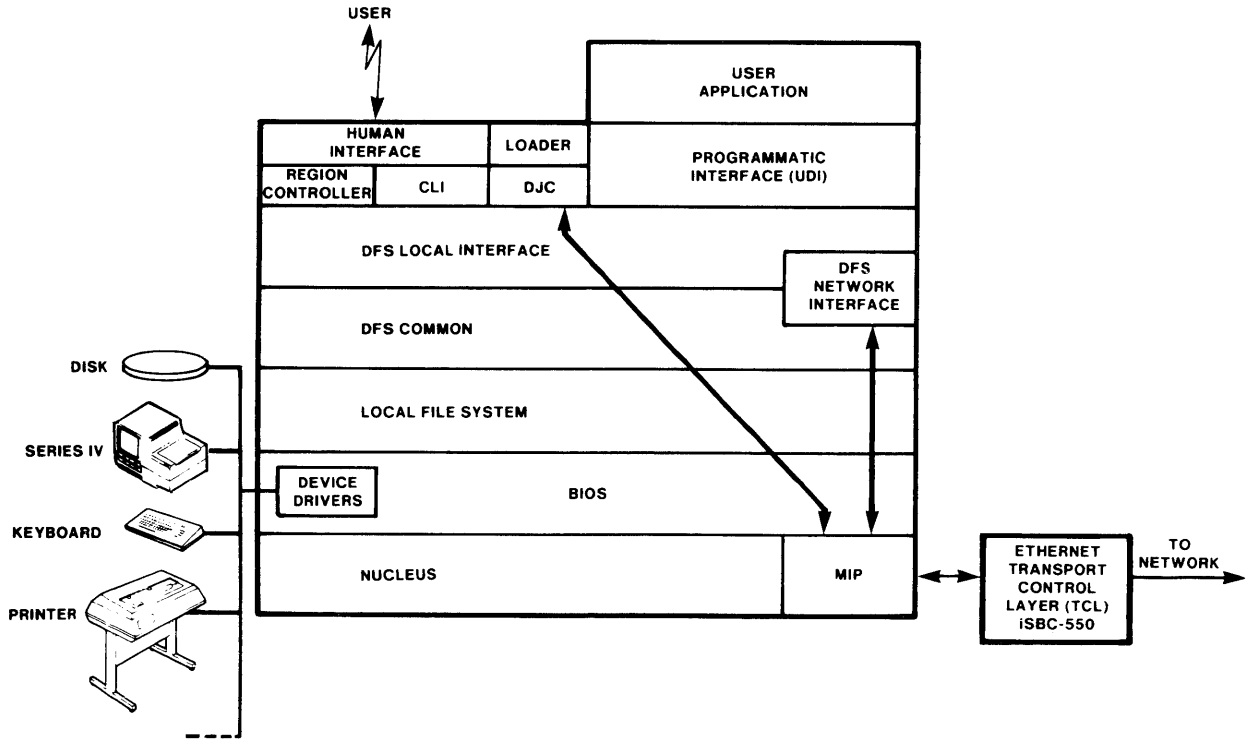


Figure 1-2. iNDX Operating System Block Diagram

- DFS Common—provides the basic distributed file system services to the higher layers of DFS; i.e., buffer management, file I/O, and local device management.

1.2.3 Local File System

The Local File System (LFS) is that portion of the iNDX operating system which provides the basic file functions. Within LFS the file names used in the upper layers of the operating system are converted to physical areas on the disk. The file structure of a disk is kept through the data structures block zero, fnode file, free space map, and bad block map. The fnode file contains essential information on every file on the disk. Pathname components are kept to maintain the directory structure of the disk. All critical system files have duplicate files to maintain device integrity.

1.2.4 Basic Input/Output System (BIOS)

The BIOS in the Series IV contains the device drivers and manager for the mass storage peripheral devices at the Series IV.

1.2.5 The Nucleus

The nucleus of the operating system provides the basic supervisory and management routines that allows the multitasking environment to execute on the Series IV. The dispatcher directs the execution of tasks and the time allowed for execution of each task. A memory manager provides memory allocation for tasks. The interrupt handler within the nucleus is the software interface for the hardware interrupts. Additionally, there is a manager that synchronizes currently executing processes and those waiting for execution. The communication between boards on the Multibus is controlled by the Multibus Interprocessor Protocol (MIP), which is also part of the Nucleus.

1.3 System Initialization

The Series IV development system may be initialized from either the flexible disk or the integrated Winchester disk drive. This section describes the initialization procedures using each device and explains user session termination and system power down.

When the development system is initialized, the power-up diagnostic tests the hardware (internal) power-up sequence.

Always power-up the Series IV before turning on any peripheral devices and turn off any peripherals before shutting down the Series IV.

If you are using a Winchester disk drive as the system device, do not shut the drive off after booting the Series IV.

Neither physically remove nor turn off the system volume after power up.

1.3.1 Booting the Series IV From the Flexible Disk Drive

If your Series IV is configured with both a single flexible disk and an integrated 5¼-inch Winchester disk drive, use the following instructions the first time you boot the operating system. Upon completion of the procedure, perform an iNDX System Build and Cusps Copy to format the Winchester disk and copy the system files from the diskette to the Winchester disk drive. Refer to Chapter 2 of the *Intellec Series IV Installation and Checkout Manual*, Order Number: 121757 for proper procedures. Thereafter, when you initialize the development system, follow the instructions in section 1.3.2.

Refer to Figure 5-1 for the location of switches on the main chassis.

1. Verify that the Configuration switches are set as follows:

   50Hz units—position 7 is ON (up), positions 1-6 and 8 are OFF (down).
   60Hz units—positions 1 and 7 are ON (up), positions 2-6 and 8, OFF (down).

2. Power up the Series IV by turning the MAIN POWER switch located at the left rear of the terminal chassis to position 1 (ON). Wait for the Power-up Diagnostics to complete (disregard any error messages). The following message should be displayed on the CRT upon successful completion of all power-up tests (approx. 40 sec.).

```
SERIES IV SYSTEM POWER-UP DIAGNOSTIC,Vx.y
CPIO PHASE I ..................................... / PASSED
CPIO PHASE II .................................... / PASSED
FLIPPY DRIVE NOT READY
SPU .............................................. / PASSED
iSBC-550 ......................................... / PASSED
CPIO/SPU MULTIBUS TEST ........................... / PASSED
IEU ..................................... / INSTALLED

SIV Boot Vx.y
Mini-floppy dr.0
-device failure
Error 3800

Attempt to reboot from : Mon88
SERIES IV CPIO MONITOR Vx.y (where x.y is the release level)
```

3. Turn on any peripheral devices, such as a Winchester peripheral chassis or a printer. When using 740 Hard Disk drives, press the STOP/START button and wait for the READY light before proceeding.

4. Insert the 5¼-inch flexible operating system diskette (iNDX.S31, No SPU; iNDX.S41 SPU) (see Figure 1-3) into the right-hand integral disk drive on the Series IV and close the disk drive latch (see Figure 1-4). The read/write access hole should face the rear of the drive and the (uncovered) write enable slot should face the left side of the disk drive.

5. Press the RESET switch (see Figure 1-5) to boot the operating system. The power-up diagnostic test results will appear on the CRT.

6. Verify that the Power-up Diagnostics execute successfully. If they did not, do not proceed with the initialization. Contact your Intel Service Representative.

7. When prompted, enter the appropriate date and time as requested. The format is MM/DD/YY for the date and HH:MM:SS for the time.

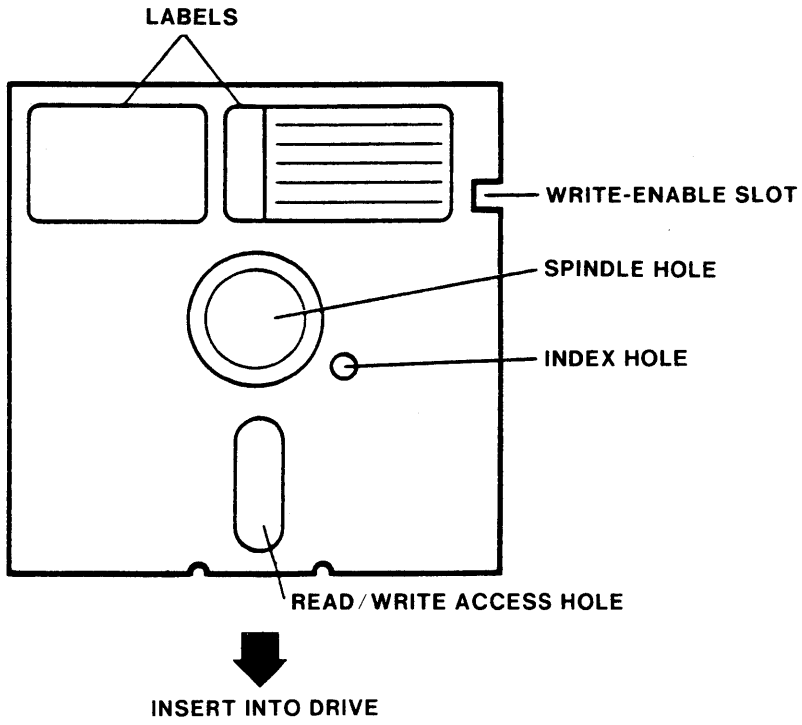
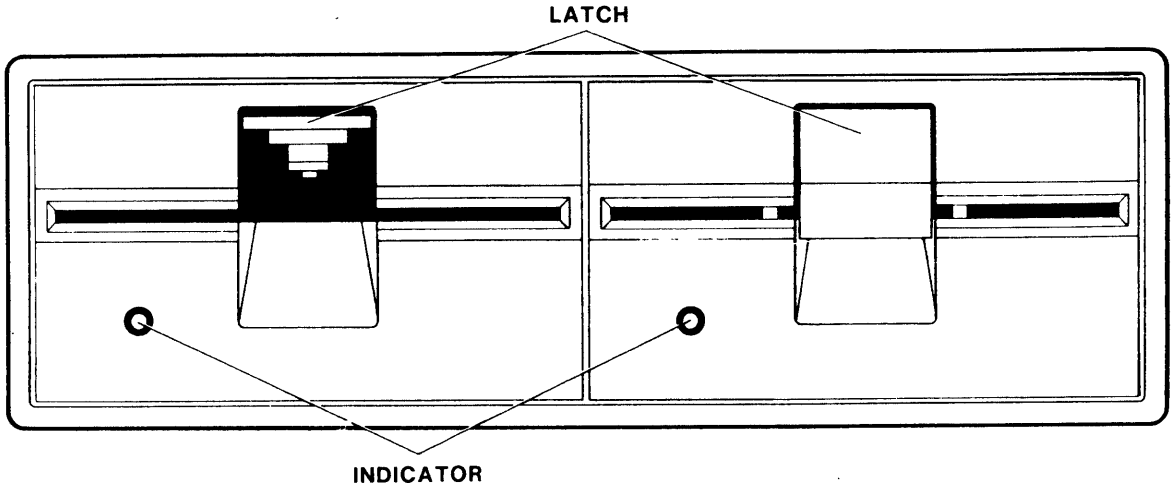


Figure 1-3. Flexible Disk



NOTE

INDICATOR(S) WILL LIGHT WHEN DRIVE IS SELECTED OR READ/WRITE OPERATIONS ARE PERFORMED.

Figure 1-4. Flexible Disk Drives

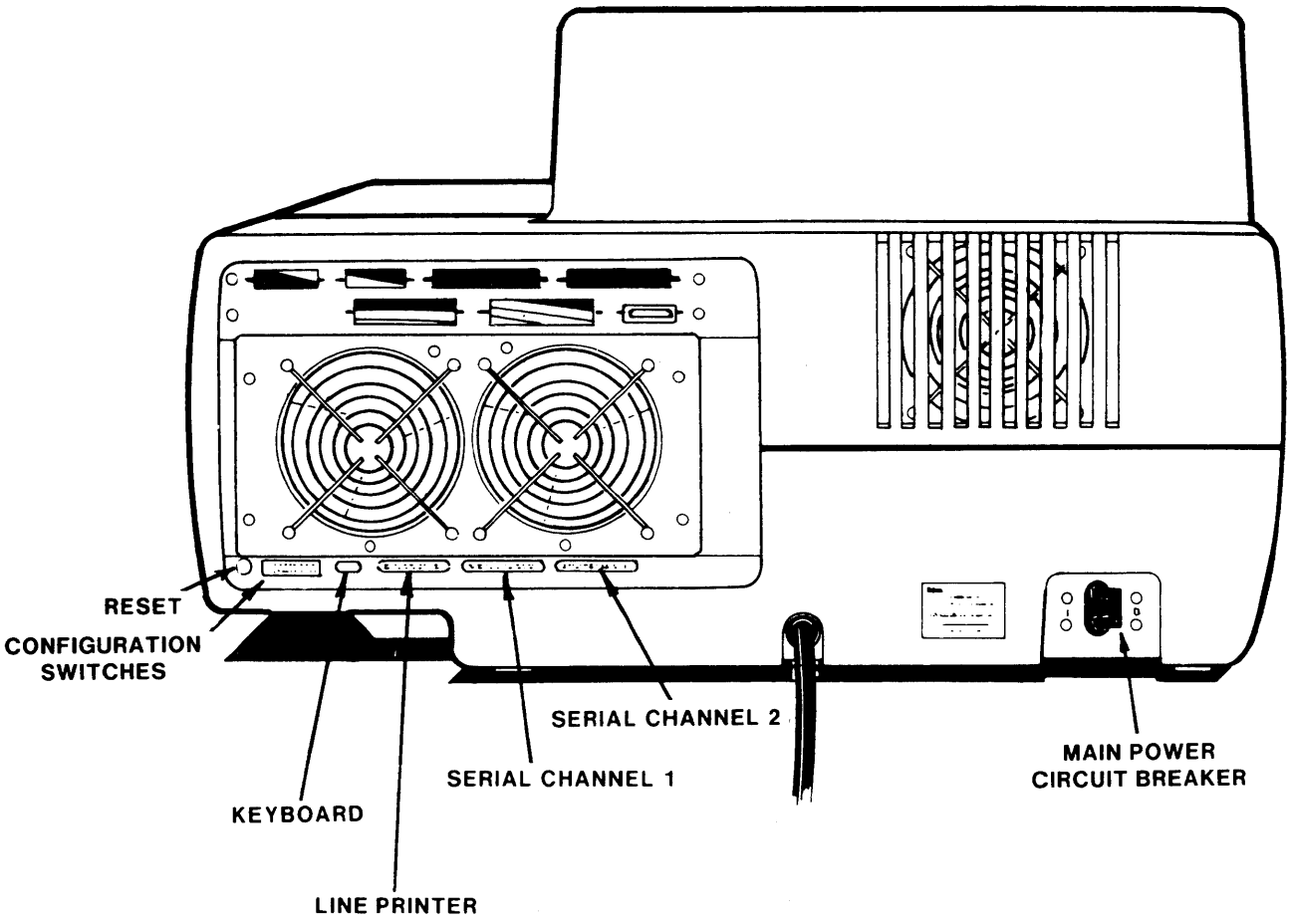


Figure 1-5. Series IV Development System (Rear View)

8. On the system keyboard, type L or F0 and enter your assigned user name. If a Winchester or Hard Disk is part of your system, also enter your password when requested. The first time you log on after installing your system, enter the user name SUPERUSER. Your associated password is PASSME. This is your system identification until you establish other users and passwords. For information on making your files and workstation accessible to others, see Section 3.5 and the USERDEF and CHPASS commands in Chapter 4.
9. Perform the iNDX System Build, Cusps Copy and ISIS build on the Winchester disk drive as instructed in Chapter 2 of the *Intellec Series IV Installation and Checkout Manual*, Order Number: 121757.

The system is now ready to accept commands. Use the facilities detailed in the *Series IV Microcomputer Development System Overview Manual*, Order Number: 121752, to assist you in learning to interface with your system.

1.3.2 Booting the Series IV From the Integrated Winchester Drive

To boot the Series IV from the integrated 5¼-inch Winchester disk drive after doing an iNDX System Build, Cusps Copy and ISIS build, follow these steps:

1. Verify that the configuration switches (see Figure 1-5) are set as follows:

   for 60Hz operation: positions 1, 5, and 7 are ON (up)
   positions 2-4, 6, and 8 are OFF (down)

   for 50Hz operation: positions 5 and 7 are ON (up)
   positions 1-4, 6, and 8 are OFF (down)

2. Power up the Series IV by turning the MAIN POWER switch located at the left rear of the terminal chassis to position 1 (ON).
3. Turn on any peripherals such as a Winchester peripheral chassis or a printer. When using 740 Hard Disk drives, press the STOP/START button and wait for the READY light before proceeding.
4. Press the RESET switch (see Figure 1-5) to boot the operating system. The Power-up Diagnostic results will appear on the CRT
5. Verify that the Power-up Diagnostic tests execute successfully. If they do not, do not proceed with the initialization. Contact your Intel Service Representative.
6. When prompted, enter the appropriate date and time as requested. The format is MM/DD/YY for the date and HH:MM:SS for the time.
7. At the system keyboard type L (logon) or F0 and enter your assigned user name, then your password.

   The first time you log on after installing your system, enter the name SUPERUSER. Your associated password is PASSME. This is your system identification until you establish other users and passwords. For information on making your files and workstation accessible to others, see "File Access and Ownership" in Chapter 3 and the USERDEF and CHPASS commands in Chapter 4.

The system is now ready to accept commands.

1.3.3 Booting the Series IV From the External Winchester Drive

To boot the Series IV from the external 8-inch Winchester disk drive after doing an iNDX System Build, Cusps Copy and ISIS Build, perform the following steps:

1. Verify that the configuration switches (see Figure 1-5) are set as follows:

   for 60Hz operation: positions 1, 6, and 7 are ON (up)
   positions 2-5, and 8 are OFF (down)

   for 50Hz operation: positions 6 and 7 are ON (up)
   positions 1-5, and 8 are OFF (down)

2. Power up the Series IV by turning the MAIN Power switch located at the left rear of the terminal chassis to position 1 (ON).
3. Turn on the External Peripheral Chassis by setting the power switch located on the front panel to position ON.
4. Turn on any other peripherals connected to the Series IV. When using 740 Hard Disk drives, press the STOP/START button and wait for the READY light to turn on before proceeding.

5. Press the Series IV RESET switch (see Figure 1-5) to boot the operating system. The power-up diagnostic results will appear on the CRT.
6. Verify that the power-up diagnostic tests execute successfully. If they do not, do not proceed with the initialization. Contact your Intel Service Representative.
7. When prompted, enter the appropriate date and time as requested. The format is MM/DD/YY for the date and HH:MM:SS for the time.
8. At the system keyboard type L (logon) or F0 and enter your assigned user name, then your password.

   The first time you log on after installing your system, enter the name SUPERUSER. Your associated password is PASSME. This is your system identification until you establish other users and passwords. For information on making your files and workstation accessible to others, see "File Access and Ownership" in Chapter 3 and the USERDEF and CHPASS commands in Chapter 4.

1.3.4 Terminating a User Session

To terminate the user session on the Series IV and allow another user to start a session, follow these steps:

1. Log off by entering the command LOGOff or by pressing the appropriate function key (see Figure 2-1).
2. The new user may now log on by entering his/her user name and a password.

NOTE

To terminate the multi-user mode enter LOGOff Exit.

1.3.5 Powering Down the Series IV

To power down the Series IV development system, follow these steps:

1. Log off by entering the command LOGOff or by pressing the appropriate function key (see Figure 2-1).
2. Turn off any peripheral devices.
3. If you are using a flexible disk(s), wait for the drive indicator light to go off (see Figure 1-4). Release the drive door latch(es) and remove the flexible disk(s).
4. Move the MAIN POWER switch located at the left rear of the terminal chassis to the 0 (OFF) position.

1.4 Disk File Types

A disk is either a system disk or non-system disk, depending on its iNDX files.

- A system disk contains the files necessary to boot the operating system.
- A non-system disk contains only the files necessary for the creation and storage of files, leaving more space for data than on a system disk.

When the system is reset with an iNDX system disk in the appropriate drive, the operating system initializes and takes control of the system.

At system start-up, only the essential iNDX files are loaded into memory. iNDX command files remain on disk until a command is entered that calls them. The required program is then loaded into memory and executed. After the command program has completed its functions, the memory it was using is again available. This allows efficient use of the operating system and memory.

The basic types of files are:

- iNDX system structure files—contain the information necessary to maintain the file system structure on each disk.
- iNDX system files—contain both the basic system programs and the command programs (data files)
- User created files (data and program).
- Directory files.

CHAPTER 2
HUMAN INTERFACE

2.1 Introduction

The human interface module of the Series IV provides a command language interpreter (CLI) that processes command input, initiates command execution and, when execution is complete, prompts for another command. A batch command processing facility is available through the SUBMIT, BACKGROUND, BATCH, and EXPORT commands to allow the execution of a series of commands stored in a file. CLI also provides line editing facilities.

The terminal session is initiated by the iNDX operating system asking the user to log on. When log on is completed, the operating system is in Command Mode (the prompt {>} is displayed). At this time, the user may enter any valid command sequence. To terminate the session, it is necessary to log off. To terminate execution of a command or submit file, use the BREAK key or CONTROL-C on the terminal keyboard.

2.2 The Intellec Terminal

The Intellec terminal is comprised of the display screen and the keyboard.

2.2.1 Display Screen

The display screen (shown in Figure 2-1) is partitioned into four display fields. Information entered in a given field never carries over into another field.

Scrolling Field: Lines 1–23 form the Scrolling Field. Each line is 80 characters wide. User-entered text will appear in the Scrolling Field. If more than 23 lines are entered, the field scrolls up one line. The topmost line disappears, additional data entries appear on Line 23.

Message Field: The Message Field occupies character positions 1–60 of line 24. Many system programs use the Message Field as a display area to avoid disturbing text that appears in the Scrolling Field.

Operating System Field: Character positions 61–80 of line 24 form the Operating System Field. The operating system uses this field to display Job Control messages, the names of mounted volumes, and the names of background and remote jobs.

Prompt Field: The Prompt Field occupies line 25. This field is used by the Syntax Guide to display Menu entries command options. A maximum of 8 entries may appear on line 25 consisting of two reverse video blocks of four entries (commands or options). Each of the entries is associated with one of the 8 Function keys.

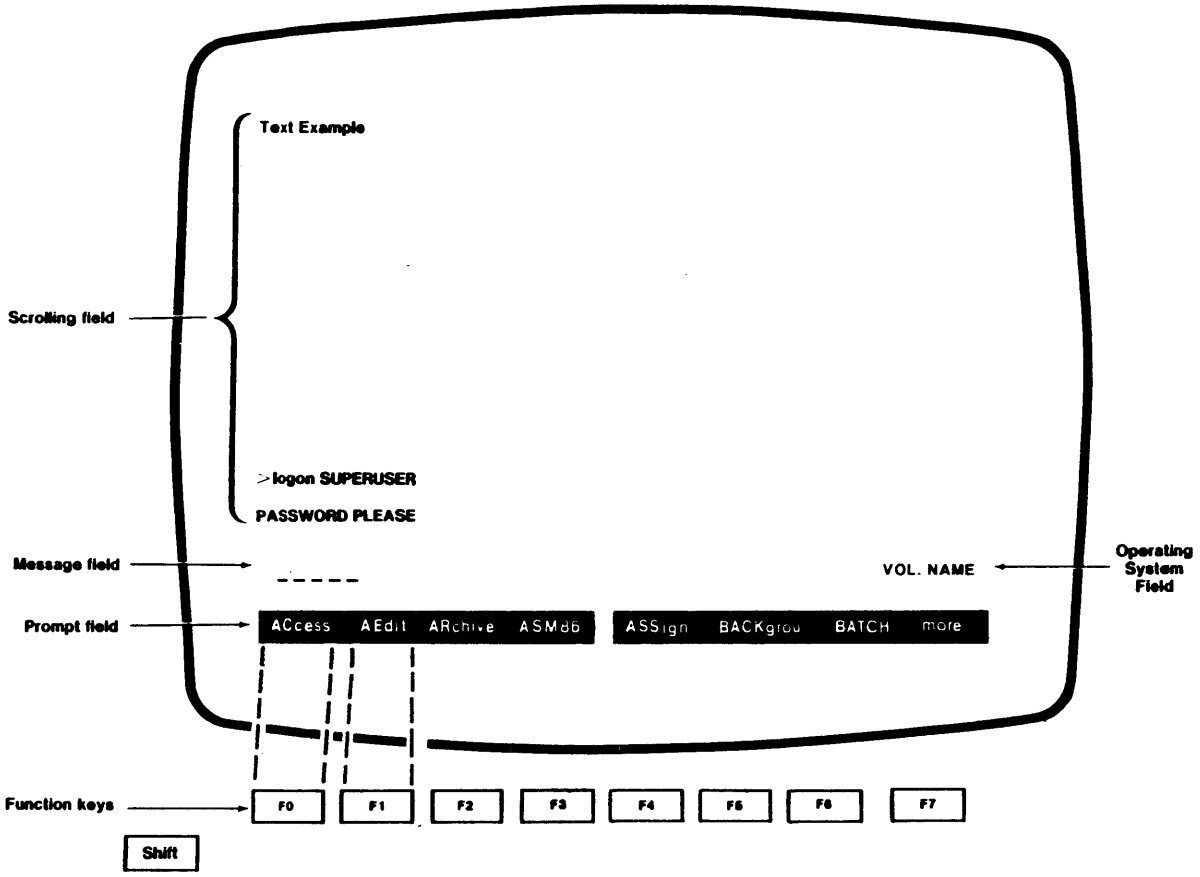


Figure 2-1. The Display Screen

2.2.2 Character Display

Characters in the display fields will appear on the screen overlined, in reverse video, blinking or highlighted.

The cursor appears on the screen as a nonblinking, reverse video rectangle. The cursor moves 1 character position to the right with each keystroke until it reaches Column 80 (the right-most column). Pressing the RETURN key causes the cursor to move to the initial (left-most) character position of the next line. If the cursor is on the bottom-most line of text (i.e., Line 23 of the Scrolling Field), the text is scrolled up one line.

2.2.3 The Keyboard

The keyboard, shown in Figure 2-2, is the user's input interface with the system. From the keyboard it is possible to control the system, enter data and commands, and request data. The data entered at the keyboard is stored in a line editing buffer until the RETURN key is depressed.

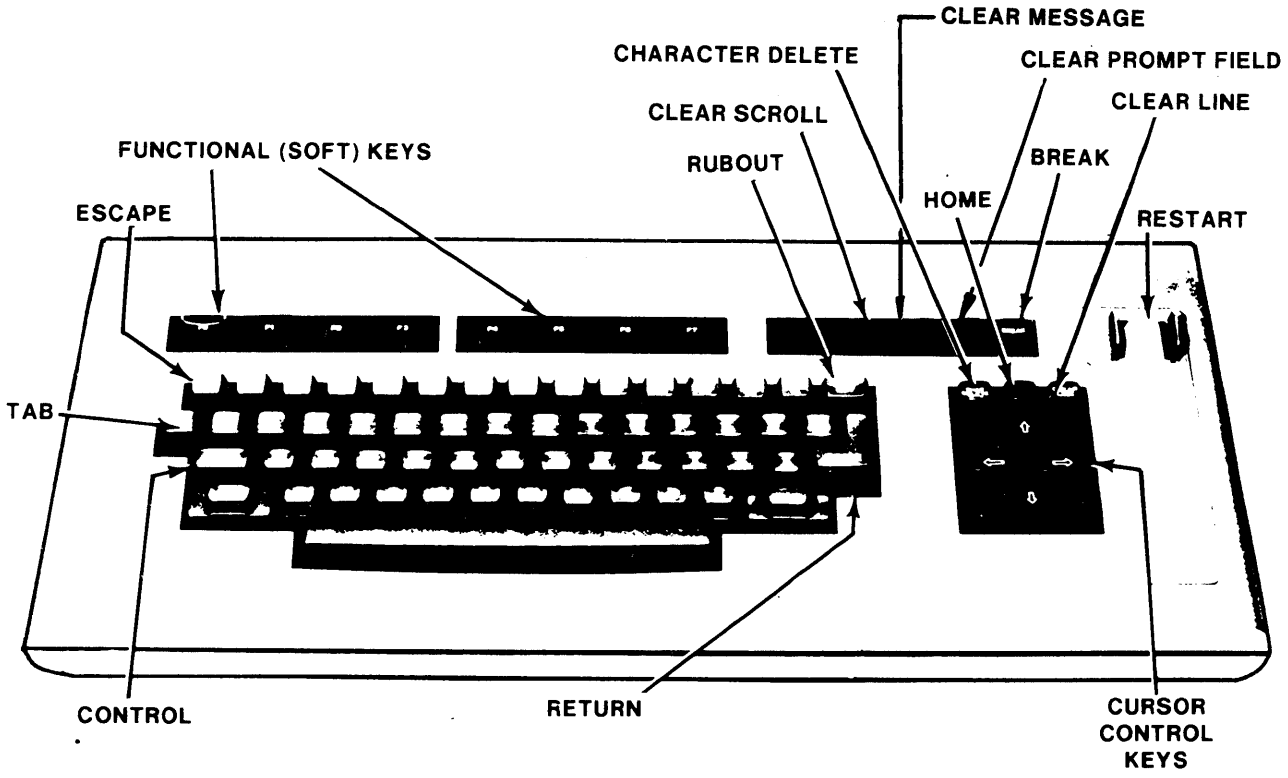


Figure 2-2. Series IV Keyboard

2.2.3.1 Key Clusters

The keyboard is organized into four clusters of keys: function keys, alphanumeric keys, reserved keys, and edit keys. Following is a description of each key cluster and several special-purpose keys.

Function Keys: When the Command Line Interpreter is executing, these keys are used in conjunction with the Menu items in the Prompt Field (line 25 of the Display Screen). Each of the eight Function keys is associated, position-by-position, with one of the Menu items. To select a given Menu item, press the key associated with that item. For example, to select the third Menu item (counting from left-to-right), press the key inscribed F2 (F0, F1, F2 being the first three keys from left-to-right). Pressing a Function key while holding down the SHIFT key produces the Help Text for the Menu entry associated with that key.

Alphanumeric Keys: Each of these keys generate the character inscribed on the keycap.

Reserved Keys: The cluster of five keys located in the upper right of the keyboard are reserved for future system use.

Edit Keys: The Edit keys consist of the keypad on the right of the keyboard and several other special keys along the right and left edges of the alphanumeric key cluster. Of the eleven keys in the right keypad cluster, only seven have inscribed keycaps. The remaining keys do not have assigned functions. The line editing keys are described in Table 2-1.

SHIFT Key: If two characters are inscribed on a given alphanumeric key, the SHIFT key must be pressed with the alphanumeric key to produce the character inscribed on the upper part of the keycap. When the SHIFT key is not used, the lower character is generated. Pressing the SHIFT key with an alphabetic key produces an uppercase character.

CAPS LOCK Key: The CAPS LOCK key functions only with the twenty-six alphabetic keys. It allows the user to enter a series of uppercase alphabetic letters without having to hold down the SHIFT key. To enter a string of uppercase characters, press the CAPS LOCK key to its lower, "locked-in" position. To generate lowercase characters again, press the CAPS LOCK key again, returning it to its upper, "unlocked" position.

BREAK Key: The BREAK key, located to the right of the reserved key cluster, aborts the execution of a job and returns the system to the interactive level (foreground).

RESTART Key: This key is reserved for use by field service personnel.

Table 2-1. Line Editor Features

Key Name	Function
RETURN	1. Terminates the line at the current cursor position. 2. Enters the command line into the system.
ESCAPE (ESC)	1. When entered as the first character in a command line, recalls the last line to the display. 2. Terminates the line at the right margin, not at the current cursor position (as with RETURN).
RUBOUT	Deletes the character to the left of the cursor and moves the cursor left one position.
CTRL X (CONTROL + X)	Deletes all characters in the current line which are to the left of the cursor. The remainder of the line is re-displayed (left-justified) with the cursor at the left margin of the line.

Table 2-1. Line Editor Features (Cont'd)

Key Name	Function
CTRL A (CONTROL + A)	Deletes all characters from the current cursor position to the end-of-line. The cursor position does not change.
DEL CHAR	Deletes the character at the cursor location. The ursor position does not change.
CLEAR LINE	Deletes the entire line. The cursor position does not change. Control remains in the line editor.
↑ (up arrow)	Moves the cursor up one line, but retains column positioning. Not functional in Command Mode.
↓ (down arrow)	Moves the cursor down one line, but retains column positioning. Not functional in Command Mode.
→ (right arrow)	Moves the cursor one position to the right but not past the current end-of-line.
← (left arrow)	Moves the cursor one position to the left but not past the starting position.
HOME	Moves the cursor position to the current end-of-line. If the last character entered was a left arrow, this key moves the cursor to the starting position.
CTRL S (CONTROL + S)	Stops output to the console.
CTRL Q (CONTROL + Q)	Resumes output to the console.

2.3 Console Operation

The iNDX operating system provides two features that help the user enter commands: the Syntax Guide and the Help Text. The Syntax Guide presents each command and its options as Menu entries. Chapter 4 provides detailed descriptions and use of the commands, and should be consulted frequently during initial system use.

The Help Text provides a functional description of a given command and its elements and options. To obtain the appropriate Help Text simply press the Function key (F0-F7) associated with that element while holding down the SHIFT key.

2.3.1 Entering Commands—Interactive Mode

The Syntax Guide displays commands and options as Menu entries in the Prompt Field (line 25 of the Display Screen). Up to eight entries, grouped into two blocks of four entries each, can be presented at one time. This arrangement emphasizes the correspondence between the Menu entries and the eight Function keys. To select a given Menu entry, either enter it from the keyboard or press the Function key spatially associated with that Menu entry. The first (left-most) Menu entry is associated with the F0 key, the second with the F1 key, and so on, to the eighth (right-most) entry, which is associated with the F7 key. Usually, the eighth Menu entry is "--more--", indicating that additional Menu entries can be displayed by pressing the F7 or TAB keys. The Menu scrolls, eventually returning to the initial display. If the keyboard entry method is used rather than the Function key method, it is necessary to know if the FILL option is operative (see the description of the FILL command in Chapter 4). If FILL is operative enter only the letters shown in uppercase in the Menu line; the system fills in

the remainder of the command automatically. If FILL is inoperative, you must enter the complete command.

The available commands and the associated function keys are as follows:

```
ACcess  AEdit    ARchive  ASM86    ASSign  Backgrou-  Batch     -more-
CAncel  CC86     CHOwner  CHPass   COPy    COUnt      CREATedir -more-
CREF86  DElete   DIR      DISmount ELse    END        ENDJob    -more-
EXIt    EXPort   FIll     FORMat   FORT86  I2ice      ICopy     -more-
IF      IMport   ISis     LIB86    LINK86  LName      LOC86     -more-
LOG     LOGOff   MAIl     MAke     MOunt   OH86       OPen      -more-
ORif    PAscal86 PDscopy  PLM86    PScope  Queue      REAd      -more-
REGion  RELab    REName   REPeat   RUn     S4fprt     SDcopy    -more-
SEArch  SET      SPace    STty     SUbmit  SVcs       SYSGen    -more-
SYSTat  Time     UNtil    USERDef  USERS   VErify     View      -more-
WAit    WHile                                                  -more-
```

F0 F1 F2 F3 F4 F5 F6 F7

For each Menu entry, the accompanying Help Text can be obtained by holding down the SHIFT key while pressing the appropriate Function Key. For example, consider the following eight Menu entries:

```
CREF86  DElete  DIR  DISmount  ELSE  END  ENDJob  -more-
```

F0 F1 F2 F3 F4 F5 F6 F7

To select the DELETE command, press the F2 key or, assuming for the example that FILL is operative, enter the characters DE. (If FILL was inoperative, enter DELETE). To obtain Help (more information) about the DELETE command, press the SHIFT and F2 keys simultaneously.

Once DELETE has been selected from the Menu line (either by pressing the Function key or by entering the command from the keyboard), the Menu line will no longer display the original eight entries. Instead, the Menu line will now contain the following message:

```
ENTER <file name>
```

where *file name* is the file to be deleted. Suppose an entry request is not clearly understood. Since the original Menu line is not on the screen, Help cannot be obtained by pressing SHIFT and a Function key. The Syntax Guide allows the user to "back up", thus recalling the previous Menu line. To back up, press the cursor left (←) key. The original Menu line will now reappear. To obtain more information about the DELETE command, press the Function key associated with the DELETE Menu entry while holding down the SHIFT key.

After reading the Help Text, return to the request for a file name by pressing the cursor right (→) key. As long as valid characters are entered, the Menu line will remain the same. The file name is terminated with a delimiter by pressing the space bar. Note that the Menu line has changed again and is now listing the options for the DELETE command, as shown below:

```
DIR QUERY --exec--
```

Select either (or both) of the options by pressing the Function key associated with them (i.e., F0 and F1), or by entering D or Q at the keyboard (DIR or

QUERY if FILL is inoperative). If more information about the two options is desired, press both the SHIFT key and Function key associated with the desired option. If a command line option is selected, but more option information is desired, use the left arrow key (◄-) to move the cursor back to the 1st letter of the option. The function key and shift can then be used to display the help files for command option.

After entering the entire command line, execute the command by pressing the Function key associated with the "--exec--" entry or by pressing the RETURN or ESCAPE key.

2.4 Command Line Interpreter

The Command Line Interpreter (CLI) is a program with direct user interaction. The CLI requests input with the > prompt, accepts a complete command from the input device, substitutes the CLI variables with the defined values, loads, and invokes the requested program. A complete command may require several input lines, each with a maximum of 128 characters. Each successive input line then, except the last, must end with the continuation character, an ampersand. When the CLI detects the ampersand, it issues a continuation prompt (> >) and allows continuation of the command entry on the next physical line. Only when a complete command line has been entered (with a carriage return <cr>) is the program loaded and invoked. If comments are to be included in a command line, use a semicolon (;). All characters between the semicolon and the return are recognized as a comment and are ignored in execution.

2.5 Command Delimiters

A command consists of a sequence of characters. When a command is examined by the system prior to execution, special characters called delimiters are used to divide the command into words that are treated as units. The following characters function as delimiters:

!	<
#	=
$	>
%	[
(	\
)	]
+	‘
,	|
-	space
<cr>	

A delimiter can be used within a string by enclosing the string within quotation marks.

Pathnames must not be broken by an ampersand. If a pathname is entered as part of a command and the pathname contains a delimiter, enclose the pathname in quotation marks so the system will treat it as a unit. For example, if the pathname /VOL1.A/A!B is entered as part of a command, enclose it within quotes:

COPY "VOL1.A/A!B" TO /VOL2.B

2.6 Command Line Input

The terminal supports a "type ahead" operation. This means that up to thirty-two keystrokes may be saved in the buffer until the system is ready for them. When the buffer is full, the system issues an audible beep; additional characters are discarded.

The command line is not complete until a RETURN or ESC key is used to terminate the line. During line input, the cursor (a reverse video block) indicates the position of the next character. In line edit, the cursor does not move off the current line or to the left of the starting position (the third character position). If more characters than can be displayed on the current screen line are entered, an exclamation mark (!) and the cursor will appear in column eighty.

2.7 CLI Variables

CLI variables are symbols that have string values associated with them. The command language allows these variables to be defined and referenced within a command file. The scope of CLI variables is restricted to the command file in which the CLI variables are defined. The CLI variable name can be a maximum of six alphanumeric characters. The first character of the name must be alphabetic. Letter case is not significant. The value associated with a CLI variable is a string with a maximum of five hundred and eight ASCII characters. A reference to a CLI variable consists of a percent sign (%) followed by the name of the CLI variable.

The CLI must be able to distinguish the variable reference from the surrounding text. One of the following techniques can be used:

1. Follow the variable name with a delimiter character.
2. Enclose the variable reference in matching quotes. The quotes are removed if the program obtains its arguments through the DQGETARGUMENT$ function (see Chapter 7).
3. Make the variable name a full six characters long. The CLI stops looking if a delimiter has not been encountered within six characters.

When a reference to a CLI variable is encountered in a command line read by CLI, the interpreter replaces the reference with the current value of the CLI variable before command execution. If the referenced CLI variable is undefined, the reference becomes a null string. Examples of CLI variable use appear in Figure 2-3.

There are two different types of CLI variables: system-defined and user-defined. No more that ten CLI variables can be defined within one command file, including both system-defined and user-defined variables.

2.7.1 System-Defined CLI Variable

One system-defined CLI variable is provided in the Series IV: 'STATUS'. Any reference to it takes the form '%STATUS'. At any given point, the value of STATUS represents the completion code returned by the last DQ$EXIT call executed (see Chapter 7). The completion code is converted to a string of ASCII decimal digits, thereby expressing the value of the completion code in decimal notation (leading zeros are suppressed). Control structures, together with the

```
Command File (compil.csd)
      OPEN FILES.NRM              ; get parameter file
      SET LINKEM TO ' '                 ; initialize
                                        ; string to
                                        ; hold link file
                                        ; name
      REPEAT
           READ NAME                    ; get module name
           IF %STATUS = 0 THEN
           WHILE %NAME < > ' '          ; exit loop at end of
                                        ; parameter file
           ;
           ;  compile one module
           ;
           PLM86 %NAME.P86 DEBUG COMPACT
           ;
           ;  build string of file names for link step
           ;
           IF %LINKEM = ' '
                SET LINKEM TO '%NAME.OBJ'
           ELSE
                SET LINKEM TO '%LINKEM,%NAME.OBJ'
           END
           END
      END
      ;
      LINK86 %LINKEM, :F0:SYSTEM.LIB TO DRIVER BIND
Parameter File (files.nrm)
      driver, mod1, mod2
Invocation Line
SUBMIT compil
```

Figure 2-3. Command File Example

STATUS variable, make possible the conditional execution of subsequent steps in a command file.

2.7.2 User-Defined CLI Variables

CLI variables may also be created by the user by means of the SET and READ commands (see Chapter 4).

2.7.3 Commands for Manipulating CLI Variables

The SET command is an assignment statement for CLI variables. If the receiving variable does not exist, SET creates it. The string concatenation of any combination of literal strings and the values of CLI variables can be assigned to a CLI variable (either system-defined or user-defined), as shown in the following example.

Assuming FILE and EXIT are user-defined CLI variables with the following values:

%FILE="aprog"
%EXT="obj"

The statement:

```
SET NAME TO ''%FILE.%EXT''
```

results in:

%NAME="aprog.obj"

In this example, the quotes surrounding the values of the CLI variables are not part of the values, but are included to separate the string values from the surrounding text.

As CLI processes a command line, the following sequence occurs: (1) the line is read, (2) the line is scanned for references to CLI variables and all substitutions are performed, (3) the line is parsed as a command. The literal quotes around the object following the TO are necessary if the values of the CLI variables being substituted contain delimiters.

2.7.4 Examples of CLI Variable Substitution

Assume the following CLI variables have been defined:

%FILE="mod"
%NUMBER="1"
%EXT = "p86"

Consider the following command lines before substitution, after substitution, and at execution (i.e., the command tail as it appears when retrieved by the DQGETARGUMENT UDI primitive).

```
before:   plm86 %FILE%NUMBER.%EXT
subst:    plm86 mod1.p86
exec:     plm86 mod1.p86

before:   plm86 "%FILE"A%NUMBER.%EXT
subst:    plm86 "mod"A1.p86
exec:     plm86 modA1.p86
```

The quotes are necessary because the CLI would interpret "%FILEA-%NUMBER" as a command line containing a reference to a CLI variable called "FILEA".

```
before:   plm86 %FILE%NUMBERA.%EXT
subst:    plm86 mod1A.p86
exec:     plm86 mod1A.p86
```

Quotes are not necessary in this example because the CLI knows that variable names are a maximum of six characters. CLI assumes the second variable reference is to "NUMBER" rather than to "NUMBERA".

If a literal percent sign needs to be included in a command line it can be surrounded by matching quotes:

```
before:   a"%"FILE.obj
subst:    a"%"FILE.obj
exec:     a%FILE.obj
```

2.8 Command Files

Commands are normally entered from the console. However, permanent files that contain lists of commands can be created and maintained. These permanent files are called command files. Command files can be executed in the foreground of the system by using the SUBMIT command (see Chapter 4), in the background of the system by using the BACKGROUND command (see Chapter 4), or at a remote workstation (NDS-II only) by using the EXPORT command (see Chapter 5).

Refer to Figure 2-3 for an example of command file usage. In addition to the normal console commands, the Series IV has control commands that provide, at run time, conditional or repetitive execution of a set of commands within a command file. CLI variables may also be defined within the command file.

2.8.1 Dynamic Command File Creation

When a command is received by the CLI from the keyboard, the syntax builder is invoked. Prompts are always displayed, although you can disable command keyword completion (refer to the FILL command in Chapter 4). If you attempt to create a command file with a standard text editor, the syntax builder prompts will not occur. If you were dependent on these, you might have to check a manual or the corresponding reference card to create a command file. To avoid this inconvenience, CLI provides the BATCH command. The BATCH command uses the syntax builder to create, write, and execute a command file. Another advantage of the BATCH command is that, since it is part of the CLI, less memory is required to invoke it than is required to invoke a separate text editor. When the BATCH command is invoked, only the keyword BATCH and the pathname of the command file is specified.

If the command file already exists, the syntax builder's editor can be used to alter the contents of the command file. If the file does not exist, it is created and then written at the end of the edit process. When the command file is complete, the BATCH command prompts the user to select one of the following options: abort, write it without executing it, execute it in the foreground or background, or export it to a remote station for execution. BATCH allows the user to write and execute the file, or execute it only (a temporary file is created and then deleted after execution). If the executed file contains references to formal parameters, BATCH prompts for the actual parameter values to be used. Refer to Chapter 4 for a detailed description of the syntax and operation of the BATCH command.

2.9 Log Files

The Human Interface provides a log facility that allows the user to specify that output written to the logical console output device should also be written to a

mass-storage file. In a foreground job, normal console output is displayed on the physical screen. If a log file is active, the same output is also written to the log file. In a background job, the normal console output goes to the Byte Bucket (see Section 2.12). A log file can be requested in either of two ways: by executing the LOG command, or by specifying the LOG option of the SUBMIT, BACKGROUND, and EXPORT commands with the LOG option (default). (The LOG command is valid only from the keyboard. See Chapter 4 for a description of the LOG command.)

The log file is the only attribute of a command file environment which is inherited by nested command files. If a log file is active when a SUBMIT command is issued, the console output from the newly submitted command file is also written to the currently active log file unless the LOG keyword is specified on the SUBMIT command. If a log file is active when a SUBMIT command with the LOG keyword is issued, the current log file is detached before the new log file is created. When the inner nested command file has finished executing, its log file is detached, the log file of the outer command environment is reattached, and the file pointer is positioned at the end-of-file. The log file of the outer command environment (either the keyboard or another command file) resumes with the next commmand after the SUBMIT (with LOG keyword).

2.10 Parameter Substitution

Actual parameters can be specifed when a command file is submitted for execution. The command file can have formal parameters of the form %n, where n is a decimal digit (0-9). At submit time, a list of actual parameters is supplied along with the name of the command file to be executed. Before the command file is executed, the CLI creates a new copy of the command file where all occurrences of formal parameters have been replaced by the corresponding actual parameters. The formal parameter, %n, is replaced by the (n + 1)st element of the list of actual parameters (%0 is replaced by the first list element, etc.). The parameter replacement is done by scanning each line in the command file once from left-to-right. Every occurrence of the string, %n, is replaced by the corresponding actual parameter in a string substitution. If the actual parameter is enclosed in quotes, the quotes are removed before the substitution is performed. If a formal parameter has no corresponding actual parameter, the replacement is performed using a null string.

2.10.1 Parameter Files

To increase the power of command files as utility tools, the command language has commands that allow the values of CLI variables to be read from mass-storage files. These files, referred to as parameter files, are manipulated using the OPEN and READ commands. Only one parameter file can be open at any given time. The OPEN statement allows any pathname to be specified as a parameter file. The READ command treats the parameter file as a byte stream subdivided into strings by delimiters, and specifies a list of CLI variable names. The READ command proceeds from the current file pointer position in the parameter file and assigns a string to each variable in the list. If end-of-file is detected and the STATUS variable is set to a non-zero value on the parameter file during a READ, the parameter file is closed. If the list contains more variables than the number of strings in the file, the variables without corresponding strings are set to null.

Suppose file "files.nrm" contains the following two logical lines:

drive, mod1
mod2, mod3

The execution of the command file fragment

```
OPEN FILES.NRM
READ FILE1, FILE2, FILE3
```

results in values of the CLI variables FILE1, FILE2, and FILE3:

%FILE1 = 'drive'
%FILE2 = 'mod1'
%FILE3 = 'mod2'

The quotes are not part of the values of the variables. If a subsequent READ command is issued, the string retrieved is 'mod3'. Thus, general purpose utility command files that can process groups of related modules, whose names are specified by parameter files, can be constructed. An example for such a command file is given in Figure 2-3.

2.11 System-Designated Device Names

The following device names are defined by the operating system:

:T1:	Serial channel #1 input
:TO:	Serial channel #1 output
:LP:	Line printer (local)
:SP:	Spool printer
:CI:	Console input (typically, Series-IV keyboard in foreground)
:CO:	Console output (typically, Series-IV display in foreground)
:BB:	Byte Bucket Though nonexistent, the byte bucket is treated as a real device by the commands. The byte bucket receives data you want to discard. Writing to :BB:, always successful, simply discards data. Reading from :BB: returns an end-of-file (i.e., a zero byte read).
FL0, FL1	Flexible disks
WM0 – WM3	Integrated 5¼-inch Winchester disk
WD0 – WD3	Winchester 35 Mb disk
HD0 – HD3	HD 5440 hard disks

2.12 Modes of Operation

The iNDX operating system has three modes of operation that allow varying degrees of interaction to the multi-tasking capabilities. There are two regions or memory partitions available where independent tasks are executed. The default mode is single-user which maintains a background region that may be used to process non-interactive batch jobs. When a second terminal is attached to serial channel 1, the Series IV may be operated in multi-user mode. This mode maintains two entirely independent regions—one for the Series IV console and one for the attached terminal. The third mode is known as toggle mode, and allows a single user to access and utilize the two regions interactively.

2.12.1 Single User Mode

In single-user mode, the foreground region is used and accessed by the user interactively and a second or background region is used to execute non-interactive command files with the Background command. A background job runs in the batch mode in the background concurrently with the foreground job. Syntactically, the BACKGROUND command is similar to the SUBMIT command: you specify the name of a file that contains a sequence of commands to be executed. When the BACKGROUND command is executed, job control creates the background job environment, and logs on as a background job. When the command file has been exhausted, job control logs off and deletes the background job. Once created, a background job has no relationship to the foreground job from which it was created; nor does it inherit any environmental information (e.g., logical names or CLI variables) from the foreground job, but does take environmental information from the INIT file.

The background job has no physical console attached to it. Thus, certain programs cannot be executed in the background and certain primitives are disallowed. A screen-oriented editor or debugger should not be invoked from the background. If a program running in the background calls DQ$TRAP$CC (see Chapter 7), the call returns the message, EXCEPT = E$OK, but no action is taken. If CONTROL-C is typed, the CONTROL-C handler of the foreground job is always invoked. A background job ignores a CONTROL-C request but the BREAK key can be used to terminate a background job.

A useful feature of the single-user mode is that with the STTY REMOTE option, the Series IV may be controlled with a second or remote terminal. The primary console is inactivated at this time. This feature may be useful to the developer with a terminal at home.

Another feature that may be used in the single-user mode is STTY TERMINAL. This option allows the Series IV to act as a terminal that may be connected to a host computer. The interface is programmable, using the configuration file as documented in Appendix D.

2.12.2 Multi-user Mode

To use the multi-user mode, a terminal must be attached to serial channel 1 and the terminal characteristics set using a configuration file (refer to Appendix D). The configuration file is incorporated into the operating system using either the SYSGEN command (where the file is incorporated upon boot up) or the STTY command. Setting the system to multi-user mode can either be done in SYSGEN or with the REGION command.

Once the system is set up in the multi-user mode, the Series IV console and the terminal processes are independent from each other. However, the Background command cannot be used and only one user (configured in SYSGEN or REGION commands) may execute ISIS-IV. To exit the Multi-user mode, the second user must use the LOGOff exit option, then the REGION or SYSGEN commands must be used to reconfigure system mode.

2.12.3 Toggle Mode

The third mode of operation is the toggle mode, which may be established using either the SYSGEN or REGION commands. This mode is characterized by the

ability of the Series IV to run two "foreground" processes at once. The key to the right of the F7 function key is the toggle key. This key determines which partition is interactive (displays on the console and accepts keyboard input). A message in the OS Field tells the user which partition is interactive (P1 or P2).

There are two options in toggle mode operation (Selected in SYSGEN or REGION). The first option allows the user to specify whether or not the screen is refreshed on every toggle. Although the option requires 4k of memory to be used, it is quite useful if performing screen oriented tasks in both regions. The second option allows the user to stop (control S) the console display when exiting a region and activate the console (control Q) when re-entering the region. An example of where this option would be useful would be in scrolling through a text file, while toggling to another process. Whenever the text file scroll is exited the screen freezes, preventing the user from missing any text that scrolls while the process is not being displayed.

CHAPTER 3
OPERATION AND MANAGEMENT

3.1 Introduction

The iNDX Distributed File System offers a hierarchical file structure that provides:

- multiple user access to shared data and directory files
- owner-controlled access (World and Owner access to the files on mass storage devices)
- a list of files that reside in the directories
- the ability to create new directory and data files while other users are accessing shared files
- flexibility in file maintenance
- an archiving facility for files stored on the shared disk

In previous versions of file systems, files and collections of files were tied to the media (disk) and the disk drive (physical device) where the files were stored. Thus, the terms disk, directory, directory identifier, logical device name, physical device name, and disk drive were functionally identical and could be used interchangeably.

The additional functionality of the iNDX structure requires redefinition of these terms. In this manual, the terms are used as follows:

- Disk—the media where directories or files can be stored.
- Directory—a logical collection of files stored on a disk.
- Directory File—a file that stores information about a directory.
- Directory Name—a user specified label for a directory (WORK.DIR).
- Directory Identifier—a logical name or fully qualified pathname used to access a directory.
- Disk Drive—a machine used to access files stored on a disk.
- Physical Device Name—a label assigned to a physical device (e.g., line printer is :LP:, flexible disk drive 1 is FL1, 8-inch Winchester disk 1 is WD1).
- Logical Device Name—a label (:BB:, :CO:, :SP:) assigned to a logical device (byte bucket, console, spooler queue).

3.2 iNDX File Structure

The iNDX file system is structured hierarchically. This structure resembles an inverted tree (see Figure 3-1).

The root or origin of the file system is called the Logical System Root (in the first tier of Figure 3-1). It "connects" the volumes within the file system (shown as the second tier). Each volume corresponds on a one-to-one basis to a physical mass storage device, with each volume name called the "volume root". Thus, in the figure, VOL 1.A could be a flexible disk, VOL 2.B a Winchester disk, and

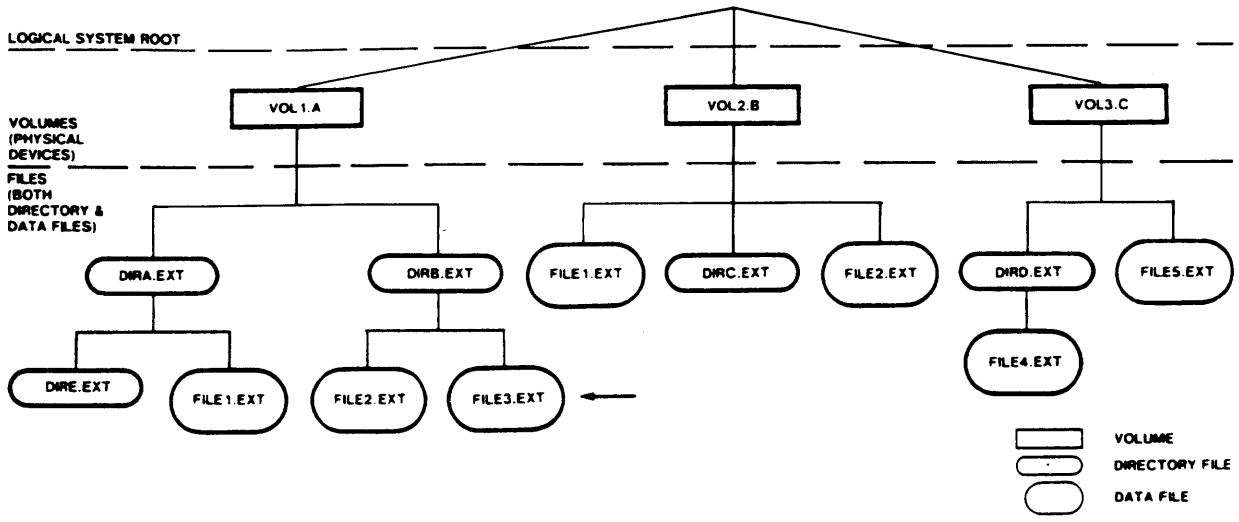


Figure 3-1. Hierarchical File Structure

VOL 3.C a hard disk. Each volume is further divided into files (shown as the third tier). Each file may be either a data file or a directory file. Data files contain only data; directory files may contain references to both data files and additional directory files.

3.2.1 Pathnames

The files (data and directory) can be traced down through the file structure by a pathname. The pathname identifies every directory from the logical system root to the data file. The pathname for the file indicated by the arrow in Figure 3-1 is /VOL1.A/DIRB.EXT/FILE3.EXT.

The pathname /VOL1.A/DIRB.EXT/FILE3.EXT is a fully qualified pathname because the slash (/) acts as a delimiter between the names of the volume and the various directories in the hierarchical path.

To directly access FILE3.EXT in Figure 3-1, the user may assign a directory identifier (X) to the fully qualified pathname of the directory with the LNAME command, as shown in the following example:

```
LNAME DEFINE X FOR /VOL1.A/DIRB.EXT
```

The pathname for this file is now X/FILE3.EXT.

For details and restrictions of the LNAME command, see Chapter 4.

3.2.2 Wildcard Filenames

The COPY, DELETE, ACCESS, DIR, CHOWNER, and ARCHIVE commands permit the use of a wildcard element to replace one or more characters in the last component of a filename. The wildcard elements can appear only in the last component of the filename.

The two wildcard elements that may be used are the asterisk (*), which matches any number of characters in the final path component, and the question mark (?), which matches any single character in the final path component.

The following conventions apply in using wildcards:

*	Matches any filename
name*	Matches filenames with name, with or without extensions.

Thus, the wildcard pathname /FAT* will match the files /FATCAT, /FATDOG, /FATCITY, etc.

The wildcard pathname /FAT?AT will match the filenames /FATCAT, /FATHAT, /FATBAT, but not /FATXHAT or /FAT.HAT.

More than one asterisk may appear in wildcard filenames. Thus, /*FAT* matches all filenames having the character string FAT appearing between any two other character strings. /AFATCAT and /INFATCITY would be matched.

The other possible combinations of the two wildcard characters such as *B?, ?B* and ?B? are also acceptable.

The * wildcard matches all files within a directory.

3.3 Directory Creation and Maintenance

A new directory can be added to the hierarchical file structure by using the CREATEDIR command. This command allows the user to add new directory files to existing volume names or directory files by specifying either the fully qualified pathname (naming all of the branches of the tree) or the directory identifier assigned to the existing directory file. A detailed description of the CREATEDIR command may be found in Chapter 4.

By adhering to the following guidelines, the user can take advantage of the versatility and flexibility of the iNDX distributed file system.

1. Minimize the number of directories in each volume by subdividing directories by project and function.
2. Keep all of the "system files" (i.e.; files that contain the information necessary to maintain the file system structure on each disk) in one directory file.
3. Create a separate directory of directory files for individual user/project "miscellaneous files."

NOTE

An extensive structure will retard file accessing and retrieval time. Figure 3-2 shows a well-defined, highly structured, multi-user environment. The file system structure should be determined by project requirements. Normally, use only two or three levels of file-depth on a small-to-medium size single-user project.

4. To minimize access rights problems, use the project id as the user name for all users requiring the same access rights. This will minimize the access commands and control required by the superuser.

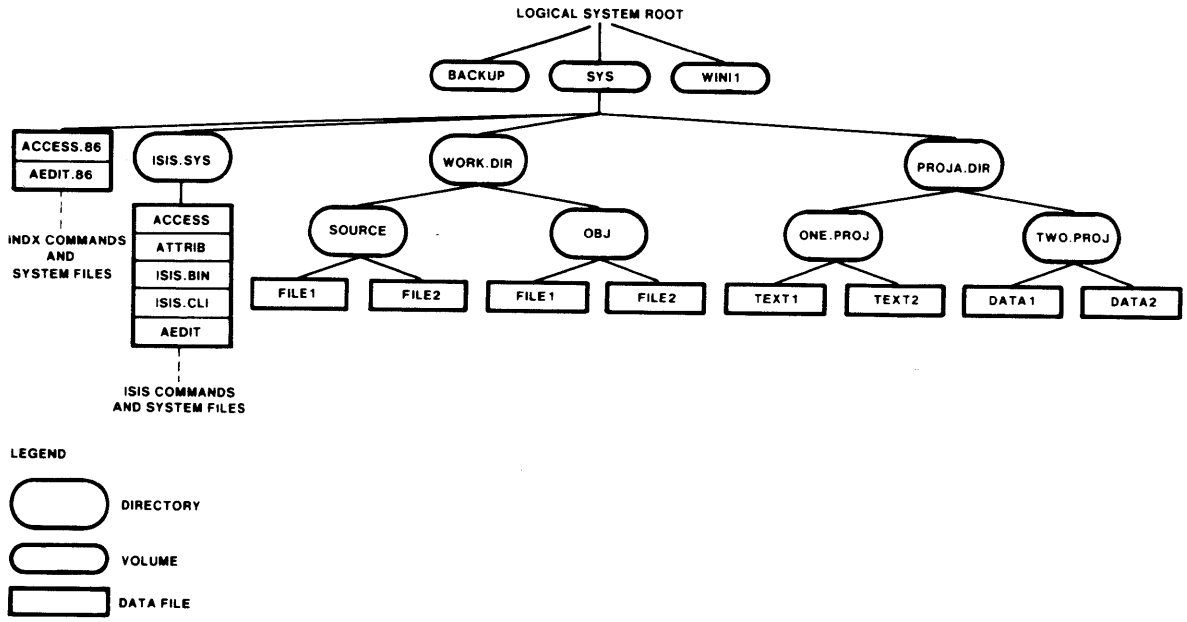


Figure 3-2. Model of iNDX File System

3.4 Directory Files

Directory files reside within each volume. Directory files can contain other directory files, or data files. Figure 3-2 shows a volume (SYS) with three directory files: ISIS.SYS, WORK.DIR, and PROJA.DIR.

All the iNDX system files are normally in the system volume root (unless an alternate directory was specified with the SEARCH command), including the iNDX commands, compilers, editors, locater, linker, etc. Some example commands shown in Figure 3-2 are ACCESS.86 and AEDIT.86.

The ISIS.SYS directory file contains all the files an ISIS user would normally look for on drive 0 (:F0:). Those files are:

- basic system files such as ISIS.DIR, ISIS.BIN, ISIS.CLI, etc.
- ISIS Command programs (ACCESS, COPY, RENAME, LOGON)
- Translators, Editors, Locators and Linkers for the ISIS environment

There are two sample project directories shown in Figure 3-2, WORK.DIR and PROJA.DIR. Typically, the prototype software (object, source and list) files could be included in the WORK.DIR and applications, documenations, etc. in the PROJA.DIR.

The PROJA.DIR directory (or subdirectories) could be used as a HOME directory (established with the USERDEF command) for the user or users associated with that project.

The directory files listed can be controlled as needed by individual programmers. Each person in the structure can have a directory name that can be used to store status reports, memos, trip reports and other miscellaneous work.

A separate directory or volume (such as BACKUP) can be used to ARCHIVE information. The ARCHIVE command can only be used to backup local devices. If on the network, the network files must be ARCHIVEd at the Network Resource Manager (NRM).

3.5 File Access and Ownership

Every data and directory file in the hierarchical file structure has an "owner": the file's creator. Every file on a shared mass storage device has separate access rights for the file owner (Owner Access Rights) and other users (World Access Rights). Creating a file (directory or data) requires that the creator have access rights to the directory where the file will reside. Users with the same access requirements should use the same user id to log on, to minimize access problems.

3.5.1 Protecting File Access on a Mass Storage Device

Access rights are not required to access files on a flexible disk. These files are protected from unauthorized access by removing and storing the media.

For files stored on Winchester or hard disks, a file protection system is provided to allow or prevent one user from accessing another's files. The person who manages this system is defined as the superuser and is responsible for providing system management for mass storage devices that the normal user cannot provide. These management functions include creating and deleting users, managing user ID numbers, and assigning passwords. The superuser can access, create, delete, or modify any file or directory within the system. Caution should be applied when acting as a superuser, to prevent unintentional acts of destruction.

The superuser should establish a user structure that allows the other users to limit access to their files as they see fit. Once this is done, each user will use his or her user name and password when logging on. Only the superuser can use the superuser logon sequence.

3.5.2 Creating and Deleting Users

The superuser initially creates (and adds) each user by assigning a unique name, a unique user/project ID number, a home directory, and a password. The user logs on by entering the user name and password at the workstation console. These identification methods guarantee that only authorized users will be able to access the file system. The user ID provides a way of tracking ownership of files throughout the system.

When a user is deleted from the system, the files belonging to that user are not deleted. An expanded directory listing shows an owner name of "NOT FOUND" for files whose owners have been removed from the structure. The superuser can then use the CHOWNER command to transfer ownership to another user or can define another user with the same user ID to become owner of the files.

The creation and deletion of users is accomplished via the USERDEF command (see Chapter 4) which only the superuser can use. The user name SUPERUSER cannot be deleted from the system. Secondary superusers can be created with distinctive user names and user ID numbers in the range 3-15. The superuser has a predefined user ID of 2. The secondary superusers have all the superuser capabilities except the ability to execute the USERDEF and USERS commands.

Secondary superusers may be removed from the structure by the primary superuser.

3.5.2.1 Managing User IDs

The primary superuser assigns user names and user ID numbers with the USERDEF command. The system keeps a record of these identification assignments. The superuser can view these via the the USERS command (see Chapter 4).

The USERS command can only be used by the primary superuser. The command displays a list of both user names and their associated ID numbers. Valid user ID numbers are in the range 1024-32767.

3.5.2.2 Assigning Passwords

The USERDEF command assigns a null password to each newly created user name. The superuser or user can then assign passwords using the CHPASS command. Passwords can be unique for each user or can be common to a project or products.

3.5.3 File System Considerations

Due to the structure of the hierarchical file system, users who have ALL WORLD access rights to a shared file need to be careful during edit sessions. For example, if you delete part or all of a shared file, it may no longer be accessed. Another example is: if user A renames a shared file without informing user B of the new name, user B can no longer access that file if WORLD access rights are not assigned.

When editing shared files (especially when using the RENAME and DELETE command), document the changes so others can understand how you have altered the files.

CHAPTER 4
COMMAND DESCRIPTIONS

4.1 Introduction

The commands available at the Series IV console through the iNDX operating system are listed in Table 4-1 and described within this chapter.

Each command description consists of:

- a syntax statement, with information on the parameters, values, and punctuation.
- a detailed description of the function and required command interaction.
- examples relating useful instances of command execution.

The command line editing features, the syntax guide, and all help facilities described in Chapter 2, Human Interface, can be used when entering the iNDX commands.

Table 4-1. Series IV Commands

Command	Description
ACCESS	Change access rights of a file
ARCHIVE	Backup network files
ASSIGN	Sets logical names to pathnames
BACKGROUND	Permits simultaneous execution of jobs
BATCH	Interactive command file executor
CANCEL	Removes a job from a queue (to CANCEL a remote job see Chapter 5)
CHOWNER	Change owner of a file
CHPASS	Change password of a user
COPY	Copy a file
COUNT	Iterative processing of a command sequence
DELETE	Delete a file
DIR	List a directory
DISMOUNT	Remove a device from the system
ELSE	Used in conjunction with the IF command
END	No-op command for ISIS compatability
ENDJOB	Terminates command file processing
EXIT	No-op command for ISIS compatibility
EXPORT*	Sends a remote job to a queue
FILL	Sets human interface attributes
FORMAT	Format and initialize a disk
FPORT	Copy ISIS files to iNDX files & vice versa
ICOPY	Copy ISIS files to iNDX files & vice versa
IF	Conditional processing in a command file
IMPORT*	Declares workstation to be public
LNAME	Manage logical names

Table 4-1. Series IV Commands (Cont'd)

Command	Description
LOG	Sends console output to a log file
LOGOFF	Terminate logon session
LOGON	Gain access to the system
MOUNT	Add a device to the system
OPEN	Opens a parameter file from a command file
ORIF	Used in the IF command
OSCOPY	Copy an operating system
PDSCOPY	Copy PDS files to iNDX files & vice versa
QUEUE*	Manage NRM job queue
READ	Reads a parameter file from a command file
REGION	Adjusts memory region size
RELAB	Relabels device volume names
RENAME	Rename a file
REPEAT	Conditional iteration in a command file
RUN	No-op command for ISIS compatability.
SDCOPY	Duplicate disks using a single drive
SEARCH	Sets priority on command directory search
SET	Sets variable values in command files
SPACE	Display available space on a volume
STTY	Sets console attributes.
SUBMIT	Execute a command file
SYSGEN	Perform system generation for the network
SYSTAT*	Display remote job status
TIME	Set system clock
UNTIL	Used in the REPEAT and COUNT commands
USERDEF	Define user
USERS	Display list of users
VERIFY	Verify device integrity
VIEW	Displays contents of specified file
WAIT	Suspends execution in a command file
WHILE	Used in the REPEAT and COUNT commands

*These commands are for use in network mode. Refer to Chapter 5 for command descriptions.

ACCESS

Syntax

ACCESS *pathname* [SET { OWNER / WORLD } { *access spec* } { QUERY }]

where:

pathname	is a pathname, wildcard pathname, or null. Null (entered as a filename) gives a list of the access rights of the directory associated with the null logical name (user home directory typically).
SET	declares the specified attributes.
access spec	is READ, WRITE, ADD, DISPLAY, DELETE, ALL, or NONE.
QUERY	initiates a prompt asking the user whether the displayed (current) access rights are to be modified in accordance with the specified arguments.

Description

The ACCESS function allows the setting and display of the user access rights to a file. Any user may use this command to display the owner and WORLD access rights of a file. The owner of a file may change both the owner and WORLD access rights. The Superuser may change access rights to any file, regardless of ownership.

When setting access rights to a file, the new rights are merged with the existing rights. For example:

```
ACCESS /A SET WORLD READ
```

adds read access to the existing world rights for file A. To clear existing rights, the keyword NONE is used (e.g., ACCESS /A SET WORLD NONE READ leaves file A with world read only, regardless of prior world access).

You may specify a combination of READ or DISPLAY, ADD or WRITE, and DELETE. READ is equivalent to DISPLAY and ADD is equivalent to WRITE. ALL indicates full access whereas NONE indicates loss of all access rights to a file. If conflicting access rights are specified, the access rights set will be the last specified, as shown in the following example:

```
ACCESS /A SET READ NONE DELETE
ACCESS /A SET NONE WRITE
ACCESS /A SET NONE READ DELETE
```

DELETE will be set
WRITE is set
READ, DELETE are set

If no user is explicitly specified, the access rights of the owner are changed. Only the access rights of the WORLD or the OWNER is altered when either one is specified.

When ACCESS is run the following screen display appears:

```
FILE_NAME     OWNER_NAME     OWNER_ACCESS     WORLD_ACCESS
XYZ           SUPERUSER      REA WRI DEL      REA
```

When ACCESS is run in QUERY mode, the following message is displayed after the existing access rights are displayed:

```
SET ACCESS RIGHTS ?
```

Enter Y, y or yes to change the access rights. The new access rights then will be displayed

Examples

1. `ACCESS /A SET READ<cr>`

   This command adds the READ access rights to the existing owner access rights.

2. `ACCESS /A<cr>`

   This command lists the access rights of the owner and world to file A.

3. `ACCESS /A SET ALL QUERY<cr>`

   This command will first display the existing access rights, then ask the user if they want to set the new access rights. If the user answers yes, the existing access rights will be replaced with full access rights.

NOTE

Access rights for spooled files cannot be modified.

ARCHIVE

Syntax

```
ARCHIVE source TO dest [ [ { INC | EXC } qualifier ] {{AND | OR} switch} ]
```

where:

source name of the source directory to be copied

dest name of the destination directory or device

INC INCLUDE specifies that the source files will be ARCHIVED if the conditions are met within the set of qualifiers following INCLUDE.

EXC EXCLUDE specifies that the specified source files will not be ARCHIVED if the conditions are met within the set of qualifiers following EXCLUDE.

qualifier command qualifiers include:

ACCessed
CREeated TIME QUALIFIER
MODified
DIRectory (*directory list, . . .*)
OWNedby (*owner*)
FILe (*pathname*)

The time qualifiers may be in the form:

```
[ { BEFORE | SINCE | ON } { TODAY | mm/dd/yy { hhmm[ss] | hh:mm:[ss] | h:mm } } ]
```

AND | OR logical operators that allow concatenation and expansion of the qualifier set.

switch command switches that include:

APpend	Name
NOupdate	Query
DElete	Update
Log *log filename*	Volume

Description

This command is designed to copy files from one directory subtree to another, primarily for file backup and restoration. ARCHIVE can be used to copy backup files onto 5¼-inch flexible diskettes, Model 740 or Winchester disk drives. Complete definitions of the ARCHIVE command arguments follow.

Qualifiers. Qualifiers give the user the ability to specifically limit groups of files to be archived according to time, location in file hierarchy, owner, and filename. There is no limitation to the order or number of qualifiers that can be entered in the ARCHIVE command. The individual qualifiers are described below.

INCLUDE indicates that files are copied if they meet the conditions defined in the qualifier set following INCLUDE. The default condition is INCLUDE.

EXCLUDE — defines a set of files that are not copied if the qualifications within the qualifier set following EXCLUDE are met. EXCLUDE takes precedence over INCLUDE.

ACCESSED — compares the time specified in the time qualifier (BEFORE/SINCE/ON) to the last access time of the file to be archived. If the file access time agrees with the time qualifier conditions, the file is qualified. The time qualifier is required for ACCESSED.

DIRECTORY — allows the user to enter directory pathnames that are used to qualify directory files. When the DIRECTORY qualifier is used, ARCHIVE will select only those files that satisfy the directory entered in parenthesis following the DIRECTORY qualifier. Logical names are allowed but wildcard characters are not. Default will copy all directories.

MODIFIED — selects files that have been modified since the last ARCHIVE operation. If the MODIFIED keyword is followed by an optional time qualifier (SINCE/ON/BEFORE), the time qualifier time will be compared to the time of last file modification. Only those files which have been modified within the time qualifier conditions will be qualified. The default time for the MODIFIED option is the time of the last ARCHIVE operation.

FILES — allows ARCHIVE to copy files selected on the basis of the pathname component. The qualifier accepts wildcard characters as part of the pathname.

OWNEDBY — instructs ARCHIVE to copy files selected on the basis of owner name.

CREATED — compares the time entered in the time qualifier to the file create time. If the file was created within the conditions of the time qualifier, the file will be qualified. The time qualifier is required for CREATED.

BEFORE — allows the user to specify files created or modified before the date entered. If time is not entered, the default time is 00:00:00.

SINCE — allows the user to specify files created or modified since the date entered. If the time is not entered, the default time is 00:00:00.

ON — specifies the start of a 24-hour period.

TODAY — obtains the current date from the operating system and defines a 24-hour period beginning at midnight. TODAY can be used with BEFORE/SINCE/ON.

DATE — is required for BEFORE/ON/SINCE and must consist of the date in the form *mm/dd/yy*, and, optionally, the time of day in the form *hh:mm:ss*. The time should be entered in 24-hour form with midnight as 00:00:00. The default time is midnight.

Operators. There are two operators in the ARCHIVE command, AND and OR. They allow the concatenating and extending of the qualifying conditions within a qualifying set.

AND — provides the capability of concatenating several requirements on the files selected.

OR — operator provides the capability of extending requirements on the file set desired. AND and OR may not be intermixed within a qualifier set defined by an INCLUDE or EXCLUDE.

Switches. Switches in the ARCHIVE command affect the way files are archived, and the interface with the user. The switches affect the entire command (unlike qualifiers that are only effective within the qualifier set), and must be entered at the end of the command string. Following is a list of switches.

APPEND	instructs ARCHIVE to add the ARCHIVE operation onto the cartridge tape as a separate logical volume. APPEND is only allowed for the cartridge tape device. To be able to APPEND to a tape volume, an archive must have already been performed on the tape. Default is no APPEND.
DELETE	instructs ARCHIVE to delete any data file selected after the file has been copied to the destination. Default is no DELETE. A directory file cannot be deleted with this option.
LOG	LOG *log-file-name* duplicates all console messages to a specified file to provide a record of files copied, etc. The log file may be on any accessible random access device. The file is created if it does not exist, and is truncated (all space removed) and used if it does exist. If an unrecoverable error is detected on the log file, the file is closed and ARCHIVE will continue operation. Default is no LOG.
NAME	is not used by the Series IV.
NOUPDATE	instructs ARCHIVE to continue operation with no user query if a duplicate file is detected and the UPDATE switch is not specified. Default is to query the user. Within a submit file NOUPDATE is the default.
QUERY	instructs the ARCHIVE program to display the file name and then request user input before each file is copied or a destination directory created. If the file is a directory file and the user does not copy the file, the directory is not created nor is the descending source directory subtree scanned further. Default is no QUERY.
UPDATE	instructs the ARCHIVE program to automatically delete duplicate copies of a data file. The user is not queried regarding deletion. If UPDATE and QUERY are both specified, the user is queried before the operation but is not asked for permission to delete conflicting destination files. UPDATE and NOUPDATE cannot be specified within the same command. Default is NOUPDATE.
VOLUME	is not used by the Series IV.

Read access rights are required to copy source directory and data files. Add entry access rights are required to all files to existing destination directories. The superuser, by default, can read all source data files and copy them to destination directories.

Additional Notes

1. The following is a typical terminal dialog between the user and the ARCHIVE program. When the command ARCHIVE is entered, the program signs on by identifying the operating system, program revision level (V*x.y*), date and time.

```
‹operating system› ARCHIVE, Vx.y
mm/dd/yy hh:mm:ss
```

For each directory archived the screen displays:

```
DIRECTORY = <directory-name>
```

After each copy the screen displays:

```
COPIED <source-file> TO <destination-file>
```

When the ARCHIVE program has executed, the screen displays:

```
ARCHIVE COMPLETE
```

2. The source directory subtree structure which is duplicated at the destination directory includes the original owner, create time, and last modified time.

3. When a directory is encountered at the maximum supported subtree depth (maximum depth is 20), a warning message is issued to the user and ARCHIVE continues:

```
UNABLE TO COPY SUBTREE - LEVEL TOO DEEP
PATHNAME = <directory-pathname>
```

Additional ARCHIVE commands must be issued if the user wants the subtree branches to be archived.

4. If the QUERY switch is specified, ARCHIVE displays:

```
COPY <source-file> TO <destination-file> ?
```

and prompts the user for permission to perform the copy. The user can enter 'Y' to copy the file, 'R' or 'C' to copy the file and continue without further queriers, 'Q' to skip this file and all subsequent files in the current directory with no additional queries until the next directory is encountered, or 'E' to exit ARCHIVE. Any other response causes the file not to be copied.

Any response can be either upper or lower case. To exit ARCHIVE, enter a control-C.

5. If a destination data file has the same name as a source data file being copied, ARCHIVE requests permission to delete the destination file before proceeding (unless the UPDATE or NOUPDATE switch is set).

```
FILE ALREADY EXISTS
PATHNAME = <destination-pathname>
Delete Existing File?
```

The user can enter 'Y' to copy the file, 'R' or 'C' to copy the file and continue without further query, or 'E' to exit ARCHIVE. If any other response is given, the file is not copied.

6. The following information relates to archiving on flexible disks.

All diskettes should be formatted in advance and should have the same root name. Multivolume ARCHIVE operations to disk devices require that all volumes have the same directory structure, from the root directory to the destination directory that is specified on the command line.

For example, if the user enters "ARCHIVE /WINI0 TO /WINI0BACK/A", then all destination volumes must be named /WINI0BACK and must have a directory named "A" in order for the ARCHIVE operation to continue successfully. If a new destination volume is mounted and the volume name is incorrect, a warning message is displayed and a request for the correct volume is issued to the operator:

```
WRONG VOLUME MOUNTED, MOUNT: <volume-name>
CONTINUE ?
```

The user must mount the correct volume and enter a 'Y' to continue, 'E' to 'N' to exit. Any other response causes ARCHIVE to repeat the prompt.

If the disk destination volume fills up, a warning message is displayed and a request for another volume is issued to the operator:

```
<'VOLUME FULL' or 'DIRECTORY FULL'>, MOUNT NEXT VOLUME
CONTINUE ?
```

The user must mount a new volume and enter a 'Y' to continue, 'E' or 'N' to exit. Any other response causes ARCHIVE to repeat the prompt.

Examples

1.
```
ARCHIVE /source-dir to /dest-dir MODIFIED QUERY
```

This example will archive all files in the directory subtree '/source-dir' to the directory subtree '/dest-dir' that have been modified since the last ARCHIVE operation. Each operation will be verified with the user before continuing.

2.
```
ARCHIVE /source-dir to /dest-dir &
                FILES *.LST AND &
                MODIFIED BEFORE 12/25/81 &
                NOUPDATE
```

This example will archive all files in the directory subtree '/source-dir' to the subtree '/dest-dir' that match the pathname '*.LST' and were modified before December 25, 1981. It will not query the user if destination files already exist and it will not update.

ASSIGN

Syntax

```
ASSIGN [{logical name} TO pathname]
```

where:

logical name	designator specifying a logical device, may be alphanumeric (up to 14 characters), or numeric.
pathname	a directory pathname or a logical name

Description

The ASSIGN command provides the same functionality as the LNAME command, and is included in the iNDX function set to provide compatibility with ISIS.

With the ASSIGN command, pathnames of files or directories can be mapped to logical names of devices. Additionally, the ASSIGN command can be used to display the existing assignments.

The ASSIGN command, along with the LNAME command, allow the use of alphanumeric logical names. However, within the ISIS operating systems, only numeric logical names are allowed. If maintenance of ISIS compatibility is desired, then the logical names at the Series IV should be of the form:

:F*n*:
with *n* = 0 to 9

Additional Notes

The ASSIGN command may be most useful in building command or Submit files (using the Batch command or an editor) at the NRM that can be run either at a workstation or the NRM. Refer to the *Intellec Series IV ISIS-IV User's Guide* for further details on ISIS compatibility.

Examples

1. In the following example, the directory /WINI0/PROJ/WORK is assigned a logical name of WORK.

   ```
   ASSIGN WORK TO /WINI0/PROJ/WORK<cr>
   ```

2. This example shows the directory /WINI0/PROJ assigned the logical name :F1: (ISIS compatibility).

   ```
   ASSIGN 1 TO /WINI0/PROJ<cr>
   ```

3. The next example also shows an ISIS compatible assignment, with logical device 2 assigned to device 4.

   ```
   ASSIGN 2 TO :F4:
   ```

4. This example is for the display of existing assignments.

```
ASSIGN<cr>
LOGICAL NAME        PATHNAME

WORK                /WINIO/PROJ/WORK
1                   /WINIO/PROJ
```

BACKGROUND

Syntax

BACKGROUND *pathname* [(*parameters*)]... [{LOG / NOLOG} [(*pathname*) [APPEND]]]

where:

pathname	is a valid pathname.
parameters	is a list of up to 10 parameters.
LOG and NOLOG	specify whether a log is to be kept on a mass storage of all console activity.
pathname	specifies the filename of the log file.
APPEND	attaches the log file to an existing file.

Description

The BACKGROUND command, used to execute a command file, permits the simultaneous execution of a job requiring user interaction. The difference between executing a given command file in the background and executing a given command in the foreground is that background execution does not allow console interaction. The user may use the LOG option to provide a log file on a mass storage device. The BACKGROUND, EXPORT, and SUBMIT commands all have similar structures. The similar structures permit the user to execute the same command file in any of these modes.

If the name of a command file does not have an extension, the system appends the extension .CSD to the command file. If the filename ends with a period, the system will use the filename it has been given (with the period truncated). If the filename includes an extension, the system uses that filename without modifying it.

The optional parameters specified in the command line are the actual parameters to be substituted for the formal parameters embedded within the command file. A maximum of ten actual parameters may be specified in the command line.

Placing formal parameters in the command file allows the user to call the same command file with varying sets of actual parameters. The command file is preprocessed for actual parameter substitution. If the command file contains any references to formal parameters, the actual parameters are substituted for the references. If actual parameters are not specified (in a parameter file or command string), then the parameter references will be replaced by null strings. Thus (reading from left-to-right), if the user enters the actual parameters ASM, PLM, and BLT, ASM replaces the formal parameter %0, PLM replaces the formal parameter %1, and BLT replaces the formal parameter %2. The command file resulting from the parameter substitution is placed in the same directory as the command file from which it was generated. The final path component of the generated file is derived from the final path component of the generating file as follows:

- The final component of the generating file up to the third character or before the first period (whichever comes first) is truncated.
- A unique character string obtained from the system job manager and the extension .CS are appended.

When the LOG option is specified, a log of the console display is kept on a mass storage device. You may specify the log file pathname; otherwise it will automatically be set to a unique name based on the command file pathname.

When APPEND is also specified, the log file is attached to the end of an existing log file. If the specified log file does not exist APPEND has no affect. If the specified log file already exists and APPEND is not selected, the file will be overlaid by the new log file.

NOLOG specifies that no log be kept. If neither option is explicitly entered, LOG is the default condition.

When a background job is executed, the system implicitly reenacts the user's LOGON. The environment created is the same one that existed when the user initially logged on to the system. This environment is not necessarily the same as the foreground environment operating at the time the BACKGROUND command was invoked. The logical names defined in the foreground are not accessible in the background and vice versa. The amount of memory available for the background is specified by the REGION command.

Examples

Refer to the examples in the SUBMIT command description for an illustration of command file execution.

1. Do not debug in the foreground while background programs are running.
2. Never run experimental software in the foreground while programs are running in the background.
3. Never run experimental software in the background or export it to another workstation.
4. Do not run the Background while in Multi-user mode or while partition 2 is active in Toggle mode.

BATCH

Syntax

BATCH *pathname*

where:

pathname is a valid pathname.

Description

The BATCH command allows interactive creation or modification of command files while using the Syntax Guide as a text editor. The file selected in the BATCH command may be an existing file, or a file to be created. After editing, the user is asked to select an execution option for the command file.

Initially, when the BATCH command is invoked, a search is made for the specified command file. If the file is not found a new file is created as specified in the pathname. If the filename does not have an extension, .CSD will automatically be appended. If the filename does have an extension, it will not be modified. If the file does exist, it will be opened and displayed along with the following:

```
------ Enter command file:
       <esc> to Exit
```

Additional Notes

1. The BATCH command editing features include all the keyboard cursor control features of the iNDX operating system as listed in Table 2-1, plus all the features of the command language interpreter and the Syntax Guide.
2. Comments may be entered by entering a ";". Everything between the semicolon and the carriage return ending the line will be ignored when the file is run, but will be displayed on the console.
3. If variables are to be inserted in the Batch file, they should be inserted in the form:

   %*n* where *n* is 0 through 9
4. Batch file editing is terminated by pressing the ESC key. The following menu will then appear:

   ```
   SELECT EXECUTION OPTION
   Abort   Write   SUbmit   Background   Export
   ```

   a. Abort—Enter A. Returns to the operating system without saving or running the batch file.

   b. Write—Enter W. Writes the edited command file and returns to the operating system.

   c. Submit—Enter S. Asks the following questions before running the edited batch file under the SUBMIT command.

      ```
      Write command file (y or [n])?
      ```

A yes (enter y) response causes the file to be written or modified if already written. The default response is no (enter n or <cr>) causes a temporary file to be written that is SUBMITted then deleted.

If any parameters were specified in the building or editing of the Batch file, the user will be queried for those values:

```
Current parameter values:
%n: parameter value

Enter value for %n:
```

This will repeat until all parameter values are set. Then they are checked for correctness:

```
Current parameter values:
%n: parameter value

Are the current parameters correct ([y] or n)?
```

If the answer is no, the user will be asked which parameter is incorrect, and allowed to change it. Then the check will be performed again. If the parameters were correct (response = y or <cr>) the questions continue:

```
Keep log of console output (y or [n])
```

A no response causes the file to be submitted for execution immediately. A yes response causes the following question to be asked:

```
Enter log file name OR <cr>
```

Enter the name of the file that will record the console display. If a filename is not entered, a log file will be created with a name of:

AAA*nnn*.LOG

with AAA = filename
nnn = three alphanumeric characters determined by the program.

d. Export—Enter E. The export option will create a temporary job file identical to the batch file and send to a specified queue upon completion of a series of questions identical to those for the submit option, except as follows:

   After the parameter values have been entered and checked the following prompt will appear:

   ```
   Enter destination job queue:
   ```

   This requires the entering of an existing queue name.

e. Background—Enter B. The background option will create a temporary job file identical to the batch file. The same questions asked in the submit option will be asked in the background option. The job will then be executed in the background mode. The job filename will be displayed in the Operating System Field during the time the job is running in the background.

NOTE

Do not select the Background option if in Multi-user mode or in Toggle mode with partition 2 active.

CANCEL

Syntax

```
CANCEL {BACKGROUND                                          } [, . . . ]
       {[REMOTE] queue {(jobname) | (jobnumber)}             }
```

where:

BACKGROUND	specifies a background job.
REMOTE	specifies a remote job.
queue	is the queue where the remote job is queued for execution.
jobname	is the final component name of the remote job to cancelled.
jobnumber	is the assigned value of the remote job (which can be displayed via the SYSTAT command).

Description

The CANCEL command is used to cancel either a background job or a remote job (see the Cancel command in Chapter 5). If BACKGROUND is specified, the currently executing background job is aborted.

NOTE

In order for the Series IV CANCEL command syntax to be more compatible with the ISIS CANCEL command, REMOTE is the default condition. Unless the CANCEL command line ends with BACKGROUND, it is assumed that the user wishes to cancel a remote job.

Examples

1.
```
CANCEL REMOTE MYQUE(MYJOB)
CANCEL MYQUE(MYJOB)
```
are identical commands.

2.
```
CANCEL BACKGROUND
```
will abort the background job, and the command:

3.
```
CANCEL BACKGROUND (MYJOB)
```
will attempt to cancel a remote job in a queue named "BACKGROUND".

CHOWNER

Syntax

CHOWNER *filename* TO *username*

where:

`username` is the name of the new owner of the file.

`filename` is a fully qualified pathname or logical name.

Description

This function permits a change in ownership of a file. Only the superuser or the owner is authorized to use this command.

Ownership of spooled print files cannot be changed.

Examples

1. `CHOWNER /SYS/MEDIA TO PAUL`

   This example changes the ownership of the MEDIA file, giving it to user PAUL.

2. `CHOWNER /SYS/BOOK/ATEXT TO ART`

   This example changes ownership of the BOOT/ATEXT file, giving it to user ART.

CHPASS

Syntax

CHPASS *username*

where:

username is the assigned user's name.

Description

This command changes the password associated with the specified user. Any superuser may invoke CHPASS for any other user. However, the command will not execute for ordinary users unless the old password is correctly entered in response to the query produced. Superusers need not enter the old password.

Examples

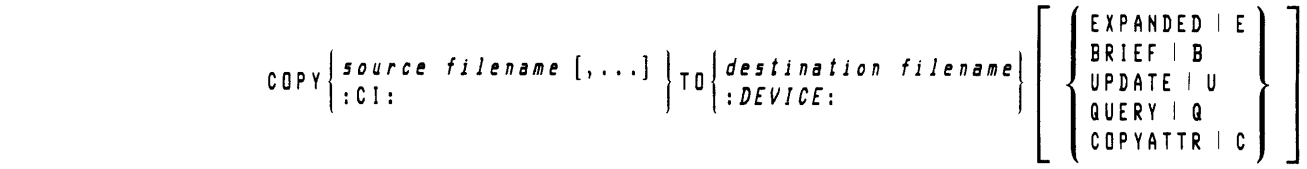
1. CHPASS FRED

```
Old Password?
pass1
New Password?
pass2
Verify Password?
pass2
```

In this example, user FRED (or any other ordinary user who knows FRED's old password) changes his password from pass1 to pass2. FRED then needs to verify that he knows the new password (pass2) by responding correctly to the third query (verify?). The passwords will not be displayed when entered at the keyboard.

COPY

Syntax



where:

source filename	is a pathname or a wildcard pathname. If the source filename is a wildcard pathname, the destination filename must be a directory file.
:CI:	is a console input device.
destination filename	is either an existing directory file or a data file.
:*DEVICE*:	is an output device such as :LP: (local line printer), :SP: (spool printer) or :CO: (console output).
UPDATE (U), QUERY (Q), BRIEF (B)	are options that suppress or enable the querying process. U, Q and B are allowable abbreviations.
EXPANDED (E)	displays the fully qualified pathnames of all source and destination files when they are copied. The abbreviation E is permissible.
COPYATTR (C)	specifies that the existing file access rights will be copied along with the file. The abbreviation C is permissible.

Description

The command copies a file (or a list of files) from one position within the file hierarchy (a single device) to another position. If the source filename is a wildcard pathname, the destination filename must be a directory.

The destination filename is either an existing directory file or data file. If the destination filename is an existing directory file, a new file is created in that directory and the final component of its filename is the final component of the source filename. This operation cannot take place if the user does not have ADD Entry access rights to the directory file.

If the destination file is an existing data file and the UPDATE or BRIEF options have not been specified, the following query is displayed:

```
FILE ALREADY EXISTS
PATHNAME = name of existing file
DELETE EXISTING FILE?
```

Entering a Y (or y) deletes the existing file and initiates the Copy operation if the user has DELETE access rights. Any other entry aborts the operation. The UPDATE option disables the querying. In SUBMIT, only the first two lines will be printed; the prompt to delete the file will not appear.

Querying can be explicitly chosen by entering the QUERY option. In this case, each Copy operation produces the following query:

```
COPY pathname TO pathname ?
```

With the EXPANDED option, fully qualified pathnames are displayed:

```
COPY /SYSROOT/DIR/pathname TO /SYSROOT/DIR2/pathname ?
```

Entering a Y (or y) causes the copy to take place. Any other entry terminates the command. The QUERY option cannot be used in submit mode; if QUERY is entered, it produces the following error message:

```
QUERY OPTION NOT USEABLE IN SUBMIT MODE
```

Each successful copy produces the following message:

```
COPIED source-pathname TO destination-pathname
```

The file protection access rights of a file can be copied along with the file by selecting the COPYATTR option. If the option is not selected, the access rights of the destination file are the default access rights set with the SYSGEN command.

The Line Printer or spooler may be specified as the destination of the Copy operation, thereby allowing the user to obtain printed output if the workstation has a local line printer. If the network workstation does not have a local line printer, the files may be spooled to a network printer. If the system does not have a local line printer, an error message will appear.

To print locally, specify :LP: as the destination filename. To provide for network spooling, enter :SP:requestname as the destination. Copying to :CO: sends the file to the console output device.

COUNT

Syntax

```
COUNT n

[commands]

[ { WHILE } argument { = } argument ] ...
  { UNTIL }         { < > }

[commands]
END
```

where:

argument	is a CLI variable value, a CLI variable name or a parameter.
n	is the number of times the block will repeat.

Description

The COUNT command is executable only from a command file. COUNT allows specified iteration of one or more commands. A decimal number specifies the number of times the loop will be executed. The WHILE, UNTIL, or ENDJOB options can be used for a premature exit from the loop. More than one WHILE or UNTIL clause may be entered.

With the UNTIL option, the command set is executed until the logical comparison evaluates TRUE; the loop is then exited regardless of the value of the counter for the loop. In the WHILE option, the command set is executed as long as the logical comparison evaluates to TRUE. When the comparison evaluates to FALSE the loop is exited regardless of the value of the counter for the loop. The logical comparison consists of testing one string against another for equality or inequality. The string may be a CLI variable value, a CLI variable name, or a parameter. Substitutions are made before the command line is executed.

NOTE

The WHILE or REPEAT options may be used as often as necessary in a COUNT block.

CREATEDIR

Syntax

CREATEDIR *pathname*

where:

pathname is the pathname identifying the new directory.

Description

This command creates a new directory with the pathname entered. The user must have ADD-Entry access rights to the parent directory. The pathname provided must not conflict with the pathname of an existing file or directory. If a conflict in pathnames exists, the command will abort. Following proper execution of the command, the user becomes the owner of the directory created with ALL access rights. No access is conferred upon WORLD.

Example

1. CREATEDIR /DIRA/DIRB

   A new directory, DIRB, is created in the parent directory, DIRA. The user is the owner of the directory DIRB, and has all access rights to the directory.

DELETE

Syntax

DELETE { *pathname* [{ DIR / QUERY }] }, [, ...]

where:

pathname	is the pathname, the wildcard pathname, or a null string.
DIR	is mandatory for deleting non-empty directory files and is optional for deleting empty directory files.
QUERY	is an option that produces interactive querying before each delete operation is executed.

Description

The Delete command deletes both data files and directory files. In the command line, the keyword DIR is mandatory for deleting Non-Empty directory files and is optional for deleting Empty directory files. If the keyword DIR is used for data files, an error message will be returned.

A given data file will be deleted only if the user has DELETE access to that file.

In deleting a directory file, access rights to the given directory file are checked. If the user does not have DELETE access, the operation terminates immediately. Otherwise, every file in the directory will be deleted except:

1. Non-empty directory files.
2. Files that the user cannot delete (no DELETE access rights).
3. Any file with existing connections at the time of delete operation. In this case, directory is not deleted and new connections to the file are not permitted. The file will be physically deleted from the medium when the last connection to it is detached.

If the null string is entered as the logical directory name (i.e., DELETE <cr>), an attempt is made to delete the directory associated with the null logical name.

Spool requests are deleted by specifying the name of the spool request to be deleted:

```
DELETE :SP:REQUESTNAME
```

With the exception of the Superuser (who may delete any spool request), a spool request may be deleted only by the owner.

A directory specified in the DELETE command will not be deleted if any file within that directory is being accessed by another user when the DELETE command was attempted.

The user may specify the QUERY option with any DELETE command.

In Interactive Mode: every delete (in the given delete invocation) produces the following query:

DELETE *pathname* ?

To delete that file, respond by entering either Y or y followed by <cr>. Any other characterwill be interpreted as a "no" response.

In Submit Mode: the query option causes termination of the delete operation. The following message appears on the console:

```
QUERY OPTION NOT USEABLE IN SUBMIT MODE.
```

Each successful deletion of a file is followed by the following console message:

```
DELETED pathname
```

Successful directory deletion produces the following console message:

```
DELETED DIRECTORY pathname
```

Examples

1. `DELETE /A/* Q`

   This example deletes all files in directory A with a query before each file deletion.

2. `DELETE /A/B/C`

   This example deletes File C in the Directory /A/B.

3. `DELETE /A/B DIR`

   This example deletes the directory file /A/B. Assuming /A/B is a non-empty directory file, the specifier DIR will be required.

4. `DELETE :SP:PROGRM.LST`

   The spooled print file PROGRM.LST is deleted.

5. `DELETE /WORK/*.LST Q,/WORK/*.OBJ`

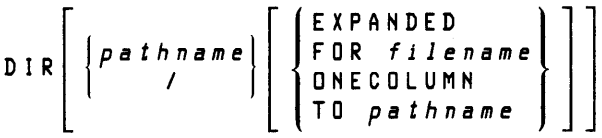
   The user is queried to delete all *.LST files in the /WORK directory. Then the *.OBJ files in /WORK directory are deleted with no query.

Additional Notes

1. In a multi-programmed/shared file environment, certain time- dependent situations may cause a delete operation on a directory to fail. For example, if a file is created within a directory at the same time a delete operation is being performed on the directory, the file may not be encountered by the search operation and the delete will fail. If connections exist to one or more files within a directory at the time a delete is attempted, the delete may fail.
2. A directory will not be deleted if it is the parent of a non empty directory file.
3. Wildcard pathnames may not be used with the DIR specification.

DIR

Syntax



where:

pathname	is either the pathname or a logical name.
FOR *filename*	specifies directory information of filename to be displayed.
EXPANDED	specifies that the complete information described below in the "Description" should be displayed for the directory pathname entered.
TO	specifies a file where the complete information is to be written (instead of to the console).
ONECOLUMN	specifies that the displayed directory will appear in one column.
/	displays the volume names of all volumes currently mounted (logically attached) to the logical system root.

Description

The DIRectory command allows the user to display the names of files within a directory. When used without the EXPANDED modifier, the DIRectory command produces a listing of file names only. Omitting the pathname produces a listing of the names of all files in the directory designated by the null logical name.

The EXPANDED modifier produces the following information:

- File name
- Owner name
- Length (in bytes)
- File Type (Data or Directory)
- Owner access rights
- World access rights

A null entry cannot be used with the EXPANDED or TO Modifier. To obtain expanded information on the file designated by the null logical name, enter " " as the directory name.

Information about the file system configuration can be obtained by performing a DIRectory operation on "/". For each volume of the file system which is accessible from the given node, the volume name and the volume location are returned. The location is specified as being Public, Local or Shadowed. A volume is Public if the World can access it; use of that volume name as the first component of a fully qualified pathname results in a file on that volume being accessed (i.e., the public). A volume name is Local if only the owner can access it; use of the pathname as the first component of a fully qualified pathname

results in a file on that volume being accessed. A volume is Shadowed if it resides in a shared file and a volume of the same name exists locally. The local volume shadows the public volume, i.e., use of this volume name as the first component of a pathname accesses the local volume, not the public volume.

The directory command can provide information about the state of the spool queue at a public node. The spool is designated by the name :SP:. SP may also be used as an abbreviation for the spool if it has not been defined as a logical name.

```
DIR :SP: or DIR SP
```

displays the contents of the spool. The format of the display is the same as for files. All DIR options can be applied. The order of the files listed is unrelated to the printing order in the print queue.

The Directory command sends output to the console as a default condition. However, the modifier TO can be used instead to designate any disk file or device (such as :SP:filename or file).

When TO is used to output data to a new file, it will be subject to the restrictions of any other action that creates/updates a file, i.e., ADD-ENTRY, DELETE, WRITE access rights must be updated. Access rights on the new file will not necessarily be compatible with access rights in the original file.

Additional Note

Wildcard pathnames cannot be used with the Directory command.

`DIR / EXPANDED` is not a valid command.

Examples

1. `DIR /A EXPANDED`

   This example displays full information about all files in directory A.

2. `DIR /A`

   This example displays only file names for directory A.

3. `DIR /A TO /A/B`

   This example outputs file names of directory A to file /A/B.

4. `DIR :SP:`

   This example displays files in the spool directory.

5. `DIR /WORK.DIR FOR PROGA.SRC EXPANDED`

   This example displays expanded information about the file PROGA.SRC, which resides in the directory WORK.DIR.

6. `DIR /WORK.DIR FOR *.SRC`

   This example displays all filenames in the directory WORK.DIR which have the extension .SRC.

7. `DIR /SAMPLE ONECOLUMN`

   This will cause the directory to display the files in one column instead of the usual three.

8. `DIR ' ' EXPANDED`

   This example displays all filenames in the directory that has has the null logical name assigned to it using the LNAME command.

DISMOUNT

Syntax

DISMOUNT *devicename*

where:

devicename is one of the following:

FL0, FL1 for Flexible Disks
WM0 for the integrated Winchester Disk
WD0–WD3 for a Winchester 35-Megabyte Disk
HD0–HD3 for HD5440 Hard Disks

Description

This command makes a mass storage device inaccessible to the file system and detaches all connections previously made to the file. The Dismount program guarantees that the specified volume will no longer be accessed by the file system. Never dismount the system volume. An error message is returned if the specified device has already been dismounted.

Examples

1. `DISMOUNT HD0`

   This example removes the hard disk fixed platter from the file system.

2. `DISMOUNT WD0`

   This example removes the 8-inch Winchester disk from the file system.

NOTE

Do not dismount the system device. DISMOUNT does not allow dismount of the system device; however, the user must not physically dismount the system device, either.

ENDJOB

Syntax

```
ENDJOB [(argument)]
```

where:

argument is a CLI variable value, a CLI variable name, or one of the ten parameters %0 to %9.

Description

ENDJOB is executable only from a command file. The ENDJOB command allows the user to terminate the processing of a command file before it reaches its normal end. Usually, the command will appear in an IF statement that is used to check for proper execution. In that case, the ENDJOB command is used to provide an error escape and a return to the calling command file.

The optional value, if specified, is assigned to the predefined CLI variable name in the calling command file (STATUS). For example, assume that the decimal number 5 is established to reflect a malfunction. If the calling program detects a malfunction STATUS = 5, the ENDJOB command was executed and a malfunction occurred in the called file.

If the optional value is not specified and the called program executes ENDJOB, and if the program terminated normally, the value 0 will be assigned to STATUS in the calling program. If a parameter file is opened within the current command file, the parameter file is automatically closed and detached. Command files are called (or submitted) either by other command files or interactively from the keyboard by the SUBMIT, EXPORT, or BACKGROUND commands.

NOTE

The EXIT command is a non-operational command provided for ISIS compatibility. When running the same command file in the ISIS mode, EXIT will be used to return to the program instead of ENDJOB.

FILL

Syntax

```
        ⎧ ON    ⎫
FILL    ⎨ OFF   ⎬
        ⎩ SPACE ⎭
```

where:

ON	enables the FILL command.
OFF	disables the FILL command.
SPACE	allows the user to press space bar to complete the entering of a command.

Description

The FILL command allows the user to enable and disable the Command Completion feature on the Syntax Guide. The portion of the command displayed in uppercase letters in the Syntax Guide is the only part that must be entered when the FILL option is on. The syntax driver automatically fills in the missing lowercase letters. FILL ON enables the feature; FILL OFF disables it. The ON option is the default case.

The FILL SPACE option lets the user enter as many characters of the command as desired (minimum of the uppercase letters shown in the prompt field), press the space bar to complete the command. The FILL SPACE feature is recommended with the Display TYPE-AHEAD feature (refer to STTY command) for optimal ease of use.

The Syntax Guide also has a Noise Word Fill feature. When the Syntax Guide enters options on the Menu Line and only one option is available, the Syntax Guide will automatically enter that word (called a redundant or "noise" word, hence the term Noise Word Fill). When FILL is off, Noise Word Fill is also inoperative.

FORMAT

Syntax

```
                                    ┌──────────────────────────┐
                                    │ AGRAN (number)           │
                                    │ FNODES (number)          │
                                    │ INTERLEAVE (number)      │
FORMAT physical-device volume-name  │ NODUP                    │
                                    │ NOINIT                   │
                                    │ NOVERIFY                 │
                                    │ OVERRIDE                 │
                                    │ RESERVE (option, ...)    │
                                    │ UPDATE                   │
                                    └──────────────────────────┘
```

where:

physical-device is:

FL0, FL1 are flexible disks
WM0 is a 5¼-inch Winchester disk
WD0...3 are 35 megabyte Winchester disks
HD0, HD2 are fixed platter hard disks
HD1, HD3 are removable platter hard disks

volume-name is the volume root directory name of the physical device and is limited to a maximum length of ten characters.

Description

The FORMAT command formats a local disk volume and initializes it with the basic files required on each volume. These files do not include any system programs.

The disk device must be physically attached to the Series IV and is either a Winchester disk, a hard disk, or the integral flexible disk.

The FORMAT command verifies each block on the disk. If any block cannot be read, the block is marked in the free space bit map and the bad block bit map to prevent its allocation to the file. If the bad block is in the fixed area reserved for the volume label, the FORMAT command terminates after displaying an appropriate message.

Format Options

The following options may be included in the Format command line.

AGRAN The Allocation GRANularity option specifies the granularity for a device. The default granularity is four. The allocation granularity specifies the number of blocks that make up one logical unit. The allowable range of values is from 1 to 64. If the default value is not acceptable, the AGRAN value would normally be changed to the average file size in terms of 512 byte blocks. The default value is satisfactory for most applications. It may be useful to format volumes used for archiving with an AGRAN of 1, since no fragmentation problems occur and all available space can be utilized.

FNODES The FNODES parameter specifies the number of filenames to be reserved. If this is not specified, a default value is used.:

Device	Minimum	Default	Maximum	Default Interleave
FL*n*	20	200	400	1
HD*n*	20	1000	2000	3
WD*n*	20	3000	6000	3
WM*n*	20	2000	4000	6

INTERLEAVE The INTERLEAVE option specifies the sector interleave for a disk drive. It is strictly a performance parameter and for most systems need not be changed. Optimum performance will very likely be either one less or one greater than default, depending on workload, file sizes, and disk fragmentation. NOINIT cannot be specified with INTERLEAVE. The possible range of values is from 1 to 8.

NODUP The NODUP option specifies that no duplicate file information is to be maintained. Duplicate information is useful in the event of a disk error, thus making recovery possible. This option should only be specified on devices single-user systems.

NOINIT The NOINIT parameter specifies that only the file structure is to be initialized. This option is used in reformatting disks that have been previously formatted. It speeds up the formatting process because the formatting of disk sector ids is not performed. FORMAT will override the NOINIT keyword when the specified disk is one that has not been previously formatted.

NOVERIFY The NOVERIFY option disables the read verification of the device. Normally, the FORMAT command will perform a read of each track to ensure that each allocated track can be read correctly. NOVERIFY disables this function for faster FORMAT execution.

OVERRIDE This option allows the user to perform a format on a device when both regions are active (Toggle or Multi-user modes or Single-user running Background). Without the override option, attempting to Format a device with both regions active generates an error message.

The override option may disrupt the operation of a second user or background process.

RESERVE The RESERVE option specifies the number of blocks to be reserved for the operating system and operating overlays. FORMAT does not write data into these file blocks; OSCOPY writes into these files. The number of blocks to be reserved may not exceed 1000 blocks and may be explicitly specified. The default numbers are 256 for the operating system and 192 for overlay. Do not use the WOS and WOV options.

UPDATE — The UPDATE option disables the verification query normally performed by FORMAT. Under default conditions, if the disk being formatted has been previously formatted, the user will be asked if this is the correct disk. UPDATE disables this feature, and is the default condition within a SUBMIT file.

Examples

1. `FORMAT HD0/ONE.VOL <cr>`

   This command will format the hard disk volume and name it ONE.VOL. When the return is entered the following message signs on the screen:

   `iNDX-G11 FORMAT, Vx.y`

2. `FORMAT HD0/SAMPLEXFLE NOINIT`

   This command is used to format a disk that has previously been used on the Series IV.

3. `FORMAT WD0 SYSTEMDIR FNODES(5000) RESERVE (OS(512),OV(256))`

   This command formats a Winchester disk, reserves system space for 5000 filenames and allocates 512 blocks for the Series IV operating system and 256 blocks for operating system overlays.

Additional Notes

Use Control C to abort the FORMAT command.

FPORT

Syntax

```
{S4FPRT}  {UP   iNDX-source-pathname TO destination-pathname                          }  [ {UPDATE} ]
{S2FPRT}  {EXIT                                                                       }  [ {EXIT   } ]
          {DOWN [disk-dir] ISIS-source-pathname TO iNDX-destination-pathname          }  [ {QUERY  } ]
```

where:

S4FPRT and S2FPRT	inform the operating system that the user is initiating the command from a Series IV or Series II, respectively.
iNDX-source-pathname	is a valid iNDX pathname or wild-card pathname.
destination-pathname	is a disk-directory or an ISIS-filename (optionally preceded by a disk-directory). ISIS-filename and disk-directory are as defined in the *ISIS-II User's Guide*, Order Number 9800306.
disk-dir	is a disk-directory as defined in the *ISIS-II User's Guide.*
ISIS-source-pathname	is an ISIS-filename or an ISIS-wildcard filename as defined in the *ISIS-II User's Guide.*
iNDX-destination-pathname	is a valid iNDX directory file or a valid Series-IV pathname.
UPDATE, EXIT, and QUERY	are options that determine if the user is to be queried prior to each copy operation.

Description

The FPORT utility program allows the user to copy ISIS files to a specified location within the iNDX file structure and vice-versa. The copy is accomplished by transmitting data through a serial line connecting the Series IV system and the ISIS system. While the FPORT program is executing, the ISIS system acts as a slave of the Series IV system and waits for commands to reach it from the Series IV. Either the Exit function of FPORT or the EXIT option must be used to return the ISIS system to independent operation.

The FPORT command has three functions: UP, EXIT, and DOWN. Each function has its own set of qualifiers.

Physical Interaction

To execute the FPORT command, first boot up the ISIS operating system on the Series II system and then enter the ISIS-II File Port program. The program will then return the following message and enter a wait state:

```
ISIS-II FILE PORT, Vx.y
ENTER FPORT COMMAND at the SERIES-IV CONSOLE
```

From this point on, enter all commands only from the Series IV keyboard.

Copying iNDX Files to ISIS Files

The UP function allows the user to copy an iNDX file to an ISIS file. If an iNDX wildcard-pathname is specified as the iNDX source pathname, the destination-pathname must be a disk-directory. If the iNDX source-pathname is not a wildcard-pathname, the destination-pathname may be an ISIS-filename, a disk-directory or both.

If the destination-pathname is a disk-directory, the constructed ISIS file takes the name of the source file. If the file name is invalid under ISIS, an error message is returned. If the destination-pathname conflicts with the name of an existing file, you will be queried as to whether the user wants to delete the existing file and copy the source file to that file name. An existing file can be deleted by writing over it only if the file's write and format attributes are not set. By using the UPDATE option, the user can disable the querying process.

Copying ISIS Files to iNDX Files

The DOWN function allows the user to copy an ISIS file to a specified location within the iNDX hierarchy. If the ISIS source pathname is an ISIS wildcard filename, the iNDX destination pathname must be an iNDX directory name. If the ISIS source pathname is not a wildcard filename, the iNDX destination pathname may be either a data file or a directory file.

If the iNDX destination file exists and is a directory file, a file of the same name as the ISIS source file is created within the iNDX directory file if the user has ADD-entry access to that directory file. If the pathname constructed for the file exceeds the iNDX limit of 127 characters, an error message is returned.

If the iNDX destination file exists and is a data file, the user will be queried as to whether you want to delete the existing file and assign that name to the new file. Such deletion can occur only if you have DELETE access for the existing file. The user can suppress the querying by specifying the UPDATE option in the control line.

If the destination filename does not already exist, a new file of the specified name is constructed within the parent directory if you have ADD-entry access to the parent directory.

Returning the ISIS System to Independent Operation

During execution of the FPORT command, the ISIS system acts as a slave of the iNDX system. To return the ISIS system to independent operation, the user must use either the Exit option or the EXIT function of FPORT. If the user has a number of files to copy, enter the Exit option after copying the final file. If only one file is to be copied, enter the EXIT function in the command line and the ISIS system will return to independent operation as soon as the file has been copied.

Transmission of the file begins when the user terminates the command line on the Series IV system. When the command line is terminated, FPORT presents the following sign-on message:

```
operating system name FILE PORT, Vx.y
```

FPORT then performs a syntax check and checks the validity of the pathnames entered. If any errors are detected, an error message is generated and the command is aborted. In this case, you must re-enter the entire command.

User Interaction

If no errors are detected, the querying process begins. If the destination filename already exists, the user will be queried to see if they want to delete the existing file by copying the new file to this filename. The following message will appear if the user is in the interactive mode:

```
FILE ALREADY EXISTS
PATHNAME is pathname
DELETE EXISTING FILE?
```

If the user replies by entering a Y (or y) followed by <cr>, the command will continue to execute and the existing file will be deleted by being written over (provided they have delete access to the existing file). Any input other than Y (or y) causes the command either to go on to the next file to be copied (if the source file is a wildcard pathname) or to terminate. Since the responses to the queries are line edited, each response must be followed by <cr>. The querying process can be eliminated by specifying the UPDATE option in the command line.

The user specifies the QUERY option with any invocation of the FPORT command. Using this option causes the following query to appear before each file is copied:

```
COPY pathname TO pathname ?
```

If the user responds by entering a Y (or y) followed by <cr>, the FPORT operation will execute. Any other response causes the operation to go on to the next file to be copied (in the case of a wildcard pathname) and otherwise terminates unsuccessfully. Enter a line terminator after each console response or the response will not be accepted.

In submit mode, the QUERY option is not allowed; if the QUERY option is attempted, the following error message will appear:

```
QUERY OPTION NOT USEABLE IN SUBMIT MODE
```

Each successful FPORT copy operation results in the following message:

```
COPIED pathname TO pathname
```

If at any time the expected data is not received by either system, the following error message will appear:

```
DEVICE TIMEOUT ERROR
```

All operations will terminate and both FPORT programs (ISIS and iNDX) will be aborted. (Both programs loop while waiting to receive responses from the other system; if no response is received, the FPORT program will terminate (timeout) and control will return to ISIS or iNDX.)

Additional Notes

Program Restrictions:

1. FPORT must be used as a foreground job only.
2. FPORT does not support output devices :CO: and :BB: as valid pathnames.

Hardware Restrictions:

FPORT requires that the Interface cable (P/N 134514) shipped with the Series IV be connected between the J1 CH1/TTY connector on the back of the Series II and Serial Channel 2 on the back of the Series IV. The IEU must be in slot 2 (connector J9).

Error Handling:

The error handling is similar to and consistent with that defined for the copy utility program.

1. Syntax Error: Syntax is checked before operations begin. Invalid syntax causes a fatal error; an error message will be displayed before the command aborts. You must re-enter the command from the beginning.
2. Timeout Error: Whenever either system (Series II or Series IV) is expecting data but does not receive data, a timeout error occurs. Timeout errors and fatal errors abort operation on both systems (Series II and Series IV).

Examples

Observe the prompts at the Series II system and enter the responses shown below.

1. `S2FPORT`

   ```
   SERIES-II FILE PRT, Vx.y
   ENTER FPORT COMMAND at the SERIES-IV CONSOLE
   ```

   Enter the commands at the Series IV system and observe the console messages shown below.

2. `S4FPORT DOWN :F1:A TO /A/B`

   ```
   COPIED:F1:A TO /A/B
   ```

   This command copies the Series II file :Fl:A to a file of the Series IV called/ A/B. A message verifying the copy operation appears on the console of the Series IV.

3. `S4FPORT UP /A/B TO :F1:A`

   ```
   COPIED /A/B TO :F1:A
   ```

   This command copies the Series IV file /A/B to the Series II file :Fl:A. A message verifying the copy operation appears on the console of the Series IV.

4. `S4FPORT DOWN :F1:*.* TO /A UPDATE EXIT`

```
COPIED :F1:JUNK TO /A/JUNK
COPIED :F1:JUNK.LST TO /A/JUNK.LST
```

In this example, the ISIS wildcard pathname *.* is used. It specifies that all files in the Series II directory file (:F1:) are to be copied to the iNDX directory (/A). The UPDATE option suppresses the querying and writes over existing files. After each file in the directory has been copied, a verification message appears. The EXIT option specifies that after the last file in the directory has been copied, the Series II system should be returned to independent operation.

5. `S4FPORT UP /A/* TO :F1: QUERY`

```
COPY /A/JUNK TO :F1:JUNK ?
Y<cr>
COPIED /A/JUNK TO :F1:JUNK
COPY /A/JUNK.LST TO :F1:JUNK.LST ?
Y<cr>
COPIED /A/JUNK.LST TO :F1:JUNK.LST
```

In this example, the iNDX wildcard character (*) is used to specify that all files within directory /A are to be copied to the ISIS directory :F1:. The QUERY option specifies that the user should be queried before each copy operation.

6. `S2FPORT EXIT`

This command releases the Series II system from its slave status and returns it to independent operation.

Possible Error Messages

```
FILE DOES NOT EXIST
PATHNAME = pathname

INCORRECT FILE TYPE
PATHNAME = pathname
DIRECTORY FILE EXPECTED

FILE ALREADY EXISTS
PATHNAME = pathname

INSUFFICIENT ACCESS RIGHTS
PATHNAME = pathname
ADD ENTRY ACCESS TO DIRECTORY OR DELETE ACCESS TO
EXISTING FILE REQUIRED.

INSUFFICIENT ACCESS RIGHTS
PATHNAME = pathname
READ ACCESS REQUIRED

MASS STORAGE EXCEEDED
PATHNAME = pathname

PATH COMPONENT NOT A DIRECTORY FILE
PATHNAME = pathname

INVALID WILD CARD PATHNAME
PATHNAME = pathname

ILLEGAL COMMAND SYNTAX
```

```
DISALLOWED USE OF A WILDCARD CHARACTER IN PATHNAME
PATHNAME = pathname
DEVICE TIMEOUT ERROR
```

ICOPY

Syntax

```
ICOPY [READ  ISIS-source-pathname TO iNDX-destination-pathname ] [ [QUERY ] ]
      [WRITE iNDX-source-pathname TO destination-pathname       ] [ [UPDATE] ]
```

where:

ISIS-source-pathname	is an ISIS filename or an ISIS wildcard filename as specified in the *ISIS-II User's Guide*, Order Number 9800306. The ISIS-source-pathname may optionally be preceded by a disk-directory.
iNDX-destination-pathname	is a valid iNDX directory file or iNDX pathname.
iNDX-source-pathname	is a valid iNDX pathname or wildcard pathname.
destination pathname	is a disk-directory, an ISIS filename, or an ISIS filename preceded by a disk-directory.
QUERY and UPDATE	are options that determine if querying is to occur before files are copied.
disk directory(:Fn:)	may be :FO:, :F1:, :F2:, or :F3: for Floppy disks or :F6:, :F7:, :F8:, or :F9: for Hard disks.

Description

The ICOPY command has two forms: READ and WRITE. READ allows the user to copy ISIS files to a specified location in the iNDX file structure. WRITE allows the user to copy iNDX files to an ISIS file and, optionally, to an ISIS file pointed at by a disk directory.

Copying ISIS Files to iNDX

The READ function of the ICOPY command copies files from an ISIS source file to a destination file within the iNDX file hierarchy. If an ISIS wildcard filename is specified as the source file, the iNDX destination file must be an iNDX directory file. If the source file is not an ISIS wildcard file, the destination may be either an iNDX data file or a directory file.

If the iNDX destination file specified already exists and is a directory file, a file of the same name as the ISIS source file is created and placed in the iNDX directory file if the user has ADD-entry access to the directory file. An error message is returned if the filename so constructed exceeds the iNDX limit of 127 characters. If the iNDX destination file specified already exists and is a data file, the user will be queried to see whether they want to destroy the existing file by writing over it. If the user wants to write over the existing file, and has delete access rights to the existing file, the existing file will be deleted and the new data file will be created. The user may suppress the querying by specifying the UPDATE option in the command line.

If the iNDX destination file specified does not exist, a new file with the same name as the ISIS source file will be created within the parent directory if the user has ADD-entry access to the parent directory.

Copying a File From iNDX to ISIS

The WRITE function of the ICOPY command allows the user to copy an iNDX source file to an ISIS destination file. If the user specifies an iNDX wildcard pathname as the iNDX source file, the ISIS destination file must be disk-directory. If the source file is not an iNDX wildcard pathname, the ISIS destination file may be an ISIS file, a disk-directory, or an ISIS file pointed at by a disk-directory.

If the ISIS destination file is a disk-directory, a file of the same name as the iNDX source file will be constructed. If the filename so constructed is an invalid ISIS filename, a pathname syntax error message will be returned.

If the ISIS destination pathname already exists, the user will be queried to see whether they want to delete the existing ISIS file by writing over it. If they reply by entering yes, the existing file will be deleted and the new file will be written to that filename unless the write and the format attributes for the existing file have been set. You may disable the querying by specifying the UPDATE option in the command line.

If the ISIS destination file does not yet exist, a new file will be created within the ISIS directory.

If the user selects a hard disk as the disk-directory, ICOPY will read the disk-directory to determine if the disk is an iNDX disk or an ISIS disk. If the disk is an iNDX disk, all files copied to it will be Public files (i.e., their Public attribute, but no other attribute, will be set).

Regardless of the hardware configuration, ICOPY always assumes the disk-directory configuration to be the iNDX configuration, as shown below:

:F0: to :F3: — represent Floppy drives 0 to 3, respectively, with :F0: as the default.

:F6: — represents the first Hard disk fixed platter.

:F7: — represents the first Hard disk removable platter.

:F8: — represents the second Hard disk fixed platter.

:F9: — represents the second Hard disk removable platter.

Physical Interaction

Physical interaction is required to specify the type of ICOPY operation to be performed. The command begins to execute as soon as the user terminates the command line by entering <cr>. ICOPY then returns the following sign-on message to the console screen:

```
operating-system name ICOPY, Vx.y
```

ICOPY then checks all pathnames for valid syntax. If any syntax error is detected, the command aborts and the user must re-enter the command from the

beginning. If the destination file already exists and is a data file, and the UPDATE option has not been selected to disable the querying process, the following query will appear on the screen:

```
FILE ALREADY EXISTS
PATHNAME = iNDX-pathname
DELETE EXISTING FILE ?
```

If the user replies by entering a Y (or y) followed by <cr>, the command will continue to execute and the existing file will be deleted by being written over (provided they have delete access to the existing file). Any input other than Y (or y) causes the command either to go on to the next file to be copied (if the source file is a wildcard pathname) or to terminate. Since their responses to the queries are line edited, each response must be followed by <cr>. The querying process can be eliminated by specifying the UPDATE option in the command line.

The user specify the QUERY option with any invocation of the ICOPY command. Using this option causes the following query to appear before each file is copied:

```
COPY source-pathname TO destination-pathname ?
```

If the user replies by entering Y (or y) followed by <cr>, the file will be copied. Any response other than Y (or Y) causes the command to either go on to the next file to be copied (if the source file is a wildcard pathname) or to terminate. Since responses are line edited, they must be followed by <cr>.

If the QUERY is used when in Submit Mode, the following error message will result:

```
QUERY OPTION NOT USEABLE IN SUBMIT MODE.
```

Each successful copy operation produces the following verfication message:

```
COPIED source-pathname TO destination-pathname
```

Additional Notes

Program Restrictions:

1. IEU must be present at the top of the user memory.
2. The hard disk controller is configured differently for ICOPY than it is for the Series IV Operating System. Both programs cannot access the hard disk at the same time.
3. ICOPY can be invoked in whichever partition has ISIS assigned to it. If ISIS (IEU memory) is assigned to the background, but the background is not active, ICOPY may be executed from the foreground.
4. ICOPY will copy only Public files from any NDS-I files.
5. ICOPY does not support single density disks.
6. In both the Read and Write functions of ICOPY, physical output devices such as :CO:, :BB:, :TO:, :LP:, etc., are not supported.
7. Wildcard pathnames used as destination pathnames are not supported.

Hardware Requirements:

ICOPY requires that the Series II hardware disk controller be installed in the Series IV chassis. Before installing the controller boards in the Series IV, do the following:

A. For Double Density Floppy Disk Controller:

1. Set the switches on the CHANNEL board to 78H port address as in ISIS. (Switches 1-3 and 8 ON, 4-7 OFF.)
2. Set the rotary interrupt switch on the board marked INTERFACE completely in the clockwise direction-turning it past interrupt number 8.
3. Plug the boards into the Series IV chassis. The board marked INTERFACE must be slot 9, 8 or 6 (where slot 1 is closest to the CRT and slot 10 the farthest).

B. For Hard Disk Controller:

1. Set the switches on the board marked CHANNEL to 70H port address. (Switches 4, 6, 7 and 8 ON, all others OFF.)
2. Set the rotary interrupt switch on the CHANNEL board completely in the clockwise direction, turning it past interrupt number 8.
3. Set the switches on the board marked INTERFACE as follows:

   SW1-set all switches to the OFF position.
   SW2-set all switches to the OFF position.
4. Plug the board into the Series IV. The board marked CHANNEL must be in slot 9, 8 or 6 (where slot 1 is the slot closest to the CRT and slot 10 the farthest).

Error Handling:

Error handling is the same as that defined for the COPY command.

Syntax Errors: Syntax is checked before any operations begin. Invalid syntax is a fatal error; an error message will be displayed before the command aborts. You must then re-enter the command from the beginning.

Disk Errors: Disk errors are fatal errors resulting from unsuccessful disk I/O operations on the controller boards. Such errors result in an error message; the command aborts.

IF

Syntax

```
IF argument { = } argument
            { < > }
    commands
   [ORIF  argument { = } argument [...]]
            { < > }
   [ELSE commands]
END
```

where:

argument is a CLI variable value, a CLI variable name or a formal parameter.

commands is a set of one or more console commands.

Description

The IF command is executable only within a command file. IF provides conditional execution of one command set. The command set choice is based upon the result of successive logical evaluations. In each evaluation, the first argument value is checked for equality or inequality against the second argument value. Each value may be a CLI variable value, variable name, or parameter. Substitutions for CLI variable names and for formal parameters are made at command file invocation. Generally, the value of a CLI variable name or parameter represents a filename to be tested to determine if the command set is to be executed on that file. The command set consists of one or more commands.

If the IF comparison evaluates to TRUE, the command set immediately beneath it is executed. If the IF comparison does not evaluate to TRUE, the ORIF comparison is evaluated. If the ORIF comparison evaluates to TRUE, the command set immediately beneath it is executed. If the ORIF comparison does not evaluate to TRUE, the command set immediately beneath the ELSE line is executed. If only the IF comparison is used and it evaluates to FALSE, no commands are executed. The ORIF and ELSE comparison lines are optional. More than one ORIF line may be used. Also, the ORIF comparisons are checked sequentially. Only one ELSE line is allowed.

Examples

1.
```
IF %STATUS < > "0"
   READ %ABC
END
```

This example checks to see if the completion code returned by the operating system (which is referenced by the predefined CLI variable name STATUS) is equal to 0. If the returned completion code equals 0, an open parameter file is read and assigns the value read to the CLI variable whose name is defined by %ABC.

LNAME

Syntax

```
LNAME [ { DEFINE logical name FOR pathname [UPDATE] }
      [ { REMOVE logical name                       }
      [ { PATH                                      } ]
```

To create a logical name:

```
LNAME DEFINE logical name FOR pathname [UPDATE]
```

where:

logical name	is a user defined string of up to fourteen characters which can be used to reference a directory. A logical name has the same syntax as a fully qualified pathname.
pathname	is the pathname for a directory.
UPDATE	is an option that automatically executes the command even if the logical name has previously been assigned; UPDATE refers the logical name to the new pathname.

To delete a logical name:

```
LNAME REMOVE logical name
```

where:

logical name	is the user defined string used as a logical name.

To display a logical name:

```
LNAME [PATH]
```

where:

PATH	displays each logical name and its associated directory pathnames.
LNAME ‹cr›	displays a list of currently defined logical names—no directory pathnames displayed.

Description

Logical names are user-defined character strings that are equivalent to pathnames. Logical names are easier to remember than pathnames and often have greater meaning to the user. The LNAME command assigns a logical name string to a pathname. If the logical name chosen conflicts with an already existing logical name, the user is queried:

```
LOGICAL NAME ALREADY EXISTS, REDEFINE?
```

Entering a Y (or y) causes the command to be executed and causes the existing logical name to be redefined. Any response other than Y (or y) aborts the command. The UPDATE option displays this querying.

Additional Notes

Logical names may only be defined for directory files. Logical names are defined for the duration of a log-on session only; they do not transfer to background and remote jobs. Since the background and foreground are totally separate, the same logical name may be used in both areas.

Examples

1. `LNAME DEFINE X FOR /A/B`

   The directory specified by the pathname /A/B is assigned the logical name X and can be accessed as such.

2. `LNAME REMOVE DATA`

   The logical name DATA is deleted. The directory previously accessed by this logical name must now be accessed by its pathname.

3. `LNAME DEFINE '' '' FOR /FILE/A/C`

   The null logical name is assigned to the directory /FILE/A/C. That file can now be accessed simply by entering a space. Assume that data files /DATA1 and /DEBT are in the directory file /FILE/A/C. After defining the null logical name for /FILE/A/C, these data files can be accessed as simply (space) DATA1 and (space) DEBT.

4. `LNAME PATH`

   Each logical name and its associated pathname is listed.

LOG

Syntax

```
LOG {:BB:    } [APPEND]
    {pathname}
```

where:

pathname	is a fully qualified pathname or a logical name.
APPEND	adds the log file to an existing file.
:BB:	is the Byte Bucket.

Description

The Log command creates a log of all console output within the file specified. If the output filename specified is the same as an existing filename, APPEND will add the log file to the existing file. If APPEND was not specified, the existing file will be destroyed. The log continues to function until a second command is issued to disable it. The Byte Bucket (:BB:) is used to disable the log process.

Examples

1. `LOG /ONE.VOL/MARK.DIR/FILES3.EXT/FILES3.LOG`

   A log of console activity is created under the filename ONE.VOL/MARK.DIR/FILES3.EXT/FILES3.LOG.

2. `LOG:BB:`

   The current log file is closed and further logging is disabled until another LOG command is received.

3. `LOG /SYS.VOL/BIGLOG APPEND`

   This example appends the log file to the end of the existing file /SYS.VOL/BIGLOG.

LOGOFF

Syntax

```
LOGOFF [Exit]
```

Description

This command terminates the current user session. All environment information, including logical names, is deleted.

When the Exit option is used to Logoff from Partition 2 (in Toggle mode) or User 2 (in Multi-user mode) the Series IV is automatically set in single-user mode and the memory allocated to the partition or region 2 then becomes available to the foreground. The Exit option has no effect if specified in the User 1, Partition 1, foreground, or background regions. After executing the Exit option, to make the region available to the second user or second partition, either the Region command must be invoked or the system rebooted (if the Multi-user mode is configured in SYSGEN).

LOGON

Syntax

```
LOGON username [INIT [filename]]
               [NOINIT         ]
```

where:

username	identifies the user to the operating system.
filename	is a pathname.
INIT and NOINIT	indicate whether the user environment is initialized with a command file (INIT is the default).

Description

LOGON allows an authorized user entrance to the iNDX operating system.

If your system has a configured mass storage device, the following prompt appears after the user name is entered:

```
PASSWORD PLEASE
```

The user's assigned password, followed by <cr>, must then be entered. The password does not appear on the screen. Also, the carriage return must be typed before the system will verify the LOGON. Line editing is not allowed when entering the password. If an incorrect character is entered, press <cr>, observe the new prompt, and enter the password correctly. When a mismatch occurs, the system will prompt for the correct password three times. If all attempts fail, the user will have to enter the LOGON command again.

System access will be granted if the entered username is found in the predefined system USERDEF file and the entered password corresponds to the username.

A successful logon also results in two logical name assignments being automatically assigned to the user's home directory; they are the null logical name, " " (blank), and ":WORK:".

The INIT option determines which command file will be executed to initialize the user's execution environment. Initialization normally consists of assigning logical names, selecting certain system options, etc.). If neither INIT nor NOINIT are specified, the default will be INIT and a default filename of INIT.CSD will be used. Unless NOINIT is specified, the following will occur:

1. If a filename is specified, the existence of the file will not be verified. The filename will be submitted for execution as the initialization file.
2. If INIT was specified and a filename was not entered, a default file INIT.CSD is searched for in the home directory. If INIT.CSD does not exist in the home directory, a search will be made to determine whether it exists in the System Volume Root Directory. If it does not exist in either directory, no initialization will be performed.
3. If a filename entered ends with a period, the filename (with the period truncated) will be used as an initialization file.
4. If the filename entered does not end in a period, the suffix .CSD is appended.

Additional Notes

1. LOGON is implicitly performed for BACKGROUND and IMPORT jobs. In these cases, initialization of the user environment will take place exactly as if you had logged on interactively in Foreground and did not specify either NOINIT or an initialization file i.e., if an INIT.CSD file exists in your home directory or in the System Volume Root Directory, it will be executed as an initialization file.
2. If an INIT.CSD file is present on your home directory or on the Volume Root Directory but you do not want that file executed, NOINIT or another initialization filename must be specified.
3. If an invalid username is entered, or an invalid password is entered 3 times in succession, the LOGON process is restarted. Consequently, the INIT/NOINIT filename options must be re-entered.

Examples:

```
LOGON HARRY NOINIT
PASSWORD PLEASE BLUE
```

HARRY logs on to the network and correctly enters the password BLUE. No initialization file is searched for.

```
LOGON HARRY INIT (SET.)
PASSWORD PLEASE BILE
PASSWORD PLEASE BLUE
```

This time HARRY logs on and specifies that initialization information is to be obtained from file SET. The password BLUE is incorrectly entered as BILE. The prompt reappears and the password is then correctly entered.

```
LOGON HARRY INIT (ENV)
PASSWORD PLEASE BLUE
```

In this example, HARRY successfully logs on and specifies ENV as the initialization file. Since the filename is not followed by a period, the system searches the user home directory for a file named ENV.CSD to use as an initialization file.

```
LOGON HARRY INIT
PASSWORD PLEASE BLUE
```

HARRY successfully logs on and specifies the INIT option but does not specify an initialization file. The operating system attempts to execute INIT.CSD as the INIT file. The same results could be achieved without specifying INIT.

MOUNT

Syntax

```
MOUNT device-name
```

where:

device-name is:

FL0, FL1 for 5¼ flexible disks
WM0 for the integrated Winchester disk
WD0–WD3 for a Winchester 35-Megabyte hard disk
HD0–HD3 for a fixed platter hard disk

Description

The MOUNT Command allows the user to explicitly request a device to be mounted; each device represents a different volume in the iNDX file system. The Mount command enters the device's volume root directory into the system root directory. If the name of the volume root directory being entered conflicts with an existing volume name of the local file system, the new disk will not be mounted. If the new volume name conflicts with a public filename, the mount will take place because the new volume name "shadows" the existing public volume name. When the user enters a pathname that contains the volume name as a component, the system will access the local filename. The public volume name is effectively hidden and is not accessible (shadowed).

The mount operation may be performed only on mass storage devices. Unless a device has been operationally dismounted (DISMOUNT), it can usually be accessed without using the MOUNT command. A floppy volume is mounted when it is inserted into the drive. Hard disk volumes are mounted when the device is ready after power-up. If the Mount operation is requested on an already mounted device, the following message is displayed:

```
DEVICE ALREADY MOUNTED
```

This is not an error condition; the device will remain mounted. The file structure on the volume being mounted is integrated into the existing directory structure by logically adding the volume root directory as an entry in the system root directory. Entering DIR / will display all mounted device names.

Examples

1. `MOUNT WD0`—Mounts the 35-Megabyte Winchester disk.
2. `MOUNT HD1`—Mounts the removable hard disk.

OPEN

Syntax

OPEN *pathname*

where:

pathname is a valid pathname.

Description

The OPEN command is executable only in a command file. Before character strings can be read from a file on a mass storage device and assigned to CLI variables, the file must be opened. Only one such file can be open at any one time in a given command file. If a parameter file has already been opened in the current command file, it will automatically be detached and closed by the second OPEN command before the second file is opened.

OSCOPY

Syntax

OSCOPY *source device* TO *destination device* [OVERRIDE]

where:

source device	physical device where the operating system resides
destination device	target system device where the operating system is to be installed (a hard disk, WM0, WD0–WD3, HD0–HD3).
OVERRIDE	allows the user to execute from either memory partition with both partitions active.

Description

The OSCOPY command is used to install an operating system at system installation time. This command is used in the submit file SYSTEM.COPY, which is performed during installation. Also, the OSCOPY command is used to install the Series IV workstation operating system on the NRM system device.

The OVERRIDE option must be used in the Multi-user or Toggle modes or from the background region in Single-user mode. If the OVERRIDE option is not specified while both regions are active, an error condition will occur.

Example

```
>OSCOPY FL0 TO WD0<cr>
```

The operating system is copied from the flexible disk drive to the system Winchester in this example.

PDSCOPY

Syntax

```
PDSCOPY { READ (disk-directory) PDS-source TO iNDX-destination } [ { QUERY } ]
        { WRITE iNDX-source TO PDS-destination               } [ { UPDATE } ]
```

where:

PDS-source	is a valid PDS filename or a wildcard filename that can optionally be preceded by a disk-directory.
iNDX-destination	is a valid iNDX directory file or pathname.
iNDX-source	is a valid iNDX pathname or wildcard pathname.
PDS-destination	is a valid PDS pathname, optionally preceded by a disk-directory.
QUERY	is an option that produces a user query before each PDSCOPY operation.
UPDATE	is an option that disables the automatic querying of PDSCOPY.

Description

PDSCOPY allows the user to copy Personal Development System (PDS) files to a specified location within the iNDX file structure (READ) and to copy iNDX files from the iNDX file structure to PDS files (WRITE).

Copying PDS Files to iNDX Files

READ allows the user to copy a PDS file to a specified location within the iNDX hierarchy. If a PDS wildcard is entered as the source, an iNDX directory file must be designated as the destination file. If a wildcard is not designated as the source, the user may specify either a data or a directory file as the destination.

If the iNDX destination file exists and is a directory file, a file with the same name as the PDS source file will be created within the iNDX directory. If the pathname constructed for the destination file exceeds the iNDX limit of 127 characters, an error will be returned. The Read operation executes only if the user has ADD-entry access to the destination directory.

If the destination file exists and is a data file, the user will be queried before the delete takes places. Specifying the UPDATE option disables the querying. If the destination does not exist, a new file having the specified name will be created within the specified iNDX parent directory. The user must have ADD-entry access to the parent directory to execute this command.

Copying iNDX Files to PDS Files

WRITE allows the user to copy an iNDX file to a PDS file. If an iNDX wildcard pathname is designated as the source, the PDS destination must be a disk-directory; otherwise, the source may be either a PDS filename or a disk-directory.

If the destination is a disk-directory, a file with the same name as the source file will be created as the destination. If the pathname constructed is not a valid PDS pathname, an error will be returned.

If the destination pathname exists, the user will be queried prior to deleting the existing file. Deletion will take place only if the user requests it and the Write and Format attributes of the existing file have not already been set. If the Update option is specified, the querying will not take place before the deletion. If the file does not exist, a new file will be created within the PDS directory.

Regardless of the specific hardware configuration, the disk directory configuration is always taken by PDSCOPY to be one of the following:

:F0: — flexible disk drive 0 (the default condition)

:F1: — flexible disk drive 1

User Interaction

After the user has entered the command line for PDSCOPY, the following message appears on the console:

operating system name PDSCOPY, V*x.y*

PDSCOPY then performs a syntax check of the filenames entered. If any errors are detected, an error message will be returned and the command will abort. You must re-enter the command line with the correct filenames. If the filenames are correct, the command proceeds as described in the following paragraphs.

If the destination file already exists and is a data file, and you have not specified the UPDATE option (disabling query), the following query will appear:

```
FILE ALREADY EXISTS
PATHNAME = destination pathname
DELETE EXISTING FILE ?
```

If the user responds by entering a Y (or y) followed by <cr>, the command will proceed and the existing file will be deleted if the user has access to the file. Any input other than Y (or y) terminates the command and no files are copied. If a wildcard file was the source, the command will then query the user for the next file. If no more files exist, the command proceeds. Note that as the input for the PDSCOPY is line-edited, the user must enter a <cr> at the close of the command lines. If the QUERY option is specified, the following query will appear before each copy takes place:

COPY *source pathname* TO *destination pathname* ?

If the user responds by entering a Y (or y) followed by <cr>, PDSCOPY executes. Any response other than Y (or y) terminates the command. If the source was a wildcard, the command will query you for the next file. If no more files exist or the source was not a wildcard, the command will be exited. As the input is line-edited, the user must end each command line with a <cr>. In Submit Mode, the QUERY option causes the following message:

```
QUERY OPTION NOT USEABLE IN THE SUBMIT MODE.
```

Every successful copy causes the following confirmation message to appear:

COPIED *source pathname* TO *destination pathname*

Additional Notes

1. PDSCOPY can be invoked only from a foreground job.
2. PDSCOPY does not support the copying of spare files.
3. Wildcard pathnames cannot appear as the destination file.
4. For both READ and WRITE, the devices :CO:, :BB:, :TO:, :LP:, etc., are not supported.
5. Upon insertion of a PDS diskette into a Series IV, the iNDX Operating System attempts to mount the PDS diskette. This attempt fails, and a NOT SUPPORTED exception is returned. This message should be disregarded.
6. The possible errors are:

   Syntax Error: syntax is checked before operations begin. Invalid syntax is a fatal error and is displayed before aborting. You are required to re-enter the command from the beginning.

   Disk Error: fatal error. The program performed an unsuccessful Disk I/O operation on the controller boards. After this message is displayed, PDSCOPY is aborted.

READ

Syntax

```
READ variable-name [, . . .]
```

where:

variable-name is either the system defined CLI variable name (STATUS) or any valid user defined CLI variable name.

Description

The READ command is executable only in a command file. After the OPEN command has been used to access the appropriate file residing on a mass storage device, the READ command is used to take character strings from the file and assign them to the variable names given in the READ command. The READ command begins at the current position in the file and reads a character string for each variable name in the command. The nonalphanumeric characters are used to define where one string ends and another begins.

If the variable names given in the command line outnumber the successive strings available in the file, the variable names for those strings not present are each assigned a null character string.

If a read is attempted but no file has been opened, an error message will appear on the standard output device, all variable names listed in the READ command will be assigned null strings, and the STATUS variable will be set to a non-zero value.

Examples

1.
```
OPEN /SYSTMRT/DOGS
READ NAME1, NAME2, NAME3, NAME4
```

Assume that file /SYSTMRT/DOGS contains only the character string FIDO#ROVER$"OLE-BLUE". The variable name NAME1 will be assigned to the character string FIDO and NAME2 will be assigned the character string ROVER. The character string OLE-BLUE has been placed in quotes so the hyphen will be interpreted not as a delimiter, but as the literal hyphen character. Therefore, NAME3 is assigned the character string OLE-BLUE. Since no other character string is available from the file, variable name NAME4 is assigned a null string.

If the character string OLE-BLUE did not appear in quotes, the hyphen would have been interpreted as a delimiter, NAME3 would have been assigned the character string OLE, and NAME4 would have been assigned the character string BLUE. Failure to use the quotation marks would thus cause two dogs to be given incorrect names.

REGION

Syntax

```
REGION
```

Description

The REGION command is used to adjust the size of the memory regions, the priority of the regions, and the operating system mode. When REGION is invoked a display appears providing region status information:

```
REGION          SIZE(K)              CONSOLE     PRIORITY
-------------------------------------------------------
FOREGROUND       96                  PRIMARY     HIGHER
BACKGROUND       96     ISIS-IV      NONE        LOWER

SYSTEM OPERATING MODE =     SINGLE USER
SYSTEM PRIORITY       =     UNEQUAL

Oneuser Multiuser Toggle Priority   IsisIV Size Abort Exit
```

NOTE

In Toggle mode only the Priority and Toggle parameters may be changed. In Multiuser mode the parameters may not be changed, only displayed.

Abort—allows the user to leave the Region command without modification. Any changes that were done will be lost.

Exit—allows the user to leave the Region command and causes the changes to get incorporated into the operating system.

Selecting Operating Mode

Three operating modes are available at the Series IV—multi-user, single-user, and toggle. The mode may be selected by entering M for multi-user, T for toggle, or O for single-user.

NOTE

When in multi-user or toggle modes, to return to single-user mode requires the second user or second partition to Logoff with the Exit option (see the Logoff command). The operating system will then automatically return to single-user mode.

Unless otherwise specified in SYSGEN, single-user is the default mode. The following paragraphs provide descriptions of each mode.

Single-user mode. In this mode, the regions are identified as Foreground and Background. The console displays Foreground, and the Background region is accessed via the Background command. Only non-interactive jobs may be executed in the background. To change from another mode to single user, enter O (Oneuser) in the Region command. The default priority in single-user mode is foreground higher.

If only one job is running in the foreground, the system temporarily combines the two regions into one and allocates the combined region entirely to the foreground. Thus, it is not necessary to change the region size frequently to achieve maximum utilization of memory. This also ensures that if no background job is currently running, an 8-bit program (ISIS-IV) can always be executed in the foreground.

Multi-user mode. The multi-user mode should be selected when a second or remote terminal is attached to serial channel 1. In multi-user mode, the regions are identified as User 1 and User 2. The Series IV console is always User 1 and serial channel is always User 2. The regions are not accessible to each other. Batch is the default priority in multi-user mode.

Toggle mode. The toggle mode allows a single user to move from one region (called partition 1 and partition 2) to another by pressing the toggle key located to the right of the F7 function key. Batch is the default priority in toggle mode. There are two options wthin the toggle mode that affect the console display. The first prompt when selecting the Toggle option is:

```
Refresh screen when switching partitions?

    Yes    No
```

If yes is selected, pressing the toggle key causes the screen to be refreshed (the exited partition display is cleared from the console and replaced with the entered partition display). Selecting this option requires 4K of memory. When no is selected information from both partitions is displayed. The next option is:

```
Automatic control - S on unselected partition?

    Yes    No
```

If the response is yes, when the toggle key is pressed the exited partition display will freeze (control S) and the entered partition display will be activated (control Q). This option is useful when both partition operations are screen intensive. If, however, non-interactive processes are operating in a partition, it is much more effective for the unselected partition to continue running (select no). After entering the options, they will be displayed in the operating mode status line:

```
SYSTEM OPERATING MODE = TOGGLE Refreshed Enabled, Control-S Disabled
```

After the Toggle mode is configured, the message "TOGGLE MODE" will display in the OS Field. After the Toggle switch is pressed, however, partition information is displayed in the OS Field. Either "P1 has console" or "P2 has console" is displayed.

Selecting Console Priority

The priority option allows the user to specify which partition has first CPU access when processes are waiting for the CPU. Using the priority option disables existing default priorities. The options are Same, Unequal, Import, and Batch. When Same is selected, the priority is displayed as the same. When Unequal is selected, region 1 always has higher priority than region 2. When Import is selected, the region that is not in the import mode has higher priority. If neither or both regions are importing, the priority is the same. With the Batch option,

the region that is executing interactive processes has priority over the region running a Batch or Submit file, otherwise the two regions have equal priority.

Selecting ISIS-IV Capability

The region selected for ISIS-IV capability is the only region that can execute the ISIS-IV operating system. When I is pressed, ISIS-IV capability is switched to the region that currently cannot run the 8-bit operating system.

Selecting Region Size

The region size may be specified by entering S (for Size), the region (1 or 2, Foreground or background, depending on operating mode), and size in kilobytes. When the size for one region is chosen, the other region is automatically allocated the remaining memory. The minimum size allowed for a region is 59K, with the maximum limited by the amount of memory available. However as a practical constraint, many commands require 96K to execute.

Examples

1.
```
>REGION<cr>
iNDX-S31 (Vx.y) REGION VERSION x.y
REGION        SIZE(K)           CONSOLE    PRIORITY
-----------------------------------------------
FOREGROUND     96               PRIMARY    BATCH
BACKGROUND     96    ISIS-IV    NONE       BATCH

SYSTEM OPERATING MODE = TOGGLE Refreshed Enabled, Control-S Disabled
SYSTEM PRIORITY       = SAME
BOTH SYSTEMS ARE ACTIVE
■
-----Select Option                                TOGGLE MODE

                Toggle   Priority                 Abort   Exit
```

This example is a display of the Region screen when in the toggle mode. The mode cannot be switched unless Partition 2 is Logged off with the Exit option (refer to the Logoff command).

2.
```
>REGION<cr>
iNDX-S31 (Vx.y) REGION VERSION x.y
REGION         SIZE(K)            CONSOLE    PRIORITY
-------------------------------------------------
PARTITION 1     96                PRIMARY    SAME
PARTITION 2     96     ISIS-IV    NONE       SAME

SYSTEM OPERATING MODE = SINGLE USER
SYSTEM PRIORITY       = BATCH
■
-----Select Option

Oneuser  Multiuser  Toggle  Priority  Isis-IV  Size  Abort  Exit
```

This example is a display of the Region screen when in the single-user mode.

3.
```
>REGION<cr>
iNDX-S31 (Vx.y) REGION VERSION x.y

REGION          SIZE(K)             CONSOLE     PRIORITY
-------------------------------------------------------
USER 1           96                 PRIMARY     SAME
USER 2           96     ISIS-IV     SECONDARY   SAME

SYSTEM OPERATING MODE = MULTIUSER
SYSTEM PRIORITY       = BATCH
>
```

This example is a display of the REGION screen when in the Multi-user mode. The parameters can only be displayed, not changed, when in multi-user mode.

RELAB

Syntax

RELab *physical device* TO *volume name* [OVERRIDE]

where:

physical device	is the disk device name: FL0-FL1 5¼-inch flexible disks HD0-HD3 fixed and removable platter hard disks. WD0-WD3 35 MB Winchester 8-inch disk WM0 integrated 5¼-inch Winchester disk.
volume name	is the new volume root directory name of the physical device.
OVERRIDE	command option allowing Relab to operate while in a multitasking mode.

Description

The RELAB command provides the capability to relabel local devices. The RELAB command performs a logical dismount of the device to be relabeled and therefore cannot be performed on the system device. Additionally, because of the dismount, all logical names associated with the relabeled device must be reassigned.

The OVERRIDE option allows the user to relabel a device from either region 1 or region 2 when both regions are active. If the RELAB command is executed with both regions active and without the OVERRIDE option, the command will abort with an error message.

RENAME

Syntax

```
RENAME old-pathname TO new-pathname [UPDATE]
```

where:

old-pathname	is the pathname that presently identifies the file.
new-pathname	is the last path component or pathname that will identify the file subsequent to execution of the RENAME command.
UPDATE	disables the user interactive querying.

Description

The RENAME command renames a directory or data file by changing the last element of its path to the component specified by the new-pathname. The new-pathname may be just the last path component that is to be the new filename or a pathname. If a pathname is supplied, the directory it identifies must be the same as that identified by the old pathname. To use the RENAME command the user must have DELETE access to the given file.

If the new name to be assigned to the file conflicts with an existing filename, the following query is displayed:

```
PATHNAME = pathname
DELETE EXISTING FILE?
```

where:

pathname	is the conflicting pathname.

If the user responds to the query by entering a Y (or y), the existing file will be deleted and the rename will take place (provided the user has proper access rights). Any response other than Y (or y) will abort the command and the existing file will not be deleted. The UPDATE option disables the querying process. If the conflicting file is a directory file, the rename cannot take place and the following message will appear on the console:

```
FILE ALREADY EXISTS
NAME CONFLICTS WITH EXISTING DIRECTORY FILE

PATHNAME = pathname
```

Additional Notes

1. Wildcard pathnames cannot be used to specify either old or new pathname.
2. Directory files cannot be deleted as the result of a rename.
3. Renaming an existing file in submit mode without specifing the Update option results in an error message.
4. When more than one user can access the same file, using the RENAME command can prevent other users from accessing that file.

5. A file can be renamed while it is being accessed by a second user.

Examples

1. `RENAME /X/Y TO Z UPDATE`

   File /X/Y is renamed to Z. The user is not queried before the change takes place.

2. `RENAME :F3:PROGRM.SRC TO :F3: WORK.SRC`

   PROGRM.SRC, with its directory identified by logical name :F3:, is renamed to WORK.SRC. This example shows the use of pathname in newpathname, a useful application for ISIS compatibility.

REPEAT

Syntax

```
REPEAT
      [commands]
      [ [WHILE|UNTIL] argument [=|<>] argument [THEN] ]
      [commands]
END
```

where:

argument	is a CLI variable value, a CLI variable name, or a parameter.
commands	is a set of one or more commands.

Description

The REPEAT command is executable only in a command file. REPEAT allows one or more commands to be looped. The REPEAT command has two optional forms: the WHILE form and the UNTIL form. In the WHILE form, the command set is repeatedly executed (in loop fashion) as long as the logical comparison evaluates to TRUE. When the comparison evaluates to FALSE, the loop is exited.

In the UNTIL option, the command set is executed as long as the logical comparison evaluates to FALSE (that is, until it evaluates to TRUE). When the comparison evaluates to TRUE, the loop is exited. If neither a WHILE or UNTIL line is used, indefinite command set execution will occur.

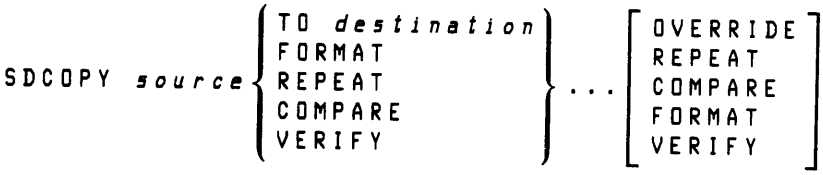
The logical comparison involves testing one string for equality or inequality against another string. Each string may be a CLI variable value, a CLI variable name, or a parameter. Substitutions are made before the command line is executed.

NOTE

The WHILE or REPEAT options may be used as often as necessary in a REPEAT block.

SDCOPY

Syntax



where:

source	source device name, i.e., either FL0 or FL1.
destination	destination device name i.e., either FL0 or FL1.
FORMAT	option that specifies disk formats of destination device before SDCOPY is performed.
VERIFY	option that specifies verification of the destination device after it has received the copy of the source.
COMPARE	option that specifies comparison of the destination to the source.
REPEAT	option specifying a repetition of the preceding operation (i.e., option) using the same source.
OVERRIDE	allows the user to perform an SDCOPY in Multiuser or Toggle modes with both regions active.

Description

SDCOPY allows the user to duplicate 5¼-inch flexible diskettes using a single drive system. SDCOPY provides track-for track copies of one disk to another with options that permit formatting the destination device and/or verification after the copy is made. Another option allows two diskettes to be compared for equality. Also, the SDCOPY command allows the user to change the volume name of the destination device (see Note 4). SDCOPY cannot operate in Submit mode.

User Interaction

After you have entered the SDCOPY command, the following message will appear:

```
operating system ID SDCOPY, Vx.y
PLEASE LOAD type DISK INTO device name
Type <CR> to continue, E to exit
```

where *type* is either SOURCE or DESTINATION.

When SDCOPY has completed an operation, the following confirmation message will appear:

```
SDCOPY COMPLETE
```

If the VERIFY or the COMPARE options were selected, one of the following messages will appear (depending on the results of the operation):

```
VERIFICATION OK
```

or

```
VERIFICATION FAILURES
```

SDCOPY assumes it has only one drive to work with unless you specify the optional TO clause. The single drive configuration is used when no TO clause is entered or is used if the destination specified in the TO clause is the same as the source. If the device-name is that of a system device, SDCOPY aborts and the following message appears:

```
CANNOT COPY DISKS USING A SINGLE SYSTEM DEVICE
```

If a Winchester disk is available, SDCOPY will use it as temporary storage. The entire source disk will be copied to the Winchester and only a single operation is required to accomplish the copy. The number of disk swaps is calculated as the total size of the source disk divided by the size of the work buffer. This number will be displayed and the following prompt requesting permission to continue will appear:

```
IT WILL BE NECESSARY TO SWAP DISKS number TIMES.
DO YOU WISH TO CONTINUE? ( [Y] or [N] )
```

Any response other than N (or n) is treated as Y, (i.e., permission granted).

If the TO option is specified with different devices for destination and source, no disk swaps are necessary. For example, with FL0 as the source and FL1 as the destination, the following message will appear:

```
PLEASE LOAD SOURCE DISK INTO FL0
PLEASE LOAD DESTINATION DISK INTO FL1
Type <CR> to continue, E to exit
```

Any entry other than E (or e) is taken as permission to continue.

If REPEAT option is specified, the same source may be used for multiple operations without requiring you to reload the source before each operation. This option is useful if a Winchester disk can be used as a workfile or if two operable flexible disks are available.

Possible Error Messages

All I/O and syntax errors that take place during SDCOPY operation are fatal. In case of a syntax error, the correct syntax will be displayed and the command will terminate.

Verification errors are recoverable. Operation will continue after one of the following errors is displayed:

```
VERIFY FAILED AT TRACK track number SECTOR sector number
```

or

```
MISCOMPARED AT TRACK track number SECTOR sector number
```

Additional Notes

1. Duplicating disks using a single system device is not allowed because the system device cannot be dismounted if the system is to remain operational.
2. SDCOPY can execute in Foreground or Background. To run in background the OVERRIDE option must be specified. This also includes the P2 region in toggle mode and User 2 region in multi-user mode. If the OVERRIDE option is not specified the following message appears:

   ```
   CANNOT RUN IN BACKGROUND PARTITION.
   ```

3. SDCOPY cannot be run in Submit mode because it queries the user. If you attempt to run SDCOPY in Submit mode, the command will terminate and the following message will appear:

   ```
   CANNOT RUN UNDER SUBMIT MODE.
   ```

4. SDCOPY allows you to change the name of the volume root Directory on the Destination Floppy Disk. When the copy is completed (Prior to SDCOPY complete message), SDCOPY will ask the following question:

   ```
   DO YOU WANT TO CHANGE THE VOLUME ROOT DIRECTORY NAME?
   ```

   A response of y or yes (uppercase or lowercase) will result in the following message:

   ```
   CURRENT VOLUME ROOT DIRECTORY NAME IS XVOL
   PLEASE ENTER NEW VOLUME ROOT DIRECTORY NAME
   ```

   You can enter a new volume root name of up to 10 characters.

SEARCH

Syntax

SEARCH [{ *pathname* or *logical name* / OFF }]

where:

pathname or *logical name*	designates a new system command directory for a user. The directory pathname length is limited to 14 characters.
OFF	sets the current search path to the default directory.

Description

The SEARCH pathname or logical name designates the directory that is searched for a command when that command is entered by the user. If a pathname or logical name is not specified, then the SEARCH command displays the existing search priorities. When the Search OFF option is entered, the default search condition is set. In the default condition, when the user enters a command, it is searched for in the following order:

1. the system volume root directory
2. null logical name (if defined)

When the SEARCH command is invoked, the search order is altered by the user to the following:

1. user specified search path
2. system volume root directory
3. null logical name (if defined)

In the event that the desired pathname is longer than 14 characters, a logical name may be specified in the Search command. The null logical name may be specified with the LNAME command (refer to the LNAME Description in this chapter).

When the Search command is entered to display the search priority (a carriage return after Search), the pathnames are shown in priority of the search path.

Examples

1. To simply display the existing search priorities, type:

```
>SEARCH<cr>
SEARCH PATH IS:  /ROOTVOL/CUSP
                 /ROOTVOL
                 /ROOTVOL/HOMEDIR
```

2. In the following example, the existing search priority is altered from the previous example to /ROOTVOL/SYSDIR. As shown below, first the logical name is defined (since the pathname is longer than 14 characters), the search priority is altered, then the search priority is displayed:

```
>LNAME Define 1 For /ROOTVOL/SYSDIR<cr>
>SEARCH 1<cr>
>SEARCH<cr>
 SEARCH PATH IS: /ROOTVOL/SYSDIR
                 /ROOTVOL
                 /ROOTVOL/HOMEDIR
```

3. In the final example, the Search path is set to the default condition and then displayed:

```
>SEARCH OFF<cr>
>SEARCH<cr>
 SEARCH PATH IS: /ROOTVOL
                 /ROOTVOL/HOMEDIR
```

SET

Syntax

```
SET variable-name TO ["] value ["]
```

where:

variable-name	is a valid CLI variable name (i.e., a string of up to six characters, alphabetic and numeric, the first character of which must always be alphabetic), or the predefined variable name STATUS.
value	is a character string. The quotation marks are necessary only if the string contains non-alphanumeric characters.

Description

The SET command is executable only in a command file. SET assigns a character string of up to 508 characters to a user-chosen variable name or to the system predefined name (STATUS). The character string may be a simple string or a conglomerate of CLI variable names, CLI variable values, and parameters. In the latter case, the system will first substitute names or parameters with their respective strings, then assign the resulting new name to a pure string. This process is made clearer in the following examples.

Example:

1. Assume the CLI variable names %X, %Y, and %Z have been previously assigned to character strings by the following commands.

```
SET X TO "for"
SET Y TO "peg thankx"
SET Z TO "1234"
```

X references the string for, Y references peg thankx and Z references 1234. If the user now enters the command:

```
SET A TO "%X%Y"
```

the CLI variable name A refers to the string formed by the concatenation of %X and %Y. Thus, A is defined as a reference to the character string "for peg thankx". The optional quotation marks are used to prevent any of the characters of the string from being interpreted as delimiter characters.

2. Now, consider this command:

```
SET B TO " "
```

The CLI variable name B refers to the null string (i.e., a string of zero characters). The null string can be useful when testing job execution. For example, the IF command can be used to test filenames against the null string. As long as the overall routine continues to find files, it will execute the rest of the procedure; once the null string is recognized, an exit from the loop will be taken.

3. `SET TNT TO "%Z" a.c %X`

Using the values from the first example, the string "1234" is substituted for the previously defined name, %Z, and the string "for" is substituted for the previously defined name, %X. Between these, the character string "a.c" is entered. The variable name %TNT can now be used to reference the character string "1234 a.c for".

It is also possible to redefine a variable name using the previously defined variable name as part of the new definition. For example, if the variable name %AK47 has previously been assigned to the character string "BANG", then the following command:

```
SET %AK47 TO %AK47"--"%AK47
```

reassigns the name %AK47 to be the character string "BANG--BANG."

SPACE

Syntax

```
SPACE /volume-name
```

where:

volume-name is the volume root directory for the given physical device.

Description

The Space command returns information about the amount of available space on a specified disk. Information is returned in the following format:

```
VOLUME GRANULARITY = number   (512 bytes per sector)
FREE BLOCKS        = number   (block = 512 bytes)
TOTAL BLOCKS       = number
FILES AVAILABLE    = number
TOTAL FILES        = number

MM/DD/YY      hh:mm:ss
```

where:

MM/DD/YY hh:mm:ss is the month, date, year, hours, minutes, and seconds at which the space information was generated.

number is a decimal integer.

Space will not accept logical names because volume names must be used.

Example

```
SPACE /A
```

The information is provided for the volume whose root directory is A.

STTY

Syntax

```
STTY [Baudrate (value) [GO]] [DISPLAY | NODISPLAY] [Terminal] [Primary | Secondary] [Remote | Local]
  [Config (filename)]
```

where:

`Baudrate (value)`	sets the baudrate indicated by (*value*) for serial channel 1.
`Go`	supresses the required carriage return <cr> when entering baudrate (for use in Submit mode).
`Config (filename)`	specifies the configuration file containing the configuration commands interfacing a terminal to serial channel 1 or the Series IV to a host computer.
`Terminal`	sets the Series IV into terminal mode.
`Display/Nodisplay`	enables/disables the display of the type-ahead feature.
`Primary/Secondary`	specifies which terminal will be affected by STTY execution.
`Remote/Local`	Remote switches the console to the serial channel, local returns the console to the Series IV keyboard and CRT.

Description

The STTY command allows the user to configure terminal characteristics and options into the operating system. This command also allows the user to switch control from the system console to a terminal connected to the serial channel 1.

Baudrate

The baudrate can be used to set a second terminal baudrate, or to match the baudrate of a modem to allow remote stationing of a terminal. The primary Series IV console baudrate cannot be changed.

To set the terminal to a new baudrate, type:

```
>STTY Baudrate ( XXXX )<cr>
```

where ***XXXX*** is the baudrate. A message will appear, allowing time to set any terminal baudrate switches, then enter a carriage return <cr> to execute. When the Go option is specified the baudrate is automatically set (without <cr>). This allows the user to set the baudrate for a remote terminal from a Submit file.

Configuring New Terminals

The CONFIG option provides the capability to configure user supplied terminals on the serial channel 1 of the Series IV. The config file can also be used for

Series IV keyboard customization and video attribute alteration (such as reverse video). The config file specified in this option should contain the configuration commands (escape sequences) used to control specific terminal functions. For full details on these commands and the configuration file, refer to Appendix D or the HELP.CFG located on the system volume root directory.

Type Ahead Display

The type ahead display function allows keystrokes entered while the system is executing a previous command to be displayed on the 25th line (right above the Syntax Guide). If the display is disabled, the keystrokes typed in are not displayed. The default condition is "Display". The Primary or Secondary option must be used along with the type ahead option to specify which terminal's type ahead characteristics will altered. If the Primary or Secondary option is not specified, the Secondary terminal will be changed by default.

Console Control

The Remote and Local options within STTY allow the user to specify which terminal controls the Series IV. The Remote option switches control to the terminal attached to serial channel 1. The terminal configuration must be entered (using the STTY Config option or in SYSGEN) before invoking the Remote option to ensure proper operation of the terminal. Also, the Series IV cannot be in the Multi-user mode (refer to Region or Sysgen commands) to use the Remote option. Local returns console control from the terminal to the Series IV integrated CRT and keyboard.

Primary and Secondary Terminal Operations

The Primary and Secondary options are used to specify which terminal the STTY command will affect. Primary is always specified as the Series IV console and the terminal attached to the serial channel 1 is Secondary. Using these options, the configuration characteristics of one can be altered without affecting the other. The Primary and Secondary options are used to specify which terminals are affected by the Baudrate, Configuration file, and Display/Nodisplay options.

Terminal Option

With the Terminal option, the Series IV can be used as a terminal and can be connected to other computers. The Series IV must be in single-user mode in order for this option to operate. Once enacted the Series IV interfaces through the serial channel 1 to a host computer (refer to Appendix D). To exit the terminal mode enter a tilde, then a period (~.).

Examples

1. `>STTY Baudrate ( 1200 ) Config (Term.cfg) Remote<cr>`

   This example sets the serial channel 1 baudrate to 1200 and the configuration file, then sets the console control to the remote terminal.

2. `>STTY Nodisplay<cr>`

   Disables the type ahead display on line 25.

3. `›STTY Baudrate ( 1200 ) Config (NRVDIO.cfg) Go‹cr›`

   This example sets the baud rate to 1200 and reconfigures the terminal with the Config option to disable reverse video (refer to Appendix D for details on the disable reverse video configuration file). The Go option allows this command to execute without the user having to enter a carriage return for the baudrate.

4. `›STTY Config (SIVKEY.CFG) Primary‹cr›`

   The Series IV keyboard (primary console) is customized as set by the SIVKEY.CFG file, the secondary console is not affected.

SUBMIT

Syntax

SUBMIT *pathname* [*parameters*] ... [{LOG *pathname* [APPEND] / NOLOG}]

where:

pathname	is the pathname of the command file to be executed.
parameter	is an actual value that replaces a formal parameter (a variable) in the command file. The maximum number of parameters allowed is 10.
LOG/NOLOG	are used to specify whether a log of the console output should be kept for the execution of the command file. The default is NOLOG.
pathname	is the pathname of an existing log file to be used with the append option.
APPEND	specifies that the console ouput will be appended to an existing log file.

Description

The SUBMIT command allows the user to automatically batch-process iNDX operating system commands in a command file. Before the SUBMIT can be executed, the user must create the command file using the editing capabilities of the AEDIT text editor or the BATCH command.

Additional Notes

1. The command file may contain any iNDX command. Control commands can be used to perform conditional and indexed looping, conditional command execution, and variable input from parameter files. The available control commands are:

COUNT	OPEN	SET
ELSE	ORIF	UNTIL
END	READ	WAIT
ENDJOB	REPEAT	WHILE
EXIT	RUN	

   Refer to the individual command descriptions for further details on these commands.

2. When the command file is created, formal parameters can be used to pass values from the SUBMIT command to the command file. Additionally, parameters can be passed to the command file by reading a parameter file.

   The optional parameters specified in the SUBMIT command line are actual parameters to be substituted for formal parameters imbedded in the command file. In creating the command file, formal parameters may consist of two characters, *%n*, where *n* is a digit from 0 through 9. Formal parameters may be placed anywhere in this file. To enter a percent sign that is not to be interpreted as a formal parameter, enclose it in single quotes.

In the SUBMIT command, a parameter is a character string of up to 36 characters. All ASCII characters with values between 20H and 7EH are legal except for the delimiters listed in Chapter 2. If a parameter contains a delimiter, it must be enclosed in quotation marks. To omit a parameter from the SUBMIT list, enter a comma in its place, i.e.,

(12,,BA)

indicates %0 = 12, %1 is null, and %2 = BA.

3. If LOG and no parameters are specified, a log file of the console ouput will be created with a filename the same as the command file with the extension changed to .LOG. If LOG and parameters are specified, the log filename created is the same as the temporary file created when SUBMIT is invoked (see note 4 below), except the extension is .LOG. To APPEND a console output to an existing log file, the pathname of the log must be entered, then APPEND.

4. If the file name of the command file does not have an extension, SUBMIT adds the extension .CSD before accessing the file. If the pathname has a trailing period, the period is removed, but no extension is added.

   When the SUBMIT command is invoked, the command file is preprocessed to substitute actual parameters (parameters specified in the command line) for formal parameters (parameter variables in the command file). If there are no actual parameters specified, the formal parmeters will be replaced by null strings. After parameter substitution, a file is created identical to the command file with a different filename. The substitute file is placed in the same directory as the command sequence file, but the last component of its pathname is created from the last component of the command sequence file's pathname as follows:

   - the original filename is truncated after the third character or before the first period, whichever comes first.
   - a unique 3-character alphanumeric string, assigned by the system, is appended to the filename.
   - the extension .CS is appended to the filename.

   The substitute file is then executed and deleted upon completion.

5. The SUBMIT command is also valid within a submit file. A SUBMIT command within a submit file is referred to as "nested." The depth of such nesting is limited to 100.

6. When using a submit file, a CTRL C will stop execution of the current commmand line and not the entire file.

7. If the same command file is simultaneously executed in the foreground and background with no parameters specified in the SUBMIT command, and no log file name specified in either BACKGROUND or SUBMIT command, only one log file will be produced. For example, the commands:

```
SUBMIT 1/A LOG
BACKGROUND 1/A LOG
```

   will both create a log file with the name 1/A.LOG. To prevent this from happening, the user may enter a null parameter (' ') in the SUBMIT command to create a log file with a different filename as shown in the following example.

```
SUBMIT 1/A (' ') LOG
```

Examples

1. `SUBMIT /SYS/MY/CMDFILE LOG <cr>`

```
Pathname of command file:    /SYS/MY/CMDFILE.CSD
Pathname of executed file:   /SYS/MY/CMD02U.CS
Pathname of log file:        /SYS/MY/CMDFILE.LOG
```

In this example, since the command file does not contain an extension or terminate in a period, the extension .CSD is appended to the command file. Since no actual parameters were specified, any formal parameters in the command file are replaced with null strings. A temporary file CMD02U.CS is created and executed. Since no parameters were specified, the log file is the same as the command file with the log extension, CMDFILE.LOG.

2. `SUBMIT /SYS/MY/CMDFILE. LOG (/SYS/MY/MYJOB.LOG)<cr>`

```
Pathname of command file:    /SYS/MY/CMDFILE
Pathname of executed file:   /SYS/MY/CMDXXY.CS
Pathname of log file:        /SYS/MY/MYJOB.LOG
```

Initially the period terminating the command file is removed. With no actual parameters specified, null strings are substituted for any formal parameters within the command file. The preprocessed command sequence is written to the new file /SYS/MY/CMDXXY.CS, and the console log is written to MYJOB.LOG.

3. `SUBMIT /SYS/MY/CMDFILE.XXX LOG <cr>`

```
Pathname of command file:    /SYS/MY/CMDFILE.XXX
Pathname of executed file:   /SYS/MY/CMD02I.CS
Pathname of log file:        /SYS/MY/CMDFILE.LOG
```

4. `SUBMIT /SYS/MY/CMDFILE (/VRD,1) LOG <cr>`

```
Pathname of command file:    /SYS/MY/CMDFILE.CSD
Pathname of executed file:   /SYS/MY/CMD01E.CS
Pathname of log file:        /SYS/MY/CMD01E.LOG
```

When the preprocessing begins, the command file has the .CSD extension added. The specified parameters /VRD and 1 are then substituted into the command file and the temporary file is created and executed. Since parameters were specified, the log file name is obtained from the temporary file with the log extension added.

5. `SUBMIT /SYS/MY/CMDFILE. (/VRD,1) LOG (/SYS/MY/MYJOB.LOG)<cr>`

```
Pathname of command file:    /SYS/MY/CMDFILE
Pathname of executed file:   /SYS/MY/CMD2US.CS
Pathname of log file:        /SYS/MY/MYJOB.LOG
```

This example illustrates the console log pathname specified in the command line.

6. `SUBMIT /SYS/MY/CMDFILE.XXX (/VRD,1) LOG (SYS/MY/MYJOB.LOG) APPEND<cr>`

```
Pathname of command file:    /SYS/MY/CMDFILE.XXX
Pathname of executed file:   /SYS/MY/CMD022.CS
Pathname of log file:        /SYS/MY/MYJOB.LOG
```

The APPEND switch in this example causes the console log to be added to the existing log file /SYS/MY/MYJOB.LOG. If /SYS/MY/MYJOB.LOG does not exist, it will be created.

SYSGEN

Syntax

```
SYSGEN
```

Description

The SYSGEN command configures operating system parameters and peripheral devices into the Series IV operating system. The system must be re-booted before the configuration files modified within SYSGEN will be modified.

NOTE

The Sysgen information documented in this manual is only for configuring a standalone Series IV. If the Series IV is in an NDS-II network, refer to the Sysgen information in the *NDS-II Network Resource Manager User's Guide.*

The options that can be configured at the Series IV include:

- adding a Winchester Peripheral box or Model 740 hard disk into the system.
- altering the default access rights to new user files.
- establishing caches for the operating system overlays for performance enhancement.
- setting up buffers that allow the Series IV processes to be traced, in the event of an operational error.
- defining the operating system mode when the Series IV is initialized.
- setting the default region that is ISIS-IV compatible.
- defining the default characteristics of the memory partitions.
- establishing the characteristics of the primary and secondary terminals.

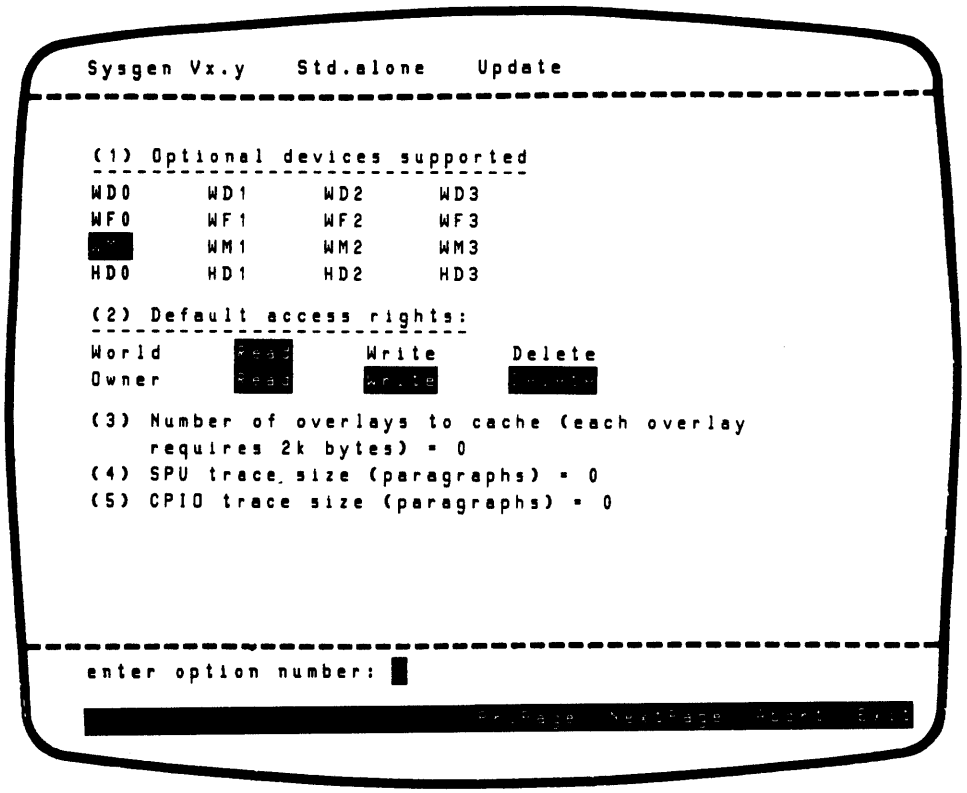
NOTE

All entries that appear on the prompt field (reverse video—bottom line) may be entered by pressing the associated function key.

SYSGEN Screen

After entering SYSGEN, the following screen will appear.

SERIES IV OPTION SCREEN



When the prompt appears:

```
Enter option number_
```

Enter the option number (one through five) and <cr>, or N (for next page), or P (for previous page). Page 2 has options 6 through 9 and Page 3 has options 10 and 11. A description of options 1 through 5 is given below.

OPTION 1 Configures the peripheral devices attached to Series IV workstation. All local disk devices attached to the Series IV should be entered. When option ! is selected, the following prompt appears:

```
Select/de-select devices, (selected devices are highlighted).
 Select    De-select                          Abort    Exit    Return
```

Position the cursor with the arrow keys to the device name that you want to select or de-select. The following list provides an explanation of the table entries.

Device Name	Drive Type
WD0-WD3	35 MB Winchester disk drives.
WF0-WF3	Not used at the standalone Series IV
WM0-WM3	10 MB integral Winchester disk drive.
HD0-HD3	10 MB Model 740/743 hard disk drive.

Select

Press S or s.
Press after positioning the cursor at the device name to be configured. The selected device is highlighted immediately. This device selection process can be repeated until a carriage return <cr> is entered to return to the option screen.

De-select

Press D or d.
Press after positioning the cursor at the highlighted (selected) device name to remove the device from the operating system configuration.

OPTION 2

The second option is the selection of the default access rights to all files created on the system. The default access rights to his own files (OWNER) and other users default access rights to files (WORLD) can be selected in this option. Suggested access rights are WORLD NONE and OWNER ALL. The ACCESS command overides these default access rights. When this option is selected, the prompt line displays:

```
Select/ Deselect, (selected access rights are high-lighted).
Select      De-select                          Abort    Exit    Return
```

Move the cursor to the access right with the arrows keys, then select (press S or s) or deselect (press D or d) the acess rights chosen. Press <cr> to return to the select option prompt.

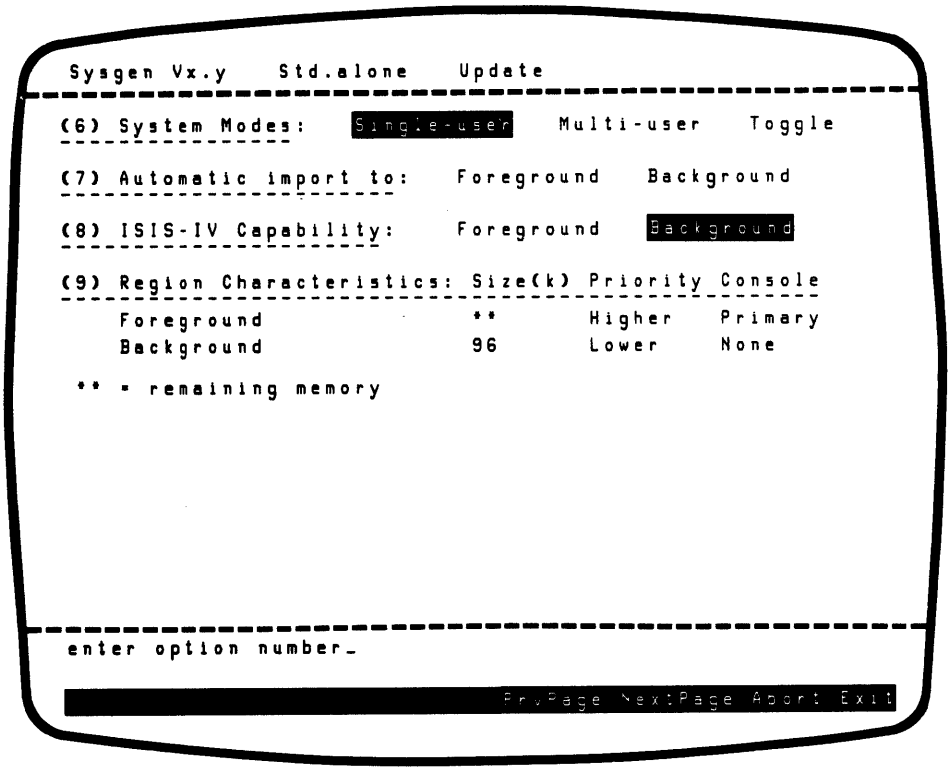
OPTION 3

The third option, numbers of overlays to cache, gives the user the ability to specify the number of operating system overlays that will be held in cache. The possible range is 0 to 40 overlays, with each overlay taking 2K of memory. As a guide for optimum performance, it is best to use between 7 to 12 overlays, but maintain a minimum of 192K of user memory.

OPTIONS 4 & 5

The SPU and CPIO trace size, allows the operations performed on those boards to be recorded, in case of a system fault. Normally the trace size should be set to zero.

SERIES IV OPTION SCREEN—PAGE 2



OPTION 6

The system mode option determines the use of the memory partition (established with option 9 or the Region command). The modes include:

Single-user Mode—allows partition 2 to be used as a "background ". The user accesses the background by using the Background command.

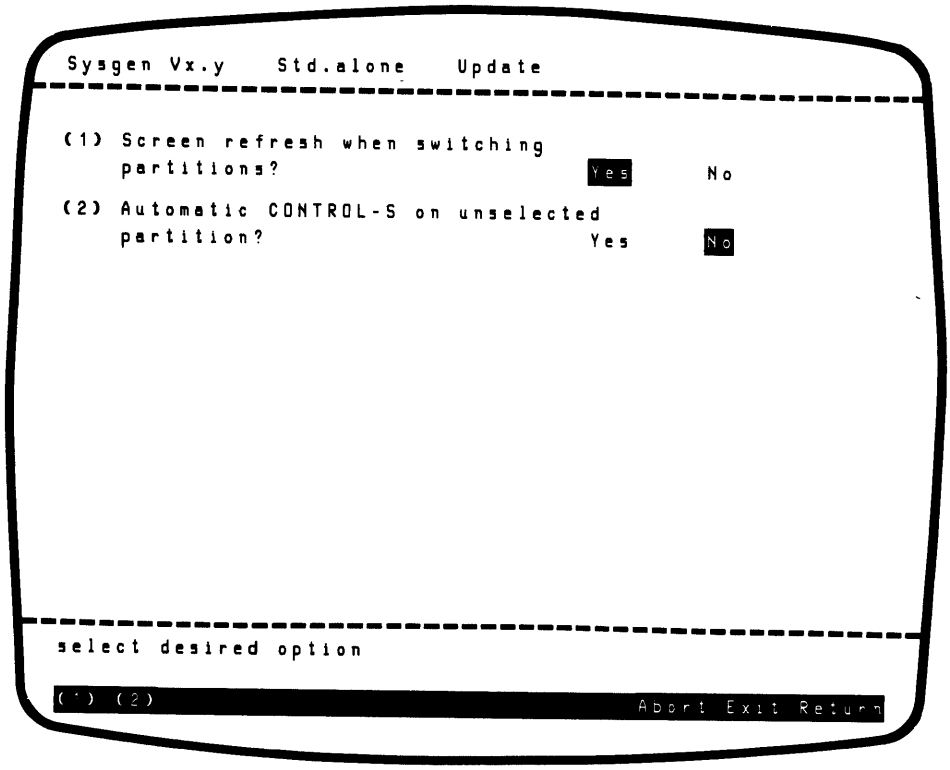
Multi-user Mode—the second partition is used by a second user (User 2). The second user feature is a system option, with a separate terminal attached to serial channel 1. The second partition then becomes inaccessible to User 1.

NOTE

When selecting the Multi-user mode, the baudrate must be set with either SYSGEN option 11 or the STTY command. If the second user terminal baud rate is not set, the second user will hang.

Toggle Mode—allows a single user to switch between partitions. When the toggle mode is selected the following Toggle Option Screen appears:

TOGGLE OPTION SCREEN



To change or select a toggle option enter the toggle option number (1 or 2) then <cr>.

Toggle option 1—select option 1 to specify screen refresh on toggle. If yes (Y) is selected, the screen is refreshed—when the toggle key is pressed the exited partition display is erased and the entered partition display is put on the screen. If no is selected (N), the console continues to display the exited partition as well as the entered partition.

NOTE

The toggle mode screen refresh option requires 4K of memory at the Series IV. If adequate memory is available (some commands require 96K), it is recommended that the refresh be used. Without the refresh on, the screen may appear confusing after a few toggles have been performed.

Toggle option 2—toggle option 2 determines if the screen is frozen (Control S) upon exiting the partition, and re-activated in entering a partition (control Q). If performing screen oriented tasks it may be useful to enter Y to cause the display to freeze when exiting the partition (pressing the toggle key). Entering N allows the screen to continue after the partition has been exited.

OPTION 7

Option 7 is only for use in the NDS-II network, and does not apply to a standalone Series IV.

OPTION 8

Option 8, ISIS-IV capability, specifies which region may run ISIS-IV. Only one region at any given time can have ISIS-IV capability. The regions will appear as Foreground/Background (single-user mode), User 1/User 2 (multi-user mode), or Partition 1/Partition 2 (toggle mode) depending on the system mode (option 6). When option 8 is selected, the prompt appears:

```
Select desired option
Use arrow keys to position cursor.
Select  De-select  Abort Exit Return
```

Use the arrow keys to move the cursor to the desired region, and enter S to select a region or D to deselect a region. Only one region may be selected at a time (SYSGEN automatically de-selects a region if the other region is selected).

OPTION 9

Allows the user to set the default region characteristics on system initialization. The REGION command will override these conditions. Characteristics include the size and priority of each region. The system mode set in option 6 determines if the regions are labeled Foreground/ Background, User 1/User 2 or Partition 1/Partition 2 for Single-user, Multi-user, or Toggle modes. When option 9 is selected (in Multi-user mode), the prompt appears:

```
Select desired option

 1st User 2nd User Abort Exit Return
```

Press 1 or 2 to allow region alteration of the respective regions Then select the characteristic to be changed:

```
Select desired option

Size     Priority   Abort  Exit  Return
```

Size. Press S to alter the region memory size or P to change the region priority. When the Size is changed SYSGEN displays:

```
Enter new number
```

Enter the amount of memory (in kilobytes) desired for the specified region. When one region's memory is specified, the alternate region is automatically allocated the remaining memory. A practical guide to the minimum amount of memory to allocate may be 96K, since some commands require that to operate.

SYSGEN will not indicate if memory size of the region specified is larger than existing memory. If this condition occurs, the Series IV will not boot and SYSGEN must be reperformed.

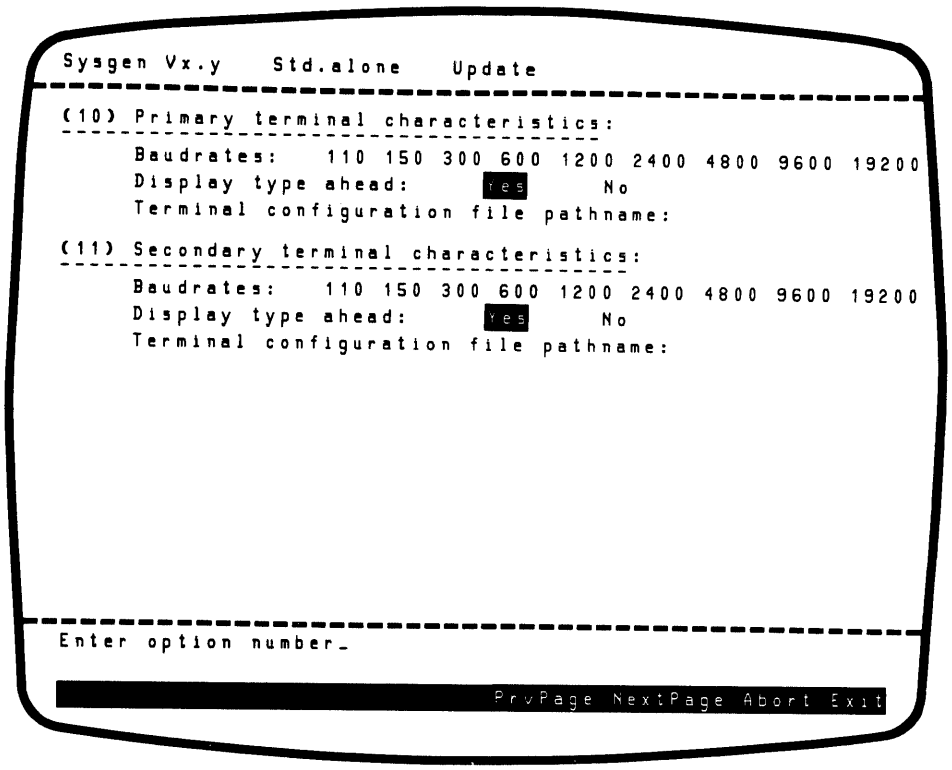
Priority. When both regions are active, the priority determines which region takes precedence when contending for the CPU. When priority is selected SYSGEN displays:

```
Select desired option

Unequal Same  Import Batch  Abort Exit Return
```

When unequal is selected (press U) the 1st region is always higher priority. Unequal priority is most useful when running in the single-user mode, with non-interactive processes running in the background. Selecting Same (enter S) gives equal priority for both regions. This may be most useful in multi-user or toggle modes where both regions are running interactive processes. With Import priority (enter I), the process not in the Import mode has priority. This is useful in all modes of operation (single-user, multi-user and toggle) for providing the best interactive program response whenever a region is running a remote job. The Batch option provides highest priority to the region executing interactively, lowest priority to the region running a Batch or Submit file. If both regions interactive, then the priority is the same. When return <cr> is pressed SYSGEN returns to region selection. Press return again to exit option 9.

SERIES IV OPTION SCREEN—PAGE 3



OPTION 10 This option allows the user to specify the characteristics of the primary Series IV console. This option specifies the default conditions present at initialization and can be over-ridden with the STTY command. When option 10 is selected the prompt follows:

```
Select desired option

Baudrate Typeahead Config  Abort Exit Return
```

At the present time, the primary console baudrate is set and cannot be altered. To select the type ahead feature enter T and SYSGEN will display:

```
Select desired option

Yes    No          Abort  Exit  Return
```

Enter Y to display type ahead or N for no display.

When the "Select desired option" prompt appears, the configuration file pathname may entered or altered by entering C. The following prompt will be displayed:

```
Enter new configuration file pathname_
```

The configuration file is used to set terminal display and keyboard parameters. Refer to Appendix D for details on the configuration file.

OPTION 11 This option allows the user to specify the characteristics of the secondary terminal on a Series IV. The displays, prompts and selection is identical to Option 10, with the exception that the secondary terminal baudrate can be set. Press <cr> to return to the option selection screen.

When the new SYSGEN information has been entered, type EXIT to create the new configuration. After the iNDX prompt (>) appears you must logoff (type LOGOFF<cr>), then reboot the system from the system device SYSGEN'ed in order for the new configuration information to take effect.

TIME

Syntax

```
TIME
```

Description

The program invoked by the TIME command will sign on as follows:

```
system id TIME (Vx.y)
DATE: mm/dd/yy
TIME: hh:mm:ss
```

where:

system id	is the operating system's identification.
mm/dd/yy	is the current date.
hh:mm:ss	is the current time.

If the user is logged on as SUPERUSER or working at a standalone Series IV, the system will prompt:

```
ENTER COMMAND (SET/EXIT):
```

A command may now be entered in line-edit mode. All keyboard entry to the TIME command is in full line-edit mode. TIME recognizes abbreviations of the commands. If the command entered is not valid, TIME will display the following message:

```
INVALID COMMAND
```

and the ENTER COMMAND prompt will be redisplayed.

If the command EXIT <cr> is entered, the program terminates and control returns to the Command Line Interpreter.

If the command entered is SET <cr> or any of its abbreviations, the following prompt will be issued:

```
ENTER DATE (mm/dd/yy <cr>):
```

The user enters the current date in *mm/dd/yy* format. If yy is less than 78, the date is assumed to be in the 21st century. If any of the characters in *mm,dd,yy* are not numeric, the following message will be displayed and control will be returned to the ENTER COMMAND prompt:

```
INVALID ENTRY
```

If the entry is valid, TIME will prompt for the current time:

```
ENTER TIME (hh:mm:ss<cr>):
```

Enter the current time in military format (i.e., 1 PM = 13:00:00). If any of the characters in the time sequence are not numeric, the following message will be displayed and control will be returned to the ENTER COMMAND prompt:

```
INVALID ENTRY
```

If the entry is valid, the system clock will be set to the requested date and time. If the date and time entered do not represent a valid combination, one of the messages below will be displayed and followed by the ENTER COMMAND (SET/EXIT) prompt:

```
EXCEPTION 201AH: INVALID TIME
EXCEPTION 201BH: INVALID DATE
```

After the SET command is executed, the current setting of the system clock is displayed and control returns to the CLI.

Additional Notes

1. When invoked within SUBMIT file processing, TIME will display the current system Date/Time and exit, returning control to the Command Line Interpreter.
2. In the network mode, only the Superuser may set the system date/time. If the date/time is set at a workstation, it will also be set at the NRM and all other workstations to ensure a synchronized network.

   If the TIME cusp is invoked by a non-superuser in the network, the current system Date/Time will be displayed and the TIME cusp will return control to the CLI.

USERDEF

Syntax

```
USERDEF {DEFINE username {ID userID | DIR filename}}
        {REMOVE username                           }
```

where:

username	is the name by which the system identifies the user.
userID	is the ID-number by which the system identifies the user. (3–15 or 1024–32767)
DIR	is used to define the user's home directory.
filename	is the pathname of the user's home directory.

Description

This function allows the superuser to add or delete a user who has access to files on mass storage devices. An attempt to create a new user will fail if the specified user name or user ID is already used. When creating a new user, the superuser has the option to simultaneously create or assign a directory for that user.

The superuser has a predefined name within the system-SUPERUSER. This name is used by the primary superuser and cannot be deleted. The primary superuser may create secondary superusers with the USERDEF function by specifying a user ID in the allowable range (3–15). Other valid user IDs are in the range 1024–32767. Secondary superusers have full superuser capabilities, except they may not execute USERDEF and they may be deleted from the system by the primary superuser.

The user's home directory is assigned the null logical name (" ") whenever the user logs on. If the directory does not exist when specified in USERDEF, a new directory is created. The owner of the new directory is the user defined in the USERDEF command. If the directory specified in the USERDEF command already exists, it will be user's home directory, but the directory ownership will not change. Whenever an existing directory is used as a home directory, it should have access to all world rights.

When a user is deleted from the system, only the user name is deleted. If that user is logged on the system, that session continues. All files that belong to the deleted user remain in the system. An expanded directory listing returns a user name of NOT FOUND for those files. The CHOWNER command must be used to transfer ownership of these files.

A list of all users and user IDs assigned via the USERDEF command can be obtained through the USERS command. For each user created, a home directory should be created or assigned on an available volume. The user should be told the fully-qualified pathname for that directory. The user should place all of his subsequent directories and files on his assigned home directory.

Additional Notes

1. The initial password for each user is " " (null) and is entered as a <cr>. The superuser should immediately use CHPASS to assign a password after assigning a user name.

2. The USERDEF command manipulates three system files: UDF, HOME, and PUBLIC.UDF. These files are located in the volume root directory of the system volume. Do not change them.

NOTE

Backup copies of the UDF and HOME files should always be made simultaneously to ensure system consistency.

3. USERDEF can be used to change the home directory of an existing user by entering the following command:

```
USERDEF DEFINE existing username DIR new filename
```

Examples

1. ```
   USERDEF DEFINE PAUL ID 1025
   ```

   This example adds user name PAUL with ID 1025.

2. ```
   USERDEF DEFINE SHEILA ID 3273 DIR /A/B
   ```

   This example adds user name SHEILA with ID 3273 and a home directory named /A/B.

3. ```
   USERDEF REMOVE BARRY
   ```

   This example removes user name BARRY.

4. ```
   USERDEF DEFINE LINDA DIR /FINANCE/ACCT
   ```

   This example gives the existing user LINDA a new home directory named/ FINANCE/ACCT.

USERS

Syntax

```
USERS /volume-name
```

where:

volume-name is the volume root directory name of the system device.

Description

This function lists in tabular form the user names, associated user IDs, and home directories for all users who have access rights to the system device. The volume-name specified should be the volume root directory name of the system volume.

NOTE

This command can only be used by the superuser.

Example

```
USERS /ALPHA
```

This example requests a list of the users on volume ALPHA. The following list is displayed:

```
USERNAME      USERID      HOME DIRECTORY

PAUL          1025        /WINI0/PAUL.DIR
SHEILA        3273        /WINI0/SHEILA.DIR
```

VERIFY

Syntax

VERIFY *physical device* [FIX / FAST] [T0] [T1] [T2] [T3] [T4] [OVERRIDE]

where:

physical device	is one of the following: FL0 = flexible disk WD0...3 = 35 megabyte Winchester disks WF0...3 = 84 megabyte Winchester disks HD2,HD0 = fixed platter hard disk HD3,HD1 = removable platter hard disk
FIX	is an option to mark bad blocks and clear up affected files as instructed.
FAST	is an option allowing a limited verification in a shorter time period and should not be run with FIX.
T0–T4	allows a specific verification test to be run. See the description below on details on the individual tests.
OVERRIDE	allows the user to verify the disk while both regions are active.

The VERIFY command dismounts the disk before executing. This makes the disk inaccessible to users during the verification. The VERIFY command cannot reside on the disk to be verified. VERIFY OVERRIDE may disrupt a second user or background process. If a file on the target drive is open when the command is entered, subsequent operations on the file will return an error indicating the disk has been dismounted.

Description

The VERIFY command allows you to check the file integrity of a specified disk. Also, VERIFY is able to correct some disk errors if the FIX option is enabled. The VERIFY command uses the fnode file data and allocation information at each file to perform its verifications. Each fnode in the fnode file contains a data structure describing the location, allocation, file type and other information about each file. VERIFY runs five tests to determine disk integrity. The tests are defined in the following table.

Table 4-2. Disk Integrating Tests

Test Number	Description
T0 with [FIX]	Verifies that reserved filenames are consistent with the FORMAT command. If reserved filenames are in error, they are changed to be consistent with the FORMAT command.

Table 4-2. Disk Integrating Tests (Cont'd)

Test Number	Description
T1 with [FIX]	Verifies that specific fields within the system files are compared to expected values. If defective fields within system files are found, they are corrected.
T2 with [FIX] without [FAST] without [FAST] with [FIX] with [FIX]	Verifies the hierarchical file structure: 1. Reads all allocated files. 2. If file is a directory, reads and marks all files within the directory. 3. Checks to see if all files were marked. 4. Reports unmarked files. 5. Attaches lost files to reserved directory /VERIFYFIX. 6. Verifies all entries in the directory file by tracing from parent fnode to child fnodes and back again. 7. Displays any invalid entries. 8. Deletes any invalid entries. 9. Asks if you want to assign a pathname to any files with fnodes assigned to more than one directory. If not, file will automatically be assigned to /VERIFYFIX directory.
T3 with [FIX] with [FIX]	Verify File System Bitmaps. This is a five part test that verifies file system bitmaps by: 1. Comparing the file allocation bitmap to the file allocation field in each file fnode. 2. Checks for valid file type for each file. Files must either system, or directory types. 3. Verifies that deleted files were completely removed. Removes any files incompletely deleted. 4. Checks for bad blocks within files. 5. Verifies that all bad blocks marked in the bad block bitmap are marked in the free space bitmap. Corrects bitmap errors.
T4 with [FAST] with [FIX]	Verify Used Disk Block Allocation. Verifies that the used disk blocks are allocated correctly by performing the following: 1. All allocated files are read and disk blocks marked. If any file is marked more than once, an error will be displayed. 2. Only the filenames and last modification date/time are displayed. 3. Marks allocated bad blocks (removing them from the file system). 4. You will be asked if you want to delete any files with incorrect block allocation.

At the end of each test, a message will appear indicating pass or failure. If a test fails, refer to the error messages in Chapter 5. Any combination of these tests may be run using the Tx option in the VERIFY command line. If no Tx option is specified, then all tests are run.

When the VERIFY FAST option is used, the command does not check the validity of directory entries, does not trace or display files with fnodes assigned to more than one file, nor does the command trace or display blocks allocated to more than one file. It is recommended not to use the FAST option when FIXes are desired.

When the VERIFY FIX option is chosen, and file errors occur, in many cases you will be queried as to whether you want to delete or reassign a file. In these cases, no access rights are observed.

The VERIFY program disregards any access rights that a file may have. Make sure you want to delete the file queried before responding with a yes.

Within the VERIFY FIX routine any duplicate files will automatically be recovered if possible. Any fnode or bitmap file data can be re-created if the primary or duplicate files were not destroyed in the same location.

The system disk cannot be verified while mounted. To verify the system device, you must boot the system from the iNDX.S31 or iNDX.S41 diskette (the Winchester then is no longer the system device).

The OVERRIDE option allows the Verification of devices even though both regions are active (Multi-user, Toggle, or Single-user using the Background). is option also allows the user to run VERIFY from the second region. ithout the OVERRIDE option an error message will be generated if both regions are active.

Examples

1. ›VERIFY FL0 ‹cr›

```
TEST #0 :
Verifying Formatted File Names
VERIFICATION ''PASSED''

TEST #1 :
Verifying Predefined Fields in System Files
VERIFICATION ''PASSED''

TEST #2 :
Verifying Hierarchical File Structure
DOING FILE XXXXH to XXXXH
VERIFICATION ''PASSED''

TEST #3 :
Verifying File System Bitmaps
DOING FILE XXXXH to XXXXH
VERIFICATION ''PASSED''

TEST #4 :
Verifying Used Disk Block Allocation
DOING FILE XXXXH to XXXXH
VERIFICATION ''PASSED''

DISK VERIFIES
```

This example runs VERIFY on the flexible disk. A specific test was not specified, so all tests were executed. All tests passed in this case.

2. ›VERIFY HD0 FIX T3‹cr›

```
TEST #3 :
Verifying File Sytem Bitmaps
DOING FILE XXXXH to XXXXH
VERIFICATION ''FAILED/FIXED''

fix: bad file bitmap corrected

DISK VERIFIES
```

Verify the file system bitmaps on drive 0 and fix any detected errors. This message indicates that the VERIFY command found the file allocation bitmap to be incorrect. VERIFY then changes the bitmap to the correct values.

3. ›VERIFY WD0 FAST T4‹cr›

```
TEST #4 :
Verifying Used Disk Block Allocation
DOING FILE XXXXH to XXXXH
```

```
ERR: DISK BLOCK(S) ALLOCATED TO MULTIPLE FILES
The following files contain multiply allocated block(s)
DATA FILE -- /SYSDIR/MYDIR/DATA.CAN
             Last modification: 03/15/83   09:31:23

DATA FILE __ /SYSDIR/DIR2/FILE.2
             Last modification: 03/15/83

VERIFICATION ''FAILED''
DISK DID NOT VERIFY
```

In this example, a 35-MB Winchester (Periheral Chassis) disk drive is verified with the FAST option which allows the display the pathname, file type, and last modification of the files found with files containing blocks allocated to more than one file.

4. `>VERIFY WF1 FIX T4<cr>`

```
TEST #4 :
Verifying Used Disk Block Allocation
DOING FILE XXXXH to XXXXH

ERR: DISK BLOCK(S) ALLOCATED TO MULTIPLE FILES
The following files contain multiply allocated block(s)
DATA FILE -- /SYSDIR/MYDIR/DATA.CAN
             Last modification: 03/15/83   09:31:23
     Multiply allocated blocks are :
          XXXXH BLOCKS AT BLOCK # XXXXXXXH
          XXXXH BLOCKS AT BLOCK # XXXXXXXH

DELETE DATA FILE?Y<cr>

DATA FILE -- /SYSDIR/DIR2/FILE.2
             Last modification: 04/12/83   14:07:04
     Multiply allocated blocks are :
          XXXXH BLOCKS AT BLOCK # XXXXXXXH
          XXXXH BLOCKS AT BLOCK # XXXXXXXH

DELETE DATA FILE?Y<cr>
ALL UNWANTED FILES DELETED?Y<cr>
VERIFICATION ''FAILED/FIXED''
BAD DISK FIXED
```

In this case, the same problem as in example three was corrected with the FIX option, which also displayed the multipli-allocated blocks.

VIEW

Syntax

VIEW *pathname*

where:

pathname is a valid pathfname.

Description

VIEW is an interactive command that allows the user to examine the contents of the specified file. The VIEW command must be entered frosm the keyboard and connot be fplaced in a command file. Whenaa the command is entered frosmt he keyboard,a the specified file is attached and itsl first page is displayed in AEDIT full screen format. The text of the page appears on Lines 1–23 of the display. Line 24 (the AEDIT message line) displays:

----- VIEW *pathname*

where *pathname* is the name of the file being read.

Line 25 (the AEDIT prompt line) displays the subset of AEDIT commands available for use with the AGAIN, FIND, -FIND, JUMP, QUIT, ROLL, SET, VIEW.

WAIT

Syntax

WAIT [*wait message*]

where:

wait message is an optional message that will be displayed when WAIT is invoked.

Description

The WAIT command is used to delay the execution of a submit file until a carriage return is entered from the keyboard. When the WAIT command is executed within a submit file, any characters in the type ahead buffer will be deleted. Next, the message:

```
----- Waiting for <cr> ...
```

will be displayed in the message line. If a wait message was entered in the WAIT command, then it will be displayed flashing in reverse video in the prompt line. When a carriage return is entered from the keyboard, a message will momentarily appear:

```
Proceeding ...
```

The submit file will then continue executing.

Examples

1.
```
>SUBMIT EXAMPLE.CSD<cr>
>WAIT 'Insert NEW.src mini floppy in f10 to be formatted'
----- Waiting for <cr> ...
Insert NEW.src mini floppy in f10 to be formatted
```

   The example submit file is waiting for a carriage return to continue, while displaying the wait message as typed.

2.
```
>SUBMIT EXAMPLE.CSD<cr>
>WAIT Insert NEW.src mini floppy in f10 to be formatted
----- Waiting for <cr> ...
INSERT NEW.SRC MINI FLOPPY IN F10 TO BE FORMATTED
```

   In this example, the submit file is delayed and the wait message is in capital letters because single quotes (') were not placed around the message in the command.

CHAPTER 5
NETWORK OPERATION

5.1 Introduction

The Series IV development system may be used as a workstation in the NDS-II network. As a workstation, the facilities of the network are available to you, the Series IV user. Within this chapter, the following information is provided:

- A guide to the resources available at the Series IV through the NDS-II.
- Procedures for accessing and using the NDS-II resources.
- Detailed descriptions of iNDX commands invoked at the Series IV that utilize the NDS-II.

The Series IV, when installed on the NDS-II network, uses a Network Resource Manager (NRM) device as the system device (i.e., the device that has the operating system resident). For the Series IV to operate, you must ensure that your Series IV operating system (either iNDX.W31 or iNDX.W41) is installed at the NRM. Refer to the *NDS-II Network Resource Manager User's Guide*, 134300, Chapter 2, for the installation procedures. Additionally, the SYSGEN command must be run during installation time at the NRM, to configure the Series IV into the network operating system at the NRM.

After installation is complete, when the Series IV is initialized, the portion of the operating system that must be resident in Series IV memory is downloaded from the NRM to the Series IV. Each time a command is invoked from the Series IV, it accesses the system device at the NRM to load that command (unless the directory priorities have been changed with the SEARCH command—refer to section 5.6).

5.2 Network Resources

NDS-II offers distributed processing with local workstation resources and remote network resources.

The network also acts as a host for software tools (ASM, Fortran, Pascal, C-86 and PL/M), instrumentation tools (ICE In-Circuit Emulators and iLTA), program management tools (MAKE and iSVCS), and electronic mail features for local area communication.

Each workstation in the Network shares concurrent access to the shared file system, the remote execution job queues, and the printer queue.

The distributed network file system offers several advantages: large mass storage capability, peripheral device sharing enabling higher performance at a lower cost per workstation, file sharing among project members and with other projects, distributed hierarchical file system, and archival facility for files stored on the shared disks.

5.3 Remote Jobs

Remote job execution allows a workstation user to execute a batch job on another NDS-II workstation. A remote job is executed by exporting it to a job queue. The NRM job controller then sends the job to an importing workstation.

There are three types of stations: the NRM, private workstations, and import workstations. The NRM Distributed Job Controller maintains state information of remote job queues and the status of the workstations.

5.3.1 Private Workstations

A private workstation can send jobs to a queue located at the NRM. The NRM then sends the jobs from the queue to have them executed by other public (import) workstations in the network. However, a private workstation does not accept jobs from the NRM. When you power up a Series IV workstation, you have the options of LOGON or IMPORT. If you enter LOGON, then it is a private workstation.

5.3.2 Public Workstations

If you enter IMPORT, the Series IV becomes a public workstation and can accept jobs from the NRM. IMPORT can also be invoked from within the operating system, where you may use the background to import jobs. A station remains an import station until the either the BREAK key is pressed from the workstation keyboard or the station is importing from a single job queue and it is deleted at the NRM. When the Break key is pressed, the import station continues executing until the current job is finished.

As an example, to perform an export from a Series IV workstation to another workstation, enter the following commands:

1. To list any existing queues, type:

```
QUEUE List<cr>
NAME OF QUEUE        # OF SERVERS    # OF WAITING JOBS
EXAMPLEQ1                  1                1
EXAMPLEQ2                  1                2
```

2. If an appropriate job queue doesn't already exist, you may create one (in this example MYQUE) by entering:

```
QUEUE Add (MYQUE)<cr>
```

3. Then to send a job to the queue for remote execution, type the following:

```
EXPORT /DIR/MYJOB TO MYQUE<cr>
```

If there are no workstations accepting jobs from the queue you exported too, you will see the message:

```
WARNING: NO SERVER EXISTS FOR QUEUE MYQUE
```

To import a job from the NRM's queue, perform the following steps:

1. To enter the import mode, enter:

```
>IMPORT FROM EXAMPLEQ1<cr>
```

or for the background mode:

```
>IMPORT FROM EXAMPEQ1 TO BACKGROUND<cr>
```

2. To exit from the import mode, press the BREAK key, then either enter F for foreground or B for background.

When the job is finished, the workstation becomes private again. A Series IV can import into either foreground, background, or both; thus, a physical station may appear as two stations to the NRM distributed job controller.

A network may consist of several different types of workstations, which means that queues must be accessible to only compatible workstations. To prevent any problems between incompatible workstations, only export to and import from known Series IV queues while working at a Series IV workstation.

NOTE

The distributed job controller at the NRM does not verify compatibility between workstations and queues. For proper remote job processing, you must ensure that the workstation exporting to the queue and the workstation importing from the queue are compatible.

A remote job sent to a queue may or may not be executed immediately, depending upon the availability of a server station.

Any external resources or files required by an EXPORT command must be made available at the importing workstation that will execute the job; local resources and files are not transported to the importing station as part of the EXPORT command.

5.4 I/O Differences

You can print from the network printer, allowing you to send your output to the spool printer and to use your workstation to initiate another job. Note, however, that the device name for the printer in this case is :SP:, not :LP:, as it is in the standalone mode. :LP: designates the local line printer, not the network spool printer. Any output from workstations in the network mode should be sent to the spool printer (:SP:).

5.5 Initiating Workstation Operation

The following procedure assumes the System Generation procedure has been implemented and the NDS-II network configuration has been verified. (See the *NDS-II Network Resource Manager User's Guide,* 134300.)

To initiate workstation operation:

1. Check that the Network Resource Manager is operative. See the *NDS-II Network Resource Manager User's Guide.*
2. Verify that the Series IV Configuration switches are set as illustrated below.

   Switch — 1 2 3 4 5 6 7 8
   Setting — ON - # # # # # ON

   Switch 1 when up selects 60 Hz. Switch 2 is reserved. Switches 3 and 4 select boot device unit address. Switches 5, 6 and 7 select boot device.

NOTE

If network communications are lost, the system will boot from the address selected by switches 3 through 7.

3. Power up the Series IV by turning on the circuit breaker switch located at the left rear of the terminal chassis.
4. Turn on any peripherals such as a Winchester disk drive or a printer. When using 740 Hard Disk Drives, press the STOP/START button and wait for the READY light before proceeding.
5. Verify the Power-up Diagnostics test passed. If it did not, refer to the *Intellec Series IV Installation and Checkout Manual,* 121757.
6. Verify the sign-on message is displayed on the Series IV. If it is not, press the RESET switch (see Figure 1-5) to boot the operating system.
7. Enter LOGON plus your assigned user name. For example, with a username of FRED, enter:

```
>LOGON FRED<cr>
```

8. Enter your password.

You are now in the operating system, and any of the iNDX operating system commands may be entered.

5.6 Using SEARCH on the Network

The SEARCH command is particularly useful when operating on a network. The SEARCH command specifies the directory that the command line interpreter will search for a command when invoked at the Series IV. To illustrate, the usefulness of SEARCH assumes that the root volume directory at the NRM is named /ROOTVOL, and the Series IV directory that contains the system files when they are downloaded from the NRM is called /SIVDIR. First, copy commonly used system commands to the Series IV workstation:

```
>COPY /ROOTVOL/DIR.86 TO /SIVDIR<cr>
```

Next specify the directory to be searched when a command is invoked:

```
>SEARCH /SIVDIR<cr>
```

Now when a command is invoked, SIVDIR is searched first. If the command is not found in SIVDIR, then the root volume directory /ROOTVOL at the NRM is searched. To display the actual SEARCH path used, enter:

```
>SEARCH<cr>
```

And the display will follow:

```
SEARCH PATH IS: /SIVDIR
                /ROOTVOL
```

Greater performance can be realized when the most commonly used commands are maintained at the Series IV, rather than at the NRM. This is because with each command that is performed at the Series IV, only a local disk operation occurs, rather than a network operation.

5.7 Terminating Operation

5.7.1 Terminating User Session

To terminate the user session and to allow another user to start a session, perform the following:

1. Log off by entering the command LOGOFF or by pressing the appropriate soft key.
2. The new user may now log on by entering his username and password. The system is now ready to accept commands from the new user.

5.7.2 Powering Down Development System

To power down your workstation, perform the following:

1. Log off by entering the command LOGOFF or by pressing the appropriate soft key.
2. Turn off any peripheral devices.
3. If you are using a flexible disk(s), wait for the drive indicator light to go off (see Figure 1-4). Release the drive door latch(es) and remove the flexible disk(s).
4. Set the circuit breaker switch located at the left rear of Series IV to the off position.

5.8 Remote Job Commands

The remote job commands are:

CANCEL QUEUE EXPORT SYSTAT IMPORT NETMSG

They are described in the following pages.

CANCEL

Syntax

```
CANCEL | BACKGROUND                                                    | [,...]
       | [REMOTE] queue {(jobname) | (jobnumber)}                    |
```

where:

queue	is the queue where the remote job is queued for execution.
jobname	is the final component name of the remote job to be cancelled.
jobnumber	is the assigned value of the remote job (which can be displayed via the SYSTAT command).
BACKGROUND	specifies that the currently executing background job is to be aborted.
REMOTE	specifies that a remote job is to be aborted.

Description

The CANCEL command is used to cancel a background job (see the CANCEL command in Chapter 4) or a remote job. If you want to abort a remote job, you must enter the job name or job number that will be aborted. *Jobname* is the final component name used in the EXPORT command that placed the job on the queue. You can only cancel jobs you have sent to the queue.

If the job name is selected and multiple instances of *jobname* are in the queue, the first job name encountered is deleted (this may not be the first one queued). To avoid this confusion, use the unique job number, assigned at the time the job was exported.

EXPORT

Syntax

EXPORT *pathname* [*parameters*] TO *queue* [[LOG [(*pathname*) [APPEND]] | NOLOG]]

where:

pathname	is a valid pathname.
parameters	is a list of up to ten parameters.
queue	is the queue to which the job is to be sent.
LOG, NOLOG	specifies whether a log is to be kept on a mass storage device of any console activity.
pathname	specifies an alternate pathname of the LOG file.
APPEND	specifies that the current console log be added to an existing log file, rather than writing over it.

Description

The EXPORT command allows a command file composed at one workstation to be executed on a remote workstation. The command file specified must be on a public volume so that a public workstation can have access to it. The Network Resource Manager (NRM) maintains a queue of jobs to be executed at remote public workstations. The desired queue is specified in the command line. If the queue does not exist, an exception message is displayed and the job is not queued.

LOG and NOLOG determine whether a log of console activity is to be maintained on a mass storage device. LOG is the default condition.

The BACKGROUND, REMOTE, and SUBMIT commands all have similar structures. These similar structures permit you to execute any given command file with any of these commands.

If your command file name has no extension, the system appends the extension .CSD to the command file. If your filename ends with a period, the system will use the filename you have given (with the period truncated). If your filename has an extension, the system uses that filename without modifying it.

The optional parameters specified in the command line are the actual parameters to be substituted for the formal parameters embedded within the command file. A maximum of ten actual parameters can be specified in the command line. Placing formal parameters in the command file allows you to call the same command file for varying sets of actual parameters.

The actual parameters in the second file are sequentially substituted for the formal parameters. Thus, (reading from left-to-right) if you enter the actual parameters ASM, PLM, and BLT, ASM replaces the formal parameter %0, PLM replaces %1, and BLT replaces %2. The command file resulting from the parameter substitution is placed in the same directory as the generating command file. The final path component is derived from the generating file by:

- Truncating the final component of the generating file up to the third character or before the first period (whichever comes first).
- Appending a unique system specified character string plus the extension .CS are appended.

When the LOG option is specified, a log of console output is routed to a mass storage device. When LOG is specified, the pathname of the LOG file will be obtained either from the command line when specified by you, or by changing the command file name extension from .CSD to .LOG. However, if parameters are specified in the command file, and you have not specified the log filename, the LOG filename will be the .CS filename (the temporary file created for run-time) with the extension changed to .LOG.

NOLOG specifies that no such log is to be kept. If neither option is explicitly entered, LOG is the default condition.

The APPEND option causes the log file that is created in the current EXPORT operation to be added to the end of the existing specified log file (usually a filename is specified).

Examples

See the SUBMIT command (Chapter 4).

IMPORT

Syntax

```
IMPORT FROM queue [,queue] ... [ TO BACKGROUND ]
                               [ TO FOREGROUND ]
```

where:

queue	is a character string up to 14 characters long which names the queue at the point where the job awaits execution. Up to five queues may be specified in one command.
TO BACKGROUND	is an option that will execute the imported job in background mode.
TO FOREGROUND	is an option that will execute the imported job in the foreground mode.

Description

The Import command must be invoked interactively from the foreground of the given workstation. You can invoke the IMPORT command while logging on or any time after logging on. The IMPORT command declares the given workstation to be a public workstation. A public workstation is one that can receive jobs maintained by the Network Resource Manager (NRM). If you enter the name of a queue that does not exist at the NRM, an exception message is displayed. The queues are searched for jobs according to the order in which the queues were listed in the command line (left-to-right). If jobs are available in the queues, the importing station will begin processing them. The importing station starts by performing an implicit logon for the owner of the job first taken off the Queue and then processes commands within the command file. At the end of the command file, the importing station logs you off and looks for another job from the queue to process.

All output messages from the remote job are displayed on the screen of the importing station and may be put on a log file of a mass storage device if the LOG option is specified in the EXPORT command.

NOTE

IMPORTED jobs are implicitly logged on. Initialization options are identical to those specified in LOGON.

When a station becomes a public workstation, it becomes unavailable for any local processing. To return the station to private operation and thus stop it from accepting jobs from the queue, hold down the CONTROL key and press the C key (CONTROL-C).

Each public workstation can service up to five queues. Each network can contain up to sixteen queues.

NETMSG

Syntax

```
NETMSG
```

Description

The NETMSG command enables the superuser at the Series IV to send messages to all Series IV users on the network. After the NETMSG command has been entered, the prompt will appear:

```
Type in message (to terminate type "." on a line)
```

Enter the desired message, then either type "." or type a control Z to terminate the message. The entered message will then be sent to all logged on workstations. No matter what process is occurring at the workstation, the message will be displayed.

Example

1. ›NETMSG‹cr›

```
Type in message (to terminate type "." on a line)
THIS IS THE MESSAGE.
.
```

This example will send the message "THIS IS THE MESSAGE" to all logged on Series IV workstations.

QUEUE

Syntax

```
QUEUE [ ADD    (queuename) , ... ]
      [ DELETE                   ]
      [ LIST                     ]
```

where:

ADD	adds a queue to the queue list.
DELETE	removes a queue from the queue list.
LIST	provides a list of existing queues.
queuename	designates the name(s) of the queue(s) to be added or removed from the queue list.

Description

The QUEUE command is used for management of the queues present at the NRM. This command may be entered in two modes: interactive or command line. In the interactive mode, enter the singular command:

```
QUEUE<cr>
```

The command then displays the name, number of jobs outstanding, and the number of servers. After this information is displayed, you are prompted:

ADD DELETE LIST EXIT

The ADD option creates new queues. Sixteen queues can exist at one time. The DELETE option lets you delete a queue. The LIST option re-displays previously listed information. The EXIT option terminates use of the QUEUE command.

In the command line mode, enter the command and the chosen option in the command line, for example:

```
QUEUE ADD (queuename)
```

Example

```
Queue List<cr>
```
(In this example, existing queues are listed.)

```
NAME OF QUEUE      # OF SERVERS      # OF WAITING JOBS
BIGQ                      1                   3
LITTLEQ
REDQ                      2
BLUEQ                     1
```

SYSTAT

Syntax

SYSTAT [{QUEUE / MYJOB} (*queuename* [, ...])] [TO *pathname*] [EXPAND] [ALL]

where:

queuename	designates the name(s) of the queue(s) for which jobs are to be listed.
pathname	designates the file where the information will be listed.
QUEUE	displays information for all queues or for only those queues explicitly listed after the QUEUE specifier. If this option is specified, the queue names must be separated by commas.
MYJOB	parallels the QUEUE option but lists information about jobs belonging only to you.
EXPAND	specifies that complete information is displayed for each job. If EXPAND is not specified, condensed information will be displayed. Complete and condensed information are demonstrated below in examples 4 and 2, respectively.
ALL	displays appropriate information for all jobs (cancelled, executed, aborted, executing and waiting) in the specified queue(s). If ALL is not specified, information is displayed only for jobs that are waiting or executing.

Description

The SYSTAT command displays the status of the Distributed Job Control (DJC) queues and the jobs within the queues. If SYSTAT is entered with no specifiers, a list of currently defined DJC queues is displayed.

The QUEUE option specifies that jobs of all users from the specified queue list be displayed. The MYJOB option specifies that only your jobs are displayed. If no queue list is specified, the requested information is displayed for all queues. If you specify more than one queue, you must separate the queue names by commas.

If you use the specifier ALL, information is provided for all jobs in the selected queues regardless of their status. If you do not specify ALL, information is provided only for those jobs that are executing or waiting. The display of job information from each queue may be halted by entering a Control-C.

Examples

1. SYSTAT

```
SYSTAT cusp version Vx.y
QUEUE      # OF JOBS      # OF IMPORT
NAME       WAITING        STATIONS
QUEUE1         2              0
QUEUE2         1              1
```

This command will list all currently defined job queues.

2. `SYSTAT QUEUE`

```
Systat Queue version Vx.y
JOB STATUS FOR: QUEUE2
  JOB NAME    JOB #    OWNER    DATE       TIME       STATUS
  ABAK        1003     A        11/11/83   11:27:30   WAITING
  ABAK        1001     A        11/12/83   12:21:00   WAITING
JOB STATUS FOR: QUEUE3
  JOB NAME    JOB #    OWNER    DATE       TIME       STATUS
  ABAK0       2001     A        11/21/83   9:35:01    WAITING
```

This command will list condensed status information on all jobs currently executing or waiting in all queues.

3. `SYSTAT QUEUE ALL`

```
SYSTAT cusp version vx.y
JOB STATUS FOR: QUEUE2
  JOB NAME    JOB #    OWNER    DATE       TIME       STATUS
  ABAK        1003     A        11/11/83   11:27:30   WAITING
  ABAK        1002     A        11/11/83   11:21:00   CANCELED
  ABAK        1001     A        11/11/83   11:16:00   WAITING
  ABAK        1000     A        11/11/83   11:10:00   CANCELED
JOB STATUS FOR: QUEUE3
  JOB NAME    JOB #    OWNER    DATE       TIME       STATUS
  ABAK        2001     A        11/21/83   9:35:01    WAITING
  ABAK        2000     A        11/11/83   9:10:00    CANCELED
```

This command will list condensed status information on all jobs in all queues.

4. `SYSTAT QUEUE EXPAND`

```
SYSTAT cusp version Vx.y
JOB NAME: ABAK                              STATUS:   WAITING
JOB NUMBER:1003      QUEUE: QUEUE2          OWNER:    A
DATE: 11/11/11       TIME REC'D:            11:27:49
LOG FILE: ABAK123.LOG
JOB NAME: ABAK                              STATUS:   WAITING
JOB NUMBER: 1001     QUEUE: QUEUE2          OWNER:    A
DATE: 11/11/11       TIME REC'D:            11:16:40
LOG FILE: ABAK124.LOG
JOB NAME: ABAK                              STATUS:   WAITING
JOB NUMBER: 2001     QUEUE: QUEUE           OWNER:    A
DATE: 11/11/11       TIME REC'D:            12:03:46
LOG FILE: ABAK132.LOG
```

This command will list complete status information on all jobs currently executing or waiting in all queues.

5. `SYSTAT MYJOB`

```
SYSTAT cusp version Vx.y
JOB STATUS FOR: QUEUE2
JOB NAME    JOB #    OWNER    DATE       TIME       STATUS
ABAK        1003     A        11/11/11   11:27:49   WAITING
ABAK        1001     A        11/11/11   11:16:40   WAITING
JOB STATUS FOR: QUEUE3
JOB NAME    JOB #    OWNER    DATE       TIME       STATUS
ABAK        2001     A        11/11/11   12:03:26   WAITING
```

This command will list condensed status information about all jobs currently executing or waiting that belong to the user.

6. `SYSTAT QUEUE (QUEUE2)`

```
SYSTAT cusp version Vx.y
JOB STATUS FOR: QUEUE2
JOB NAME    JOB #    OWNER    DATE        TIME        STATUS
ABAK        1003     A        11/11/11    11:27:49    WAITING
ABAK        1001     A        11/11/11    11:16:40    WAITING
```

This command will list condensed status information about all jobs in QUEUE2 that are currently executing or waiting.

CHAPTER 6
PROGRAMMING INTRODUCTION

6.1 Introduction

An operating system is a group of programs that provide the functional (as opposed to physical) environment in whichram. By managing overlays and error conditions, you can extend the range of functions (and recoveries) available to your programs. Service routines designed to handle exceptions allow early detection and handling of unwanted conditions arising during execution.

6.2 Functions of the iNDX and ISIS-IV Operating Systems

In addition to device management and overlay capability, the iNDX and ISIS-IV operating systems also have command scanning and dynamic file or device manipulation (i.e., under program control during execution).

Command scanning allows your program to pick up options specified on the input line that invokes program execution, or to treat specially formatted files as if they were input from the console.

Dynamic file control during program execution allows you to maintain a list of twelve files or devices that are used by your program (e.g., written or read); but, you can only use six at any one time. The ability to immediately access twice as many files and devices can make input/output operations more efficient.

The iNDX operating system also provides memory management and an interactive symbolic debugging aid.

Memory management during program execution allows you to allocate memory for specific processes as they arise and to free those blocks when they are no longer needed.

The debugging tool is called DEBUG-88. Its interactive language is similar to that of Intel's ICE-86 or ICE-88 emulators. Using DEBUG-86, you can insert breakpoints into your program, execute until some predefined condition is encountered, and halt-thereby allowing you to examine the state of processor registers or variables in your program. For more information, consult the *DEBUG-88 User's Guide,* Order Number: 121758.

6.3 Program Development Cycle

The program development cycle begins with an idea and ends with a fully debugged, operational program. The idea evolves into a design and, ultimately, into program specifications. Specifications are split up into smaller groups of functions, each performed by a single module of code.

As work progresses, problems may arise, due perphaps, to unforeseen gaps or complications within the base design or difficulties in design implementation.

Modules may require expansion, modification, or integration with other modules. Functions may merge or be abandoned, and parameter lists may change.

To minimize such problems and the resulting rework and redesign, following these guidelines:

1. Develop clear and specific program goals which have been agreed to by designers, implementers, and users alike.
2. Isolate every non-trivial function of the system or program into separate modules; even isolate difficult design decisions.
3. Write clear and concise documentation for every module, including liberal comments in the code.
4. Write clear standards for module implementation, including conventions for naming and passing parameters.

Modules developed along these guidelines will form the units of actual programming work which are separately specified at the detail level. They are coded, translated, and tested, both individually and in logical groups.

When these logical groups are combined and tested, the complete program approaches final integration. Using input data that reflects the ultimate usage as closely as possible, the final tests explore every major option defined by the original program specifications.

At each stage of individual and multi-module testing, the operating system debugger aids in isolating the source of unexpected results. Under the iNDX operating system, DEBUG-88 permits: use of symbolic names for debugging output; instruction references by line number; access to the processor's registers and flags; and alternate execution modes with or without the use of breakpoints. Under the ISIS-IV operating system, the Monitor debugger or an In-Circuit Emulator provide similar functions.

6.4 Specific System Services for Target Environments

When you are developing a program to run on the Series IV, you must choose one of the two environments provided for program execution: the 8086/8088-based environment or the 8080/8085-based environment. Each has similar built-in facilities to aid in the developing and testing of your program products, including standard system services that can be called from your programs (e.g., I/O). These I/O routines free you from rewriting routines already embedded in the operating system and provide a standard interface for all modules or systems you develop. However, the interface for each operating system environment is unique; the calls and parameters differ.

6.4.1 The 8086/8088-Based Environment

In the 8086/8088-based environment, you develop the modules using the languages that run on the 8086/8088 chip and produce code that works on the 8086/8088 (for example, the resident FORTRAN86, PASCAL-86, ASM-86 or PL/M-86 translators). Earlier versions of these translators ran on the 8080/8085 chip but produced code to run on the 8086/8088 chip. In some cases, modules compiled or assembled with these prior versions of the translators can be used unchanged. Appendix D clarifies the circumstances in which this situation may occur.

Over two dozen system service routines are available in the 8086/8088-based environment. These routines enable you to use the capabilities of the iNDX operating system for resource management in the 8086/8088-based environment.

A conceptual introduction of expected parameters and results of routine execution appears early in Chapter 7. Chapter 7 also contains a complete description of each parameter with the sample PL/M-86 declarations for these external procedures. The description of each routine ends with syntax examples and the list of exception conditions that can occur during the routine's execution. Brief, combined usage examples appear after these discussions.

Whenever you write a module that uses one of these service routines, you simply declare it as an external procedure. LINK86 then provides the correct address to the resident system program. You specify the library appropriate to the PL/M-86 model of segmentation you programmed for: SMALL.LIB, COMPAC.LIB, or LARGE.LIB.

After you link and locate groups of compatible modules in your final system, testing can be conducted on a module or a group. The DEBUG-88 feature described in the *DEBUG-88 User's Guide,* Order Number: 121758, can help you isolate and correct defects.

6.4.2 The 8080/8085-Based Environment

For the 8080/8085-based environment under the ISIS-IV operating system, use the standard 8085-based versions of these same translators to build your modules. These modules will then run under ISIS-IV on the 8080/8085. In this environment, the modules may call upon ISIS routines for a variety of input/output services that already exist as part of the ISIS-IV facilities. Many of these routines operate similarly to those of the 8086/8088-based operating system; some, however, are unique to ISIS-IV operating system. Fourteen ISIS-IV routines and nine Monitor routines are available.

Whenever you write a module using one of these service routines simply declare it an external procedure. When the module is processed by LINK, the correct address to the resident system program is provided. The DEBUG feature of the Monitor can aid the program analysis and correction process. (Refer to the *ISIS-IV User's Guide,* Order Number: 121880, for further discussion of the 8080/8085-based environment).

6.5 Built-in Service Routines

The parameters appropriate to each service routine include the address of a word (filled by the system) indicating whether the desired operation finished successfully. Usually, your call should be followed by code that tests this word—permitting error recovery, alternate processing, or exit, depending on the operation's results. To help you understand the available service routines, the following three tables name all of the 8086/8088-based system routines:

1. Alphabetically (Table 6-1).
2. In groups by function (Table 6-2).
3. By sequence of use in a hypothetical program. Table 6-3 shows the nearest functional equivalent under ISIS-IV. (Some procedures used under the Monitor of ISIS-IV have no direct counterpart in the 8086/8088-based environment.)

All 8086/8088-based procedures begin with DQ$.

Table 6-1. iNDX Service Routines

DQ$ALLOCATE	DQGETSIZE
DQ$ATTACH	DQGETSYSTEMS$ID
DQ$CHANGE$ACCESS	DQGETTIME
DQ$CLOSE	DQ$OPEN
DQ$CREATE	DQ$OVERLAY
DQ$DECODE$EXCEPTION	DQ$READ
DQ$DECODE$TIME	DQ$RENAME
DQ$DELETE	DQ$RESERVE$IO$MEMORY
DQ$DETACH	DQ$SEEK
DQ$EXIT	DQ$SPECIAL
DQ$FILES$INFO	DQ$SWITCH$BUFFER
DQ$FREE	DQ$TRAP$EXCEPTION
DQGETARGUMENT	DQ$TRUNCATE
DQGETCONNECTION$STATUS	DQ$WRITE
DQGETEXCEPTION$HANDLER	

Table 6-2. Service Routines by Functional Groups

Utility and Input Scanning	
DQ$DECODE$TIME DQGETARGUMENT DQGETSYSTEM$ID DQ$GET$TIME DQ$SWITCH$BUFFER	
Memory Management	
DQ$ALLOCATE DQ$FREE DQGETSIZE DQ$RESERVE$EXCEPTION	
File Management	
Program Connection and File Existence	**Program Usage**
DQ$ATTACH DQ$CREATE DQ$DELETE DQ$DETACH DQ$FILE$INFO DQGETCONNECT DQGETCONNECTION$STATUS	DQ$CLOSE DQ$OPEN DQ$READ DQ$SEEK DQ$SPECIAL DQ$TRUNCATE DQ$WRITE
Naming	**Program Control**
DQ$CHANGE$ACCESS DQ$CHANGE$EXTENSION DQ$RENAME	DQ$EXIT DQ$OVERLAY
Exception Handling	
DQ$DECODE$EXCEPTION DQGETEXCEPTION$HANDLER ENDQ$TRAP$CC DQ$TRAP$EXCEPTION	

Table 6-3. Hypothetical Steps in Program Execution & Service Routines Relevant to Each Step

Steps	Service Routine Names	
	8086/8088 Environment	8080/8085 Environment
1. Finds out date and system id for logging/reporting purposes	DQ$DECODE$TIME DQGETTIME DQGETSYSTEM$ID	none
2. Allocates a memory work area for immediate calculation	DQ$RESERVE$IO$ MEMORY DQ$ALLOCATE DQ$GET$SIZE	none
3. Determines if console input is transparent or line-edited	DQ$SPECIAL	none: console is always line-edited
4. Rescans last command (at first, the one invoking this program)	DQGETARGUMENT DQ$WITCH$BUFFER	RESCAN
5. Asks user to enter needed data or parameters at the console	DQ$WRITE DQ$READ	WRITE READ
6. Writes to a file	DQ$WRITE	WRITE
7. Loads overlay to process next phase or user response	DQ$OVERLAY	LOAD
8. Checks if required files are on-line; gets status, including file pointer position	DQ$ATTACH/DQ$OPEN DQGETCONNECTION$ STATUS	SEEK/OPEN
9. Creates files as needed for program reads/writes	DQ$CREATE/DQ$OPEN	OPEN
10. Opens the files for reads/writes	DQ$OPEN	OPEN
11. Reads file(s) or seeks to desired position in file	DQ$READ DQ$SEEK	READ SEEK
12. Calculates	user supplied	none
13. Frees the memory work areas no longer needed	DQ$FREE	none
14. Closes and/or deletes files	DQ$CLOSE DQ$DELETE	CLOSE DELETE
15. Writes new or old file(s)	DQ$WRITE	WRITE
16. Renames certain files or changes extensions on filename string and changes file protection attributes	DQ$RENAME DQ$CHANGE$EXTENSION	RENAME none
17. Change or check file access rights	DQ$CHANGE$ACCESS DQ$FILE$INFO	ATTRIB/GETATT
18. Obtains file device directory information	none	GETD
19. Truncates and/or closes files no longer needed	DQ$TRUNCATE DQ$CLOSE	CLOSE
20. Detaches files not currently needed	DQ$DETACH	none
21. Errors, exceptions, or unwanted conditions can occur at each step. An implicit order after any step is detection/handling of such conditions	DQ$TRAP$EXCEPTION DQ$DECODE$EXCEPTION DQ$TRAP$CC DQGETEXCEPTION $HANDLER	ERROR
22. Exits when job is complete or cannot continue	DQ$EXIT	EXIT

CHAPTER 7
THE 8086/8088-BASED ENVIRONMENT

7.1 Introduction

This chapter discusses each system service routine available in the 8086/8088-based environment of the Series IV. The routines provide a variety of capabilities to programs running on the Series IV. The routines do not, however, require user development and verification because they are part of the operating system.

7.2 Conceptual Consideration

The system service routines, which embody a variety of usage expectations, constitute a model of the way programs interact with files, the console, and each other. The expectations are directly reflected in the parameters you must supply when calling the routines. Following are some of the key concepts underlying the parameters.

7.2.1 Command Tail Arguments

An 8086/8088-based program such as PROGRM can be invoked by typing PROGRM. PROGRM may have options that can be specified on the invocation line. If so, the remainder of that line, including any continuation lines, is called a "command tail."

This command tail is accessible to PROGRM via the DQGETARGUMENT system service routine, which you call to get each option in the command tail. The first parameter of this call tells the system where to put the next option found (i.e., the address of the name you declared in PROGRM as the string to receive these options).

Successive calls to DQGETARGUMENT return successive options, each separated by some delimiting character such as a blank or a parenthesis. (Details for using this routine appear later in this chapter.) The concept of the command tail is basic to the discussion of that routine and can influence your program design.

7.2.2 Memory Management

Memory management routines monitor which memory areas are in use and which are free to be allocated to new uses. Free space memory management is handled by the service routines DQ$ALLOCATE, DQ$FREE, and DQ$GETSIZE.

The Series IV does not support absolute object modules, but does support two types of relocatable object modules: position-independent-code (PIC), and load-time-locatable (LTL). Segment register changes do not occur in PIC modules. The code can work wherever it is ultimately loaded. LTL modules contain special records to resolve program references that do require segment register changes (e.g., an intersegment jump).

When a relocatable object module (PIC or LTL) is loaded, the lower limit of the free space pool is set before the load. Memory required to load the segments is then allocated from the initial free space pool by the free space manager.

A request for memory (i.e., invoking DQ$ALLOCATE) will return the lowest-addressed segment within the requesting job's region. When a segment is freed, it is automatically combined with adjacent free memory to form the largest contiguous area possible.

7.2.3 Connections

The operating system maintains a list of twelve devices or files your program can use during its execution (i.e., a list of "connections"). A connection is a word, named by you, filled by the DQ$ATTACH or DQ$CREATE system service routines. (Only six connections may be open at once, although multiple openings of a single device count as only one of the six.)

You use this word to specify a file or device whenever you need to perform any operation on either of them. For files that already exist, DQ$ATTACH and DQ$DETACH can add or delete connections. New files are connected with DQ$CREATE.

For example, when your program performs console input and output, the connections for :CI: and :CO: must be on this list. The list permits efficient specification and manipulation of devices or files during execution.

Only objects on this list can be opened or closed, read or written. Use the connection rather than the actual device or file name. During execution, your program may perform these functions on multiple files but only six files and devices may be open at one time (not counting :CO:).

Some Series IV service routines include an "internal" open as part of their operation. When you use such a routine, you may need to close another file or device temporarily to avoid exceeding the limit of six open files-plus-devices at once. However, you may open a physical device more than once because it counts only as one open file.

Before a file can be read or written (by DQ$READ or DQ$WRITE), it must be connected and opened (by DQ$OPEN). When the activity to that file is completed, it can be closed (by DQ$CLOSE). These four routines can be used only with connections established earlier.

Output devices are created; input devices are attached. For example, workfiles (defined below) and console output :CO: must be created, not attached. Console input is the opposite; :CI: must be attached rather than created.

7.2.4 Buffers

Buffers are areas reserved for expediting disk input/output. A request to buffer a device will be ignored except when :CO: has been redirected to a disk file.

If a series of read operations can be interspersed with calculation, more efficient operation will result. This occurs when the data being read in are not used immediately. Similarly, if a sequence of write operations can be interspersed with calculation, efficiency rises.

When you open a file, buffers are allocated according to your specifications. Your program will read (only), write (only), or update (both). When a file is opened for write, buffer use begins with the first call to DQ$WRITE. When a file is opened for read or update, buffer use begins with the call to DQ$OPEN. When a file is open for update, the most efficient use is clustering the reads (interspersed with calculations) separately from the writes.

You are responsible for indicating the optimal number of buffers for the type of usage you see for a file. For seldom-used files and random I/O, one buffer is best. With sequential console input/output, zero buffers is the correct specification. In all other cases, double buffering is appropriate.

7.2.5 Workfile

Many programs need temporary files to store immediate results while processing. These files need to be available each time the user logs on. On the Series IV, the designator :WORK: is used to represent these files.

Prior to creating any temporary files, the logical name :WORK: must exist and be assigned to the user's home directory using the LNAME command in the LOGON INT file. This command assigns a directory that allows temporary files to be created and stored under the directory named :WORK:.

Temporary files may be created any number of times. Each time DQ$CREATE is involved with a pointer to the string "6,:WORK:," a new workfile is established.

Workfiles must be created, opened, closed, and detached like any new file. They are automatically deleted when they are detached.

7.2.6 Exception Conditions and Exception Handling

Exceptions, or errors, are detected when an indicated operation cannot be completed. The Series IV operating system classifies errors as either avoidable or unavoidable. Every system service routine except EXIT returns an error code through a pointer (referred to as excep$p in the descriptions later in this chapter). Your programs can test this code to check if the operation you called for completed successfully. Appendix C contains the standard error names and values.

An error code of zero means your call executed successfully. For example, when you call DQ$OPEN to prepare a file for input/output, you supply a connection number. A zero error code would be returned if the connection representing that file were successfully opened.

A non-zero error code indicates an inappropriate event during a routine's execution. For example, a write operation would fail if the connection representing the desired file indicated that the file had already been closed or detached. A non-zero error code would be returned. Your program should always check for this.

7.2.6.1 Unavoidable Errors

Unavoidable errors generally arise from environmental conditions outside program control. Examples include insufficient memory for a requested operation, or inability to find an expected file. These errors always return a non-zero value. Often, appropriate program action can be taken. In other instances, nothing can

be done until the correct disk is found and inserted, or until the available memory is increased by adjusting other program functions.

7.2.6.2 Avoidable Errors

Avoidable errors typically are caused by coding errors such as inappropriate parameters or unusable numeric results. Hardware-detected errors, which also fall into this category, include division by zero (interrupt 0), overflow (interrupt 4), and interrupts generated by the 8087 Numeric Data Processor. These errors cause the system's default exception handler to be executed. (See Appendix C.)

However, you may establish your own routine to handle hardware-detected exceptions by using the system routine DQ$TRAP$EXCEPTION and supplying a pointer to your exception-handler. (The state of the stack when your routine gets control is discussed under DQ$TRAP$EXCEPTION.)

One special kind of exception is defined into the system: when a CONTROL-C is typed at the physical console input device by default, the system cancels the current foreground job. You can program your own response to a CONTROL-C via the system routine DQ$TRAP$CC and thereby cause the system to use the CONTROL-C to provide a pointer to your private routine.

7.2.7 Data Types and Register Convention

The descriptions of the system service routines that appear later in this chapter assume data types similar to those of PL/M-86. Your calls to system service routines must supply parameters that meet the following specifications:

BOOLEAN — A BYTE object taking the values TRUE (OFFH) or FALSE (00H). The BOOLEAN specification assumes the following literal definition (in PL/M-86 terms):

```
DECLARE BOOLEAN LITERALLY 'BYTE' ;
```

BYTE — Equivalent to PL/M-86.

CONNECTION — A token representing a connection to a file or device. The CONNECTION specification assumes the following literal definition (in PL/M-86 terms):

```
DECLARE CONNECTION LITERALLY 'WORD' ;
```

DWORD — Four-byte unsigned integer.

POINTER — Equivalent to PL/M-86. Two bytes in the SMALL model of segmentation, four bytes in all others. SELECTOR Equivalent to PL/M-86.

STRING — A sequence of bytes, the first of which contains the number of bytes following in the sequence (i.e., not including the length byte). A length of zero indicates the null string.

WORD — Equivalent to PL/M-86.

The operating system follows the conventions for interfacing PL/M-86 programs with assembly language programs, saving only registers CS, DS, SS, IP, SP, and BP on a call. Other registers and flags may be used by the operating system routines; upon return to your program, they have no predefined value.

7.3 External Procedure Definitions for Series IV System Service Routines

7.3.1 Introduction

Any module using a service routine must first declare it an external procedure and then link the final object module with the appropriate interface library. Following are the appropriate PL/M-86 declarations with syntax and usage examples for all Series IV routines.

The service routines are presented alphabetically in five categories as follows:

- Exception handling
- File management
- Memory management
- Program control (overlays, exit)
- Utility/command parsing

The file management category has three subclasses:

- Connection to files
- Naming of files
- Use of files

Appendix B lists the routines and parameters in alphabetic order.

Exception Handling Routines

Syntax

```
DQ$DECODE$EXCEPTION: PROCEDURE (exception$code,message$p,excep$p) EXTERNAL
   DECLARE exception$code WORD,
           message$p POINTER,
           excep$p POINTER;
END;
```

Description

exception$code is a word containing an excepion code. *message$p* is a pointer to the 81-byte area (minimum) you declared to receive the error message decoded by the operating system. The first byte of the message is the length of the string. The word whose address is *excep$p* will contain zero unless an unexpected problem in decoding causes a non-zero code to be returned.

The routine returns a string containing the following information:

```
EXCEPTION nnnnH message
```

nnnn is the exception code value.
message is the exception message.

If the *exception$code* you supply as a parameter is not a recognized system error number, an exception number, not a message, will appear.

Example

```
CALL DQ$CODE$EXCEPTION (ERRNO, @ERRMESS(0), @ERR)
```

Exception Codes Returned

```
E$OK
```

Syntax

```
DQ$GET$EXCEPTION$HANDLER: PROCEDURE (handler$p, excep$p) EXTERNAL;
   DECLARE handler$p POINTER,
           excep$p POINTER;
END
```

Description

handler$p must point to a four-byte area the system can fill with a long pointer to the current avoidable-exception handler. A four-byte pointer is always returned even if it is called from a program compiled under the SMALL model of segmentation.

If called, this pointer will be the address specified in the last call of DQ$TRAP$EXCEPTION.

Example

```
CALL DQ$GET$EXCEPTION$HANDLER (@WHICH_HANDLER, @ERR);
```

Exception Codes Returned

EOK, EPTR

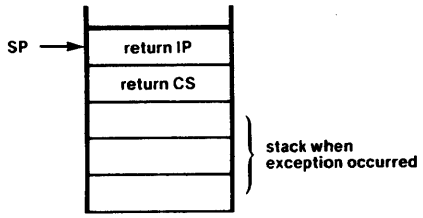
Syntax

```
DQ$TRAP$CC: PROCEDURE (handler$p, excep$p) EXTERNAL;
   DECLARE handler$p POINTER,
           excep$p POINTER;
END;
```

Description

Whenever CONTROL-C is pressed at the physical console device, the system executes the default handler or your specially coded routine, whose address is specified as *handler$p* via this system service call.

When your CONTROL-C routine receives control, the registers and flags are the same as in the interrupted program. At that time, the stack looks like the following figure:



Write the CONTROL-C routine in assembly language. The program must save the 8086/8088 processor flags and registers and load the DS register with the data segment value of the CONTROL-C routine. Before returning to the interrupted program, the CONTROL-C outline must restore the registers and flags and execute a long return.

The default CONTROL-C handler closes files and terminates program execution. If the key was pressed while the system was performing a system service routine (other than a DQ$READ of :CI:), the routine is completed and the CONTROL-C routine is not executed until just before the system returns to the calling program. If a DQ$READ of :CI: is being serviced, the CONTROL-C handler is executed immediately and the DQ$READ returns an actual count of zero to the calling program. If the job is being processed in BACKGROUND mode, the CONTROL-C request will be ignored.

If :CI: has been redirected to another device or a disk file (e.g., under SUBMIT, the special meanings of CONTROL-C and CONTROL-D will not be recognized from the SUBMIT file); they will be treated as ordinary characters unless they come from the cold start console.

Example

```
CALL DQ$TRAP$CC (@SPECIAL1_C1_PTR,@ERR_C);
```

Exception Codes Returned

```
E$OK
```

Syntax

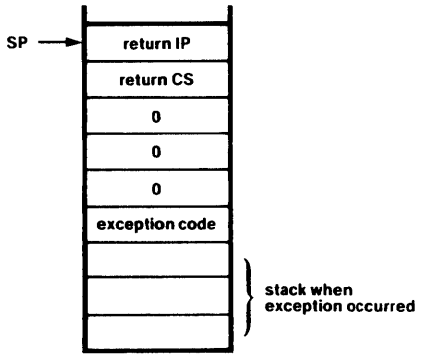
```
DQ$TRAP$EXCEPTION: PROCEDURE (handler$p, excep$p) EXTERNAL;
   DECLARE handler$p POINTER
           excep$p POINTER;
END;
```

Description

handler$p is the address of a four-byte area containing a long pointer to the entry point of your exception handler. Programs compiled under the SMALL model of segmentation have no access to CS and thus cannot create the long pointer directly

Hardware-detected exceptions will cause the exception handler to be executed. The state of the stack upon entry looks as though the instruction pushed four words and then executed a long call to the exception handler. The first word pushed is the condition code. The next three words are reserved for future use by the operating system and the numeric data processor.

Upon entry to the exception handler, the stack looks like the following diagram:



See Appendix C for exception code values and descriptions.

The default system action displays an error message, closes files, and terminates program execution. Following is the message format:

```
*** EXCEPTION nnnnH error message
    CS:IP = xxxx:yyyy
```

Example

```
EXCEP_ROUT=DQ$TRAP$EXCEPTION (@USER_HANDLER, @ERR);
```

Exception Codes Returned

```
E$OK
```

File Management Routines

Connection Routines

Syntax

```
DQ$ATTACH: PROCEDURE (path$p, excep$p) CONNECTION EXTERNAL;
   DECLARE path$p POINTER,
           excep$p POINTER;
END;
```

Description

path$p points to a string containing a pathname. This string must begin with a number, which denotes the length of the character string. The standard format for defining these strings in the Series IV operating system is x, "string". If the pathname consists only of a logical name, the connection created will refer to the directory file indicated by the logical name.

Only input devices and disk files that are not workfiles can be specified via the pathname. Attempting to attach :CO: (or :LP:) will cause an E$SUPPORT exception condition. (However, you may DQ$ATTACH, DQ$CREATE, or DQ$SEEK the Byte Bucket, :BB:.)

The console input device must be attached only via the :CI: pathname. The console output device must be created only via the :CO: pathname.

A maximum of twelve connections can be maintained by Series IV; multiple connections to physical devices (e.g., :LP:) and logical devices (e.g., :BB:, :CI:) are allowed.

DQ$ATTACH operates as a function (i.e., a typed procedure). It returns a connection to an existing file in the variable to the left of the equal sign. If the named file does not exist, the operation will fail and return a non-zero error code at the address pointed to by *excep$1*.

Examples

```
DECLARE TAX_CONNECTION WORD, ERR WORD;
TAX_CONNECTION = DQ$ATTACH (@(10, 'FEDTAX.JUN') , @ ERR);
```

Exception Codes Returned

```
E$FNEXIST, E$OK, E$SYNTAX, E$MEM, E$LIMIT, E$TYPE,
E$SUPPORT, E$DEVICE$NOT$READY, E$DEVICE$ERROR,
E$COMM$ERROR, E$NODE$NOT$READY, E$MARKED$DELETED, E$PARAM
```

Syntax

```
DQ$CREATE: PROCEDURE (path$p,excep$p) CONNECTION EXTERNAL;
   DECLARE path$p4, POINTER,
           excep$p POINTER;
END;
```

Description

DQ$CREATE is a type procedure that returns a connection to a new file. If a file of the same name exists and is not connected, an attempt will be made to delete the existing file and create a new file with this name. This function will fail if the user does not have delete access to the existing file. A non-zero error code will then be returned in the location pointed to by *excep$p*.

path$p points to a string containing a pathname.

Example

```
DECLARE NEW_CONNECTION WORD, ERR WORD;
NEW_CONNECTION = DQ$CREATE (@ (15, '/TAX/NEWTAX.AUG') , @ERR) ;
/*A connection number will be created for*/
/*the named file and stored in NEW_CONNECTION.*/
```

Exception Codes Returned

```
E$SHARE, E$FACCESS, E$OK, E$SYNTAX, E$SPACE, E$LIMIT,
E$MEM, E$SUPPORT, E$FNEXIST, E$FTYPE, E$PARAM,
E$CONNECTION$EXIST, E$DEVICE$NOT$READY, E$COMM$ERROR,
E$DEVICE$IO$ERROR, E$NODE$NOT$READY, E$SPACE, E$PARAM,
E$FEXIST
```

Syntax

```
DQ$DELETE: PROCEDURE (path$p, excep$p) EXTERNAL;
   DECLARE path$p POINTER
           excep$p POINTER;
END;
```

Description

This routine deletes the file specified by *path$p*. *path$p* points to a string containing a pathname.

File deletion is a logical operation rather than a physical one. The deletion actually occurs when the last connection to the file is detached.

In addition, the supplied pathname must contain a file name part, and the file to be deleted must not have the write-protect set. A directory file will not be deleted if the directory contains files.

If this operation fails, a non-zero error code will be returned in the location pointed to by *excep$p*.

Example

```
CALL DQ$DELETE (@FILE$PTR, @ERR ) ;
/*The file pointed to by FILE$PTR will be deleted,*/
/*assuming it meets the above conditions.*/
```

Exception Codes Returned

```
E$FACCESS, E$OK, E$PTR, E$SUPPORT, E$SYNTAX, E$FNEXIST,
E$DEVICE$NOT$READY, E$PARAM, E$DEVICE$IO$READY,
E$COMM$ERROR, E$NODE$NOT$READY
```

Syntax

```
DQ$DETACH: PROCEDURE (conn, excep$p) EXTERNAL;
   DECLARE conn CONNECTION,
           excep$p POINTER;
END;
```

Description

DQ$DETACH breaks the connection created by DQ$ATTACH or DQ$CREATE. If the connection is open, it is closed before being detached.

conn is a word representing the connection to be detached (i.e., removed from the current list of attached devices or files).

Example

```
CALL DQ$DETACH ( PAY_FILE_CONNECTION, @ERR);
/*The file whose connection is PAY_FILE_CONNECTION*/
/*will be closed and removed from the list of*/
/*connected/attached files, i.e., it will be detached.*/
```

Exception Codes Returned

```
E$EXIST, E$OK, E$PARAM, E$DEVICE$NOT$READY,
E$DEVICE$IO$ERROR, E$COMM$ERROR, E$NODE$NOT$READY
E$DETACHED
```

Syntax

```
DQ$GET$CONNECTION$STATUS: PROCEDURE (conn, info$p, excep$p) EXTERNAL;
   DECLARE conn CONNECTION,
           info$p POINTER,
           excep$p POINTER;
END;
```

Description

This routine supplies the connection status of a connection established earlier. *conn* is a connection established earlier via attach or create. As seen in the following example, the parameter *info$p* points to a structure you have declared to receive the connection data found by this routine.

```
DECLARE INFO STRUCTURE
(OPEN0     BOOLEAN,
ACCESS0    BYTE,
SEEK0      BYTE,
FILE$PTR$ DWORD);
```

The fields just listed have the following interpretations:

OPEN True if connection is open, otherwise false.

ACCESS Access privileges of the connection. The rights are granted if the corresponding bit is on.

Data Files		Directory Files	
Bit	**Access**	**Bit**	**Access**
0	delete	0	delete
1	read	1	display
2	write	2	add-entry
3	update		

SEEK Following are the types of seek supported:

Bit	Seek Types
0	seek forward
1	seek backward
2–7	undefined

FILE$PTR Current position of the file pointer is interpreted as a four-byte unsigned integer (DWORD) representing the number of bytes from the beginning of the file. This file is undefined if the file is not open or if seek backward is not supported.

When the connection you specified is established on a physical device or a logical device, the access value returned depends on the nature of the device. For example, access privilege of the line printer is write. For a disk file with the write protect attributes set, access is read; for workfiles, access is read, write, and update. All other disk files have access privileges of delete, read, and write.

Physical devices and the console do not support any type of seek. For the byte bucket (:BB:), the returned file pointer is 0.

Example

```
CALL DQ$GET$CONNECTION$STATUS (INVENTORY_CONN,
@FILE_STATUS, @ERR);
```

Exception Codes Returned

E$EXIST, E$OK, EPTR, EPARAM, E$COMM$ERROR

Syntax

```
DQ$FILE$INFO: PROCEDURE (conn, mode, file$info$p, excep$p) EXTERNAL;
   DECLARE conn CONNECTION,
           mode BYTE,
           file$info$p POINTER,
           excep$p POINTER;
END;
```

Description

DQ$FILE$INFO returns the file information normally associated with user security and accounting.

Following are the input parameters:

conn	identifies the connection of a currently attached file.
mode	indicates whether the file owner is to be identified. Byte Value 0 indicates that the owner name or identification is not to be returned. Byte Value 1 indicates that the owner name or identification is to be returned.
file$info$p	points to a table used for output.

As seen in the following example, the output of *file$info$p* points to the structure you declare to receive the file information.

```
DECLARE FILE$INFO                STRUCTURE
(OWNER(15)                       STRING,
LENGTH0                          DWORD,
TYPE                             BYTE,
OWNER$ACCESS                     BYTE,
WORLD$ACCESS                     BYTE,
CREATE$TIME                      DWORD,
LAST$MODE$TIME  DWORD,           RESERVED(20)
BYTE);
```

The fields just listed have the following interpretations:

- OWNER is a string that identifies the system name of the owner of the file.
- TYPE indicates the type of file. 0 for a data file, 1 for a directory file, and 2–255 reserved.
- OWNER$ACCESS and WORLD$ACCESS describe the access rights of the file owner and the world.

  Bit 0 = delete access.
  Bit 1 = read access (for data files) or display access (for directory files).
  Bit 2 = write access (for data files) or add_entry access (for directory files).
  Bit 3 = update (read and write) access.
- CREATE$TIME, LAST$MODE$TIME indicates the date and time of creation and last modification for a file. If a file has been created but not modified, the LAST$MODE$TIME should be the same as the CREATE$TIME. A modification consists of a write or update since January 1, 1978. These dates may be decoded into an ASCII string with DQ$DECODE$TIME.

Example

```
CALL DQ$FILE$INFO (INVENTORY_CONN,0,@FILE_STATUS,@ERR);
```

Exception Codes Returned

```
E$OK, E$SUPPORT
```

Naming Routines

Syntax

```
DQ$CHANGE$ACCESS: PROCEDURE (path$p, class, access,excep$p) EXTERNAL;
   DECLARE path$p POINTER,
           class BYTE,
           access BYTE,
           excep$p POINTER;
END;
```

Description

DQ$CHANGE$ACCESS changes the owner and world access rights to a file. The privilege to use this primitive is assured for the owner of the file. The granting of this privilege to other users is operating-system dependent. If the privilege is not granted, the error E$FACCESS is supported. E$FNEXIST indicates the file does not exist; E$SUPPORT indicates an attempt to change the access rights of a non-disk file. The access rights of the file will be changed immediately but will not affect connections to the file until they are detached.

Following are the input parameters:

path$p points to a string containing the pathname of the file whose access rights are to be changed.

class specifies the class of users whose access rights are to be changed. 0 indicates the owner, 1 the world, and 2–255 are reserved.

access specifies the type of access to be granted to the class of file users specified. If all bits are set to 0, the specified user's access to the file will be denied. If any bits are set to 1, the following access is granted:

Bit 0 = delete access (for data files and directory files).
Bit 1 = read access (for data files), display (for directory files).
Bit 2 = write access (for data files), add-entry access (for directory files).
Bit 3 = update access (read and write).

No output is returned by this call.

Example

```
CALL DQ$CHANGE$ACCESS (@FILE_NAME, 0, 2, @ERR);
```

Exception Codes Returned

EOK, EMEM

Syntax

```
DQ$CHANGE$EXTENSION: PROCEDURE (path$p, extension$p, excep$p) EXTERNAL;
   DECLARE path$p POINTER,
           extension$p POINTER,
           excep$p POINTER;
END;
```

Description

DQ$CHANGE$EXTENSION replaces any existing extension on the file name with the supplied extension. *path$p* points to a string containing the pathname to be changed. *extension$p* points to a three-character extension that is to become the extension in the pathname. These characters may not be delimiters. This procedure changes only the specified string and performs no file operations.

If the first character addressed by *extension$p* is a blank, any prior extension of the file-name, including the trailing period, will be deleted. Trailing blanks are allowed (i.e., the third character or both the second and third characters of the new extension may be blanks).

Examples

1.
```
CALL DQ$CHANGE$EXTENSION (@(8, 'TASK.QRY'),
@ ('ANS'), @ERR$P);
/*Filename string will be changed from*/
/*TASK.QRY to TASK.ANS*/
```

2.
```
CALL DQ$CHANGE$EXTENSION ( FILE$PTR, @('OBJ'), EXCEP$P);
/*This will change the extension on the filename*/
/*pointed to by FILE$PTR to be .OBJ*/
```

Exception Codes Returned

EOK, ESTRINGBUF, EPTR, E$SYNTAX

Syntax

```
DQ$RENAME: PROCEDURE (old$p, new$p, excep$p) EXTERNAL;
   DECLARE old$p POINTER,
           new$p POINTER,
           excep$p POINTER;
END;
```

Description

DQ$RENAME changes the name of a file within its parent directory. The new file name is not restricted to a single path component; however, the old and new pathnames should differ only in the last component. *old$p* and *new$p* are pointers to the strings containing the existing pathname and the new pathname, respectively.

An exception condition occurs if a file with a new name already exists or if the file to be renamed has the write-protect attribute set. Renaming a file on which a connection is established is valid, and the connection does not need to be re-established.

Example

```
CALL DQ$RENAME (@FILE_PTR(3), @(9, 'TERMS.NOV'), @ERR);
```

Exception Codes Returned

```
E$FACCESS, E$FEXIST, E$FNEXIST, E$OK, E$PTR, E$SUPPORT,
E$SYNTAX, E$MEM, E$FTYPE, E$DEVICE$NOT$READY,
E$COMM$ERROR, E$NODE$NOT$READY, E$PARAM
```

Usage Routines

Syntax

```
DQ$CLOSE: PROCEDURE (conn, excep$p) EXTERNAL;
   DECLARE conn, CONNECTION,
           excep$p POINTER;
END;
```

Description

DQ$CLOSE waits for completion of input/output operations (if any) taking place on the file, ensures output buffers are empty, and frees buffers. Once closed, a connection may be either re-opened or detached. CLOSE does not truncate the file; the original extent (or the new extent enlarged by writes) is maintained.

conn represents a connection established earlier using attach or create and opened via OPEN.

Programs that attach :CI: and create :CO: should open, close and detach them. :CI: and :CO: do not count towards the limit of open files.

Example

```
CALL DQ$CLOSE (TAX;_CONNECTION, @ERR);
/*The file whose connection number is in*/
/*TAX_ENCONNECTION will be closed.*/
```

Exception Codes Returned

```
E$EXIST, E$NOPEN, E$PARAM, E$OK, E$DEVICE$NOT$READY,
E$DEVICE$IO$ERROR, E$COMM$ERROR, E$MODE$NOT$READY,
E$SPACE
```

Syntax

```
DQ$OPEN: PROCEDURE (conn, access, num$buf, excep$p) EXTERNAL;
   DECLARE conn CONNECTION,
           access BYTE,
           num$buf BYTE,
           excep$p POINTER,
END;
```

Description

conn represents a connection established earlier via attach or create.

Value	Type
1	read access only
2	write access only
3	update (both read and write)

num$buf indicates the optimal number of buffers and should be 0, 1 or 2. Zero means that no buffering should occur; physical I/O should occur during a DQ$READ or DQ$WRITE. For seldom-used files, *num$buf* should be 1. In all other cases, it should be 2 for double buffering. If program computation cannot be interspersed as described earlier in this chapter, num$buf should be 0 to maximize performance.

Files can be opened "externally" using this routine or "internally" as part of the operation of other system service routines. The limit of six open files and devices includes those opened internally by the system. DQ$ATTACH, DQ$CREATE, DQ$TRUNCATE, and entry to DEBUG-88 all perform an "internal open." Entering DEBUG-88 internally opens :CI:.

Since it is possible to establish multiple connections on the same device, multiple opens of such a device are also permitted. The device still counts as only one open device on the list of six. The console input device and output devices do not count toward this limit.

If *access* is write or update, the file represented by the connection must not have the write-protect or format attributes set. The file pointer is set to 0 (i.e., the beginning of the file). If the next access to this file is write, writing begins at the first byte, destroying earlier contents. You must read or seek to the end of the file first, and then write to it to append information. The use of DQ$OPEN must not violate physical limitations (e.g., the line printer must not be opened for read or update.

Example

```
CALL DQ$OPEN (EMPLOYEE_CONN, 3, 2, @ERR);
```

Exception Codes Returned

```
E$EXIST, E$FACCESS, E$OK, E$PARAM, E$OPEN, E$SIX,
E$SHARE, E$FNEXIST
```

Syntax

```
DQ$READ: PROCEDURE (conn, buf$p, count,except$p) WORD EXTERNAL;
   DECLARE conn CONNECTION,
           buf$p POINTER,
           count WORD,
           excep$p POINTER;
END;
```

Description

conn represents an open connection established earlier via attach or create. *buf$p* points to a buffer area at least count bytes long which you have declared to receive the data read.

This routine is used as a function (i.e., a typed procedure returning the number of bytes actually transferred. This number will equal count unless an error occurred or an end-of-file was encountered.

count bytes are read from the current location of the file pointer and placed in your buffer. If the procedure returns a value less than count and an exception code of E$OK, end-of-file was encountered.

If your buffer is not long enough to receive the number of bytes requested, this routine will over-write the memory locations that follow the buffer.

DQ$READ will not recognize CONTROL-C and CONTROL-D as having any special meaning if the console has been assigned to a disk or device other than the cold start console. (See also DQ$TRAP$CC.)

Example

```
DECLARE ACTUAL WORD ENTRIES (256) BYTE, ERR WORD;
ACTUAL = DQ$READ (JOURNAL_CONN, @ENTRIES(0), 256, @ERR);
```

Exception Codes Returned

E$EXIST, E$NOPEN, EOK, EOWRITE, E$PARAM, E$PTR

Syntax

```
DQ$SEEK: PROCEDURE (conn, mode, offset, except$p) EXTERNAL;
   DECLARE conn CONNECTION,
           mode BYTE,
           offset DWORD,
           except$p POINTER;
END;
```

Description

conn represents a currently open connection established earlier via attach or create.

mode indicates the type of seek required.

Value	Type
1	move file pointer back by offset
2	set pointer to offset
3	move file pointer forward by offset
4	move file pointer to end-of-file minus offset

offset forms a four-byte unsigned integer (DWORD) representing the number of bytes required to move the file pointer.

If the seek goes beyond the end-of-file and a subsequent write occurs, a file opened for write or update will be extended with nulls. For a file opened for read, such a seek will position the pointer beyond the end-of-file. If a seek would move the pointer to a position before the start of the file, the pointer is set to the beginning of the file. Seeks are invalid on connections to physical devices or the console.

Example

```
CALL DQ$SEEK (IONS_CONN, 3, 22, @ERR);
```

This call does a seek from the current position forward—by an offset of 22 x 2 to the 16th power bytes—on the file whose connection is IONS_CONN.

Exception Codes Returned

```
E$EXIST, E$NOPEN, E$OK, E$PARAM, E$SUPPORT, E$SPACE,
E$DEVICE$IO$ERROR, E$DEVICE$NOT$READY, E$COMM$ERROR,
E$NODE$NOT$READY
```

Syntax

```
DQ$SPECIAL: PROCEDURE (type, parameter$p, excep$p) EXTERNAL;
   DECLARE type BYTE,
           parameter$p POINTER,
           excep$p POINTER;
END;
```

Description

This routine determines whether subsequent console input is transparent or line-edited.

type = 1 indicates the subsequent console input is transparent (i.e., not line-edited).

type = 2 indicates the subsequent console input will be lineedited. The initial default, when a job begins, is type two.

type = 3 is identical to a type of 1 except that a DQ$READ of :CI: will return only the single character already in the system buffer.

type = 4 sets the word pointed to by PARAM_P to 1. This indicates to the calling program that it is running on a Series IV.

parameter$p must point to a connection representing a DQ$ATTACH of :CI:.

:CI: can be assigned to a disk file with a SUBMIT system call. This routine returns E$SUPPORT if type 1 or type 3 is specified; this occurs even if :CI: is temporarily restored to the cold start console via control E.

A call to this routine changes :CI: from line-edited to transparent and causes all characters currently in the physical console buffer to be discarded. The key buffer can be cleared by reading 32 characters in polling (DQ$SPECIAL type 3) made from :CI: and discarding the results.

Example

```
CALL DQ$SPECIAL (1, @CI_CONN, @ERR);
```

Exception Codes Returned

EOK, EEXIST, E$PARAM, E$PTR, E$SUPPORT

Syntax

```
DQ$TRUNCATE: PROCEDURE (conn, excep$p) EXTERNAL;
   DECLARE conn WORD,
           excep$p POINTER;
END;
```

Description

This routine truncates the file represented by *conn* at the current file pointer and frees all previously allocated disk space beyond that pointer value. (If the pointer is at the end-of-line or past the end-of-file, truncation has no effect.)

conn represents a connection established earlier via attach or create and currently open for write or update.

Example

```
CALL DQ$TRUNCATE (INTERIM_CONN, @ERR);
```

Exception Codes Returned

```
E$OK, E$EXIST, E$NOPEN, E$SHARE, E$SPACE, E$PARAM, E$MEM,
E$OPEN$MODE, E$SUPPORT, E$DEVICE$NOT$READY,
E$NODE$NOT$READY
```

Syntax

```
DQ$WRITE: PROCEDURE (conn, buf$p, count, excep$p) EXTERNAL;
   DECLARE conn CONNECTION,
           buf$p POINTER,
           count WORD,
           excep$p POINTER;
END;
```

Description

conn represents an open connection established earlier via attach or create. Access must be write or update.

buf$p points to the start of the data to be written out.

count is the number of bytes to be written.

If the *count* exceeds the remaining length of your buffer, the contents of memory locations following the buffer will be written to the device or file represented by the connection you supplied.

Writing begins at the current value of the file-pointer, which is 0 if no prior reads, seeks, or writes to this file have occurred.

A write to a pre-existing file you have opened will destroy earlier contents unless the file-pointer is first positioned (by a seek) at the end-of-file or after the end-of-file. A subsequent close, however, does not truncate the file; the original extent (or the new extent enlarged by writes) is maintained.

Example

```
ALL DQ$WRITE (INVENTORY_CONN,@PHYSICAL, 256, @ERR);
```

Exception Codes Returned

```
E$EXIST, E$NOPEN, E$OK, E$PARAM, E$MEM, E$OPEN$MODE,
E$SPACE, E$DEVICE$NOT$READY, E$DEVICE$IO$READY,
E$COMM$ERROR, E$NODE$NOT$READY, E$DEVICE$IO$ERROR
```

NOTE

DQ$WRITE can return the error message E$OK when the DQ$READ call actually results in I/O errors. To detect this error, a subsequent DQ$READ, DQ$WRITE, DQ$SEEK, DQ$CLOSE, or DQ$TRUNCATE call should be made to the same file connection.

Memory Management Routines

Syntax

```
DQ$ALLOCATE: PROCEDURE (size, excep$p) SELECTOR EXTERNAL;
   DECLARE size WORD,
           excep$p POINTER;
END;
```

Description

size is the number of bytes of memory desired. A size of zero means a request for 64K bytes. If enough memory is available, this function returns a SELECTOR representing the base of the acquired memory block»at is, the segment part of the pointer to the acquired area. (The offset of this pointer is zero.) If the operation fails, this SELECTOR will be 0FFFFH.

When a program allocates memory using DQ$ALLOCATE, the program must not try to access more memory than that allocated by the DQ$ALLOCATE call. DQ$ALLOCATE is used as a function (i.e., to the right of an equal sign in a PL/M-86 assignment statement. The variable to the left of the equal sign is filled with the segment token.

Example

```
PAY_REC_STRUC_BASE = DQ$ALLOCATE (12, @ERROR1_PAY);
   IF (PAY_REC_STRUC1_BASE = 0FFFFH) THEN CALL MY_ERRCHK (PAY-
   _REC_STRUC_BASE, LESS_MEM);
IF (ERROR_PAY 0) THEN CALL MY_ERRCHK (ERROR_PAY, MEM_ERR);
```

Exception Codes Returned

```
E$OK, E$MEM
```

Syntax

```
DQ$FREE: PROCEDURE (segment, excep$p) EXTERNAL;
   DECLARE segment SELECTOR,
           excep$p POINTER;
END;
```

Description

segment is a SELECTOR representing a memory segment acquired earlier from DQ$ALLOCATE. The indicated segment is freed. This liberation should be done at the end of the task that allocated the segment, or whenever the segment will no longer be needed.

DQ$EXIT automatically frees all memory segments allocated by DQ$ALLOCATE. Therefore, you need only DQ$FREE a segment when the program needs to reclaim the space for another purpose. Once a program has freed a particular segment, do not access memory in that freed segment.

Example

```
CALL DQ$FREE (PAY_REC_STRUC_BASE, @ERROR_LESS)
```

Exception Codes Returned

```
E$OK, E$BAD$SEGMENT
```

Syntax

```
DQ$GET$SIZE: PROCEDURE (segbase, excep$p) WORD EXTERNAL;
   DECLARE segbase SELECTOR,
           excep$p POINTER;
END;
```

Description

This function returns the size of the segment in bytes; zero means 64K bytes. This segment must have been previously allocated with the DQ$ALLOCATE routine.

segbase is a SELECTOR for a memory segment.

The loader uses DQ$ALLOCATE to get the memory it requires to load your program from the memory manager. When you link your program, you can specify an "expanding segment", which means the size will be determined when the program is loaded. The size depends upon the amount of memory available.

Relocatable PL/M-86 programs compiled under the SMALL model of segmentation can use an expanding data segment whose size is determined with a statement that has the following form:

SIZE = DQGETSIZE (STACKBASE, @EXCEP);

In SMALL model programs, the stack segment base and the data segment base are the same. Thus, the above PL/M-86 statement passes the token representing the data segment base and obtains the data segment size.

Example

```
DECLARE ARRAY_SIZE WORD;
ARRAY_SIZE = DQ$GET$SIZE (ARRAY_BASE, @ERR);
```

Exception Codes Returned

EOK, EBAD$SEGMENT

Syntax

```
DQ$RESERVE$IO$MEMORY: PROCEDURE (number$files, number$buffers, excep$p)
EXTERNAL;
   DECLARE number$files WORD,
           number$buffers WORD,
           excep$p POINTER;
END;
```

Description

DQ$RESERVE$IO$MEMORY informs the operating system of the maximum number of files to be attached, and the maximum number of buffers to be requested during the execution of a particular program. The call requests the system to reserve enough memory to assure that the Creates, Attaches and Opens will be successful. The default value for the two variables specified in this function is zero. Furthermore, you may assume that at least twelve Attaches and six Opens will be supported by calling DQ$RESERVE$IO$MEMORY although some operating systems may allow for more.

You can use this to anticipate calls to DQ$ATTACH and DQ$OPEN which you will need to make. By warning the operating system of these calls, the system can reserve memory so that intervening calls to DQ$ALLOCATE do not prevent the Attaches and Opens from being successful. Successive calls to this function are legal; they simply change the current number of buffers requested for the corresponding program. A request to increase the number of buffers can fail due to a lack of memory, especially if calls to DQ$ALLOCATE have been made since the previous call to DQ$RESERVE$IO$MEMORY. The call to DQ$RESERVEIOMEMORY should occur before the first call to DQ$ALLOCATE, thereby maximizing the probability that it will be successful.

The input parameters are *number$files* and *number$buffers*. *number$files* is the maximum number of files to be attached at any one time. If this value is exceeded, the application takes the responsibility for ensuring that the additional memory is preconfigured or available for allocation when the calls to DQ$ATTACH and DQ$CREATE are made. *number$buffers* is the maximum number of buffers to be required for any concurrent set of open files. *number$buffers* limits the total number of *number$buffers* parameters allowable in any set of calls to DQ$OPEN for which corresponding calls to DQ$CLOSE have not been made. If this parameter is exceeded, the application takes the responsibility of ensuring that the additional memory is available for allocation when the calls to DQ$OPEN are made.

For compatibility when porting to other operating systems, DQ$RESERVE$IO$MEMORY allows the maximum number of connections and opens allowed by that operating system to be used. The default case in the Series IV is a maximum of 12 connections and 6 opens.

Example

```
CALL DQ$RESERVE$IO$MEMORY (NUMBER_FILES, NUMBER_BUFFERS, @ERR);
```

Exception Codes Returned

EOK, EMEM

Program Control Routines

Syntax

```
DQ$EXIT: PROCEDURE (completion$code) EXTERNAL;
   DECLARE completion$code WORD;
END;
```

Description

Exit terminates a job. All files are closed and all resources are freed. If ISIS-IV was in control, it prompts for another command.

The *completion$code* indicates whether termination was normal. Following are its values and interpretations:

0—normal termination
1—warning messages were issued
2—errors were detected
3—fatal errors were detected
4—job aborted

completion$code has no exception pointer argument because it never generates an exception.

Example

```
CALL DQ$EXIT (COMPL);
```

Exception Codes Returned

None.

Syntax

```
DQ$OVERLAY: PROCEDURE (name$p, excep$p) EXTERNAL;
   DECLARE name$p POINTER,
           excep$p POINTER;
END;
```

Description

This routine causes the loading of the overlay whose name is the string pointed to by *name$p*. Only one level of overlays is allowed. This routine may be called only from the root (non-overlaid) phase.

You must define the overlay name with the LINK86 OVERLAY control. The name must not exceed 40 characters. The string used in the call must match the name used in LINK86.

(See the *iAPX 86, 88 Family Utilities User's Guide,* 121616, for a full discussion of overlays.)

Example

```
CALL DQ$OVERLAY (@(10 'MYPROG.OV2'), @ERR);
```

Exception Codes Returned

```
E$OK, E$EXIST, E$PARAM, E$SIX, E$SYNTAX, E$UNSAT,
E$ADDRESS, E$BAD$FILE
```

Utility and Command Parsing Service Routines

Syntax

```
DQ$DECODE$TIME: PROCEDURE (dt$p, excep$p) EXTERNAL;
   DECLARE dt$p POINTER
           excep$p POINTER;
END;
```

Description

DQ$DECODE$TIME decodes the operating system dependent time and date DWORD into ASCII data and time strings. It returns the current date and time in binary DWORD format and/or as a decoded ASCII string.

dt$p is a pointer to a user declared structure of the following form:

```
DECLARE DT STRUCTURE
 (SYSTEM$TIME DWORD,
   DATE(8)  BYTE,
   TIME(8)  BYTE);
```

system$time the output parameter, is an operating system dependent formatted DWORD containing the time and date. If *system$time* is zero, the system clock is first read to obtain the current date and time. If *system$time* is non-zero, it is simply decoded into the ASCII date and time string.

system$time will contain the binary format of the current date and time as selected by the input value zero. The specified format is in seconds, beginning with January 1, 1978.

DATE has the form *MM/DD/YY* for month, day, and year. TIME has the form *HH:MM:SS* for hours, minutes, and seconds. The value for hours is in the range 0–23.

Example

```
CALL DQ$DECODE$TIME (@TIME-FILE,@ERR);
```

Exception Codes Returned

EOK, ESUPPORT

Syntax

```
DQ$GET$ARGUMENT: PROCEDURE (argument$p, excep$p) BYTE EXTERNAL;
   DECLARE argument$p POINTER,
           excep$p POINTER;
END;
```

Description

This function returns the arguments in the command line or usersupplied buffer. Each successive call returns the next argument.

argument$p points to an 81-byte area you have declared to receive a strong argument from the command tail (see the section on command tail arguments at the beginning of this chapter).

This typed procedure is used as a function (i.e., on the right side of an equal sign in a PL/M-86 assignment statement). The variable to the left of the equal sign is filled with the delimiter found by DQGETARGUMENT, as shown in the examples below. If the exception code returned is zero, the call functioned properly. The possible delimiters include:

,) (= # ! $ % \‘
- + |] [> < <cr> SPACE (20H)

or have a hexadecimal value of 0–20H for ASCII characters. For non-ASCII characters, possible delimiters also include hexadecimal values ⩾ 80H.

The command line is pre-scanned when your program is invoked. At that time, the system makes the following changes to the command line before it is saved in a system buffer:

1. Each continuation line sequence is converted to a blank. A continuation line sequence begins with an ampersand (&) and ends with the line terminator.
2. A comment is removed entirely. A comment begins with a semicolon (;) and ends with the character preceding the line terminator.
3. Any DEBUG commands preceding the pathname are removed.

The following rules apply to the arguments and delimiters returned by DQGETARGUMENT:

1. Lowercase alphabetic characters, except inside quotes, are converted to uppercase.
2. Multiple adjacent blanks separating two arguments are treated as one blank. One or more blanks adjacent to any other delimiter are ignored. A tab is treated like a blank and returned as a blank.
3. Strings enclosed within a pair of matching quotes are considered literals and are not scanned for interpretation. The enclosing quotes are not returned as part of the argument. Quotes may be used in a quoted string by doubling the quote.

The DQ$SWITCH$BUFFER routine may be used to get arguments from a user-supplied buffer. The command line pre-scan is not performed on this buffer.

Examples

The following examples illustrate the argument returned by calls to DQGETARGUMENT:

1. `PLM86.86 LINKER.PLM PRINT(:LP:)NOLIST`

Argument	Delimiter
8,'PLM86.86'	' '
10,'LINKER.PLM'	' '
5,'PRINT'	'('
4,':LP:'	')'
6,'NOLIST'	<cr>

2. `PLM86.86 MODULE.SRC PRINT(/VOL1/THISIS.IT)OPTIMIZE(0)TITLE('MY MODULE')`

Argument	Delimiter
8,'PLM86.86'	' '
10,'MODULE.SRC'	' '
5,'PRINT'	'('
13,'/VOL1/THISIS.IT'	')'
8,'OPTIMIZE'	'('
1,'0'	')'
5,TITLE	'('
9,MY MODULE	')'
0	<cr>

3. `LINK86.86 /VOL4/X.OBJ,LLIB(MODL), SYSTEM;LIB,PUBLICS &`
   `(/VOL3/FUNNY.LIB(MOD1)) MAP;` my link command

Argument	Delimiter
9,'LINK86.86'	' '
9,'/VOL4/X.OBJ'	','
4,'LLIB'	'('
4,'MODL'	')'
10,'SYSTEM.LIB'	','
7,'PUBLICS'	'('
13,'/VOL3/FUNNY.LIB'	'('
4,'MOD1'	')'
0	')'
3,'MAP'	<cr>

Example

```
DECLARE DELIM_SCAN BYTE, ERR WORD;
DECLARE NEXT_ARG STRUCTURE
             (LENGTH BYTE, ARG (80) BYTE);
DELIM_SCAN=DQ$GET$ARGUMENT
             (@NEXT_ARG.LENGTH, @ERR);
```

Exception Codes Returned

`E$OK, E$STRING$BUF, E$PTR`

Syntax

```
DQ$GET$SYSTEMS$ID: PROCEDURE (id$p, excep$p) EXTERNAL;
   DECLARE id$p POINTER,
           excep$p POINTER;
END;
```

Description

This routine returns a string identifying the operating system. *id$p* must point to a 21-byte buffer you define in your program.

Example

```
CALL DQ$GET$SYSTEM$ID (@ID, @ERR);
```

Exception Codes Returned

```
E$OK, E$PTR
```

Syntax

```
DQ$GET$TIME: PROCEDURE (dt$p, excep$p) EXTERNAL;
   DECLARE dt$p POINTER,
           excep$p POINTER;
END;
```

Description

This routine will return the system date, which can be set by the DATE command. *dt$p* must be a pointer to a user structure of the following form:

```
DECLARE DT STRUCTURE
(DATE(8) BYTE, TIME(8) BYTE);
```

DATE has the form *MM/DD/YY* for month, day, and year. TIME has the form *HH/MM/SS* for hours, minutes, and seconds.

Example

```
CALL DQ$GET$TIME (@PT, @ERR)
```

Exception Codes Returned

```
E$OK
```

Syntax

```
DQ$SWITCH$BUFFER: PROCEDURE (buffer$p,excep$p) WORD EXTERNAL;
   DECLARE buffer$p POINTER,
           excep$p POINTER;
END;
```

Description

This routine is used as a function.

The offset value (*buffer$p*) returned from the invocation of DQGETARGUMENT points to the buffer position of the previous buffer. By keeping track of all previous buffer position pointers you can switch between user-supplied buffers when parsing a command line.

When this routine is first invoked, you cannot switch back to arguments from the input command buffer.

Example

```
DECLARE NEXT_COUNT WORD, ERR WORD;
NEXT_COUNT=DQ$SWITCH$BUFFER (@ARGLIST, @ERR);
```

Exception Codes Returned

```
E$OK
```

APPENDIX A
CLI COMMAND SYNTAX

```
ACCESS pathname [SET {OWNER | WORLD} {access spec} {QUERY}]

ARCHIVE source TO dest [{{INC | EXC} qualifier} {{AND | OR} switch}]
    [{BEFORE | SINCE | ON} {TODAY | mm/dd/yy {hhmm[ss] | hh:mm:[ss] | h:mm}}]

ASSIGN [{logical name} TO pathname]

BACKGROUND pathname [(parameters)]... [{LOG | NOLOG [(pathname) [APPEND]]}]

BATCH pathname

CANCEL {BACKGROUND | [REMOTE] queue {(jobname) | (jobnumber)}} [, . . . ]

CHOWNER filename TO username

CHPASS username

COPY {source filename [,...] | :CI:} TO {destination filename | :DEVICE:}
    [{EXPANDED | E | BRIEF | B | UPDATE | U | QUERY | Q | COPYATTR | C}]

COUNT n
[commands]

[{WHILE | UNTIL} argument {< | = | >} argument]...

[commands]
END
```

```
CREATEDIR pathname

DELETE {pathname [{DIR}]}, [, ...]
       {      [{QUERY}]}

DIR [{pathname} [{EXPANDED      }]]
    [{      /} [{FOR filename   }]]
               [{ONECOLUMN      }]
               [{TO pathname    }]

DISMOUNT devicename

ENDJOB [(argument)]

EXPORT pathname [parameters] TO queue [{LOG [(pathname) [APPEND]]}]
                                      {NOLOG                     }

FILL {ON   }
     {OFF  }
     {SPACE}

                                      [AGRAN number)                  ]
                                      [FNODES (number)                ]
                                      [INTERLEAVE (number)            ]
                                      [NODUP                          ]
FORMAT physical-device volume-name    [NOINIT                         ]
                                      [NOVERIFY                       ]
                                      [OVERRIDE                       ]
                                      [RESERVE (option, ...)          ]
                                      [UPDATE                         ]

ICOPY {READ  ISIS-source-pathname TO iNDX-destination-pathname} [{QUERY }]
      {WRITE iNDX-source-pathname TO destination-pathname     } [{UPDATE}]

IF argument {=} argument
            {<>}
   commands
   [ORIF argument {=} argument [...]]
          {<>}
   [ELSE commands]
END

IMPORT FROM queue [,queue] ... [{TO BACKGROUND}]
                               {TO FOREGROUND}

LNAME [{DEFINE logical name FOR pathname [UPDATE]}]
      [{REMOVE logical name                      }]
      [{PATH                                     }]

LOG {:BB:   } [APPEND]
    {pathname}
```

```
LOGOFF [Exit]

LOGON username {INIT [filename]}
               {NOINIT         }

MOUNT device-name

NETMSG

OPEN pathname

OSCOPY source device TO destination device [OVERRIDE]

PDSCOPY {READ (disk-directory) PDS-source TO iNDX-destination} [{QUERY }]
        {WRITE iNDX-source TO PDS-destination                } [{UPDATE}]

QUEUE [{ADD   }                          ]
      [{DELETE} (queuename , ... )       ]
      [{LIST  }                          ]

READ variable-name [,...]

REGION

RELAB physical-device TO volume-name [OVERRIDE]

RENAME old-pathname TO new-pathname [UPDATE]

REPEAT
      [commands]
      [{WHILE} argument {<} argument [THEN]]
      [{UNTIL}          {>}                 ]
      [commands]
END

{S4FPRT} {UP   iNDX-source-pathname TO destination-pathname                          } [{UPDATE}]
{S2FPRT} {EXIT                                                                       } [{EXIT  }]
         {DOWN [disk-dir] ISIS-source-pathname TO iNDX-destination-pathname          } [{QUERY }]

SDCOPY source {TO destination} ... [OVERRIDE]
              {FORMAT        }     [REPEAT  ]
              {REPEAT        }     [COMPARE ]
              {COMPARE       }     [FORMAT  ]
              {VERIFY        }     [VERIFY  ]
```

```
SEARCH [ { pathname or logical name } ]
       [ { OFF                     } ]

SET variable-name TO ["] value ["]

SPACE /volume-name

STTY [Baudrate (value) [GO]] [ DISPLAY   ] [Terminal] [ Primary   ] [ Remote ] [Config (filename)]
                             [ NODISPLAY ]             [ Secondary ] [ Local  ]

SUBMIT pathname [parameters] ... [ { LOG pathname [APPEND] } ]
                                 [ { NOLOG                 } ]

SYSGEN

SYSTAT [ { QUEUE } (queuename [, ...]) ] [TO pathname] [EXPAND] [ALL]
       [ { MYJOB } ]

TIME

USERDEF { DEFINE username {ID userID | DIR filename} }
        { REMOVE username                            }

USERS /volume-name

VERIFY physical device [ FIX  ] [T0][T1][T2][T3][T4][OVERRIDE]
                       [ FAST ]

VIEW pathname

WAIT [wait message]
```

APPENDIX B
PARAMETERS AND SYSTEM SERVICE ROUTINES

Table B-1 lists the system service routines in alphabetical order for study or reference. Table B-2 alphabetically lists the parameters used by the service routines.

Table B-1. Alphabetical List of Series IV Service Routines

```
DQ$ALLOCATE: PROCEDURE (size, excep$p) SELECTOR EXTERNAL;
    DECLARE              size WORD,
                         excep$p POINTER;
    END;

DQ$ATTACH: PROCEDURE (path$p, excep$p) CONNECTION EXTERNAL;
    DECLARE              path$p POINTER,
                         excep$p POINTER;
    END;

DQ$CHANGE$ACCESS: PROCEDURE (path$p, class, access, excep$p) EXTERNAL;
    DECLARE              path$p POINTER,
                         class BYTE,
                         access BYTE,
                         excep$p POINTER;
    END;

DQ$CHANGE$EXTENSION: PROCEDURE (path$p, extension$p, excep$p) EXTERNAL;
    DECLARE              path$p POINTER,
                         extension$p POINTER,
                         excep$p POINTER;
    END;

DQ$CLOSE: PROCEDURE (conn, excep$p) EXTERNAL;
    DECLARE              conn CONNECTION,
                         excep$p POINTER;
    END;

DQ$CREATE: PROCEDURE (path$p, excep$p) CONNECTION EXTERNAL;
    DECLARE              path$p POINTER,
                         excep$p POINTER:
    END;

DQ$DECODE$EXCEPTION: PROCEDURE (exception$code, message$p, excep$p) EX-
TERNAL;
    DECLARE              exception$code WORD,
                         message$p POINTER,
                         excep$p POINTER;
    END;

DQ$DECODE$TIME: PROCEDURE (dt$p, excep$p) EXTERNAL;
    DECLARE              dt$p POINTER,
                         excep$p POINTER;
    END;

DQ$DELETE: PROCEDURE (path$p, excep$p) EXTERNAL;
    DECLARE              path$p POINTER,
                         excep$p POINTER;
    END;

DQ$DETACH: PROCEDURE (conn, excep$p) EXTERNAL;
    DECLARE              conn CONNECTION,
                         excep$p POINTER;

DQ$EXIT: PROCEDURE (completion$code) EXTERNAL;
    DECLARE              completion$code WORD;
    END;
```

Table B-1. Alphabetical List of Series IV Service Routines (Cont'd)

```
DQ$FILE$INFO:  PROCEDURE (conn, mode, file$info$p, excep$p) EXTERNAL;
      DECLARE              conn CONNECTION,
                           mode BYTE,
                           file$info$p POINTER,
                           excep$p POINTER;
      END;

DQ$FREE:  PROCEDURE (segment, excep$p) EXTERNAL;
      DECLARE              segment SELECTOR,
                           excep$p POINTER;
      END;

DQ$GET$ARGUMENT:  PROCEDURE (argument$p, excep$p) BYTE EXTERNAL;
      DECLARE              argument$p POINTER,
                           excep$p POINTER;
      END;

DQ$GET$CONNECTION$STATUS:  PROCEDURE (conn, info$p, excep$p) EXTERNAL;
      DECLARE              conn CONNECTION,
                           info$p POINTER,
                           excep$p POINTER;
      END;

DQ$GET$EXCEPTION$HANDLER:  PROCEDURE (handler$p, excep$p) EXTERNAL;
      DECLARE              handler$p POINTER,
                           excep$p POINTER;
      END;

DQ$GET$SIZE:  PROCEDURE (segbase, excep$p) WORD EXTERNAL;
      DECLARE              segbase SELECTOR,
                           excep$p POINTER;
      END;

DQ$GET$SYSTEM$ID:  PROCEDURE (id$p, excep$p) EXTERNAL;
      DECLARE              id$p POINTER,
                           excep$p POINTER;
      END;

DQ$GET$TIME:  PROCEDURE (dt$p, excep$p) EXTERNAL;
      DECLARE              dt$p POINTER,
                           excep$p POINTER;
      END;

DQ$OPEN:  PROCEDURE (conn, access, num$buf, excep$p) EXTERNAL;
      DECLARE              conn CONNECTION,
                           access BYTE,
                           num$buf BYTE,
                           excep$p POINTER;

DQ$OVERLAY:  PROCEDURE (name$p, excep$p) EXTERNAL;
      DECLARE              name$p POINTER,
                           excep$p POINTER;
      END;

DQ$READ:  PROCEDURE (conn, buf$p, count, excep$p) WORD EXTERNAL;
      DECLARE              conn CONNECTION,
                           buf$p POINTER,
                           count WORD,
                           excep$p POINTER;
      END;

DQ$RENAME:  PROCEDURE (old$p, new$p, excep$p) EXTERNAL;
      DECLARE              old$p POINTER,
                           new$p POINTER,
                           excep$p POINTER;
      END;
```

Table B-1. Alphabetical List of Series IV Service Routines (Cont'd)

```
DQ$RESERVE$IO$MEMORY:  PROCEDURE (number$files, number$buffers, excep$p) EX-
TERNAL;
      DECLARE                  number$files WORD,
                               number$buffers WORD,
                               excep$p POINTER;
      END;
```

```
DQ$SEEK:  PROCEDURE (conn, mode, offset, excep$p) EXTERNAL;
      DECLARE                  conn CONNECTION,
                               mode BYTE,
                               offset DWORD,
                               excep$p POINTER;
      END;
```

```
DQ$SPECIAL:  PROCEDURE (type, parameter$p, excep$p) EXTERNAL;
      DECLARE                  type BYTE,
                               parameter$p POINTER,
                               excep$p POINTER;
      END;
```

```
DQ$SWITCH$BUFFER:  PROCEDURE (buffer$p, excep$p) WORD EXTERNAL;
      DECLARE                  buffer$p POINTER,
                               excep$p POINTER;
      END;
```

```
DQ$TRAP$CC:  PROCEDURE (handler$p, excep$p) EXTERNAL;
      DECLARE                  handler$p POINTER,
                               excep$p POINTER;
      END;
```

```
DQ$TRAP$EXCEPTION:  PROCEDURE (handler$p, excep$p) EXTERNAL;
      DECLARE                  handler$p POINTER,
                               excep$p POINTER;
      END;
```

```
DQ$TRUNCATE:  PROCEDURE (conn, excep$p) EXTERNAL;
      DECLARE                  conn WORD,
                               excep$p POINTER;
      END;
```

```
DQ$WRITE:  PROCEDURE (conn, buf$p, count, excep$p) EXTERNAL;
      DECLARE                  conn CONNECTION,
                               buf$p POINTER,
                               count WORD,
                               excep$p POINTER;
      END;
```

Table B-2. Alphabetical Parameter Definitions

Parameter Name	Routines Using This Parameter and Brief Definition of Parameter
access	OPEN, SEEK, CHANGE$ACCESS A number telling how you plan to use the file, e.g., read, write, or both
arg$p	GET$ARGUMENT Pointer to the 81-byte area you have declared to receive the argument from a line-edited input source
buf$p	READ, WRITE, SWITCH$BUFFER Pointer to the area you have declared for reading from (or writing to) a file; for read or write, it should be at least COUNT bytes long or unintended results will occur
class	CHANGE$ACCESS Specifies the class of users whose access rights are to be changed (0=owner, 1=world, 2-255=reserved)
compl$cod	EXIT A word telling the success of program completion and termination
conn	DETACH, GET$CONNECTION$STATUS, OPEN, SEEK, READ, WRITE, TRUNCATE, CLOSE, SPECIAL, FILE$INFO Connection to a file or device, established earlier via attach or create
count	READ, WRITE The number of bytes you want read or written
delim	GET$ARGUMENT A byte filled by the routine with the delimiter ending the current argument
dt$p	GET$TIME, DECODE$TIME Pointer to the structure you set up for date and time
excep$p	ALL ROUTINES BUT EXIT Pointer to the word you have declared to receive the exception value
exception$cod	DECODE$EXCEPTION A word into which you have placed an exception code
exception$p	DECODE$EXCEPTION Pointer to the 81-byte area you have declared for receiving the formatted message corresponding to excep$cod
extension$p	CHANGE$EXTENSION Pointer to the extension as you wish it to be
file$info$p	FILE$INFO Pointer to table used for output
file$ptr	Same as offset
handler$p	TRAP$EXCEPTION, GET$EXCEPTION$HANDLER, TRAP$CC Pointer to the entry-point of your routine to handle current exceptions (or Control C)
id$p	GET$SYSTEM$ID Pointer to the location where you want the system-name or sign-on put
info$p	GET$CONNECTION$STATUS Pointer to the pathname whose connection you need to know
mode	SEEK, FILE$INFO Value representing direction and type of seek operation
name$p	OVERLAY Pointer to the pathname of the overlay to be loaded next

Table B-2. Alphabetical Parameter Definitions (Cont'd)

Parameter Name	Routines Using This Parameter and Brief Definition of Parameter
new$p	RENAME Pointer to the pathname as you wish to have it
num$buf	OPEN Number of buffers to be used for I/O to file being opened
number$files	RESERVEIOMEMORY Maximum number of files that will be attached at any one time.
number$buffers	RESERVEIOMEMORY Maximum number of buffers that will be requested.
offset	SEEK A four-byte unsigned integer representing the number of bytes to move to the file pointer
old$p	RENAME Pointer to the pathname as it is now
path$p	CREATE, DELETE, ATTACH, CHANGE$EXTENSION, CHANGE$ACCESS Pointer to the pathname you wish to use
segbase	FREE, GET$SIZE The word containing the base of a block of bytes
size	ALLOCATE The number of bytes you want to use
type	SPECIAL A value determining whether console input should be line-edited or transparent

APPENDIX C
ERROR MESSAGES & EXCEPTION CODES

C.1 iNDX-Series IV Error Messages

A warning does not cause a command to be aborted.

```
BACKGROUND JOB ACTIVE
```

Background job is active at the time of user logoff. The logoff is delayed until the background job finishes.

Following is a set of error messages; when these occur, the current command is aborted.

```
INVALID USER
```

The user name is not recognized by the system.

```
INVALID PASSWORD
```

The password given by the user is incorrect.

```
SYNTAX ERROR
```

Illegal syntax was detected.

```
FILE NOT FOUND
```

File specified in the command cannot be found.

```
INVALID CONDITION
```

Condition specified in the IF, WHILE, UNTIL commands is not valid.

```
ILLEGAL COMMAND
```

May be one of a number of conditions:

ORIF command is not preceded by an IF command in the command file. ELSE command is not preceded by an IF command in the command file. UNTIL command is not preceded by a REPEAT or COUNT command. WHILE command is not preceded by a REPEAT or COUNT command. END command without a matching IF, REPEAT, COUNT command. Attempt to read without an OPEN command.

```
INVALID NUMBER
```

The number specified in the COUNT command is not valid.

```
THE BACKGROUND IS ALREADY ACTIVE
```

The BACKGROUND command has been executed while a job is running in the background.

```
QUEUE DOES NOT EXIST
```

The queue name specified in the SYSTAT command does not exist.

C.2 Series IV Exception Codes

C.2.1 General Exceptions

```
0000H 'SUCCESSFUL COMPLETION'
0001H 'CRITICAL REGION LOCKED TOO LONG'
0002H 'INSUFFICIENT MEMORY FOR REQUESTED OPERATION'
0009H 'FATAL EXCEPTION - A PROCESSOR HAS NOT REPONDED TO
      MIP'
0020H 'FILE ALREADY EXISTS'
0021H 'FILE DOES NOT EXIST'
0023H 'UNSUPPORTED OPERATION OR DISK FORMAT'
0026H 'INSUFFICIENT ACCESS RIGHTS'
0028H 'SHARED STATE OF FILE PROHIBITS OPEN'
0029H 'INSUFFICIENT SPACE ON DIRECT-ACCESS DEVICE'
0030H 'DEVICE BEING ACCESSED IS NOT READY'
0031H 'ILLEGAL DEVICE NUMBER'
0032H 'ALL FNODES ARE IN USE'
0033H 'NO MORE AVAILABLE SPACE ON DISK'
0034H 'LOCAL FILE SYSTEM I/O ERROR'
0035H 'NON-EMPTY DIRECTORY CAN NOT BE DELETED'
0036H 'DELETE IS PENDING ON THE REQUESTED FILE'
0037H 'REQUESTED DEVICE IS ALREADY MOUNTED'
0038H 'REQUESTED DEVICE IS NOT MOUNTED'
0039H 'BAD LOCAL FILE SYSTEM COMMAND'
003AH 'BAD REGION'
003BH 'UNABLE TO SATISFY REQUEST TO CREATE QUEUE'
003CH 'ATTAMPT TO ENQUEUE OR DEQUEUE NONEXISTENT QUEUE'
003DH 'QUEUE BUSY'
003EH 'ATTEMPT TO SEND OR RECEIVE UNIT FROM BAD
      SEMAPHORE'
003FH 'BUSY SEMAPHORE'
0040H 'BAD PRIORIY'
0041H 'NO FREE REGIONS'
0042H 'CANNOT ALLOCATE'
0043H 'BOUNDS'
0044H 'TOKEN DOES NOT POINT TO A VALID SEGMENT'
0045H 'DIRECTORY OPERATION ATTEMPTED ON DATA FILE'
0046H 'FILE REQUESTED IS ALREADY ATTACHED'
0047H 'FILE SYSTEM ATTACH TABLE IS FULL'
0048H 'UNABLE TO SATISFY REQUEST TO CREATE SEMAPHORE'
0049H 'NOT FREE'
004AH 'FILE NOT DELETABLE (SYSTEM FILE OR CONTAINS BAD
      BLOCKS)'
004BH 'FILE CAN NOT BE EXTENDED'
004FH 'DATA OPERATION ATTEMPTED ON DIRECTORY FILE'
0050H 'BITMAP ENTRY NOT TRACED TO DEVICE'
0051H 'BLOCK DOES NOT CHECKSUM'
0052H 'DIRECTORY CONTAINS INVALID FILE REFERENCE'
0053H 'ILLEGAL BLOCK NUMBER FOR FILE ON DEVICE'
0054H 'ILLEGAL FNODE NUMBER'
0055H 'ATTEMPT TO ALLOCATE PREVIOUSLY ALLOCATED FNODE'
0056H 'ATTEMPT TO WRITE A SYSTEM FILE'
```

```
0057H 'BITMAP NOT FLUSHED TO DISK'
0058H 'ATTEMPT TO WRITE BLOCK ZERO OF DEVICE'
0059H 'DEVICE LOST DUPLICATE FLIE PROTECTION'
005AH 'DEVICE NOT FORMATTED PROPERLY'
005BH 'BAD SIZE'
0081H 'STRING TOO LONG'
0100H 'INVALID PATHNAME-COMPONET EXCEEDS 14 CHARACTERS'
0105H 'INVALID ATTACH NUMBER PASSED AS A PARAMETER'
010BH 'DEVICE OR SYSTEM DOES NOT SUPPORT DEVICE OR
       SYSTEM'
010CH 'INVALID PATHNAME - COMPONENT EXCEEDS 14
       CHARACTERS'
```

C.2.2 Human Interface/UDI Layer Exceptions

```
2001H 'INVALID LOAD FILE-INVALID REGISTER INITIALIZATION
       RECORD'
2002H 'CALL NOT VALID IN THIS CONTEXT'
2005H 'OPERATION ATTAMPTED ON NON-EXISTANT FILE
       CONNECTION'
2006H 'INVALID PARAMETER'
2007H 'INVALID PATHNAME'
2008H 'TOKEN DOES NOT POINT TO A VALID SEGMENT'
2009H 'WRONG JOB TYPE'
200AH 'SYNTAX ERROR'
200BH 'END OF FILE'
200AH 'INVALID TIME'
200CH 'ILL RECORD'
200DH 'BAD FILE'
200EH 'BAD FIXUP'
200FH 'NO OVERLAY'
2010H 'UNRESOLVED'
2011H 'INVALID LOAD FILE-INVALID REGISTER INITIALIZATION
       RECORD'
2012H 'TOO MANY DEFS'
201AH 'INVALID TIME'
201BH 'INVALID DATE'
201CH 'PARAMTER TOO LONG'
201DH 'TOO MANY PARAMETERS'
201EH 'FILE ALREADY EXISTS'
201FH 'LIMIT EXCEEDED'
2020H 'TOO MANY QUEUES'
2021H 'TOO MANY SERVERS'
2022H 'QUEUE FULL'
2023H 'NO SERVERS'
2024H 'SERVER EXISTS'
2025H 'NO QUEUE'
2026H 'NO JOB'
2027H 'JOB USER'
2028H 'SYSTEM FILES'
2029H 'BAD JOB NUMBER'
2030H 'COMMAND VARIABLE ALREADY EXISTS'
2031H 'COMMAND VARIABLE UNDEFINED'
2032H 'BAD COMMAND VARIABLE NAME'
2033H 'BAD COMMAND VARIABLE TOKEN'
2034H 'COMMAND VARIABLE TABLE FULL'
2035H 'CLI STACK NOT INITIALIZED'
```

```
2036H 'CLI STACK OVERFLOW'
2037H 'COMMAND VARIABLE OVERFLOW'
2038H 'ENV_BUF_NEXIST'
2039H 'ENV_BUF_OVERFLOW'
2040H 'VERSION'
2041H 'COMM'
2050H 'E_COMM'
2051H 'E_SESSION'
2052H 'E_TIMEOUT'
2053H 'E_LOCAL_ABORT'
2054H 'E_REMOTE_ABORT
2060H 'INVALID SYSTEM CALL AT '
2061H 'DIRECTORY FORMAT ERROR DURING WILDCARD PROCESSING'
2101H 'OPERATION CONFLICT (FILE OPENED FOR WRITE)'
2102H 'OPERATION CONFLICT (FILE OPENED FOR READ)'
2200H 'TOO MANY OVERLAYS'
2201H 'OVERLAY INITIALIZATION HAS NOT BEEN COMPLETED'
2202H 'INVALID OVERLAY AREA REQUESTED'
2203H 'OVERLAY INITIALIZATION HAS NOT BEEN COMPLETED'
```

C.2.3 Loader Exceptions

```
2300H 'UNEXPECTED END OF FILE DURING LOAD'
2301H 'INVALID LOAD FILE'
2302H 'ILLEGAL LOAD RECORD'
2303H 'LOAD FILE CONTAINS ABSOLUTE LOAD ADDRESS'
2304H 'INVALID OVERLAY NAME REQUESTED'
2305H 'SPECIFIED OVERLAY NOT FOUND'
2306H 'INVALID LOAD FILE: DATA RECORD EXPECTED'
2307H 'INVALID LOAD FILE: NOT RELOCATABLE'
2308H 'LOADER LIMIT EXCEEDED: TOO MANY SEGMENTS'
2309H 'LOADER LIMIT EXCEEDED: TOO MANY GROUPS'
230AH 'LOADER LIMIT EXCEEDED: TOO MANY OVERLAYS'
230BH 'NON-RELOCATABLE SEGMENT ENCOUNTERED'
230CH 'INVALID LOAD FILE: BAD SEGMENT DEFINITION RECORD'
230DH 'NON-RELOCATABLE GROUP ENCOUNTERED'
230EH 'INVALID LOAD FILE: BAD SEGMENT IN GROUP DEF
       RECORD'
230FH 'INVALID LOAD FILE: BAD GROUP DEFINITION RECORD'
2310H 'INVALID LOAD FILE: OVERLAY DEFINITION RECORD
       EXPECTED'
2311H 'INVALID LOAD FILE: ILLEGAL LOAD TIME FIXUP'
2321H 'INVALID LOAD FILE: INVALID REGISTER INITIALIZATION
       RECORD'
2322H 'LOAD FILE CONTAINS UNRESOLVED EXTERNAL REFERENCES'
2323H 'LOAD FILE HAS BAD REGISTER INIT, UNRESOLVED
       EXTERNALS'
2324H 'INITIAL STACK FOR LOADED FILE IS TOO SMALL FOR
       STARTUP'
2325H 'LOAD FILE HAS BAD REGISTER INIT AND STACK IS TOO
       SMALL'
2326H 'LOAD FILE HAS UNRESOLVED EXTERNALS AND STACK IS
       TOO SMALL'
2327H 'LOAD FILE HAS BAD REGISTER INIT, UNRESOLVEDS,
       STACK TOO SMALL'
```

C.2.4 Job Control Exceptions

```
2400H 'QUEUE LIMIT EXCEEDED'
2401H 'SERVER LIMIT EXCEEDED'
2402H 'QUEUE FULL'
2403H 'SERVER DOES NOT EXIST'
2405H 'QUEUE DOES NOT EXIST
2406H 'JOB(S) NOT FOUND'
2407H 'USER'
2408H 'SYSTEM FILES'
2409H 'JOB NUMBER IS INCONSISTENT WITH SPECIFIED QUEUE'
240AH 'DJC ERROR: E_BUFFER_TOO_SMALL'
240BH 'DJC ERROR: E_INVALID_DJC_REQUEST'
240CH 'DJC ERROR: E_TIME'
240EH 'DJC ERROR: E_BREAK_OCCURRED
2410H 'DJC: END IMPORT'
2411H 'DJC ERROR: E_DJC_BAD_ACK'
2412H 'DJC ERROR: E_DJC_COMM_SYNCH'
2413H 'DJC ERROR: E_DJC_JOB_NOT_EXECUTING'
2414H 'DJC CONSISTENCY ERROR: QUEUE TABLE FILE INVALID'
2415H 'DJC: PROTOCOL VERSION MISMATCH'
2416H 'DJC: SUBSYSTEM FAILURE'
2417H 'DJC: SPECIFIED QUEUE ALREADY EXISTS'
2418H 'DJC: CANT DELETE,SPECIFIED QUEUE HAS JOBS
      ENQUEUED'
2419H 'CANT CREATE DJC QUEUE, DIRECTORY EXISTS OF SAME
      NAME'
241AH 'DJC NOT AVAILABLE, NRM IN STANDALONE MODE'
2602H 'DJC TCL ERROR: INVALID REQUEST'
2604H 'DJC TCL ERROR: NO RESOURCE AVAILABLE'
2608H 'DJC ERROR: E_BUFF_TOO_SHORT
260AH 'DJC TCL ERROR: TRIED TO SEND AFTER CONNECTION WAS
      CLOSED'
260CH 'DJC TCL ERROR: LOCAL CLIENT OF TCL ISSUED ABORT'
260EH 'DJC TCL ERROR: REMOTE CLIENT OF TCL ISSUED ABORT'
2610H 'DJC TCL ERROR: LOCAL ABORT TIMEOUT'
2612H 'DJC: CLOSE COMPLETE (NOT AN ERROR)'
2614H 'DJC TCL ERROR: INVALID REQUEST BLOCK POINTER'
2616H 'DJC PROTOCOL ERROR:PRB TO CLOSED CONNECTIONS
```

C.2.5 COMM Loader Exceptions

```
2A00H 'COMM LOADER: COMM BOARD NOT RESPONDING.'
2A01H 'COMM LOADER: COMM BOARD FAILURE.'
2A02H 'COMM LOADER: TCL SOFTWARE FILE TOO LARGE.'
2A03H 'COMM LOADER: NRM COMM ID DOES NOT MATCH WITH
      SYSGEN TABLES.'
2A04H 'COMM LOADER: UNABLE TO LOAD SESSION DATA BASE.'
2A05H 'COMM LOADER: UNABLE TO LOAD TCL SOFTWARE FILE.'
2A06H 'COMM LOADER: COMM BOARD LOAD/GO FAILURE.'
2A07H 'COMM LOADER: ALLOCATION OF BUFFERS FAILED.'
```

C.2.6 Remote Boot Server Exceptions

```
2A20H 'BOOT SERVER INITIALIZATION ABORTED.'
2A21H 'BOOT SERVER: ALLOCATION OF BUFFERS FAILED.'
```

```
2A22H 'BOOT SERVER: INCONSISTENT HEADER ON TCL SOFTWARE
      FILE.'
2A23H 'BOOT SERVER: EDL ADD MULTICAST ID REQUEST FAILED.'
2A24H 'BOOT SERVER: SYSTEM CONFIGURATION TABLE TOO
      LARGE.'
2A25H 'BOOT SERVER: I/O ERROR ON FILE '
2A26H 'BOOT SERVER: DEQUEUE ERROR, EXPECTED MESSAGE NOT
      RECEIVED.'
2A2AH 'BOOT SERVER: BOOT REQUEST FROM UNKNOWN WORKSTATION
      IGNORED.'
2A2BH 'BOOT SERVER: UNABLE TO COMPUTE SESSION TABLES.'
2A2CH 'BOOT SERVER: INVALID REQUEST RECEIVED: BAD CLASS
      CODE.'
2A2DH 'BOOT SERVER: INVALID REQUEST RECEIVED: BAD REQUEST
      TYPE.'
2A2EH 'BOOT SERVER: DATA TRANSMIT REQUEST FAILED.'
2A2FH 'BOOT SERVER: MIP SEND ERROR: MIP EXCEPTION CODE ='
```

C.2.7 I/O Device Specific Exceptions

For errors in the range 3000H to 380FH, the low order nibble (half byte) is a number which indicates the device on which the exception occurred. For different values of the device number a different message table is used. The error code is of the form 3*xxn*H. If *n* = 0 or 1 then it is a flexible disk error. If *n* = 2, 3, 4 or 5, then the error occurred on a 740 hard disk. If *n* = 6, 7, 8, 9, A, B, C, D, or E then the error occurred on a Winchester hard disk. If *n* = F then the error occured on the tape drive.

C.2.8 5¼-Inch Flexible Disk Device Error Messages

If *n* = 0, '*dev*' = 'FL0' (flexible disk drive zero)

```
301nH 'dev ERR, UNEXPECTED DELETED DATA ADDRESS MARK ON
      DISK'
302nH 'dev ERR, HARDWARE OR DISK FAIL, BAD DATA FIELD
      CRC'
304nH 'dev ERR, HARDWARE FAIL, ID CYL FIELD <> SEEK CYL'
308nH 'dev ERR, OS ERR, ATTEMPTING TO ACCESS NON-EXISTENT
      SECTOR'
30AnH 'dev ERR, HARDWARE OR DISK FAIL, BAD ID FIELD CRC'
30BnH 'dev ERR, PROTOCOL ERROR DUE TO OS OR CONTROLLER
      FAILURE'
30CnH 'dev ERR, CYLINDER ADDRESS OUT OF BOUNDS DUE TO OS
      OR DRIVE FAILURE'
30EnH 'dev ERR, HARDWARE OR DISK FAIL, NO ID ADDRESS
      MARK'
30FnH 'dev ERR, HARDWARE OR DISK FAIL, NO DATA ADDRESS
      MARK'
310nH 'dev ERR, MULTIBUS CONTENTION, ACCESS OVER/UNDER
      RUN'
311nH 'dev ERR, CONTROLLER RAM TEST FAILURE'
312nH 'dev ERR, CONTROLLER PROM TEST FAILURE'
313nH 'dev ERR, OS OR CONTROLLER FAILURE, SEEK IN
      PROGRESS'
314nH 'dev ERR, OS OR CONTROLLER FAILURE, TRACK TYPE
      DISALLOWS OPERATION'
```

```
315nH 'dev ERR, ACCESS BEYOND END OF MEDIA DUE TO OS OR
      CONTROLLER FAILURE'
316nH 'dev ERR, DISK BAD, INCORRECT SECTOR SIZE FOUND'
317nH 'dev ERR, CONTROLLER DIAGNOSTIC FAULT'
318nH 'dev ERR, NO INDEX SIGNAL DUE TO CONTROLLER OR
      DRIVE FAILURE'
319nH 'dev ERR, INVALID FUNCTION CODE PASSED DUE TO OS OR
      CONTROLLER FAILURE'
320nH 'dev ERR, ATTEMPT TO WRITE WITH HARDWARE WRITE
      PROTECT SET'
340nH 'dev ERR, DRIVE IS INDICATING A HARDWARE FAILURE'
370nH 'dev ERR, DISK BAD, CAN'T FIND SECTOR'
371nH 'dev ERR, UNEXPECTED BAD TRACK FLAG ON DISK'
372nH 'dev ERR, HARDWARE FAIL, SEEK DID NOT GET TO
      EXPECTED TRACK'
378nH 'dev ERR, UNKNOWN ERROR CODE FROM CONTROLLER'
380nH 'dev ERR, DISK NOT INSERTED AND SPINNING'
```

C.2.9 Device Specific Exceptions for Model 740 Hard Disks

If *n* = 2, then '*dev*' = 'HD0' (fixed platter of first 5440)
If *n* = 3, then '*dev*' = 'HD1' (removable platter of first 5440)
If *n* = 4, then '*dev*' = 'HD2' (fixed platter of second 5440)
If *n* = 5, then '*dev*' = 'HD3' (removable platter of first 5440)

```
301nH 'dev ERR, HARDWARE OR DISK FAIL, BAD ID FIELD
      CONTENTS'
302nH 'dev ERR, HARDWARE OR DISK FAIL, BAD DATA FIELD
      CRC'
304nH 'dev ERR, HARDWARE OR DRIVE FAIL, SEEK ERROR'
308nH 'dev ERR, OS ERROR, ATTEMPTING TO ACCESS NON-
      EXISTENT SECTOR'
30AnH 'dev ERR, HARDWARE OR DISK FAIL, BAD ID FIELD CRC
      CODE'
30BnH 'dev ERR, OS OR CONTROLLER FAIL, BAD PROTOCOL'
30CnH 'dev ERR, OS OR DRIVE FAIL, CYLINDER ADDRESS OUT OF
      BOUNDS'
30EnH 'dev ERR, DISK BAD, CAN'T FIND SECTOR'
30FnH 'dev ERR, CONTROLLER OR DISK FAIL, NO BEGIN OF DATA
      FIELD MARK'
310nH 'dev ERR, MULTIBUS CONTENTION, ACCESS OVER/UNDER
      RUN'
320nH 'dev ERR, ATTEMPT TO WRITE WITH HARDWARE WRITE
      PROTECT SET'
340nH 'dev ERR, DRIVE IS INDICATING A HARDWARE FAULT'
380nH 'dev ERR, DRIVE NOT READY'
```

C.2.10 Device Specific Exceptions for Winchester Disks

If *n* = 6, then '*dev*' = WD0 (first 35 megabyte device)
If *n* = 7, then '*dev*' = WD1 (second 35 megabyte device)
If *n* = 8, then '*dev*' = WD2 (third 35 megabyte device)
If *n* = 9, then '*dev*' = WD3 (fourth 35 megabyte device)
If *n* = A, then '*dev*' = WF0 (first 84 megabyte device)

If *n* = B, then '*dev*' = WF1 (second 84 megabyte device)
If *n* = C, then '*dev*' = WF2 (third 84 megabyte device)
If *n* = D, then '*dev*' = WF3 (fourth 84 megabyte device)

```
302nH 'dev ERR, HARDWARE OR DISK FAIL, BAD DATA FIELD
      ECC'
304nH 'dev ERR, HARDWARE OR DRIVE FAIL, SEEK ERROR'
30AnH 'dev ERR, HARDWARE OR DISK FAIL, BAD ID FIELD ECC'
30EnH 'dev ERR, DISK BAD, CAN'T FIND SECTOR'
311nH 'dev ERR, CONTROLLER RAM TEST FAIL'
312nH 'dev ERR, CONTROLLER PROM TEST FAIL'
313nH 'dev ERR, OS OR CONTROLLER FAIL, SEEK IN PROGRESS'
314nH 'dev ERR, OS OR CONTROLLER FAIL, TRACK TYPE
      DISALLOWS OPERATION'
315nH 'dev ERR, OS OR CONTROLLER FAIL, ACCESS BEYOND END
      OF MEDIA'
316nH 'dev ERR, DISK BAD, INCORRECT SECTOR SIZE FOUND'
317nH 'dev ERR, CONTROLLER DIAGNOSTIC FAULT'
318nH 'dev ERR, CONTROLLER OR DRIVE FAIL, NO INDEX
      SIGNAL'
319nH 'dev ERR, OS OR CONTROLLER FAIL, INVALID FUNCTION
      CODE PASSED'
31AnH 'dev ERR, OS OR CONTROLLER FAIL, DISK ADDRESS OUT
      OF BOUNDS'
320nH 'dev ERR, ATTEMPT TO WRITE WITH HARDWARE WRITE
      PROTECT SET'
340nH 'dev ERR, DRIVE IS INDICATING A HARDWARE FAULT'
372nH 'dev ERR, DRIVE FAIL, ID CYL ADDR DOESN'T MATCH
      SEEK ADDR'
378nH 'dev ERR, UNKNOWN ERROR CODE FROM CONTROLLER'
380nH 'dev ERR, DRIVE NOT READY'
```

C.2.11 I/O Exceptions Which Are Not Device Specific

```
3C00H 'DEVICE ERROR, OS FAILURE, BAD COMMAND CODE PASSED
      TO DRIVER'
3C10H 'DEVICE ERROR, OS FAILURE, BLOCKS ACCESSED OVERFLOW
      DEVICE'
3C20H 'DEVICE ERROR, MINI DRIVER FAILURE, INVALID 8272
      COMMAND'
3C30H 'DEVICE ERROR, OS MINI DRIVER FAILURE, Q IS
      DAMAGED'
3C40H 'DEVICE ERROR, OS MINI DRIVER FAILURE, 8089 NOT
      RESPONDING'
3C50H 'DEVICE ERROR, ACCESS TO DEVICE W/MISSING
      CONTROLLER HARDWARE'
3CF0H 'DEVICE ERROR, OS MINI DRIVER FAILURE, BAD 8272
      PROTOCOL'
3D00H 'BAD CRT ADDRESS'
3D01H 'B_CRT_CC'
3D02H 'B_CRT_50HZ'
3D80H 'KEYBOARD READ ABORTED'
3DC0H 'DEVICE ERROR, OS HD5440 DRIVER FAIL, BAD
      CONTROLLER PROTOCOL'
3DC1H 'DEVICE ERR, OS HD5440 DRIVER FAILURE, Q IS
      DAMAGED'
3DC2H 'DEVICE ERR, HD5440 CONTROLLER CONFIGURATION
      SWITCHES WRONG'
```

```
3DC3H 'DEVICE ERR, UNKNOWN INTERRUPT SOURCE ON MULTIBUS
      LEVEL 2'
3DC4H 'DEVICE ERR, TRANSFER OVERFLOWS 8086 SEGMENT'
3DE0H 'ILLEGAL BX_SBIOS ACCESS BY MORE THAN ONE JOB'
3DF0H 'DEVICE ERR, OS FAILURE, LIST DAMAGED'
3E01H 'DEVICE ERR, PRINTER TIMEOUT, NO RESPONSE IN 3 SEC'
3E02H 'DEVICE ERR, PRINTER HARDWARE FAIL, FAULT BEING
      SIGNALLED'
3E04H 'DEVICE ERR, PRINTER NOT READY'
3E10H 'DEVICE ERR, OS PRINTER DRIVER FAIL, PROTOCOL
      ERROR'
3F00H 'DEVICE ERR, OS WINC FAIL, DATA STRUCTURE DAMAGED'
3F01H 'DEVICE ERR, OS WINC DRIVER FAIL, BAD CONTROLLER
      PROTOCOL'
3F02H 'DEVICE ERR, UNSUPPORTED WINC CONTROLLER
      ENCOUNTERED'
```

C.2.12 Distributed File System Exceptions

```
4000H 'OPEN ATTEMPTED ON CONNECTION WHICH IS ALREADY
      OPEN'
4001H 'CONNECTION NOT OPEN'
4002H 'INCORRECT FILE TYPE'
4003H 'PARAMETER HAS INVALID SYNTAX'
4004H 'DEVICE BEING ACCESSED IS NOT READY'
4005H 'DEVICE I/O ERROR'
4006H 'COMMUNICATIONS SYSTEM ERROR'
4007H 'NETWORK FAILURE - PUBLIC FILES NOT ACCESSIBLE'
4008H 'DELETE IS PENDING ON REQUEST FILE'
4009H 'OPEN MODE OF CONNECTION PROHIBITS OPERATION'
4010H 'FILE EXISTS AND HAS CONNECTION ESTABLISHED ON IT'
4011H 'NONEMPTY DIRECTORY FILE'
4012H 'MAXIMUM NUMBER OF PERMITTED OBJECTS EXCEEDED'
4013H 'CONNECTION DOES NOT EXIST'
4014H 'OPERATION NOT SUPPORTED FOR THE CONSOLE DEVICE'
4015H 'LOGICAL NAME ALREADY EXISTS'
4016H 'ILLEGAL DEVICE ID VALUE'
4017H 'DISMOUNT ATTEMPTED ON SYSTEM DEVICE'
4018H 'LOGICAL NAME DOES NOT EXIST'
4019H 'INVALID PATHNAME SYNTAX'
401AH 'FILE IS WRONG TYPE, DIRECTORY FILE EXPECTED'
401BH 'FILE DOES NOT EXIST'
401CH 'FILE EXISTS AND HAS CONNECTION ESTABLISHED ON IT'
401DH 'EXTERNAL EVENT CAUSED DETACH OF CONNECTION'
4020H 'VOLUME WITH SAME VOLUME ROOT DIRECTORY NAME
      ALREADY MOUNTED'
4021H 'NETWORK LOGON ENABLED'
4022H 'NETWORK LOGON ENABLED'
4023H 'UNKNOWN HOST'
4026H 'NETWORK PROTOCOL VIOLATION: GENERIC CONSISTENCY
      ERROR'
4FFFH 'FATAL INTERNAL ERROR* GENERIC CONSISTENCY ERROR'
8004H 'ILLEGAL PARAMETER VALUE'
CCB4H 'M TMP SYNTAX'
CCB5H 'M FULLY QUALIFIED'
CCB6H 'M DISALLOWED QUERY'
CCB7H 'M SINGLE COMP'
```

```
CCB8H 'M DIR EXIST'
CCB9H 'M CREATE CONFLICT'
CCBAH 'M DELETE ACCESS'
CCBBH 'M DIR REQUIRED'
CCBCH 'M DISPLAY ACCESS'
CCBDH 'M ADD ACCESS'
CCBEH 'M ILLEGAL WILDCARD'
CCBFH 'M DISALLOWED WILDCARD'
CCC0H 'M PARENT DOES NOT EXIST'
CCC1H 'M DATA DIR OPTIONS'
CCC2H 'M DIR OPTIONS'
CCC3H 'M DATA OPTION'
CCC4H 'M DIR CREATE'
CCC5H 'M PASSWORD'
CCC6H 'M ILLEGAL VALUE'
CCC7H 'M USER ID'
CCC8H 'M USER NAME'
CCC9H 'M NOT COMLETE'
CCCAH 'M CANNOT CREATE'
CCCBH 'M READ ACCESS'
CCCCH 'M COMMAND SYNTAX'
CCCDH 'M SYSTEM ERROR'
CCCEH 'M PATH SYNTAX'
CCCFH 'M DISALLOWED REMOTE'
DFFBH 'E ISIS DATA PREFIX'
DFFCH 'E ISIS FILE TOO LONG'
DFFDH 'E ISIS WP CONSISTENCY'
DFFEH 'VOLUME DOES NOT EXIST'
DFFFH 'USER OPERATIONS NOT SUPPORTED ON FLIPPIES'
```

C.2.13 Syntax Guide Exceptions

```
E000H 'SYNTAX GUIDE LIMIT EXCEEDED: ARGUMENT BUFFER
      OVERFLOW'
E001H 'SYNTAX GUIDE LIMIT EXCEEDED: PARSE STACK OVERFLOW'
E002H 'SYNTAX GUIDE LIMIT EXCEEDED: REMOVE STACK
      OVERFLOW'
E003H 'SYNTAX GUIDE LIMIT EXCEEDED: EDIT BUFFER OVERFLOW'
E004H 'SYNTAX TABLE INCOMPATABLE WITH SYNTAX GUIDE'
E005H 'SYNTAX GUIDE CONSISTENCY CHECK: INTERNAL ERROR'
```

C.2.14 ISIS-IV Exceptions

```
E100H 'INTERNAL ISIS-IV ERROR, BAD MIP CONNECT'
E101H 'INTERNAL ISIS-IV ERROR, UNKNOWN MONITOR REQUEST'
E103H 'INTERNAL ISIS-IV ERROR, UNKNOWN MESSAGE
      DESTINATION'
E107H 'INTERNAL ISIS-IV ERROR, NO RECEIVE BUFFERS'
E109H 'INTERNAL ISIS-IV ERROR, CLOSED DESTINATION FILE'
E10BH 'IEU BOARD NOT RESPONDING (DAMAGED SOFTWARE
      POSSIBLE)'
E10DH 'INTERNAL ISIS-IV ERROR, BAD MESSAGE ADDRESS'
E111H 'IEU HARDWARE NOT RESPONDING'
E112H 'INTERNAL ISIS-IV ERROR, UNKNOWN MESSAGE TYPE'
E113H 'INTERNAL ISIS-IV ERROR, UNKNOWN ISIS REQUEST TYPE'
E114H 'INTERNAL ISIS-IV ERROR, BAD LNAME'
```

```
E115H 'CANNOT LOAD ISIS'
E116H 'READ FROM KEYBOARD IN BACKGROUND OR IMPORT MODE
      ATTEMPTED'
E117H 'ERROR OCCURED WHILE WRITING TO :CO: OR WHILE :CO:
      WAS CLOSED (ISIS)'
E118H 'BAD REMOTE INTERFACE COMMAND'
E119H 'UNSUPPORTED MONITOR COMMAND'
E11AH 'BAD PARAMETER TO MONITOR KIC ROUTINE'
E11BH 'BAD REMOTE INTERFACE COMMAND'
E11CH 'REMOTE INTERFACE PROTOCOL VIOLATION'
E11DH 'IEU MEMORY NOT AVAILABLE TO ISIS-IV'
E11EH 'I/O ERROR ON CONSOLE FILE'
E200H 'INSUFFICIENT MEMORY FOR VIEW'
```

C.2.15 Distributed File System Command Exceptions

```
F000H 'FATAL FILE SYSTEM CONSISTENCY ERROR'
F001H 'FATAL ERROR IN S-IV LOCAL LOGON; JOB ID ALREADY
      REGISTERED IN LOCAL LOGON TABLE'
F002H 'FATAL ERROR IN S-IV LOCAL LOGON; JOB ID LIMIT
      EXCEEDED IN LOCAL LOGON TABLE'
F003H 'FATAL ERROR IN S-IV LOCAL LOGOFF; ATTEMPTED
      DELETION OF NONEXISTENT JOB ID IN LOCAL LOGON
      TABLE'
F004H 'FATAL ERROR IN S-IV JOB MANAGEMENT; ATTEMPT TO
      RETRIEVE NRM JOB ID FOR A WS ID NOT REGISTERED IN
      LOCAL LOGON TABLE'
F005H 'UNKNOWN VIRTUAL CIRCUIT ID'
F006H 'ATTEMPTED OPERATION ON UNESTABLISHED HOST CRITICAL
      REGION'
F007H 'VC STATE IN HOST TABLE INCONSISTENT FOR REQUESTED
      ACTION'
F008H 'VC BUSY COUNT INCONSISTENCY'
FFE8H 'RESTORE UDF HOME'
FFE9H 'FATAL FILE SYSTEM CONSISTENCY ERROR'
FFEBH 'INCORRECT PASSWORD' FFECH 'USER NAME NOT KNOWN'
FFEDH 'ILLEGAL USER ID VALUE'
FFEEH 'OPERATION LIMITED TO SUPER USER/MASTER SUPER USER'
FFEFH 'USER OR HOME DIRECTORY DEFINITION FILE DOES NOT
      EXIST'
FFF0H 'FATAL IO ERROR'
FFF1H 'REBOOT REQUIRED'
FFF2H 'USER ID ALREADY EXISTS'
FFF3H 'USER NAME ALREADY EXISTS OR USER SPECIFICATION
      MISMATCH'
FFF4H 'SYSTEM FILE INACCESSIBLE'
FFF5H 'E IO'
FFF9H 'E BOUNDS
FFFAH 'MAXIMUM FILE LENGTH EXCEEDED'
FFFBH 'DELETED'
FFFCH 'END OF FILE'
```

C.2.16 MIP Exception Codes

The following exception codes indicate the status of a Multibus Interrupt Protocol (MIP) communication request. Nonzero MIP exceptions are fatal indications of a failure between processors.

```
00H MESSAGE DELIVERED TO DESTINATION PORT
01H DESTINATION PORT NOT RECOGNIZED BY DESTINATION DEVICE
05H INSUFFICIENT RESOURCES AT DESTINATION DEVICE TO
    RECIEVE MESSAGE
07H DESTINATION PORT CLOSED
09H DESTINATION DEVICE DID NOT RESPOND TO THIS REQUEST
    WITHIN 10 SECONDS, OR WAS ALREADDY DECLARED DEAD
0BH MESSAGE AT ADDRESS BELOW 1000:0 IS INACCESSABLE TO
    DESTINATION DEVICE
```

C.2.17 Communication Subsystem Exception Codes

The Transport Control Layer (TCL) of the communication sub-system indicates the status of network communication requests and reflect the state of the TCL virtual circuits (VC) following the requests. TCL exceptions are usually non-fatal at the NRM, but indicate a failure of a remote workstation. The TCL exceptions at a workstation are fatal and are reported by the message:

```
WS TCL EXCEPTION,
```

followed by the message:

```
SPU CONSISTENCY CHECK, E=XXXH CS:IP=XXXX:XXXXH
```

or:

```
CPIO CONSISTENCY CHECK, E=XXXXH
```

The TCL exception will be dislayed in the consistency check message. These exceptions will never be returned to a user program.

```
01H NO ERROR DETECTED
02H INVALID REQUEST TO TCL
03H NO ERROR DETECTED, EOM WAS SENT
04H DESTINATION NODE HAD INSUFFICIENT RESOURCE TO RECIEVE
    MESSAGE
05H NO ERROR DETECTED, FIN RECIEVED, VC CLOSED
06H ATTEMPTED TO COMMUNICATE VIA NON-EXISTENT VC
08H BUFFER TO SMALL FOR STATUS REQUEST
0AH ATTEMPT TO SEND AFTER VC WAS CLOSED
0CH VC WAS ABORTED BY LOCAL CLIENT
0EH VC WAS ABORTED BY REMOTE CLIENT
10H LOCAL TIMEOUT ABORT
12H VC CLOSE SEQUENCE COMPLETE (NOT AN ERROR)
14H TCL RECIEVED INVALID POINTER TO REQUEST BLOCK OR IN
    REQUEST BLOCK
16H ATTEMPTED TO POST RECIEVE BUFFER ON VC THAT IS CLOSED
```

C.2.18 UDI-Series IV Exception Codes

Exceptions Returned From System Calls

```
E$OK (0000H) E$ABS (2303H)
```

Program contains an absolute record.

E$ACTIVE

Open connections to the volume existed and had to be forcibly detached.

E$ADDRESS

The overlay loaded contained addresses in the operating system area. The load was not completed.

EBADFILE (2301H)

The file containing the overlay is not a valid object file.

E$COMM$ERROR

An error occurred in the communication system.

E$CONNECTION$EXIST

Connections to the volme existed and were detached.

E$CONSOLE

An attempt was made to interace a commection with the console.

E$CONTEXT (0101H)

The routine was called on an illegal context. More specifically, this includes and attempt to:

- Attach, create, or delete a file which the console was assigned.
- Attach, create, or delete the user files containing overlays
- Attach, create, delete or rename a file at the NRM when the user is no longer logged on (NDS-II workstations).

E$CROSSFS (0102H)

This operation attemped an illegal cross volume rename.

E$DEVICE$ERROR

An error occurred in the machinery.

E$DEVICE$NOT$READY

The device is not ready for use.

EDIRNOT$EMPTY

The target directory is not empty.

E$EXIST (0103H)

The specified token or connection did not exist, or the specified overlay did not exist.

E$FACCESS (0026H)

A deletion, rename, or destructive creation of a write protect format file was attempted; or the mode of open did not agree with the file attributes of device characteristics.

E$FEXIST (0020H)

The specified file exists when it is not expected to exist.

E$FNEXIST (0021H)

The specified file does not exist when expected. The deletion, renaming, or attachment of a workfile will also cause the exception.

E$FTYPE

A non-terminal path component is not a directory file.

EILLRECORD (2302H)

Illegal OMF record detected by the loader.

EILLVOLUME

An illegal volume-name was specified.

E$LIMIT

The calling job has exceeded the allowable number of attaches.

E$MARK$DELETED

The file has been marked for deletion and cannot be attached.

E$MEM (002H)

Insufficient memory for requested operation.

E$MOUNTED

The device is already mounted.

E$NODE$NOT$READY

The node is not responding to the request.

E$NOPEN (0104H)

The operation attempted to close, read, write, or seek a connection that was not opened.

E$OPEN (0105H)

The operation attempted to open a connection that was already opened.

E$OPEN$MODE

The opened mode does not allow reading to occur.

E$OREAD (0106H)

A write operation was attempted on a connection opened for read.

E$OWRITE (0107H)

A read operation was attempted on a connection opened for write.

E$PARAM (0108H)

An argument had an illegal value. This is usually the result of a bound check (e.g., 0 < connection token < 12).

E$PTR (0109H)

a pointer argument was illegal. If this was the *excep$p* argument, the operating system aborts the job and prints an error message to the cold-start console.

E$SHADOWED

The (local) volume has been mounted, but shadows the corresponding public volume.

E$SHARE (0028H)

An attempt was made to delete, rename, open, destructively create, or attach to a file on which a connection was already established. This may mean that the file is currently open by another user.

E$SIX (010AH)

An attempt was made to open a seventh connection.

E$SPACE (0029H)

The operation attempted to add a directory entry to a full directory.

E$STRING$BUF (0081H)

The string is over 45 characters long (DQ$CHANGE$EXTENSION) or the argument is over 80 characters long (DQGETARGUMENT).

E$SUPPORT (23H)

One of the following operations was attempted:

- The deletion or renaming of a physical or logical device
- The seeking of a physical device or the console
- A DQ$SPECIAL with a connection that was not established on :CI:.
- A DQ$SPECIAL with type 1 or type 3 when :CI: has been assigned to a disk file.

E$SYNTAX (010CH)

An illegal ISIS pathname was specified. This includes device-name parts not supported by Series IV, or an illegal overlay name.

E$SYSTEM$DEVICE

An attempt was made to dismount a system device.

Hardware Detected Conditions

E$ZERO$DIVIDE (8000H)

A divide by zero was attempted.

E$OVERFLOW (8001H)

An overflow occurred

E$8087 (8007H)

An 8087 error occurred.

APPENDIX D
TERMINAL CONFIGURATION

D.1 Introduction

The serial channel drivers on the Series IV are designed to use the standard Series IV console. However, terminals with characteristics different from the Series IV console can be connected to serial channel 1 and used as a second user terminal. Also a modem can be connected to serial channel 1 for remote system access. The STTY utility provides the configuration capability with the "Config" option (see the STTY command description in Chapter 4). Using the STTY Config option requires a configuration file containing specific terminal configuration information. The configuration file provided with the iNDX operating system for the standard Series IV console is SIV.CFG. Other configuration files can be developed to provide additional or different terminal capabilities. The HELP.CFG file provides more information on developing configuration files. This appendix includes the following information:

- a description of the terminal configuration commands available to use in configuration files.
- description and listings of additional example configuration files.
- information on configuring the Series IV as a terminal connected to a host computer.

Additional information on the construction and use of configuration files can be found in HELP.CFG file located in the volume root directory.

D.2 Terminal Configuration Commands

The configuration files used in the STTY command allow terminal characteristics to be controlled using specific configuration commands. These commands must be placed in a ".CFG" file that is used in the STTY Config option. To create a terminal configuration file, compare your terminal's behavior to the default values (COBRA.CFG is the default terminal). Also, refer to the terminal manual for the codes that your terminal expects and generates. To use the terminal with the Series IV, it must meet the following conditions:

- ASCII codes 20H through 7EH display some symbol that requires one column space.
- Carriage return (0DH) must cause the cursor to return to begining of the line and linefeed (0AH) must cause the cursor to move down one line.
- The down, home, left, right, and up cursor codes must have key input codes and screen output codes. Output screen codes for clear rest of screen, clear rest of line, and direct cursor addressing are desirable but not required. The default codes can be changed with the terminal configuration commands.
- The terminal accepts a blankout code that blanks out the former contents of the screen location to which it is output.
- Your terminal should not generate a linefeed with a carriage return (carriage return only). This feature can be switched on and off on some terminals.

The STTY Config or SYSGEN Config (in Options 10 and 11) must be specified to alter the existing configuration parameters. The configuration commands are

accumulative, i.e., only specified parameters are altered. For example, if the default conditions are in effect and a configuration file with one command envoked, then only that one command will be altered. All other default parameters will remain intact.

To change a command from a specific value to an unspecific condition (for example, configuring terminals that do not have a particular feature specified in the default file) use a semicolon to indicate deactivation:

AFIL=;

D.2.1 AEDIT Compatibility

The commands used in the configuration file are as much as possible the same as those used in the AEDIT terminal configuration files.

The only AEDIT terminal configuration macro file required when using AEDIT is AV=24; (or AV = whatever;). STTY ignores any configuration command in the ".CFG" file it does not recognize, assuming that it may be some AEDIT command.

D.2.2 Configuration Command Descriptions

AV

The AV command sets the total number of lines that appear on the terminal display.

Autowrap AW

Autowrap set to true (AW=T;) indicates that the terminal wraps to next line when a character is printed in column 80.

Cursor Addressing AX

The AX command should be set to true (AX=T;) if the column precedes row in cursor command addressing.

Invisible Attributes AI

Set the invisible attribute command to true (AI=T;) to indicate that a terminal has invisable attributes (e.g. start reverse video command not requiring a character position on the screen).

Character Attributes AC

Character attributes set to true (AC=T;) indicates terminal has character attributes (not field). For example, when AC=T; and reverse video is in effect, all characters will be printed in reverse video. Field attributes (AC=F;) have start and end characters, all characters in the "field" between the start and end characters will be affected by the attribute.

Cursor Address AO

Cursor address offset is the offset to add to both the row and column number for the cursor address command AFAC (listed in the Screen Output Sequence Command Section D.2.2.3).

Blank Character AFBK

AFBK specifies the blank character value. This value is used on the fill commands, like clear to end of line, clear to end of screen, etc. To change the value of the normal blank character, use the AFPA command defined below. Often both are set to the end reverse video character.

Ignore Character AFIG

The character specified by the ignore character command (AFIG) will be ignored.

Mask Key AFKM

The value supplied by the key mask command (AFKM) will be applied when reading keys from the terminal.

Print Mask AFPM

The value supplied by the print mask command (AFPM) will be applied when printing characters to the terminal.

DC1-DC3 AFDC

This command should be set to true (AFDC=T;) if your terminal expects DC1-DC3 protocol. DC3 (cntrl S) stops screen output until DC1 (cntrl Q) is received (also functional out of line edit mode).

Hang Up AFHG

This command should be set to true if you wish DTR to be dropped when you switch from serial to local console (if you want your modem not to answer when you are not using the serial channel). Thus you can set the modem in your office to answer only when your console is redirected to your remote location allowing normal voice communication during business hours.

LOGOFF AFLO

When the Logoff command is set to true (AFLO=T;) the user is automatically logged off when a switch from local to remote or remote to local occurs. This provides additional security for the Series IV.

Slow Edit Mode AFSE

If AFSE is set to true, a delay is entered in the screen scrolling process in line edit mode, allowing time for a control S to stop the screen. If set to false, the screen scroll rate is not delayed.

Keyboard Unlock Key AFKY

The AFKY command sets the string value that may be entered at the primary Series IV console to reclaim system control after a STTY Remote command has been performed (equivalent to executing an STTY Local command at the remote terminal. The key may be set to a string up to 14 characters long and single quotes should be used for lower case. This command should be placed in the primary configuration file to execute properly.

D.2.3 Serial Channel Control Command Descriptions

The following commands set control values effecting serial channel one. The value of these commands normally do not have to be modified.

Setup 51 Command AFS5—This command is used when seting up the baud rate for serial channel one. It defaults to: Async, 1 stop bit, no parity, 8 char bits and 16x baud rate factor.

Enable 51 Command AFE5—This command is used when a switch to the serial channel one is made. It defaults to: RTS, error reset, receive enable, DTR, and transmit enable.

Disable 51 command AFD5—This command is used when switching from serial channel back to local (when hangup (AFHG) is true. It defaults to: RTS, error reset, receive enable, DTR disable, and transmit enable. It is designed to drop DTR thus "Hanging up" the modem so that incoming calls will not be answered by the modem.

Rx ready mask AFRX—This value is used as a mask to determine if receive is ready (a character has arrived). Default value is '010B'.

Tx ready mask AFTX—This value is used as a mask to determine if ok to transmit next character to the screen. Default value is '0101B'.

Data overrun mask AFOV—This value is used as a mask to determine if a second character has arrived before first was read (data overrun). It defaults to '010000B'.

D.2.4 Keyboard Input Sequence Commands

The following commands are used to convert the keyboard sequences to iNDX codes. For example, on the Series IV console, the character delete key sends an "ESC, 'Q'" sequence to the serial channel driver which gets converted to the Series IV code 0E0H. Each sequence may be up to four characters. The "read as" table (see section D.2.6) is applied prior to input sequences.

In Table D-1, the input sequence commands are listed, the key function that the command affects (along with the standard control or escape code for that function), and the code that is expected by the iNDX operating system for that key function. The values shown below in Table D-1. are the default values of the input sequence commands for a Series IV secondary terminal.

NOTE

The input sequence commands AFC1 through AFC3 operate on the function keys to the right of the toggle key.

Table D-1. Input Sequence Commands

Input Sequence	Key Function	iNDX Code
AFCU=0B;	CURSER UP (VT)	(0E7H)
AFCD=0A;	CURSOR DOWN (LF)	(0E8H)
AFCL=08;	CURSOR LEFT (BS)	(0E9H)
AFCR=0C;	CURSOR RIGHT (FF)	(0EAH)
AFCH=1E;	CURSOR HOME (RS)	(0E1H)
AFXF=1B51;	CHAR DELETE (ESC Q)	(0E0H)
AFXZ=1B54;	DELETE LINE (ESC T)	(0E2H)
AFXX=18;	DELETE LEFT (cntrl x)	(018H)
AFXA=01;	DELETE RIGHT (cntrl a)	(001H)
AFXU=15;	UNDO (cntrl u)	(015H)
AR=7F;	RUBOUT (DEL)	(07FH)
AFSS=13;	SCREEN SUSPEND (cntrl s)	(013H)
AFSR=11;	SCREEN RESUME (cntrl q)	(011H)
AFCA=03;	COMMAND ABORT (cntrl c)	(003H)
AFDB=04;	DEBUG (cntrl d)	(004H)
AFJA=00;	JOB ABORT (break)	(08CH)
AB=1B;	escape (ESC)	(01BH)
AFSO=;	SCREEN ON OFF TOGGLE (cntrl O)	(00FH)
AFC1=;	CLEAR SCROLL PART OF SCREEN	(0FCH)
AFC2=;	CLEAR MESSAGE PART OF SCREEN	(0FDH)
AFC3=;	CLEAR PROMPT PART OF SCREEN	(0FEH)
AFL0=01440D;	UNSHIFTED FUNCTION KEY 0	(0F0H)
AFL1=01450D;	UNSHIFTED FUNCTION KEY 1	(0F1H)
AFL2=01460D;	UNSHIFTED FUNCTION KEY 2	(0F2H)
AFL3=01470D;	UNSHIFTED FUNCTION KEY 3	(0F3H)
AFL4=01480D;	UNSHIFTED FUNCTION KEY 4	(0F4H)
AFL5=01490D;	UNSHIFTED FUNCTION KEY 5	(0F5H)
AFL6=014A0D;	UNSHIFTED FUNCTION KEY 6	(0F6H)
AFL7=014B0D;	UNSHIFTED FUNCTION KEY 7	(0F7H)
AFU0=01640D;	SHIFTED FUNCTION KEY 0	(0D0H)
AFU1=01650D;	SHIFTED FUNCTION KEY 1	(0D1H)
AFU2=01660D;	SHIFTED FUNCTION KEY 2	(0D2H)
AFU3=01670D;	SHIFTED FUNCTION KEY 3	(0D3H)
AFU4=01680D;	SHIFTED FUNCTION KEY 4	(0D4H)
AFU5=01690D;	SHIFTED FUNCTION KEY 5	(0D5H)
AFU6=016A0D;	SHIFTED FUNCTION KEY 6	(0D6H)
AFU7=016B0D;	SHIFTED FUNCTION KEY 7	(0D7H)
AFCC=;	CAUSE INTERRUPT 3 (NMI)	(08BH)
AFTS=14;	TOGGLE SCREEN	(098H)

D.2.5 Screen Output Sequence Commands

The output sequence commands convert screen control characters from the iNDX codes to the sequence expected by your terminal. For example, the default cursor address command for the secondary terminal (ESC, '-', '0', line, char) is converted from the iNDX cursor address command (ESC, 'Y', line, char). Each output sequence may be up to four characters. If the code from iNDX is matched by these screen output sequences, then the converted value is applied directly to the terminal. If not, the print as character table is used (see Section D.2.6) for conversion.

In Table D-2, the screen output sequence commands are listed with the screen function that the command affects and the code produced by iNDX for that function. The screen output command sequence values shown below in Table D-2 represent the default values for the secondary terminal.

In Table D-3 the output sequence delay commands are listed, along with the screen function that they affect. After the screen command conversion is complete the output of the value is delayed to the terminal by the time specified in the delay command. The delay value is that used in the PLM-86 CALL TIME procedure, and the range is from 0 to FF.

Table D-2. Screen Sequence Commands

Output Sequence Command	Screen Function	iNDX Code
AFMU=0B;	CURSOR UP (VT)	(ESC A)
AFMD=0A;	CURSOR DOWN (LF)	(ESC B)
AFML=08;	CURSOR LEFT (BS)	(ESC D)
AFMR=0C;	CURSOR RIGHT (FF)	(ESC C)
AFMH=1E;	CURSOR HOME (RS)	(ESC H)
AFMB=0D;	RETURN (CR)	(0DH)
AFER=1B59;	ERASE REST OF SCREEN (ESC Y)	(ESC J)
AFEL=1B54;	ERASE REST OF LINE (ESC T)	(ESC K)
AFAC=1B2D30;	ADDRESS CURSOR LEAD IN (ESC - 0)	(ESC Y line char)
AFAT=1B47;	ATTRIBUTE CMD LEAD IN (ESC G)	(ESC L attr_char)
AFRV=;	REVERSE VIDEO START (ESC G CODE)	(ESC L if AFAT=;)
AFNV=;	REVERSE VIDEO END (ESC G CODE)	(ESC L if AFAT=;)
AFIL=1B45;	INSERT LINE (ESC E)	(NONE)
AFDL=1B52;	DELETE LINE (ESC R)	(NONE)

Table D-3. Output Sequence Delay Commands

Ouput Sequence Delay Command	Function Affected
ADMU	CURSOR UP
ADMD	CURSOR DOWN
ADML	CURSOR LEFT
ADMR	CURSOR RIGHT

Table D-3. Output Sequence Delay Commands (Cont'd)

Ouput Sequence Delay Command	Function Affected
ADMH	CURSOR HOME
ADMB	RETURN
ADER	ERASE REST OF SCREEN
ADEL	ERASE REST OF LINE
ADAC	ADDRESS CURSOR LEAD IN
ADAT	ATTRIBUTE COMMAND LEAD IN
ADRV	REVERSE VIDEO START
ADNV	REVERSE VIDEO END
ADIL	INSERT LINE
ADDL	DELETE LINE

D.2.6 Conversion Table Update Commands

There are two conversion tables that can be established within the configuration file, the Read As table and the Print As table.

Read AS Character Table

The Read As character table is modified with the AFRA command: Multiple AFRA commands used in a configuration file. Read as table is used immediately after key is pressed before any other operations are performed on the key (except key mask).

The format of AFRA is:

```
AFRA=offset value;
```

where:

offset is position in table to be updated (max is 0FFH).
value is value to be read from table.

example:

```
AFRA=801B;
```

Puts hex value 1B (ESC) into table at hex locaton 80. Thus when key giving a value of 80H is pressed it will be read as 1BH (ESC).

Print As Character Table

The Print As character table is modified with the AFPA command: Like the REad as character command, multiple AFPA commands may be specified within a configuration file. Print as table is the last event in printing characters to the screen.

Format of AFPA is:

```
AFPA=offset value;
```

where:

offset is position in table to be updated (max is 07FH).
value is value to be printed from table.

example:

```
AFPA=7B28; AFPA=7D29;
```

Puts hex value 28 "(" into table at hex locaton 7b "{" and hex value 29 ")" into table at hex locaton 7d "}". Thus when "{" is to be printed, "(" will be printed instead and when "}" is to be printed, ")" will be printed instead.

D.3 Series IV Serial Channnel 1 Default Values

The default values for the secondary or serial channel terminal are listed below and are found in the file COBRA.CFG.

AV=24;	NUMBER OF LINES (decimal)
AW=T;	WRAPPER
AX=F;	X FIRST
AI=F;	INVISABLE ATTRIBUTES
AC=F;	CHARACTER ATTRIBUTES
AO=20;	OFFSET FOR CURSOR ADDR
AFBK=20;	BLANK CHARACTER
AFIG=FF;	IGNORE CHARACTER
AFKM=7F;	KEY MASK
AFPM=7F;	PRINT MASK
AFDC=T;	DC1-DC3 PROTOCOL
AFHG=T;	HANG UP MODEM ON SW LOCAL
AFLO=T;	LOGOFF ON CONSOL SWITCH
AFSE=T;	SLOW EDIT ON
AFKY=;	KEYBOARD UNLOCK KEY

The default values for the Serial Channel 1 terminal configuration file for the input sequence and output sequence commands are included in Tables D-1 and D-2.

D.4 Series IV Primary Console Configuration

The Series IV primary console configuration file is located on the iNDX.CUSPS diskette with the filename of SIV.CFG. This file can be altered to customize the keyboard or screen functions. The default values for the primary console are listed in Tables D-4 thru D-6.

D.5 The Series IV as a Terminal

When the STTY Terminal option is specified, the Series IV is configured to act as a terminal connected to a host computer, via Serial Channel 1. When the Series IV operates in the terminal mode, it utilizes the primary configuration file, SIV.CFG (see Tables D-4 thru D-6 for default values).

Table D-4. Primary Configuration File Default Values

AV=24;	NUMBER OF LINES (decimal)
AW=T;	WRAPPER
AX=F;	X FIRST
AI=F;	INVISABLE ATTRIBUTES
AC=F;	CHARACTER ATTRIBUTES
AO=20;	OFFSET FOR CURSOR ADDR
AFBK=20;	BLANK CHARACTER
AFIG=FF;	IGNORE CHARACTER
AFKM=7F;	KEY MASK
AFPM=7F;	PRINT MASK
AFDC=T;	DC1-DC3 PROTOCOL
AFHG=T;	HANG UP MODEM ON SW LOCAL
AFLO=T;	LOGOFF ON CONSOL SWITCH
AFKY=;	KEYBOARD UNLOCK KEY
AFSE=T;	SLOW EDIT MODE

Table D-5. Primary Configuration File Input Sequence Commands

Input Sequence	Key Function	iNDX Code
AFCU=E7;	CURSER UP (VT)	(0E7H)
AFCD=E8;	CURSOR DOWN (LF)	(0E8H)
AFCL=EA;	CURSOR LEFT (BS)	(0E9H)
AFCR=E1;	CURSOR RIGHT (FF)	(0EAH)
AFCH=E0;	CURSOR HOME (RS)	(0E1H)
AFXF=E0;	CHAR DELETE (ESC Q)	(0E0H)
AFXZ=E2;	DELETE LINE (ESC T)	(0E2H)
AFXX=18;	DELETE LEFT (cntrl x)	(018H)
AFXA=01;	DELETE RIGHT (cntrl a)	(001H)
AFXU=15;	UNDO (cntrl u)	(015H)
AR=7F;	RUBOUT (DEL)	(07FH)
AFSS=13;	SCREEN SUSPEND (cntrl s)	(013H)
AFSR=11;	SCREEN RESUME (cntrl q)	(011H)
AFCA=03;	COMMAND ABORT (cntrl c)	(003H)
AFDB=04;	DEBUG (cntrl d)	(004H)
AFJA=8C;	JOB ABORT (break)	(08CH)
AB=1B;	escape (ESC)	(01BH)
AFPT=;	prompt on off toggle (cntrl p)	(010H)
AFRL=;	return last line (cntrl r)	(012H)
AFSO=;	SCREEN ON OFF TOGGLE (cntrl O)	(00FH)
AFC1=F9;	CLEAR SCROLL PART OF SCREEN	(0FCH)
AFC2=FA;	CLEAR MESSAGE PART OF SCREEN	(0FDH)
AFC3=FB;	CLEAR PROMPT PART OF SCREEN	(0FEH)
AFL0=F0;	UNSHIFTED FUNCTION KEY 0	(0F0H)
AFL1=F1;	UNSHIFTED FUNCTION KEY 1	(0F1H)
AFL2=F2;	UNSHIFTED FUNCTION KEY 2	(0F2H)
AFL3=F3;	UNSHIFTED FUNCTION KEY 3	(0F3H)
AFL4=F4;	UNSHIFTED FUNCTION KEY 4	(0F4H)
AFL5=F5;	UNSHIFTED FUNCTION KEY 5	(0F5H)
AFL6=F6;	UNSHIFTED FUNCTION KEY 6	(0F6H)
AFL7=F7;	UNSHIFTED FUNCTION KEY 7	(0F7H)
AFU0=D0;	SHIFTED FUNCTION KEY 0	(0D0H)
AFU1=D1;	SHIFTED FUNCTION KEY 1	(0D1H)
AFU2=D2;	SHIFTED FUNCTION KEY 2	(0D2H)
AFU3=D3;	SHIFTED FUNCTION KEY 3	(0D3H)
AFU4=D4;	SHIFTED FUNCTION KEY 4	(0D4H)
AFU5=D5;	SHIFTED FUNCTION KEY 5	(0D5H)

Table D-5. Primary Configuration File Input Sequence Commands (Cont'd)

Input Sequence	Key Function	iNDX Code
AFU6=D6;	SHIFTED FUNCTION KEY 6	(0D6H)
AFU7=D7;	SHIFTED FUNCTION KEY 7	(0D7H)
AFCC=;	CAUSE INTERRUPT 3 (NMI)	(08BH)
AFTS=F8;	TOGGLE SCREEN	(098H)

Table D-6. Primary Configuration File Output Sequence Delay Commands

Output Sequence	Key Function	iNDX Code
AFMU=;	CURSOR UP (VT)	(ESC A)
AFMD=0A;	CURSOR DOWN (LF)	(ESC B)
AFML=;	CURSOR LEFT (BS)	(ESC D)
AFMR=;	CURSOR RIGHT (FF)	(ESC C)
AFMH=;	CURSOR HOME (RS)	(ESC H)
AFMB=0D;	RETURN (CR)	(0DH)
AFER=;	ERASE REST OF SCREEN (ESC Y)	(ESC J)
AFEL=;	ERASE REST OF LINE (ESC T)	(ESC K)
AFAC=;	ADDRESS CURSOR LEAD IN (ESC - 0)	(ESC Y line char)
AFAT=;	ATTRIBUTE CMD LEAD IN (ESC G)	(ESC L attr_char)
AFRV=;	REVERSE VIDEO START (ESC G CODE)	(ESC L if AFAT=;)
AFNV=;	REVERSE VIDEO END (ESC G CODE)	(ESC L if AFAT=;)
AFIL=;	INSERT LINE (ESC E)	(NONE)
AFDL=;	DELETE LINE (ESC R)	(NONE)

Primary Configuration File Conversion Table Commands

AFRA=;
AFRA=CC8C;
AFRA=EC8C;
AFPA=;

D.7 Additional Terminal Configuration Files

There are additional terminal configuration files that are provided on the iNDX.CUSPS diskette shipped with the Series IV. The configuration files include:

Configuration File	Terminal
ADDS3A.CFG	ADDS Viewpoint 3
ADM3A.CFG	Lier Siegler ADM 3A.
COBRA.CFG	Zentec Cobra—deault for Serial channel 1
QUME.CFG	Qume QVT-102
SYNT88.CFG	Synco T88
TAND.CFG	Tandberg data TVD2215
TV910.CFG	Televideo model 910
TV925.CFG	Televideo model 925
TV950.CFG	Televideo model 950
ZEPHYR.CFG	Zentec Zephyr

APPENDIX E
OBJECT MODULE RELOCATION AND LINKAGE

Figure E-1 presents the valid possibilities for combining object modules created by resident and cross-product translators or by relocation-and linkage packages. ("Cross-product" here means software packages that execute on an 8080/8085-based system but create code to run on an 8086/8088 based system).

As long as LINK and LOCATE are used in sequence, object modules developed using cross-product translators can be combined with new object modules developed using resident translators. The permitted and prohibited possiblities are as follows:

Permitted

1. The resident R & L86 will successfully process any object module produced by any INTEL cross-product or resident product.

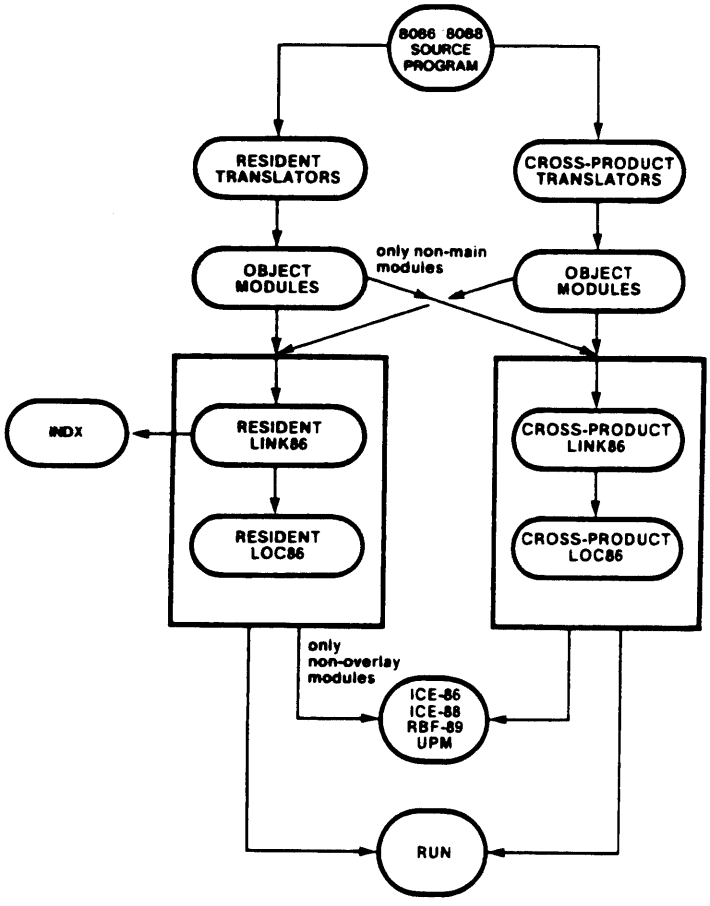


Figure E-1. Use of Relocation and Linkage Packages

2. The cross-product R & L package will successfully process any non-main object module produced by resident translators.
3. Current absolute loaders (e.g., those shown if Figure E-1) will successfully process non-overlay modules produced as well as the output of cross-products.
4. Position-Independent-Code (PIC) or Load-Time Locatable (LTL) modules do not need LOC86 and can be executed directly after being processed by LINK86.

Prohibited

1. The cross-product R & L package cannot process the following kinds of modules:
   a. Main object modules from resident translators.
   b. Modules produced by resident R & L86.
2. Current absolute loaders cannot process overlay modules produced by resident R & L86.

See the *iAPX 86, 88 Family Utilities User's Guide,* 121616, for complete details on INTEL's R & L packages.

NOTE

The Series IV will not load a module with any absolute records (a module that has been "located"), variables declared at an absolute location (@), or a procedure declared "interrupt".

APPENDIX F
BOOT DEVICE & CONFIGURATION SWITCH ASSIGNMENTS

Table F-1. Switch Assignments

Switch Numbers								Functions
1	2	3	4	5	6	7	8	
*	*n*	0	0	0	0	0	0	Skip power-up test and boot system monitor.
*	*n*	0	0	0	0	1	0	Boot system from integral floppy disk, drive 0.
*	*n*	0	1	0	0	1	0	Boot system from integral floppy disk, drive 1.
*	*n*	0	0	0	1	0	0	Boot system from 740 Hard Disk, fixed platter.
*	*n*	0	1	0	1	0	0	Boot system from 740 Hard Disk, removable platter.
*	*n*	0	0	0	1	1	0	Boot system from external peripheral chassis.
*	*n*	0	0	1	0	1	0	Boot system from integral Winchester drive.
*	*n*	0	1	1	0	1	0	Reserved for future configuration.
*	*n*	0	0	1	1	0	0	Reserved for future configuration.
*	*n*	0	0	1	1	1	0	Reserved for future configuration.
*	*n*	0	1	1	1	1	0	Reserved for future configuration.
*	*n*	#	#	#	#	#	1	Boot workstation from network.
*	*n*	0	0	1	1	1	1	Reserved (special case)

NOTES:

1. 0 = OFF (down); 1 = ON (up); *n* = DON'T CARE
2. Switch 1 (*) selects 60Hz when up (1) or 50Hz when down (0); for CRT scan rate only.
3. Switch 2 (*n*) is reserved for future configurations.
4. Switches 3 and 4 select boot device unit addresses.
5. Switches 5, 6 and 7 select boot device.
6. Switch 8 selects network communications booting.
7. # = Bit Substitute; i.e., substitute the bit pattern that corresponds to the device from which the Operating System (OS) will be booted by default. (For example, a workstation that uses a 740 hard disk, drive 0 as a defaulted boot device, would require a switch pattern of:

   * *n* 0 0 0 1 0 1

   If network communications are lost, the system will boot from the address selected by switches 3–7.

APPENDIX G
ASCII CODES

Table G-1. ASCII Code List

Decimal	Octal	Hexadecimal	Character
0	000	00	NUL
1	001	01	SOH
2	002	02	STX
3	003	03	ETX
4	004	04	EOT
5	005	05	ENQ
6	006	06	ACK
7	007	07	BEL
8	010	08	BS
9	011	09	HT
10	012	0A	LF
11	013	0B	VT
12	014	0C	FF
13	015	0D	CR
14	016	0E	SO
15	017	0F	SI
16	020	10	DLE
17	021	11	DC1
18	022	12	DC2
19	023	13	DC3
20	024	14	DC4
21	025	15	NAK
22	026	16	SYN
23	027	17	ETB
24	030	18	CAN
25	031	19	EM
26	032	1A	SUB
27	033	1B	ESC
28	034	1C	FS
29	035	1D	GS
30	036	1E	RS
31	037	1F	US
32	040	20	SP
33	041	21	!
34	042	22	"
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	'
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	,
45	055	2D	-
46	056	2E	.
47	057	2F	/
48	060	30	0
49	061	31	1
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=

Table G-1. ASCII Code List (Cont'd)

Decimal	Octal	Hexadecimal	Character
62	076	3E	>
63	077	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	^
95	137	5F	_
96	140	60	`
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~
127	177	7F	DEL

Table G-2. ASCII Code Definition

Abbreviation	Meaning	Decimal Code
NUL	NULL Character	0
SOH	Start of Heading	1
STX	Start of Text	2
ETX	End of Text	3
EOT	End of Transmission	4
ENQ	Enquiry	5
ACK	Acknowledge	6
BEL	Bell	7
BS	Backspace	8
HT	Horizontal Tabulation	9
LF	Line Feed	10
VT	Vertical Tabulation	11
FF	Form Feed	12
CR	Carriage Return	13
SO	Shift Out	14
SI	Shift In	15
DLE	Data Link Escape	16
DC1	Device Control 1	17
DC2	Device Control 2	18
DC3	Device Control 3	19
DC4	Device Control 4	20
NAK	Negative Acknowledge	21
SYN	Synchronous Idle	22
ETB	End of Transmission Block	23
CAN	Cancel	24
EM	End of Medium	25
SUB	Substitute	26
ESC	Escape	27
FS	File Separator	28
GS	Group Separator	29
RS	Record Separator	30
US	Unit Separator	31
SP	Space	32
DEL	Delete	127

REQUEST FOR READER'S COMMENTS

Intel's Technical Publications Departments attempt to provide publications that meet the needs of all Intel product users. This form lets you participate directly in the publication process. Your comments will help us correct and improve our publications. Please take a few minutes to respond.

Please restrict your comments to the usability, accuracy, readibility, organization, and completeness of this publication. If you have any comments on the product that this publication describes, please contact your Intel representative. If you wish to order publications, contact the Intel Literature Department (see page ii of this manual).

1. Please describe any errors you found in this publication (include page number).

2. Does the publication cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of publication for your needs? Is it at the right level? What other types of publications are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this publication on a scale of 1 to 5 (5 being the best rating.) _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

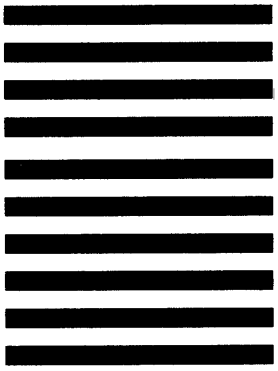
CITY _____ STATE _____ ZIP CODE _____
(COUNTRY)

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS . . .

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN U.S.A.



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1040 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
Attn: Technical Publications M/S DV2-291
2402 West Beardsley Road
Phoenix, Arizona 85027

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.

PE/0684/.95K LW