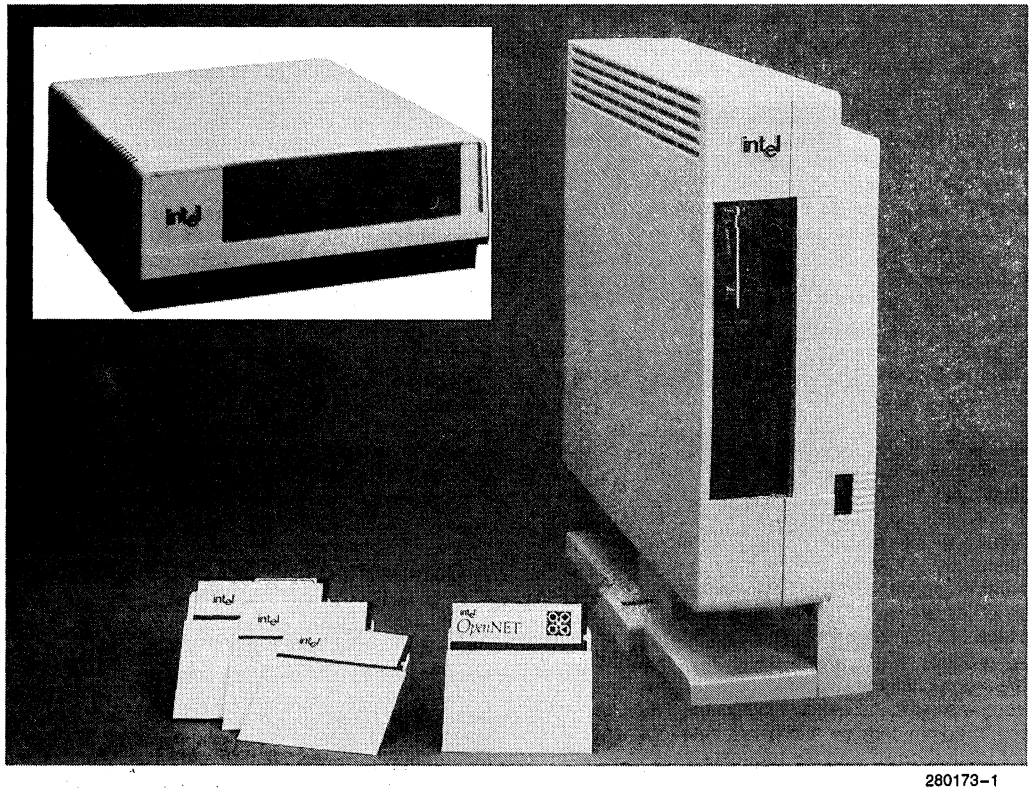
OpenNET™ NETWORK RESOURCE MANAGER (NRM) iMDX 460

- **Ethernet-Based File Server to Connect**
  - **— Intellec Series II/III/IV**
  - **— Model 800**
  - **— Compil$_e$ngine**
  - **— Intel XENIX* and iRMX™-Based System**
  - **— VAX*/MicroVAX* Running VMS**
  - **— DOS 3.1 (and up)-Based PCs**
- **Runs 8- and 16-Bit Languages**
- **Protected Hierarchical File System**
- **Remote Job Execution for Distributed Processing**
- **Shared Resources**
  - **— Central Disk Storage up to 560 MB**
  - **— 60 MB Streaming Tape**
  - **— Shared Spooled Line Printer**
- **High-Performance Hardware**
  - **— iSBC® 286/12, 8 MHz CPU Board with One Megabyte of Zero Wait-State RAM**
  - **— iSBC® 214 Disk Controller with a Dedicated 80186 CPU and Track Caching**



280173-1

*XENIX is a registered trademark of the Microsoft Corp.
*VAX, MicroVAX, VMS and DEC are registered trademarks of Digital Equipment Corp.

November 1986
Order Number: 280173-002

OVERVIEW

The OpenNET NRM is a network file server optimized for a distributed development systems environment. Workstations on the network address the hardware engineer's needs by providing the base environment for tools such as in-circuit emulators and PROM programmers. Furthermore, workstations support software engineering teams—hosting software tools such as compilers, software debuggers, and project management tools. For a project team, the OpenNET NRM creates a network that provides the dual advantage of a powerful desk-top computer and access to shared resources on a high-speed standard network. These capabilities include transparent file access to shared files, remote job execution, and print spooling. The OpenNET NRM is plug-compatible with the original NDS-II NRM.

FUNCTIONAL DESCRIPTION

The OpenNET NRM manages all workstation requests for centralized network resources. These tasks include service of workstation file access requests, print spooling, management of remote job execution queues, and network maintenance functions such as user creation/authentication, file archiving, and system configuration.

The iNDX operating system on the OpenNET NRM uses a hierarchical file system providing file sharing and protection features. With this file organization, files can be logically grouped into directories and sub-directories. File protection is implemented in the form of access rights for each file or directory.

For messages or simple file transfers among workstations, Electronic Mail is included for Series II/III/IV, Model 800 and ISIS cluster users.

In addition to the management of communications between shared disks and workstations, the NRM maximizes the use of all network resources with a remote job execution facility called Distributed Job Control (DJC). Any Series II/III/IV, Model 800, ISIS cluster or VAX/VMS user on the network can export a batch job through the NRM for remote execution on other idle workstations or a specialized system called the Compil$_e$ngine. The Compil$_e$ngine is an Intel product that specializes in the importation of large compilations and link/locates from other network workstations. Alternatively, the NRM can accept medium compile and link/locate jobs when it is not loaded down with file requests.

NDS-II Plus OpenNET™ Network Communications Enable Data Sharing in Mixed Vendor Environments

The OpenNET NRM communicates with other systems over the industry standard Ethernet network. The OpenNET NRM supports two Intel network protocols used on Ehternet: the NDS-II and OpenNET network protocols. Installed in the NRM, and iSBC 550 communications board set supports the NDS-II protocols. An iSXM 552 supports the ISO 8073 compatible iNA960 transport software for the OpenNET network. Intellec Series II/III/IV systems, Model 800 workstations, Compil$_e$ngines, communicate with the NRM via the NDS-II protocols while XENIX, iRMX and PC-DOS systems communicate via the OpenNET network. The VAX and MicroVAX running VMS connect to the NRM via VAX Link.

All data that is stored at the NRM is visible to, and can be transparently accessed by, all workstations in the network. VAX/VMS users have file transfer capability. For example, PCs or 286/310 systems (running the iRMX or XENIX operating system) can share files on the NRM with Series IVs with the capability to upload those files to the VAX for back-up.

Network Managment Facilities

The NRM provides your network administrator with tools such as hardware diagnostics, network status and configuration commands to help optimize your user environment. These capabilities include the display all network users and transaction counts, definition of the number of multitasking jobs and allocate communication memory. For example, for a network with 2–3 Series IVs and 4–5 PCs, the NRM can be configured with maximum communication buffer memory. This is to maximize throughput for a few workstations. Alternatively, for a network with 4–5 Series IVs and 20 PCs, the NRM can be configured with smaller buffers but large numbers of users. This is to maximize the number of users on the network.

A Comprehensive Set of Development Tools Supported

All current Intel development tools such as languages, assemblers, linker/locators, PROM programmers, debuggers and in-circuit emulators can be used on networked workstations. In addition, the NRM can run 8- and 16-bit compile and link/locate jobs to maximize the computing power of your NRM when it is relatively idle from communication tasks. These jobs can be sent from any workstation on the network.

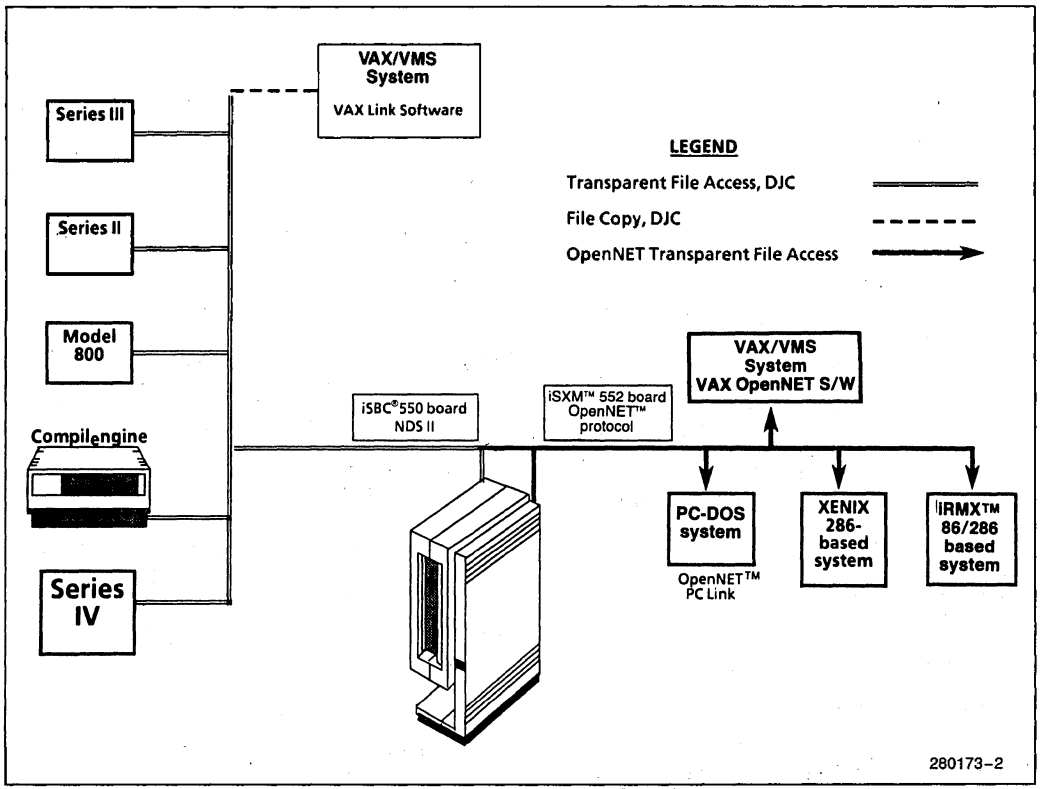


Figure 1. Share Files Transparently on the OpenNET™ NRM From Any Workstation

State-of-the-Art System Components means Performance

The OpenNET NRM is a high performance 80286-based microcomputer. Two models are offered: the MAXI and MINI. The MAXI has a fast 140 MB disk with a 60 MB streaming tape drive while the MINI has a 40 MB disk with no tape. Each system contains an iSBC 286/12 8 MHz CPU board with one megabyte of zero wait-state on-board Random Access Memory (RAM). Also contributing to the high system performance is the iSBC 214 multiprocessing peripheral controller. This controller features its own 80186 microprocessor and 32 KB of software transparent cache memory. The 80186 offloads the 80286 CPU from virtually all peripheral controller tasks, while the cache memory greatly reduces apparent access times to hard disk memory. The iSBC 214 controller is used to support the Winchester and 640 KB half-height floppy disk drives in both the MAXI and MINI models. The iSBC 214 controller additionally supports the streaming tape drive in the MAXI model.

Interconnecting Hardware Is Standard

To connect workstations to the OpenNET NRM, standard Ethernet transceivers, transceiver cables, and Ethernet coaxial cables are used. Intel's Intellink module may also be used to connect multiple workstations to Ethernet. The same (one) Intellink module can be used to connect Series II/III/IVs, VAXs via VAX Link, and OpenNET workstations, as long as cables from both the iSBC 550 and iSXM 522

boards are connected to that Intellink module. The Series II/III/IVs, VAXs will respond to messages from the iSBC 550 board while the OpenNET workstations will respond to messages from the iSXM 552 board. This helps the user optimize usage of current and new interconnect hardware.

The OpenNET™ NRM Expands To Fit Growing Development Environments

Creating a network in your development lab is accomplished by adding an NRM and upgrade kits for Intellec, 286/310 XENIX and iRMX, PC-DOS and VAX/VMS systems. The network grows with your development environment. Workstations can be added, mass storage increased, and multiple networks connected—giving your development environment maximum flexibility. You only need to acquire as much development equipment as you require today, knowing that you will be able to grow in increments tomorrow.

SPECIFICATIONS

Hardware

Dimensions: 6½" x 17" x 22"

Weight: 55 lbs.

Electrical: User selectable AC power with either 88-132V, 60 Hz or 180-264V, 50 Hz

Software

iNDX R3.2 Operating System

OPERATING ENVIRONMENT

Environmental Characteristics

Temperature: 10°C to 35°C

Humidity: 20-80% relative humidity

Altitude: Sea Level to 8000 Feet

Hardware Required

User supplied ANSI 3.64 standard terminal
Workstation interconnecting hardware:
Intellink module, transceivers, transceiver cables, Ethernet coaxial cables (depending on number of workstations and distance)

NDS-II Workstations

Workstation	Hardware/Software Products Required
Compil$_e$ngine	—
Model 800	PIMDX455
Series II/III	PIMDX455
Series IV	PIMDX456
VAX/VMS	IMDX 392 & DEUNA* board
μVAX/VMX	iMDX 392 & DEQNA* board

*A DEC product

OpenNET™ Workstations

Workstation	Hardware/Software Products Required
PC-DOS V3.1 System	PCLNK
System 310 XENIX	XNX-NET & iSXM 552 board
System 310 iRMX 86/286	iRMX-NET 8 iSXM 552 board
VAX/VMS	VMSSVR
μVAX/VMS	MVSSVR

DOCUMENTATION

iNDX User's Guide
(order #: 138809-001)

iNDX System Installation Guide
(order #: 138810-001)

ORDERING INFORMATION

Product Order Code	Description
iMDX 460-140T	OpenNET NRM (MAXI model)
iMDX 460-40	OpenNET NRM (MINI model)
iSYP 312	Floor stand which encloses either the OpenNET NRM or the SYS 311 (see peripheral upgrades section) peripheral expansion box

Interconnecting Hardware

Product Order Code	Description
PIMDX 457/458	Transceiver cables (10/50 meters) (two are required for an OpenNET NRM)
PMDX 3015	Transceiver for Ethernet coaxial cables (at least two are required unless an Intellink is used)
iDCM 91-1	Intellink module (the OpenNET NRM uses two ports)
PIMDX 3016-1/ 3016-2	Ethernet coaxial cable (25/50 meters)

Workstation Kits

Product Order Code	Description
PIMDX 455	NDS-II Workstation Upgrade Kit for any Series II/85, Series III, or Model 800 to connect to the OpenNET NRM
PIMDX 456	NDS-II Workstation Upgrade Kit for the Series IV
PIMDX 581	ISIS Cluster Board Package
MVMSSVR	VAX/VMS-OpenNET Link S/W and Controller Board
VMSSVR	MicroVAX/VMS OpenNET Link S/W and Controller Board
PCLNK	OpenNET PC Link hardware and software kit to connect the PC XT, PC AT, and compatible systems to the NRM via the OpenNET network; requires DOS 3.1 or higher
RMXNT961KITWSU	iRMX Networking Software for a 286/310 system running the iRMX 86 Operating System to connect to the NRM via the OpenNET network
SXM 5524	Ethernet-based Single Board Transport Engine for 310 systems
XNXNETKRIKIT	OpenNET-XenixNET iNA 961 iSXM 552 and XenixNET Pass-through Kit

STORAGE EXPANSION SUB-SYSTEMS

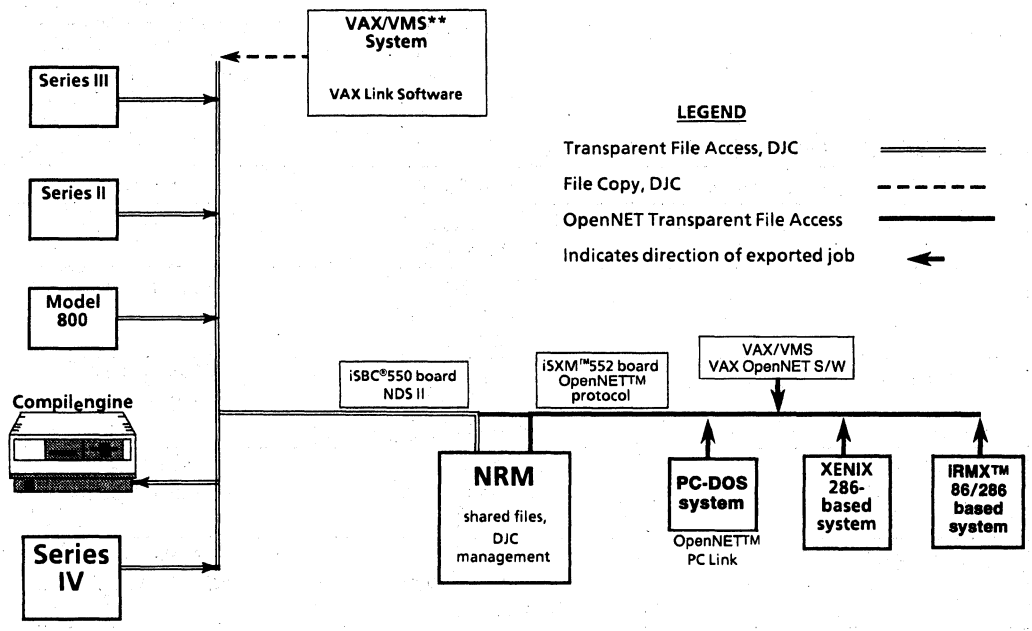
Winchester Storage Size	With Tape	No Tape
0 MB	PSYS311A02 PSXM311TCBL	—
40 MB	PSYS311A14 PSXM311WDCBL PSXM311TCBL	PSYS311A13 PSXM311WDCBL
2 x 40 MB	PSYS311A17 PSXM311WDCBL PSXM311TCBL	PSYS311A16 PSXM311WDCBL
140 MB	PSYS311A34 PSXM311WDCBL PSXM311TCBL	PSYS311A33 PSXM311WDCBL
2 x 140 MB	PSYS311A37 PSXM311WDCBL PSXM311TCBL	PSYS311A36 PSXM311WDCBL
3 x 140 MB	—	PSYS311A39 PSXM311WDCBL

NOTE:
Check product catalog (under PSYS311) for add-on Winchester disk drive ordering information

COMPILENGINE iMDX 485CE

- **Fast, Dedicated Import Station on the Network**
- **Off-Loads Compilations and Link/Locates from other Intellec® Development Workstations on the Network Using Remote Job Execution**
- **Off-Loads Compile Jobs from a DEC VAX/VMS** via the NRM and VAX Link**
- **Supports 8051, 8086, 8096, 80186, 80286 Languages Including ASM, C, PL/M, Pascal, and Fortran**
- **High Performance System Containing an 8 MHz 80286 CPU with One Megabyte of Zero Wait-State RAM**
- **Fast Disk Access via iSBC® 214 Disk Controller with 32 KB Track Caching**
- **640 KB Half-Height Floppy Disk Drive for Initial Software Load**
- **Supports Optional Standard Terminal for System Maintenance and Direct Job Execution**

The Compil$_e$ngine is a 286-based supermicrocomputer system designed to improve productivity of a networked development team. Connected to a Network Resource Manager (NRM), it is optimized to off-load large compile and link/locate jobs from the Series II/III/IV, Model 800, and DEC's VAX/VMS systems. PC-DOS and XENIX* systems connected via the OpenNET™ network to the NRM can also export compiles and link/locates onto the Compil$_e$ngine. The Compil$_e$ngine performs compilations and link/locate jobs faster than any single workstation on the network. By exporting time-consuming jobs to a shared Compil$_e$ngine, workstations are free to perform other tasks such as editing or debugging.

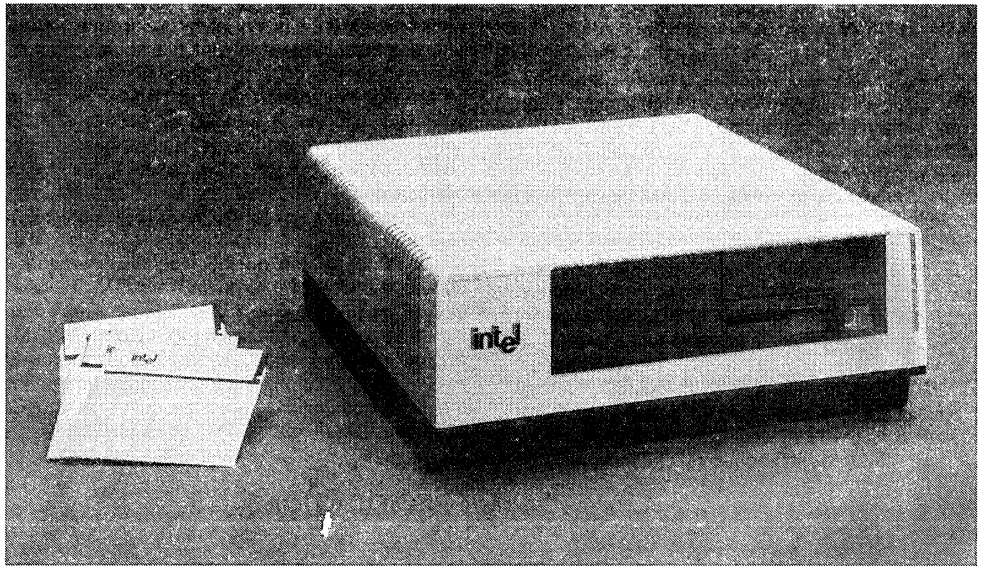


280174-1

Offload Your Non-Interactive Software Jobs from any System to the *Compil$_e$ngine*

*XENIX is a registered trademark of the Microsoft Corporation.
**VAX, VMS, DEC are registered trademarks of Digital Equipment Corporation.

October 1986
Order Number: 280174-002



280174-2

FUNCTIONAL DESCRIPTION

The Compil$_e$ngine adds more performance to your networked development environment by off-loading time consuming tasks from workstations. These tasks are execued by a powerful 286-based system hardware running the iNDX operating system. By using Intel's remote job execution facility called Distributed Job Control (DJC), Intellec Series II/III/IV and Model 800 workstations as well as VAX/VMS systems connected via optional VAX Link software can export compile and/or link/locate jobs to one or more Compil$_e$ngines. Remote job execution is also possible from OpenNET systems via an NRM supporting the OpenNET network and an export utility in the Network Toolbox product. The user can now compile or link, during the workday, those long, CPU-intensive jobs that traditionally have been executed off-hours.

Easy to Start, Easy to Use

The Compil$_e$ngine is connected to the network just like any other workstation on the NDS-II network, via the iSBC 550 communication board set (included) and a transceiver cable. Once physically connected and configured (system generated) onto the NRM, the Compil$_e$ngine starts automatically with a simple switch on of the power. The system autoboots from the floppy disk drive or the network. Sending jobs to this execution vehicle is equally as simple. By executing the EXPORT command on a batch job containing the compile or link/locate task, DJC on the NRM takes over and completes the job at the Compil$_e$ngine.

The Right Location of Files Maximizes Performance

To maximize the performance of compiles and link/locates sent to the Compil$_e$ngine, frequently accessed but stable files (compiler, linker/locators, and language libraries) should reside on the Compil$_e$ngine's local 40 MB Winchester drive. This will help reduce network traffic. On the other hand, frequently changed files such as source code, include files, and user object libraries should reside on the NRM for version control. For the best performance on large compiles, files may be copied to the Compil$_e$ngine as part of an exported job. All shared files should reside on the NRM so that the most up-to-date copy of the files are visible to all network users. During off-hours the NRM can update or replace compilers, linker/locators, or other commonly used files with the latest versions.

State-Of-The-Art System Components Means Performance

The Compil$_e$ngine is a high-performance super-microcomputer with state-of-the-art technology. Each system contains the advanced iSBC 286/12 8 MHz CPU board with one megabyte of zero wait-state on board Random Access Memory (RAM). Also contributing to the high system performance is the iSBC 214 Multiprocessing Peripheral Controller. This controller features its own 80186 microprocessor and 32 KB of software transparent cache memory. The 80186 offloads the 80286 CPU from virtually all peripheral controller tasks, while the cache memory greatly reduces apparent access times to hard disk memory. The iSBC 214 Controller is used to support a 40 MB Winchester drive and 640 KB 5¼" floppy disk drive. To communicate with the NRM, the Compil$_e$ngine uses the iSBC 550 Communication Board set for the standard high-speed (10 MB per second) Ethernet network.

Comprehensive Software Development Tools Supported

Since the Compil$_e$ngine is a very fast, specialized iNDX system, all languages, macro assemblers, and linker/locaters currently supported on the Series IV and the NRM are also supported on the Compil$_e$ngine. This includes popular high-level languages such as PL/M, Pascal, Fortran, and C, as well as powerful "high-level" macro assemblers such as ASM86. These languages support development for 8051, 8086/8088, 80186/188, 80286, and 8096 architectures.

For More Flexibility

Although a terminal is not required to operate the Compil$_e$ngine, the capability to connect an ANSI standard terminal is provided. This feature gives the customer the ability to perform file maintenance on the local storage devices (40 MB Winchester and floppy disk drives). The user can also initiate jobs directly on the Compil$_e$ngine.

APPLICATIONS

As shown in Figure 1, any workstation configured on the NDS-II network can export jobs to the Compil$_e$ngine using DJC. For example, on a Series IV, the user can simply execute the EXPORT command on a batch file containing a compile job. Then, edit or debug other programs while the Compil$_e$ngine compiles that job.

Use of the Compil$_e$ngine(s) from a VAX connected via VAX Link R2.0 is similar to use from a Series IV. The only requirement is that the source files to be compiled or the object modules/library routines to be link/located reside on the NRM. This is easily achieved by creating a batch file on the VAX. Each time a DJC command for remote compiles is executed under VMS, this batch file copies files to be compiled from the VAX onto the NRM and returns the compiled code back to the VAX. (See Figure 2.)

Systems connected on the OpenNET network can access the Compil$_e$ngine via the NRM. To use the Compil$_e$ngine, the PC or XENIX user simply copies the submit file to a special directory on the NRM. An export program in the Network Toolbox that runs on the NRM will scan that directory and send that job to the DJC queue for remote execution at the Compil$_e$ngine. (See Figure 3.)

SPECIFICATIONS

Hardware

Dimensions: 6½" x 17" x 22"
Weight: Less than 55 pounds
Electrical: User selectable AC power with either 88-132V, 60 Hz or 180-264V, 50 Hz

Software

iNDX operating system

OPERATING ENVIRONMENT

Environmental Characteristics

Temperature: 10°C to 35°C
Humidity: 20–80% relative humidity
Altitude: Sea level to 8000 feet

Hardware Required

- An NDS-II NRM (iMDX 450) or OpenNET NRM (iMDX 460) with the iSBC 550 communications board set installed.
- Interconnecting hardware (one of the following):
  - — One transceiver cable and one port on an Intellink module
  - — One transceiver cable, one transceiver and an Ethernet coaxial cable
- Optional: ANSI 3.64 standard terminal

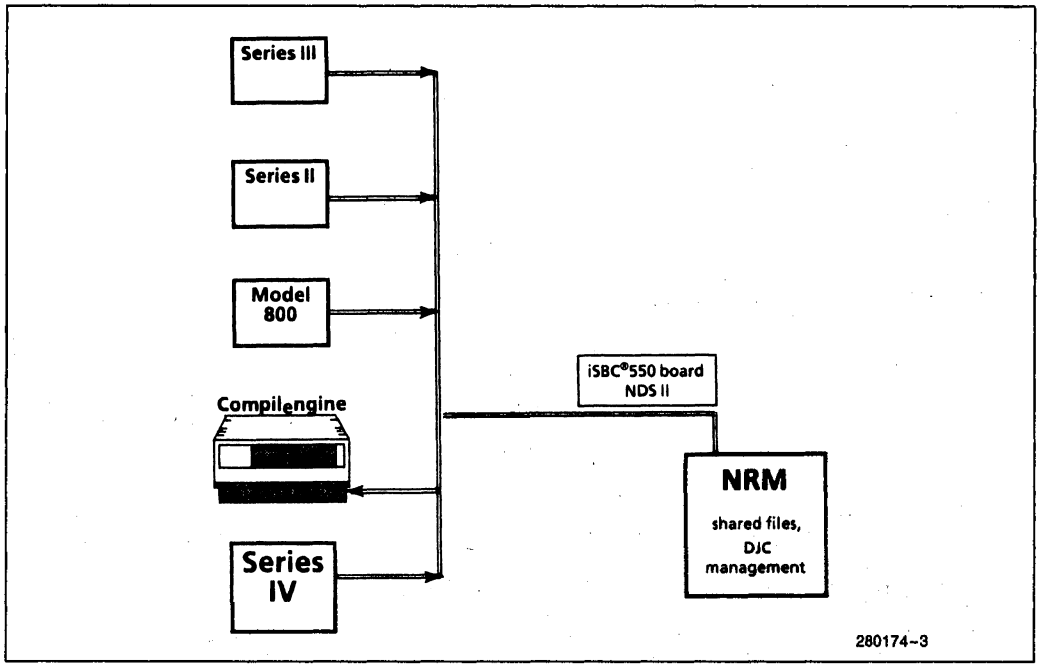


Figure 1. Series II/III/IV and Model 800 Environment

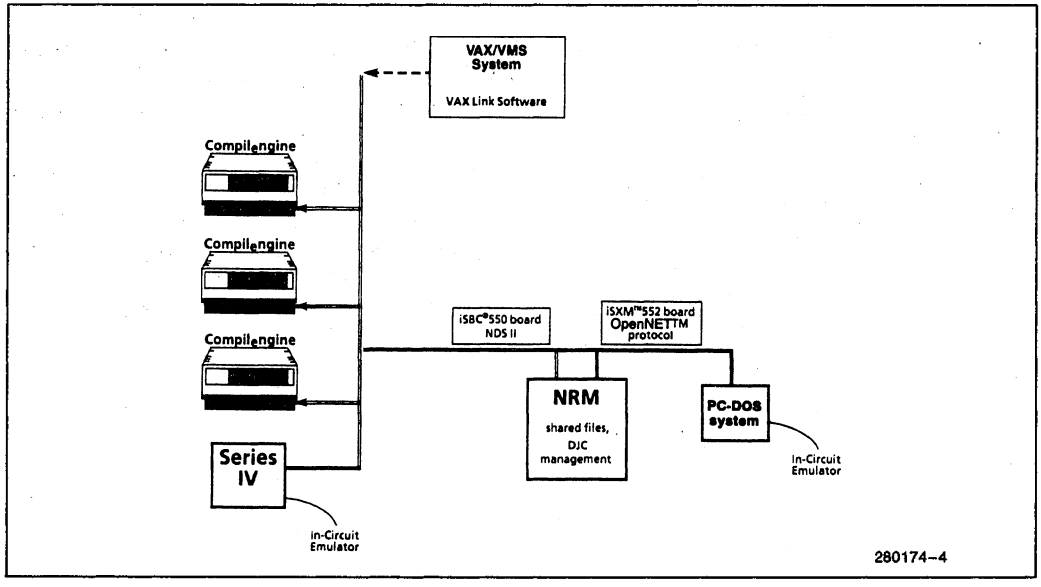


Figure 2. VAX/VMS and Other Intellec® Workstations

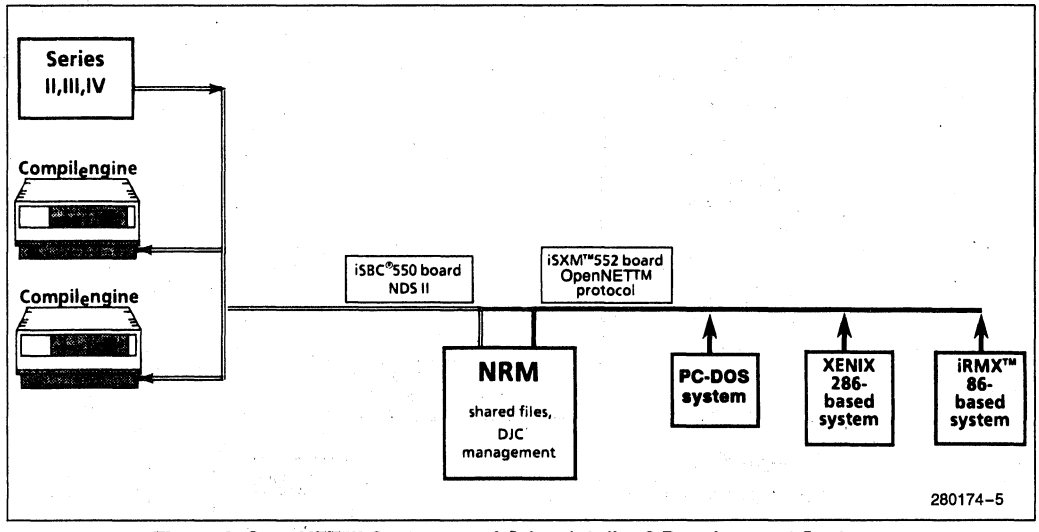


Figure 3. OpenNET™ Systems and Other Intellec® Development Systems

Software Required

High-level language compilers and/or assemblers on 5¼" iNDX-formatted (96 tpi) diskettes.

SUPPORT DOCUMENTATION

iNDX User's Guide
(order # 138809)

iNDX System Installation Guide
(order # 138810)

ORDERING INFORMATION

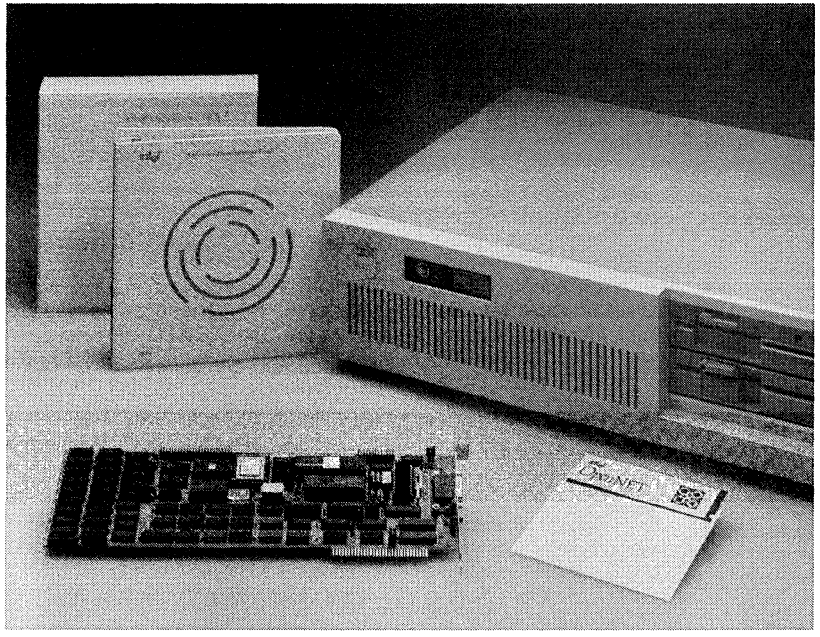
Product Order Code	Description
iMDX 485CE	Compilengine
iMDX 460-140T	OpenNET NRM (maxi model)
iMDX 460-40	OpenNET NRM (mini model)
iMDX 457/458	Transceiver cables (10/50 meters)
iDCM 911-1	Intellink module
iMDX 3015	Transceiver for Ethernet coaxial cables
iMDX 3016	Ethernet coaxial cables (25 or 50 meters)
iSYP 312	Floor stand for the Compilengine
iMDX460-140T	OpenNET NRM file server (Maxi Model)
iMDX460-40	OpenNET NRM file server (Mini Model)

OpenNET™ PERSONAL COMPUTER LINK

- **Connects an IBM* PC AT, PC XT (and PC-DOS Compatibles) to the OpenNET™ Network**
- **Works with Standard DOS Commands**
- **Interconnects a PC System to iRMX™, XENIX*, and NDS II/NRM Systems Offering OpenNET Server Capability**
- **Uses an 80186/82586 Processor-Based Network Controller Board**
- **Contains Power-Up Diagnostics**
- **Supports the ISO/OSI Seven Layer Networking Standards**
- **Enables a PC System to Access Remote Storage and Printer Devices**
- **Provides Transparent-File-Access Capability Between a PC System and Remote Servers**
- **Uses ISO 8073 Transport and Ethernet/ IEEE 802.3 Standard Communication Protocols**
- **Intelligent Board Uses Only 44K of PC's Memory**

The OpenNET Personal Computer Link (OpenNET PC Link) enables users to connect their IBM PC AT and PC XT computer systems to the OpenNET network. This connection enables a PC system to be configured as a consumer workstation on the OpenNET network, and to transparently access and share files and printers on an OpenNET network resource manager (NRM), NDS-II (with the OpenNET upgrade installed) NRM, iRMX, and XENIX-based remote server systems. The OpenNET PC Link is an 80186/82586 microprocessor-based expansion board, which is easily installed in an expansion card slot of the PC system. On-board jumpers and a user configurable software package enable the OpenNET PC Link to be used with a wide-range of expansion boards currently available for the PC system. The OpenNET PC Link incorporates the Microsoft* Networks (MS-NET) networking software and iNA 960 (ISO 8073 compatible) transport software as a part of its software package.

*IBM is a registered trademark of the International Business Machines Corporation.
*MS-DOS is a trademark of the Microsoft Corporation.
*Microsoft is a registered trademark of the Microsoft Corporation.
*XENIX is a trademark of the Microsoft Corporation.



280164-1

October 1986
Order Number: 280164-002

PRODUCT OVERVIEW

The OpenNET PC Link is a member of Intel's OpenNET networking product family. The OpenNET products incorporate a set of system and component level LAN products covering all seven layers of the ISO (International Standards Organization) Open System Interconnect (OSI) model, and the protocols on which they are based. OpenNET network protocols are established industry standards for each function. Therefore, OpenNET network products can connect and operate not only with each other, but with the most popular networking products from other vendors. OpenNET networks provide a high level of interoperability between heterogeneous systems (MS-DOS*, PC-DOS, iNDX, XENIX, and iRMX operating system versions are available). Thus, users can tailor their networks to meet their specific needs by incorporating any combination of the capabilities of these diverse systems.

The OpenNET network application protocols implemented by OpenNET PC Link software are those adopted by Intel, Microsoft, and IBM for their computer networking products. The OpenNET PC Link software is compatible with and will operate with iNDX, XENIX, and iRMX networking software at the application layer.

PHYSICAL DESCRIPTION

The OpenNET PC Link consists of a network controller board and a 5¼ inch disk that contains the software necessary for the PC system to communicate across the OpenNET network. The following sections describe the hardware and software components of the OpenNET PC Link.

OpenNET PC Link Network Controller Board

The network controller board is an adaptor board that can be installed in any available expansion slot of a PC system. The board implements the industry standard ISO 8073 transport protocol (a modified version of iNA 960) and Ethernet/IEEE 802.3 physical data link technology (see Figure 1). The board uses an Intel 80186 microprocessor in combination with an 82586 LAN communication controller. The board includes the following major components:

- 80186 microprocessor
- 82586 LAN communications controller
- 8 KB of EPROM
- 128 KB of RAM shared between the PC system and the 80186 microprocessor on the network controller board
- 82501 Ethernet serial interface
- Fujitsu MB502A encoder decoder
- 15-pin Ethernet D connector
- 8-bit parallel DMA interface and control register set
- Power-up diagnostics

The network controller board performs all network communication functions for the first two layers of the ISO/OSI model (see Figure 2). Layers three and four reside in the modified iNA 960 transport software. The remaining layers (five through seven) reside in the MS-NET networking software on the PC system.

POWER-UP DIAGNOSTICS

An effective diagnostic function is implemented in firmware on the network controller board. This function is invoked at system initializatiion during both power-up and system reset time. The following list summarizes the functions tested:

- 80186 and 82586 microprocessors
- I/O ports
- Shared memory window
- Interrupt channels
- DMA channel
- Ethernet connection

An on-board LED indicates whether the network controller board failed any of the various test functions.

OpenNET PC Link Software

The software is supplied on a 5¼ inch double-density disk (360 KB). The following files are included as part of the OpenNET PC Link:

- A specially configured version of iNA 960 transport layer software, called UBCODE.MEM, which operates on the network controller board.
- A DOS interface driver, called XPORT.EXE, which enables DOS programs to access the network controller board.
- The Microsoft Networks (MS-NET) networking software, release 1.0, which enables users to connect with and access remote file servers on the OpenNET network system.

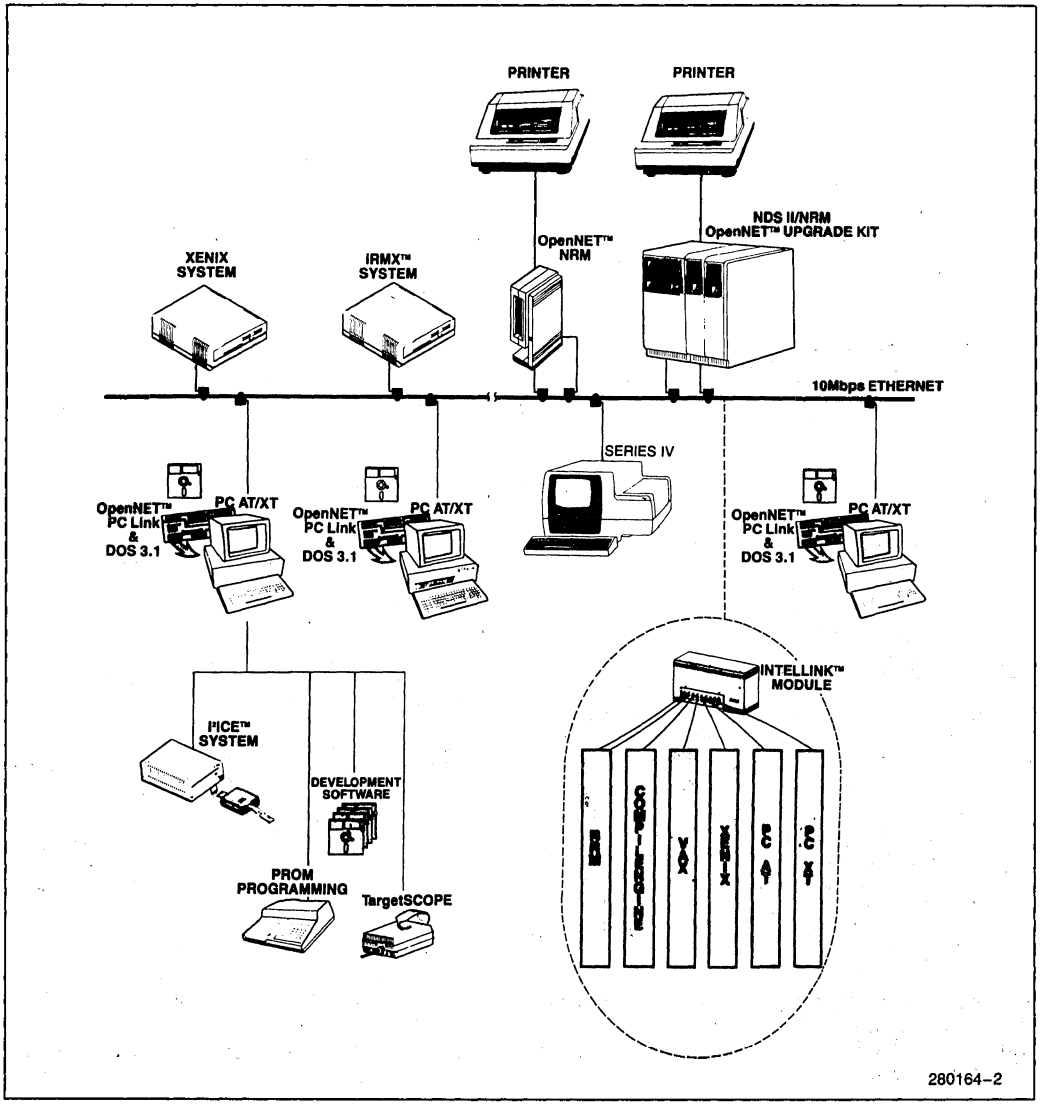


Figure 1. The OpenNET™ PC Link Environment

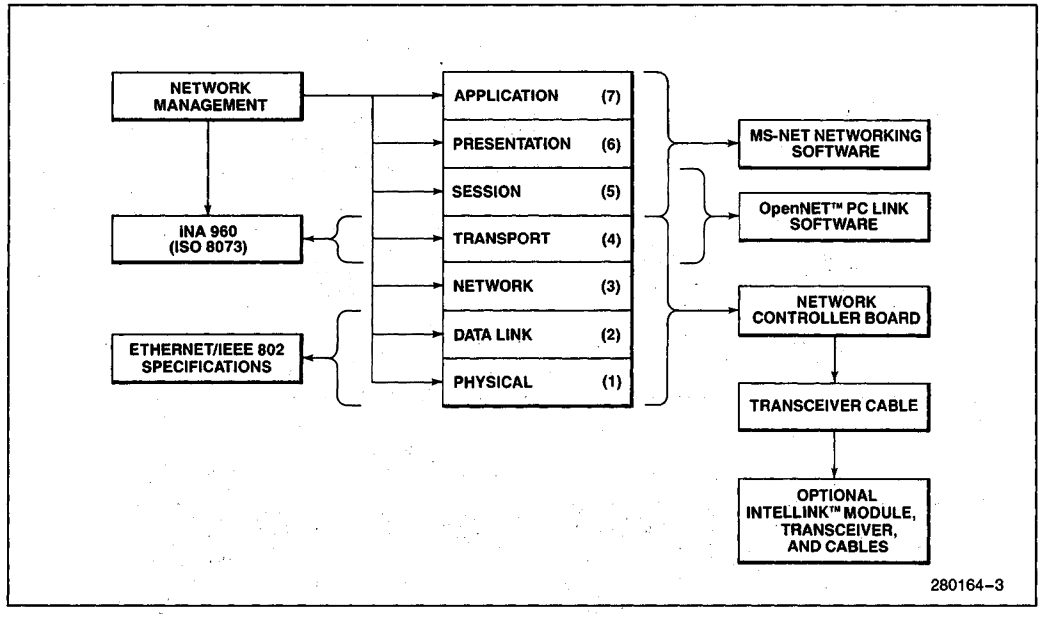


Figure 2. ISO/OSI OpenNET™ PC Link Implementation

PC SYSTEM REQUIREMENTS

For the PC system to function as a workstation on the OpenNET network, it must contain at least 192 KB of memory. A 32 KB memory window is shared between the PC system and the network controller board. The starting address of this window must be placed in an area of memory that does not conflict with the PC system's internal memory address space. The network controller board is jumpered at the factory to reflect a setting which is compatible with the PC system and most of the expansion boards available for use with the PC system.

In order for the network controller board and the OpenNET software to function properly, the PC system must use the DOS (MS-DOS or PC-DOS) operating system, version 3.1 or later.

FUNCTIONAL DESCRIPTION

The OpenNET PC Link enables a PC system to be configured as a consumer workstation in the OpenNET network environment. This enables a PC system to access and share files and remote printers on a remote file server. After establishing a connection with a remote server, the user can access different directories by connecting drive letters at the PC system to the desired directories.

Creating a PC System Consumer Workstation

The PC system is easily configured as a consumer workstation on the OpenNET network. The following steps summarize how to configure the PC system for use as a workstation:

- Install the OpenNET PC Link network controller board in the PC system and connect the PC system to an Ethernet transceiver or Intellink™ module.
- Configure the OpenNET PC Link software to reflect the name and network address assigned to the PC system and each remote server system that the PC system will access.
- Define the PC system user as a valid user of the remote server system.

To connect with and access remote resources over the OpenNET network, perform the following steps:

- Invoke the PC system's consumer networking software.
- Execute a connect-to-server command.
- Execute standard DOS commands.

The PC system user can now access remote resources (files, directories, or printers) at remote servers on the network.

The user has the option of automatically connecting to a remote server each time the DOS operating system is booted. This is done by placing networking commands in a DOS AUTOEXEC.BAT file.

Remote Server Access

The PC user gains access to a remote server by connecting an unused drive letter at the PC system to a remote home directory at the server. The server validates the PC system user by comparing the user name offered in the connect-to-server command with the server's user definition file. If the name is valid, the user is logged on to the server, and can access any file within the home directory. Multiple subdirectories may be created within the home directory. The user is restricted from accessing directories or files located above the user's home directory.

Transparent Access to Multiple Directories

A PC system user may access multiple directories at a file server. This is done by defining multiple users (giving users access to different directories) at the server. After establishing a connection to the server, the user can access different directories by connecting drive devices at the PC system to the desired directories.

Data and resource sharing are implemented via transparent remote file access. This enables the user to work with remote data files and resources residing at server systems on the network as if they were resident on the PC system. Users of a remote server may be given access to the same home directory, enabling multiple users to access and share remote data files. The access rights of remote data files can be changed to enable all or some of the users to read, write, or delete files in that directory.

Using DOS Commands Across the OpenNET Network

Once the PC system has been connected to a remote server on the network, almost any DOS command can be used with remote files and directories. The exceptions are commands that manage physical devices (e.g., FORMAT). The MS-NET software reports an error message if an invalid DOS command is sent across the network. Using DOS commands, the user can manipulate drives, files, and directories as follows:

- Look at and list remote directories and files.
- Copy files back and forth between a PC system and a remote server.
- Redirect print requests to a remote printer.
- Set and reset the read-only attribute of remote directories and files.
- Map drive assignments to remote directories.
- Set the path to remote directories and files.

Shared Printer Access

The PC system can be linked to a remote printer that is connected to a server on the OpenNET network. This enables the user to take advantage of the remote printer services, thus freeing the user from having to install a printer at the PC system.

A PC system user can print local or remote data files by first connecting the PC system's logical printer device to the remote server's printer spool. Then, the MS-NET networking software command NET PRINT is used to print the file on the remote printer device.

Table 1. MS-NET Networking Commands

Command	Description
APPEND	Locates a file which is outside the current directory.
NET CONTINUE	Restarts the disk redirector or print redirector programs.
NET HELP	Displays a help file with information about MS-NET commands.
NET NAME	Displays the name assigned to your PC system.
NET PAUSE	Temporarily halts the disk redirector and print redirector programs.
NET PRINT	Prints a file on a remote printer.
NET START REDIRECTOR	Invokes the OpenNET PC Link consumer networking software.
NET USE	Connects a device at the PC system to a remote server or printer.

Microsoft Networks Software

The Microsoft Networks (MS-NET) networking software is included as part of the OpenNET PC Link software. The MS-NET software manages the transfer of information between the PC system and a remote server. Once a connection is made between the PC system and a remote server or printer, the user uses the MS-NET software (in conjunction with DOS commands) to access and manipulate remote files or printers. Table 1 presents a list of MS-NET commands and a description of each command.

The MS-NET networking software displays a message each time a command is successfully completed. If an error is made, the software displays an error message listing the probable cause of the error and suggestions for correcting it. On-line help files enable the user to quickly reference MS-NET commands and obtain the correct syntax for entering commands.

OpenNET PC LINK SPECIFICATIONS

Host Requirements

IBM PC AT or PC XT computer system

— 192 KB of system memory

— DOS (MOS-DOS or PC-DOS) operating system, version 3.1 or later

Physical Characteristics

NETWORK CONTROLLER BOARD

Width: 13.315 in. (33.82 cm)

Height: 4.15 in. (10.54 cm)

Weight: 35 oz. (0.99 kg)

SOFTWARE

5¼ inch double-density disk (360 KB)

POWER REQUIREMENTS

+5V at 2.7 Amps

+12V at 0.5 Amps

Environmental Characteristics

Operating Temperature: 0° to 55°C (32° to 131°F)

Operating Humidity: Maximum of 90% relative humidity, non-condensing

Convection Cooling

Documentation

166664 OpenNET™ PC Link User's Guide

Optional Equipment

The following items can be ordered for use with the OpenNET PC Link:

PCLNK20F Transceiver cable, 20 ft (6.1m)

PCLNK164F Transceiver cable, 164 ft (50 m)

DCM911-1 Intellink module

iMDX3015F Transceiver

ORDERING INFORMATION

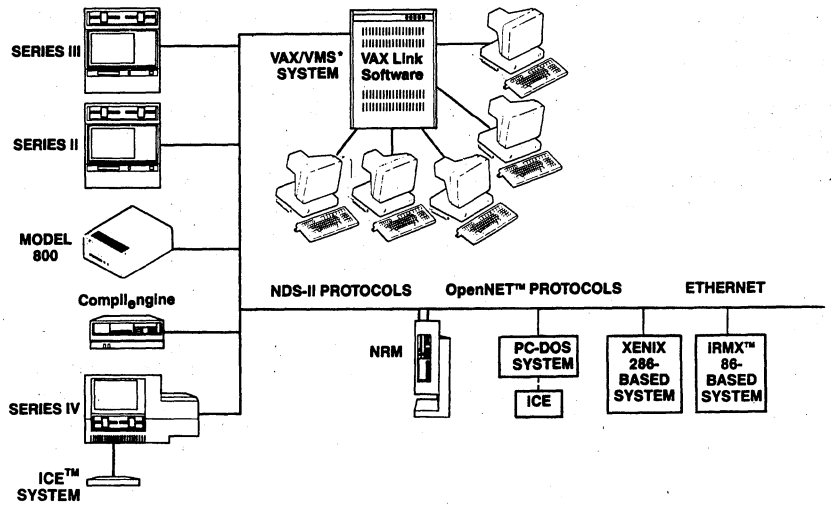
PCLNK OpenNET PC Link: Consists of a network controller board, a PC XT card support, software, and a user's guide

NDS-II/VAX* LINK NETWORKING SOFTWARE

- **Links VAX/VMS* to both NDS-II and OpenNET™ Development Environments**
- **Transfers Data via High-Speed Standard Ethernet/IEEE 802.3**
- **Enables File Transfer Between the NRM and VAX or MicroVAX II***
- **Offers VAX Users Access to All NRM File Services**
- **Optimizes Computing Resources with Distributed Job Control**
- **Authenticates User File Access Privileges for All Network Resource Manager (NRM) File Operations**
- **Requires a Digital Equipment Corporation DEUNA or DEQNA Communication Board for Operation (Not Supplied)**
- **Co-Exists with DECNET on the Same DEUNA or DEQNA Board**
- **Supports Multiple VAXs and Network Resource Managers (NRMs) in a Single Network**

NDS-II/VAX Link is an Ethernet-based communication link between Intel's Network Resource Manager (NRM) and a Digital Equipment Corporation (DEC*) VAX and MicroVAX II minicomputer. The NDS-II/VAX Link enables users to transfer files to/from the Series II/III/IV, Model 800, and the high-performance 286/310 based Compilengine. VAX users also have access to systems connected on the OpenNET network via the NRM. All data that is stored at the NRM is visible to, and can be accessed by, VAX users.

A major advantage of the NDS-II/VAX Link is its ability to optimize computing resources on the network via Distributed Job Control (DJC). DJC allows VAX users to queue jobs for remote execution upon the NRM. Similarly, NRM users can send jobs for remote execution upon the VAX. For example, CPU intensive jobs, such as compiles, can be sent from the VAX to idle Intel workstations for execution, saving valuable computational power for other activities. Or engineers using Intel development workstations can send special jobs to the VAX.



231299-2

Figure 1. NDS-II/VAX Link Enables High Speed Ethernet Data Transfers between the NDS-II, OpenNET™ and VAX Development Environments

NOTE:
All connections are on the same Ethernet cable.

*DEC, VAX, MicroVAX II and MicroVMS are trademarks of Digital Equipment Corporation.

August 1986
Order Number: 231299-003

NDS-II/VAX Link supports numerous commands that are initiated at the VAX. These commands are similar to Digital Command Language (DCL) commands and execute at the DCL command level. Users can obtain information on all commands by using the standard VMS* help facility. Commands cover general link operations (NVOPEN, NVCLOSE, NVLOGON, NVBYE, NVMESSAGE), distributed file system services (NVCOPY, NVDIRECTORY, NVCREATE, NVRENAME, NVDELETE, NVSET) and Distributed Job Control functions (NVCREATE/QUEUE; NVCREATE/IMPORT, NVEXPORT, NVCANCEL, NVSTATUS). Summaries of the more important commands follow below:

General Link Commands:

NVOPEN allows a VAX user with OPERATOR privilege to startup the link. NVCLOSE allows a VAX user to gracefully shutdown the link.

NVLOGON gives a VAX user access to the NDS-II files on a given NRM. The user must provide an NDS-II username and password. NVBYE logs a user off from a given NRM.

Distributed File System Commands:

NVCOPY copies a single file or a group of files from the VAX to the NRM or vice versa. The command accepts wildcard filename specifications and supports common sequential VAX/VMS file types.

NVDIR lists the directory entry of the NRM file(s) specified. The directory listing is in iNDX format. The user can request an expanded directory listing consisting of the filename, owner name, length, type, and owner and world access rights.

NVDELETE deletes one or more files or directories from the NRM file system. The invoker must have DELETE permission on each file or directory specified. A directory must be empty before it is deleted.

NVCREATE creates a new directory in the NRM file system. The invoker must have write access to the parent directory where the new directory is being created.

NVSET displays and/or changes the protection mask for files in the NRM file system. The user must have the appropriate access rights to the files in question when using the NVSET command.

Distributed Job Control Commands:

NVCREATE/QUEUE creates NRM queues. Queues must be created before they are used by either NVCREATE/IMPORT or NVEXPORT.

NVCREATE/IMPORT creates an import station on the VAX that serves the specified existing NRM queue. This is a privileged command that can only be executed by users having OPERATOR privilege.

NVEXPORT queues a job for execution in the NRM queue specified in the command parameters. The exported job will be executed by an import station serving the specified queue.

NVSTATUS lists NRM queues, the number of jobs waiting in the queues, and the number of import stations serving the queues. By specifying the /FULL qualifier the user can display detailed information on each job in the queue(s).

NDS-II/VAX Link supports up to 16 users on the link at a given time. With multiple users the link is operated in time-sharing fashion thus, giving each user the appearance of a dedicated connection to the NDS-II.

NDS-II/VAX Link also supports multiple VAXs and multiple NRMs in a single network. Users on separate VAXs can access the same NRM simultaneously, and users on the same VAX can access different NRMs. Multiple NRM support is only supported under VAX/VMS* version 4.0 (and later versions). Separate NDS-II/VAX Link software licenses must be purchased for each VAX connected in a multiple VAX/NRM environment.

NDS-II/VAX Link requires a DEC DEUNA-AA or DEQNA communication assembly that must be purchased from and installed by DEC. In addition, a standard external Ethernet transceiver cable is required to connect the DEUNA or DEQNA assembly to the Intel NDS-II Intellink™Module.

SPECIFICATIONS

Operating Environment:

REQUIRED HARDWARE:

NDS-II NRM or OpenNET NRM

DEC* VAX 11/730, 11/750, 11/780, 11/782, or 11/785, or MicroVAX II Minicomputer

DEC DEUNA-AA or DEQNA Assembly (from DEC)

Ethernet Transceiver Cable (Intel iMDX-457 or equivalent)

REQUIRED SOFTWARE:

iNDX Network Operating Software, Version 3.0 or later

VAX/VMS* or MicroVMS* Operating Software, Version 4.2 or later

Documentation:

NDS-II/VAX Link User's Guide (Order Number 122301-002)

Software Support:

This product includes a 90-day initial support consisting of subscription services and telephone hotline support. Additional software support services are available separately.

Future Update Kits are not covered under warranty and must be purchased separately.

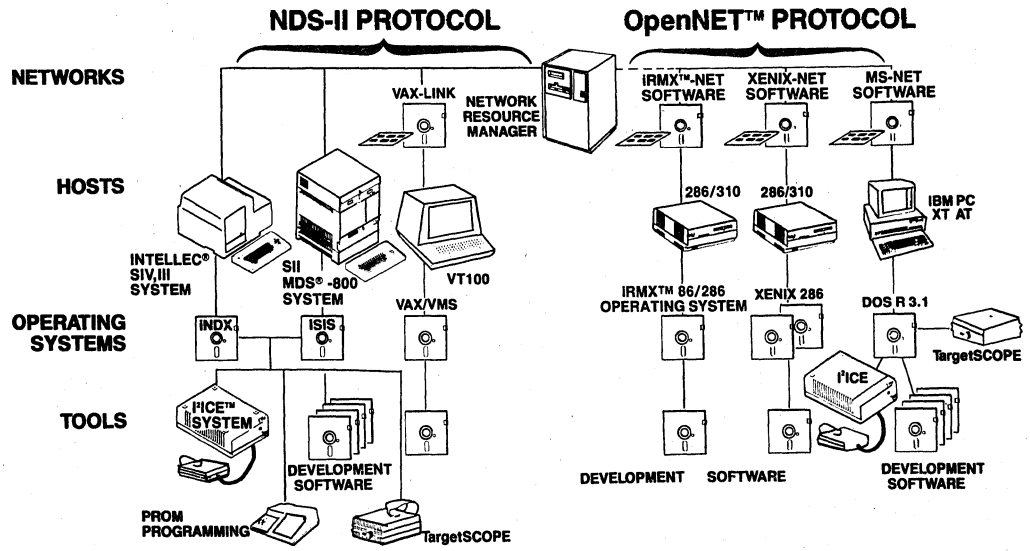
ORDERING INFORMATION

Part Number	Description
iMDX392	NDS-II/VAX* Link 9 track magnetic tape media
iMDX 393F	NDS-II/MicroVAX II Link RX 50 5¼" Floppy Media
iMDX 393T	NDS-II/MicroVAX II Link TK 50 Cartridge-tape Media

iMDX 555
NDS-II NRM OpenNET™ UPGRADE

- **Provides Series II/III/IV, Model 800 and VAX*/VMS Development Customers an Upgrade Path into the OpenNET™ Networking Environment**
- **NDS-II NRM now Becomes a File Server for OpenNET Users Including XENIX, iRMX™ 86, and PC-DOS Systems**
- **Supports Large Number of OpenNET Workstations up to the Physical Connection Limit for Ethernet (100 Direct Connections via Transceivers or over 800 via Intellink™) Boxes with the Ability for 30 to Simultaneously Access the NRM**
- **Transparent File Access to the NDS-II NRM from XENIX*, iRMX 86, and PC-DOS Systems**
- **Additional OpenNET Workstations Can Be Added without Reconfiguring the Network**
- **Authenticates OpenNET User File Access Privileges for all NRM Resources**
- **Shared Resources, e.g., Spooled Line Printer, up to 336 MB Winchester Central Disk Storage, Tape Archive**
- **Uses ISO 8073 Transport and Ethernet/IEEE 802.3 Standard Communication Protocols**

The NDS-II NRM OpenNET Upgrade contains software and hardware that allow the Network Resource Manager (NRM) to function as a file server in an OpenNET network environment. The Intellec® Series II/III/IV and Model 800 development systems, and Digital Equipment Corporation's VAX minicomputers can now share files residing on the NRM with XENIX, iRMX 86, and PC-DOS systems. XENIX, iRMX 86, and DOS system users can also take advantage of the NRM resources such as the spooled line printer, tape archive, and fast disks.



280141-1

NDS-II NRM Links Series II/III/IV and VAX/VMS to iRMX™ 86, XENIX, and DOS Systems Via OpenNET

*VAX is a registered trademark of Digital Equipment Corporation
*XENIX is a trademark of Microsoft
*IBM is a trademark of International Business Machines

October 1986
Order Number: 280141-002

FUNCTIONAL DESCRIPTION

The NDS-II NRM OpenNET Upgrade provides the capability for existing NDS-II users to expand into the OpenNET networking world. OpenNET network file access is based on protocols developed jointly by Intel, Microsoft, and IBM* to interconnect systems running different operating systems. This includes systems running XENIX, iRMX 86, PC-DOS, VAX/VMS, and now iNDX (on the NRM) operating systems. Alternatively, new NDS-II owners can integrate development tools running on XENIX, iRMX 86, and PC-DOS workstations, with the NRM file server.

These capabilities are achieved by combining the iNDX OpenNET software and OpenNET transport engine—the iSXM™ 552 board. The iSXM 552 board implements the industry standards ISO 8073 transport protocol (iNA 961) and Ethernet/IEEE 802.3 physical data link technology.

The NDS-II and OpenNET networks utilize separate communication boards (i.e., iSBC® 550 and iSXM 552 boards, respectively); therefore, there is no contention for communication resources.

Some limitations (e.g., formatting remote disks from a local workstation) do apply to iNDX OpenNET. See iNDX OpenNET User's guide for specific command support.

Number of Users Supported on the OpenNET™ Network

Each time an OpenNET workstation connects or "logs on" to the NRM (e.g., for file or printer access), a virtual circuit is created between the workstation and the NRM. iNDX OpenNET supports up to 30 simultaneous NRM users (virtual circuits) on the OpenNET side (one virtual circuit per OpenNET consumer node). When that limit is reached, no new circuit will be established until one of the existing circuits is closed. This means that although the number of physical OpenNET workstations on the network is potentially the limit of Ethernet connections (e.g., 100 via transceivers or over 800 via cascaded Intellink boxes), only 30 can access the NRM at any given time.

Because of this limit, iNDX OpenNET implements a least-recently-used algorithm to maximize virtual circuit availability. An idle OpenNET consumer (no outstanding file requests or log-ons) is automatically disconnected when the 31st network connection request is made. Alternatively, a workstation that is turned off without disconnecting from the NRM is automatically disconnected approximately 10 minutes after it is turned off.

Under normal file usage, the number of users on the OpenNET side should not affect the number of users on the NDS-II side (i.e., Series II/III/IV, VAX/VMS etc.)

NRM User Creation/File Access

The NDS-II NRM OpenNET Upgrade allows OpenNET workstations to transparently access files on the NRM without physically copying those files onto local disk. Files to be shared between NDS-II workstations (e.g., the Model 800, Series II/III, Series IV) and OpenNET workstations (e.g., PC-DOS, XENIX, iRMX) reside on the NRM.

Every OpenNET user is created on the NRM console with two simple commands:

- USERDEF DEFINE gives each user a unique log-on ID and home directory (it is mandatory to specify a home directory)
- CHPASS creates the user password.

These two commands allow the NRM Administrator (i.e., SUPERUSER) to control access by other OpenNET users to files as well as other resources (e.g., print spooler) residing on the NRM. Each time an OpenNET user attempts to access an NRM resource, an automatic log-on process occurs. Those users who do not have the proper log-on ID and password will be denied access to the requested NRM resource.

For file sharing at the NRM, users can be created with the same home directory. Then, the access rights of the home directory can be changed to allow all the users to read, add or delete files in that directory. The iNDX hierarchical protected file system supports owner and world access rights to files (READ, WRITE and DELETE) and directories (DISPLAY, ADD-ENTRY, and DELETE). These access rights may be set by the owner of the file/directory, or by the SUPERUSER (on the NRM). The SUPERUSER on the NRM terminal has access rights to all resources at the NRM as well as the authority to create and delete users.

iNDX supports concurrent read access to files and single write access to files (i.e., while a file is opened for writing by a user, no other user can read from or write to that file).

OpenNET™ Consumers

To become a valid network user under OpenNET, the user configures his/her OpenNET workstation

as an NRM OpenNET consumer. This involves performing three simple steps at the workstation:

- Adding the NRM's communication address in the workstation's session data base
- Activating the workstation's consumer network software
- Executing a connect-to-NRM command

This establishes connection between the workstation and the home directory specified during user creation at the NRM (with the USERDEF DEFINE command).

A user may access different directories residing on the NRM. This can be done by defining multiple NRM users (giving access to different directories) at the NRM console. After establishing a connection with the NRM, a user at that workstation can access different directories with different log-ons.

NRM Print Spooler Access

An OpenNET user can print files on the NRM by first connecting the workstation's printer logical device to the NRM print spooler (with the NET USE command). Then, local and/or network files are printed by using the workstation's local network print command.

For example, a DOS user would enter the NET PRINT command from the DOS workstation. A XENIX user, on the other hand, would simply use the RPRINT command with the specified NRM destination to print a file at the NRM. Alternatively, the XENIX user can use the copy command (CP) to copy local print files to the NRM print spooler (:SP:).

Once a file has been sent to the NRM print spooler, that print job can be cancelled with the DELETE command at the NRM terminal by the network administrator.

The OpenNET™ Communications Engine (iSXM 552 Board)

The iSXM 552 board integrates a high performance processor, the 80186, and a powerful Local Area Network (LAN) coprocessor, the 82586, on a single MULTIBUS® board. With the iNA 961 transport software, this solution set provides the implementation of the first 4 layers of the ISO OSI (7-layer) network communications model.

This Intel LAN solution offers reliability and easy expandibility. iNA 961 provides internal detection and correction of communication mediums and workstations so that a malfunction at a given point will not cause total network failure. Moreover, stations can be added or deleted from an existing network without reinitialization or reconfiguration of all other workstations.

OpenNET™ and Interconnection Hardware

Although the OpenNET connection requires a communications board (iSXM 552) different from NDS-II (DFS/ISIS) communications boards (iSBC 550 board), all existing Intellink™ boxes, transceivers, and cables remain the same. Furthermore, the same (one) Intellink can be used to connect Series II/III/IVs, VAXes, and OpenNET workstations, as long as cables from both the iSBC 550 and iSBC 552 boards are connected to that Intellink box. The Series II/III/IV's, VAXes will only respond to the messages from the iSBC 550 board while the OpenNET workstations will only understand the messages from the iSBC 552 board. This helps the user maximize usage of current interconnect hardware.

SPECIFICATIONS

Hardware Supplied

iSXM 552A OpenNET communication board
iSBC028A 128K RAM board
Internal cable assembly

Software Supplied

iNDX OpenNET software
iNDX Operation System

Operating Environment

- Hardware required:
  NDS-II NRM with 2 unoccupied MULTIBUS board slots
- OpenNET workstation connection requirements
  Personal Computer:

PCLNK	OpenNET PC Link hardware and software kit to connect the PC XT, PC AT, and compatible systems to the NRM via the OpenNET network; requires DOS 3.1 or higher

XENIX System:

XNXNETNRIKIT	OpenNET-XenixNET, iNA 961, iSXM 552 and Xenix-NET Pass-through Kit

RMX System:

RMXNETKITWRI iRMX Networking Software for a 286/310 system running the iRMX 86 R6.0 Operating System to connect to the NRM via the OpenNET network, SXM 552S.

VAX or MicroVAX running VMS:
—VAX OpenNET server s/w and controller board

- Documentation:

  NDS-II OpenNET User's Guide
  (order no. 138809-001)

ORDERING INFORMATION

Part Number	Description
iMDX 555	NDS-II NRM OpenNET Upgrade Package

iMDX-581
ISIS CLUSTER BOARD PACKAGES

- **Converts Spare Slots in Series II, III, IV, or Model 800 Workstations into Additional Workstations**
- **Up to Seven Additional NDS-II Workstations May Reside in One Development System Host**
- **Utilizes the Powerful ISIS-III(C) Operating System**
- **Supports 16-bit Development with Local ASM-86 and PL/M-86, and Via NDS-II Distributed Job Control**
- **Supports 8-bit Macroassemblers and High-Level Languages**
- **Supports all 8-bit ISIS-Based Software Development Tools Including the AEDIT-80, Text Editor, Program Management Tools, and NDS-II Electronic Mail**
- **Provides Execution Environment for 8085-Based Application Programs**
- **Compatible with a Variety of 9.6K or 19.2K Baud Terminals**

The ISIS Cluster Board Package is an NDS-II upgrade that cost effectively supports incremental software workstations on the network. Each Cluster board provides an 8085 CPU, 4K of ROM and 64K of RAM, and must reside in a Series II, Series III, Series IV, or Model 800 development system host. When attached to a user-supplied terminal, an ISIS Cluster workstation will boot onto the NDS-II and provide an ISIS environment which allows users to log on to the network and run Intellec®-supported 8-bit software, as well as "export" jobs to other network resources.

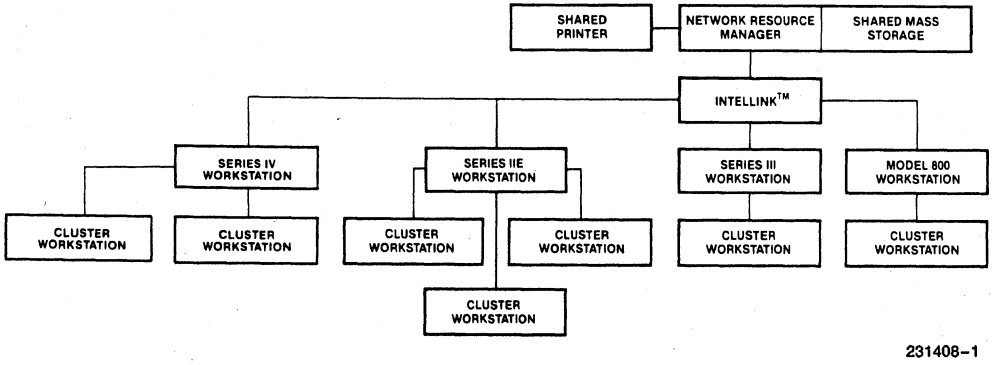


Figure 1. Example of an NDS-II Configuration

December 1985
Order Number: 231408-001

FUNCTIONAL DESCRIPTION

Summary: The ISIS Cluster board is a single-board computer centered around an 8085AH-2 CPU running at 4.0 MHz. 64K bytes of dual-ported RAM are provided on-board, along with 4K of ROM preprogrammed with a bootstrap program and self-test diagnostics.

The ISIS Cluster MULTIBUS® interface provides data and address interface latches. The serial I/O interface provides a full duplex RS232C serial data communications channel that can be programmed to handle serial data transmission at 19.2K or 9.6K baud. Software reset may be accomplished using the BREAK key on the terminal.

A block diagram of the ISIS Cluster board is shown in Figure 2.

Central Processing Unit

Intel's powerful 8-bit 8085AH-2 CPU running at 4.0 MHz is the central processor for the Cluster board. It is fully software compatible with all 8-bit ISIS-based languages and utilities which run on the Intellec Model 800, Series II/80, Series II/85, or Series IIE.

System ROM

4K bytes of non-volatile read only memory are included on the Cluster board using Intel's 2732A EPROM. Preprogrammed with the ISIS Cluster Boot program, the system ROM provides boot-up and diagnostic capabilities, and a generalized I/O system.

The Boot program communicates with the operator via an interactive console. Upon reset of the Cluster system, execution is handled by the bootstrap PROMs which overlay 4K bytes of system RAM, initialize Cluster board devices, run self-test diagnostic, and perform a communication handshake before prompting the user.

RAM

The Cluster board uses eight 2164 RAMs and a dual port RAM controller to provide 64K of dual-ported dynamic read/write memory. Slave RAM decode logic allows extended MULTIBUS addressing with a 1 Megabyte address space, so that RAM accesses may occur from either the Cluster board or from the network communication boards interacting via the MULTIBUS interface. Since on-board RAM accesses do not require MULTIBUS accesses, the bus is available for other concurrent operations. Dynamic RAM refresh is accomplished automatically by the Cluster board.

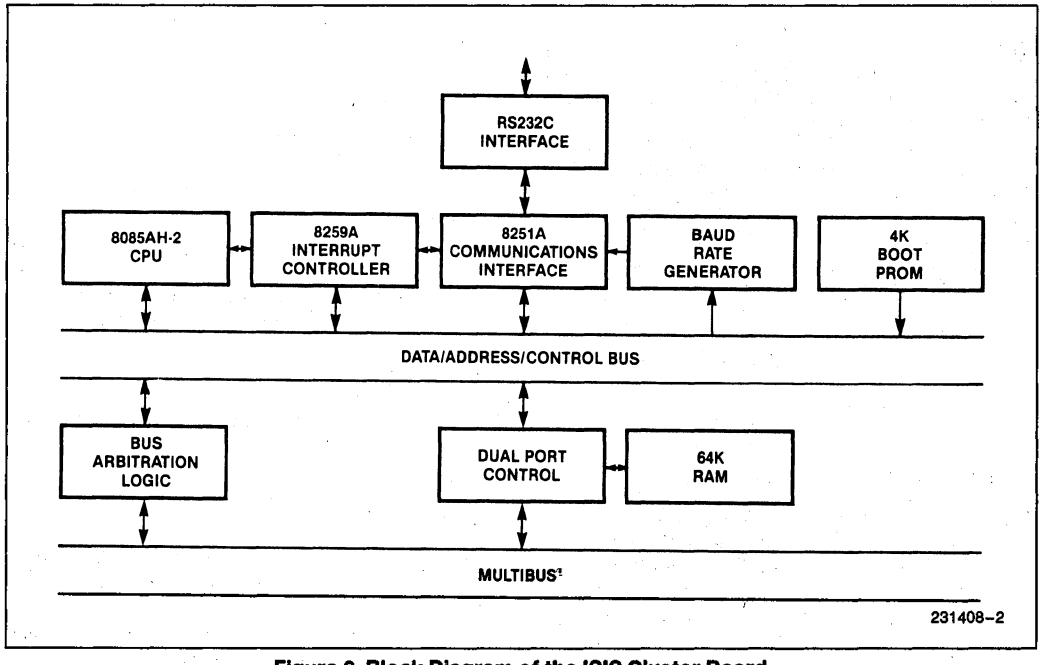


Figure 2. Block Diagram of the ISIS Cluster Board

Serial I/O

A programmable communications interface using the Intel 8251A USART (Universal Synchronous/Asynchronous Receiver/Transmitter) is on the Cluster board, and provides a full duplex RS232C serial communications channel. The transmit and receive lines are link exchangeable to enable a data set or data terminal to be used with the Cluster board. The board is pre-set for 9600 baud, but may be jumpered for 19.2K baud.

Programmable Timers

The interval timer capability is implemented with an Intel 8254 Programmable Interval Timer. The 8254 includes three 16-bit BCD or binary counters. The first two counters are not used. The output from the third counter is applied to the serial I/O interface and provides the baud rate frequency for serial communications.

Interrupt Controller

The Cluster board also includes an Intel 8295A Interrupt Controller. It is pre-configured with Interrupt 1 triggered by the BREAK key on the user-supplied terminal.

MULTIBUS® Interface

The Cluster board is a complete computer on a single board, capable of supporting a variety of 8-bit development tools. For applications requiring additional processing capacity, the Cluster board provides full MULTIBUS arbitration control logic. The bus arbitration logic operates synchronously with a MULTIBUS clock. All memory references made by the CPU refer to the on-board RAM. The Cluster board cannot access devices local to the host development system, but all of the shared network resources are accessible.

The Cluster board communicates with the Network Resource Manager via the MULTIBUS interface and the network communication board set in the host development system.

System Configuration

Each ISIS Cluster board requires one master slot in an Intellec cardcage. The host development system may be a Model 800, Series IV, Series II or IIE, or Series III or IIIE with an optional expansion chassis. A Series II or IIE with an expansion chassis will support a maximum of seven ISIS Cluster workstations, since the Integrated Processor Card and Network Communication boards occupy three of the ten cardcage slots. A Model 800 will support a maximum of 2 ISIS Cluster workstations, and Series IV workstation will support a maximum of 4 ISIS Cluster workstations. Each ISIS Cluster workstation counts as one additional network workstation, so the maximum number of Cluster workstations on a network is constrained only by the total number of users supported by the NDS-II Network Resource Manager. NDS-II iNDX Release 2.8 or later will support ISIS Cluster workstations in any Intellec development system host, including the Series IV.

Programming Capability

The Cluster workstation's ISIS environment supports all 8-bit Intellec-supported ISIS-based software, including the programmer-oriented AEDIT-80 text editor, PMT-80 Program Management Tools, NDS-II Electronic Mail, 8-bit macroassemblers, and PL/M, FORTRAN, PASCAL, and BASIC high-level 8-bit languages. 16-bit development is supported by the ASM86 cross assembler and the PL/M-86 cross compiler, or by "exporting" any 16-bit job to a 16-bit workstation for execution.

SPECIFICATIONS

CPU: 4.0 MHz 8085AH-2

MEMORY:

On-board RAM, 64K bytes, dual-ported

On-board ROM, 4K bytes preprogrammed with the ISIS Cluster Bootstrap Program

Interfaces

SERIAL I/O:	RS232C compatible, programmable interface
BUS:	MULTIBUS compatible, TTL level
TIMER:	3 programmable 16-bit BCD or binary counters, 1 used as baud rate timer
INTERRUPTS:	1 interrupt level available to user via the BREAK key on the terminal

Physical Characteristics

Two-sided printed circuit board fits into Intellec cardcage:

Length:	12 inches
Width:	6.75 inches
Depth:	0.062 inches

Internal flat ribbon cable connects ISIS Cluster board edge connector to the development system rear panel.

External 10-foot RS232C compatible cable connects the development system rear panel to a user-supplied terminal.

Electrical Characteristics

DC Power Requirements (from Mainframe)

V_{CC} = +5V, 4.5 Amps
V_{DD} = +12V, 25 mA
V_{AA} = −12V, 23 mA

Environmental Specifications

Operating Temperature: 0°C to 55°C
Humidity: up to 90%, without condensation

Documentation

iMDX-581 Installation, Operation, and Service Manual (#122293)

NDS-II ISIS-III(C) User's Guide Supplement (#122098)

Equipment Required

Recommended Terminals* (one per ISIS Cluster Board)

The following terminals have been tested and found to be interface compatible with the ISIS Cluster board; configuration files are provided for these terminals. Customers are advised to select terminals to meet their own environmental specifications.

Hazeltine, Model 1510
Televideo, Model 910+, 925, 950
Lear Seigler, Model ADM 3A
Adds Viewpoint, Model 3A+
Qume, Model 102
Zentec, Model ZMS-35, Cobra

*All of the recommended terminals run at 9.6K or 19.2K baud.

CAUTION: Other RS232C-compatible devices may not meet Intel environmental specifications, and could degrade overall system performance.

Host Development System (requires one open 6.75 x 12 in. master slot in system cardcage per ISIS Cluster board):

Series II/85 or Series IIE*
Series III or Series IIIE*
Model 800**
Series IV
*with optional Expansion Chassis
**supports maximum of 2 ISIS Cluster Boards

Workstation Upgrade Kit (one per host system):

iMDX-455 for Series II, III, or Model 800
iMDX-456 for Series IV

NDS-II Network Resource Manager with Hard Disk Mass Storage

Software Required

For Series II, III, or Model 800 Host:

NDS-II iNDX Operating System, Release 2.0 or later

ISIS-III(N)/III(C) Operating System, Version 2.0 or later*

For all Development System Hosts, Including Series IV:

NDS-II iNDX Operating System, Release 2.8 or later

Series IV iNDX Workstation Operating System, Release 2.8 or later**

ISIS-III(N), version 2.2 or later**

ISIS-III(C), version 2.2 or later**

*included with NDS-II Release 2.0
**included with NDS-II Release 2.8

ORDERING INFORMATION

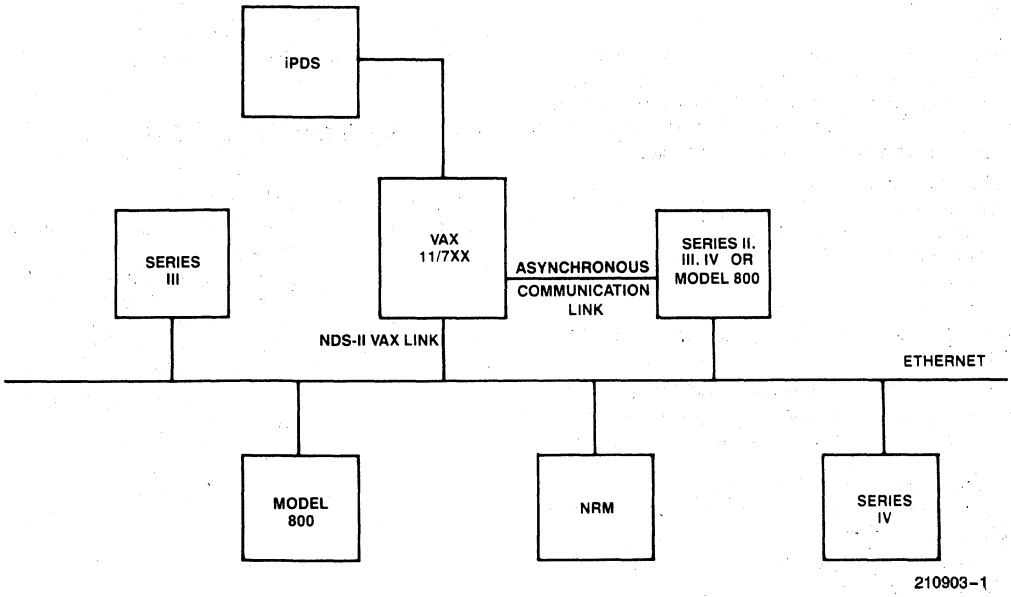
Part Number	Description
iMDX-581	ISIS Cluster Board Package for Series II, Series III, Series IV, or Model 800—includes processor board, cables, and documentation. Must be installed on NDS-II in a Model 800, Series II, Series III, or Series IV workstation and connected to a user-supplied terminal.

INTEL ASYNCHRONOUS COMMUNICATIONS LINK

- **Communications Software for VAX* Host Computer and Intel Microcomputer Development Systems**
- **Compatible with VAX/VMS* and UNIX† Operating Systems**
- **Supports Intel's Model 800, Intellec® Series II, Series III, Series IV and iPDS™ Microcomputer Development Systems**
- **Supports NDS-II Workstations**
- **Allows Development System Console to Function as a Host Terminal**
- **Operates through Direct Cable Connection or over Telephone Lines**
- **Software Selectable Transmission Rate from 300 to 9600 Baud**

Intel's Asynchronous Communications Link (ACL) enables Intel microcomputer development systems to communicate with a Digital Equipment Corporation VAX family computer. The link supports Intel Model 800, Intellec Series II, Series III, Series IV or iPDS development systems. Programmers can use the editing and file management tools of the host computer and then download to the Intel microcomputer development system for debugging and execution. Programmers can use their microcomputer development system as a host terminal and control the host directly without changing terminals.

NDS-II Example



210903-1

NOTE:
NDS-II VAX Link is an Ethernet Link between VAX*/VMS and Intel's Network Resource Manager. This product is also available from Intel.

*VAX and VAX/VMS are trademarks of Digital Equipment Corporation.
†UNIX is a trademark of Bell Laboratories.
††VADIC is a trademark of Racal-Vadic Inc.

October 1986
Order Number: 210903-003

FUNCTIONAL DESCRIPTION

The Asynchronous Communications Link (ACL) consists of cooperating programs: one that runs on the VAX computer, and others that run on each microcomputer development system. The development system programs execute under the ISIS-II or ISIS-III(N), ISIS-IV, ISIS-II(W), or ISIS-PDS operating system. They invoke the companion program on the VAX-11/7XX, which runs under either the VAX/VMS or UNIX operating system.

The link provides three modes of communication: on-line transmission, single-line transmission, and file transfer. In on-line mode, the development system functions as a host terminal, enabling the programmer to develope programs using the host computer's editing, compilation, and file-management tools directly from the development system's console. Later, switching to file transfer mode, text files and object code can be downloaded from the host to the development system for debugging and execution. Alternatively, files can be sent back to the host for editing or storage. In single line mode, the programmer can send single-line commands to the host computer while remaining in the ISIS environment.

The user can select transmission rates over the link from 300 to 9600 baud. The link transmits in encapsulated blocks. The receiver program validates the transmission by checking record-number and checksum information in each block's header. In the event of a transmission error, the receiving program recognizes a bad block and requests the sender to retransmit the correct block. The result is highly reliable data communications.

SOFTWARE PACKAGE

The Asynchronous Communications Link Package contains either a VAX/VMS or UNIX compatible magnetic tape, a single 8", double 8", Series-IV 5¼", and PDS 5¼" diskette compatible with the Intellec development system, and the *Asynchronous Communications Link User's Guide* containing installation, configuration and operation information.

HARDWARE CONNECTION

The Link sends data over an RS232C cable. The communication line from the host computer connects directly to a development system port.

TELECOMMUNICATIONS USING THE LINK

The ACL is ideal for cross-host program development using a commercial timesharing service. This configuration requires RS232C compatible modems and a telecommunications line. Depending on the anticipated level of usage, wide-area telephone service (WATS), a leased line, or a data communications network may be chosen to keep operating overhead low.

NDS-II ACCESS USING THE LINK

The ACL is ideal for interconnecting VAX host computers with NDS-II. This configuration requires that an NDS-II workstation be connected to the VAX host computer using the RS232C interface and to NDS-II using the Ethernet interface.

All three modes of communication operate identically on NDS-II. In the on-line mode, the development workstation operates as a host terminal, and concurrently, as an NDS-II workstation. It is an easy transition between the VAX and ISIS operating system environments as LOGON/LOGOFF sequences are not required to re-enter environments.

In file transfer mode, text and object files can be transferred from the VAX directly to the Winchester Disk at the NRM without first copying the files to the workstation local floppy disk. Similarly, files residing on the NDS-II Network File System (the Winchester Disk at the NRM) can be transferred directly to the VAX without using local workstation storage.

Using the EXPORT/IMPORT mechanisms of NDS-II, a network workstation which is not directly connected to the VAX can cause files to be transferred between the VAX and NRM. For example, any NDS-II workstation can "EXPORT" ACL commands to

another "IMPORT"ing NDS-II workstation which is physically connected to a VAX. The "IMPORT"ing workstation executes the ACL command file causing the desired action to occur.

VAX ACCESS USING THE LINK

Users who want multiple workstations concurrently operating as VAX terminals (ONLINE mode) must physically connect each workstation to the VAX. However, users who want multiple workstations to be able to upload/download files, for example, must only physically connect one workstation to the VAX. By using the EXPORT/IMPORT mechanism of NDS-II as described above, the user can have multiple workstations accessing the VAX using only one connection.

SPECIFICATIONS

Software

Asynchronous Communications Link development system programs

VAX/VMS or UNIX companion program

Media

Single- or double-density ISIS 8" and Series-IV, iPDS 5¼" compatible diskette

600-ft. 1600 bpi magnetic tape, VAX/VMS or UNIX compatible

Data Transfer Speeds

All systems up to 9600 bps

Online Terminal Mode Speeds

Series II, Series III, Series IV — 2400 bps max PDS — 9600 bps max

Model 800 — equal or less than the Terminal speed

Manual

Asynchronous Communications Link User's Guide Order No. 172174-001

Required Host Configuration

VAX-11/7XX running VAX/VMS (Version 4.1 and later) or fourth Berkley distribution of UNIX 4.2

Required Intel Development System Configuration

Model 800, Series II, Series III, Series IV, or iPDS under ISIS

Required Connection

RS232C compatible—cable 3M-3349/25 or equivalent; 25-pin connector 3M-3482-1000 or equivalent

Recommended Modems for Telecommunications

300 baud—Bell 103 modem; VADIC†† 3455 modem or equivalent

1200 baud—Bell 202 modem; VADIC 3451 modem or equivalent

9600 baud—Bell 209A (full duplex, leased line) or equivalent

NOTE:
Since one of the two Model 800 ports uses a current loop interface, Model 800 users need a terminal or modem that is current loop compatible, or a current loop/RS232C converter.
The model 800 might require modification by a qualified hardware technician. Intel does not repair or maintain boards with these changes.

ORDERING INFORMATION

Product Name

Asynchronous Communications Link

Ordering Code‡

iMDX 394 for VAX/VMS systems

iMDS 395 for UNIX systems

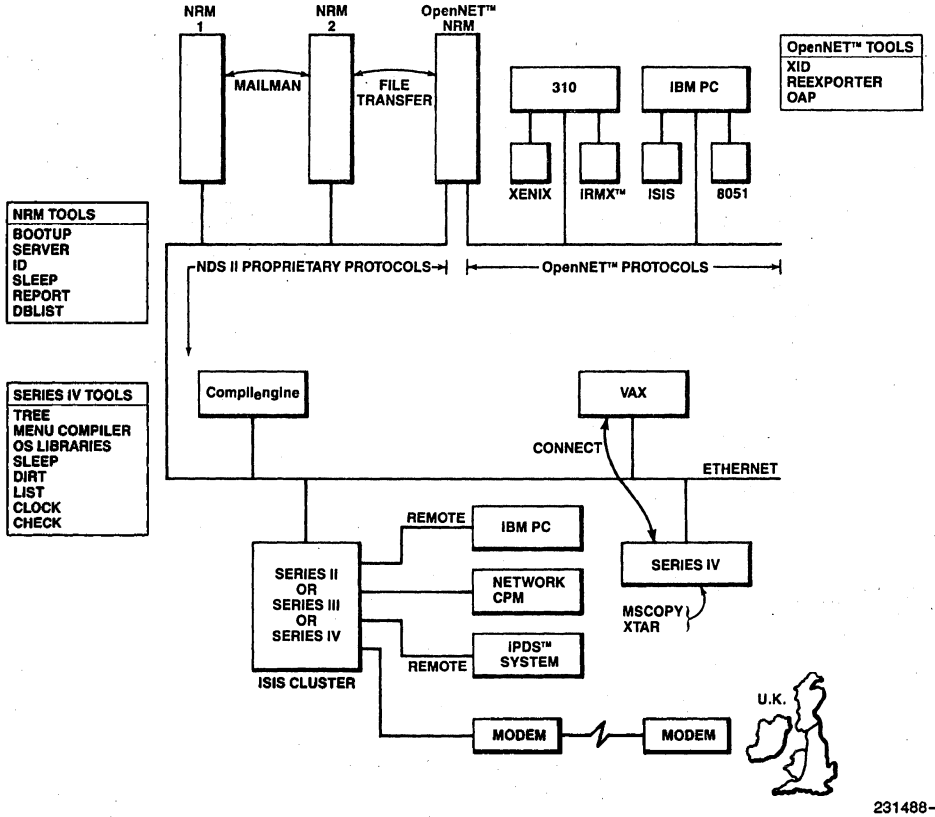
‡ See price book for proper suffixes for options and media selections.

NDS-II/Series-IV/OpenNET™ Toolbox

- **Multiple NRM Communication**
- **Remote Series-IV from VAX* Terminal**
- **Series-IV Menu Compiler**
- **MS-DOS*/Series-IV Disk Read Utility**
- **XENIX Services for Any Workstation that can Access an OpenNET™ NRM**
- **Access to NDS II DJC for OpenNET™ Workstations**
- **Allow 8080 Based Intel Tools on 8086 Family Systems**

The NDS-II/Series-IV/OpenNET Toolbox is a software only product that contains valuable collection of tools developed for the NDS-II, Series-IV and OpenNET user. These tools have been designed to make hybrid development system environments work together and to more fully automate the software developer's task. Many tools are provided with source to allow the engineer to customize these products to their own environment.

Note: However, this is not a supported product.



231488-2

Example of the Many Possible Connections Available with NDS-II/Series-IV/OpenNET™ Toolbox

*MS-DOS is a trademark of Microsoft Corporation
*VAX is a trademark of Digital Equipment Corp.
CP/M® is a registered trademark of Digital Research Inc.

May 1986
Order Number: 231488-002

CONNECT

CONNECT allows software developers to use their VAX terminal as a virtual terminal for their Series-II or Series-IV work station. Software developers can now run PSCOPE, ICE™ and I²ICE™ emulators from their VAX terminal, eliminating the need to switch terminals when debugging a program. This serial communications based program runs at 9600 baud for the Series-II and 2400 baud for the Series-IV. Complete support of the Series-IV menu line is available on the VAX terminal. CONNECT does not provide file transfer capability, this is provided for in either the VAX Link or ACL products. A separate serial cable, not supplied with Toolbox, is required for connecting the development system to the VAX. Source and generation are provided.

NRM to NRM Communications

The NRM to NRM communications package provides file transfer and printer spooling from one NDS-II network to another via ethernet. Two new commands are provided, NNCOPY and NNDIR, for Series-IVs running iNDX version 2.5 or greater. These commands do not function on the MDS-800 development system, ISIS Cluster, Series II, or Series III; although an ISIS work station may use export to run NNCOPY or NNDIR remotely. Full file protection is provided by this application. This product also requires that the NRM terminal run the slave program, NNL. The system administrator can prevent access to the NRM from remote systems by not executing NNL.

TREE

TREE is a program for the SIV or NRM that provides: ARCHIVE over the network, listing of a directory tree, searching a directory tree for a specified file, deletion of an entire directory tree, wildcard deletion of files from a directory tree, or displaying the total disk space used by a particular user or directory tree. Commands provide for OWNedby, MODIFIED-BEFORE or SINCE controls.

MENU COMPILER

Allows users of the Series IV or NRM to modify the command level menu to include their own commands or to remove commands not often used. Source for the current Series-IV menu line is provided as well as the additional information needed to add Toolbox commands. Menu compiler input is provided in the form of an LL1 parse tree which will require some knowledge of compiler technology to modify.

MSCOPY

MSCOPY is an iNDX utility that allows manipulation of an MS-DOS disk on a Series IV or NRM. Using this program, the Series-IV or NRM can read and write MS-DOS files. Source and generation are provided.

NETWORK CP/M-80

Network CP/M is a package that allows a Series-II or ISIS cluster to run CP/M®-80 and use the NDS-II as a remote file server. A separate license is required for CP/M on each work station. This package is only an interface that allows to use the NDS-II as a file server, the CP/M operating system is not provided. CP/M is available separately as Intel part number SD01CPM80-B-SU. Source is provided for utilities only.

NETWORK CP/M UTILITIES

CP/M	— loads Network CP/M onto the Series II or ISIS cluster.
MAKDSK	— creates a blank Network CP/M disk image.
CDIR	— gives directory of a Network CP/M disk image or CP/M-80 diskette in drive 1 of a Series II.
ADDSYS	— adds CP/M OS to disk image A: created using MAKDSK.
CCOPY	— allows an ISIS user access to CP/M files.
CPMOMF	— converts a program developed under ISIS to a CP/M executable program.
SUCPM	— SUPERUSER facility for CP/M.

BOOTUP

BOOTUP allows an iMDX-580/581 ISIS cluster board to be used in any SBC chassis instead of only a microcomputer development system. BOOTUP is a special monitor PROM which is installed on a standard ISIS cluster board. This board is then installed into any SBC system chassis to provide a diskless work station. The cluster board accesses the NDS-II file system via an iSBC®550 communication controller also installed in the system chassis. Additional ISIS cluster boards may be installed in the same chassis to provide for more users instead of using a Series-II, III, or IV. Up to eight clusters can be used in a single system chassis.

NOTE:
Only object files are provided, the customer must provide his own 2732A PROM. Object files are provided for all formats of Intel PROM programmers.

SERVER

SERVER allows an ISIS Cluster to automatically logon to the NDS-II network by supplying a username and password from PROM. ISIS will then execute the corresponding initialization file (:f9: ISIS.INI). A useful application of SERVER is to provide additional spooled printer capability to the network by executing PRINCE, another Toolbox application, in an infinite loop from the ISIS initialization file. Some source and generation are provided. All object files are supplied.

PRINCE

PRINCE is an ISIS based spooling program for use with a Series-II, III, IV, or ISIS cluster board in either the stand-alone or networked environments. PRINCE provides support for both parallel and serial printers, including complete XON/XOFF or DTR/DSR printer ready protocols. PRINCE is most effective when used with an ISIS Cluster board and the SERVER PROM. The program features extensive logging capabilities. Source and generation are provided.

PRMSLO

PRMSLO is a PROM image for an ISIS Cluster that sets the default baud rate to 300 or 1200 baud. This enables the cluster board to be used with a modem. Object files only supplied.

UDXCOM.LIB

System library for iNDX specific UDI extensions. This library provides support for MULTIBUS® hardware and software interrupt calls, enable/disable interrupts, read directory expanded, and more. Object code and documentation are provided for this library.

OSXCOM.LIB

System library for internal iNDX operating system extensions. This extensive internal system library provides many system level calls, such as create directory, enable/disable break, change access, change owner, change password, MIP communication calls, and many more. Object code and documentation are provided for this library.

BVOSX.LIB

This library provides operating system support for C language programs in the SMALL model. Normally the programmer would use OSXCOM.LIB and the COMPACT model of compilation. The functions in BVOSX.LIB are the same for the corresponding calls in OSXCOM.LIB, although not all functions are provided. Source and generation are provided.

BVCLIB

BVCLIB is a useful set of C language functions contained in the libraries BVCSLB.LIB and BVCLLB.LIB. BVCSLB.LIB is SMALL model, and BVCLLB.LIB is large model. Functions included are: parse, wmatch, strtok, valid, creat, open, read, write, seek, close, conn_num, str_to_uppercase, str_to_lowercase, plm_to_c_str, c_to_plm_str, err_chk, mark_end. All references to strings are assumed to be C format strings. Source and generation are provided.

SLEEP

This program puts a Series-IV or NRM to SLEEP for the time specified in the (time) parameter. SLEEP can be used in a submit file to execute a program at a certain time. For example, automatically archiving at midnight and then returning to sleep until the next day at midnight and repeating the archive. Source and generation are provided.

ID

ID is an iNDX utility that lists the name of the current user to the current console. It is useful if you forget who you are or need to know who is executing a particular submit file (MAILMAN is a good example of this). Source and generation are provided.

MDS-800 FPORT

INIT800.86 and FPRT are iNDX and ISIS utilities that allow file transfer between an MDS-800 development system and Series-IV over a serial line. Requires S4FPRT.86 (supplied standard with the Series-IV). Source and generation are provided.

DBLIST

DBLIST is an ISIS utility that enhances the operation of the SVCS programming tool set. It can list the entire SVCS database to an output device and may be used to remove deleted variants from a data base directory. Source and generation are provided.

REMOTE Communication with iPDS, Series-II, III, IV

This program gives the remote iPDS, Series-II, III, or IV user complete access to an NDS-II system through an ISIS Cluster board; including file upload and download capability. The program is menu driv-

en and includes: serial channel select, 8253 clock select, break-key select, baud rate select, modem present/not present select, dial/touchtone select, add-to-out call list option. Source and generation are provided.

REMOTE Communication with IBM PC running MS/DOS

This program enables an IBM Personal Computer running MS/DOS to act as a dumb terminal for an ISIS Cluster board connected to an NDS-II network. The ability to upload and download files from the PC to the network is supplied. Source and generation are provided.

REPORT

REPORT is an ISIS utility that reports back on the status of a job that has been EXPORTED to the NDS-II network for execution on a remote job station. The user can add messages to the command file at appropriate positions in the job sequence, and these messages are returned to the ISIS user when encountered. Source and generation are provided.

DIRT

DIRT is an iNDX utility which provides a directory listing with time and date of file creation and modification. Source and generation are provided.

VIEWPASS

VIEWPASS is an iNDX utility provided exclusively for the SUPERUSER. It lists all the usernames on the system, their associated passwords, and their id number. Source and generation are provided.

FDUMP

FDUMP is an iNDX utility that is used to print the contents of a file on the console in one of four possible formats: HEX, BINARY, OCTAL, or DECIMAL. The default is HEX if no option is specified; all formats include a display of the file in ASCII (reverse video on the Series-IV). Source and generation are provided.

CLOCK

CLOCK is a desk clock for use when you have nothing else to worry about. CLOCK displays the current system time on the console of a Series-II, III, IV (iNDX) or ISIS Cluster. Eight and sixteen bit versions are supplied for ISIS and iNDX systems. Source and generation are provided.

IFILES

IFILES is an ISIS-III (N) utility used to identify date/time stamped files in a directory. All of the files that conform to the defined specification will be listed in a savefile. This file can further be used in command files for manipulating the identified files. Source and generation are provided.

LIST

LIST is a utility that copies files to the system printer (:SP: or :LP:). LIST has the following features as enhancements over a normal copy to :SP:.

1) No form feed at the very beginning of a file.
2) Assumes '.LST' for an extension if none is given.
3) Supports multiple copies.
4) Supports multiple files.
5) Supports page breaks.
6) Supports printing of the filename at the beginning of the listing.
7) Converts tabs to spaces if necessary.

Source and generation are provided for ISIS and iNDX versions.

TA

TA is an ISIS based type-ahead utility for the Series-II/III. TA provides a 255 character type-ahead buffer on the Series-II/III. TA requires the iMDX-511 enhanced IOC upgrade, available on most systems manufactured after 1983. Source and generation are provided.

MAILMAN

MAILMAN is an extensive command file that supports multiple network electronic mail when used with more than one NRM and NRM to NRM communications. Source is provided.

CHECKEXIST and CHECKTIME

CHECKEXIST and CHECKTIME are iNDX utilities used to assist the automation of iNDX command files. CHECKEXIST provides a true or false system variable (%status) depending upon the existance of a specified file. A following check of %status within the command file will control the flow of the command file based upon the existance of the specified file. CHECKTIME provides a greater or less than %status by comparing an input time with the system clock for conditional execution of commands in the command file at specified times. Source and generation are provided.

XID

XID, (pronounced "zid") the (X)enix (I)mport (D)aemon, provides XENIX services for any workstation that can access an OpenNET NRM. Thinking of it in another way, XID provides yet another resource for NRM users; a resource much like a spooled line printer or mass storage. In this case, the resource provided is "any job or service that a XENIX box can do; you, as an NDS-II user can gain access to". Source and generation are provided.

REEXPORTER

Reexporter is an iNDX utility that allows OpenNET users (PC's, Xenix, iRMX Systems) to execute batch jobs on NDS-II systems (i.e., VAX/VMS, Model 8001 Series II, III, IV). The utility will execute on a Series-IV, Compilengine or the NRM itself. In brief, it scans special user directories on the NRM looking for command files. If a command file is found, it re"EXPORT"s the command file to a DJC job queue. A log file is generated to allow the OpenNET user to check the success/failure of the job. Source and generation are provided.

XTAR

XTAR is a program that will let you manipulate a XENIX tar diskette at a Series IV. XTAR works only with disks formatted by the /dev/dvf0 device driver on a 286/310 box, or with the /dev/fd048ds96 device driver on a PC/AT. This version will not handle files physically bigger than a single flippy (367104 bytes). Source and generation are provided.

ISIS

The ISIS environment is designed to allow 8080 based Intel tools (such as ASM, PLM LINKER/LOCATOR) to run on an 8086 family system, either iRMX or PCDOS based system. The ISIS environment does not support all ISIS calls, but is sufficient to run 8051 translators and utilities. Hosting ISIS on Xenix-286 systems is possible and installation instructions are included. All object files are supplied.

OAP

OAP is a utility that for security reasons masks the username and password in the PC-Link net use command for increased security. The utility also displays all available servers, by looking at the NETADDR file. Source and generation are provided.

SPECIFICATIONS

Operating Environment

ISIS, iNDX, RMX, XENIX, or PC-DOS operating system. Check description of each tool for specific requirements.

Documentation

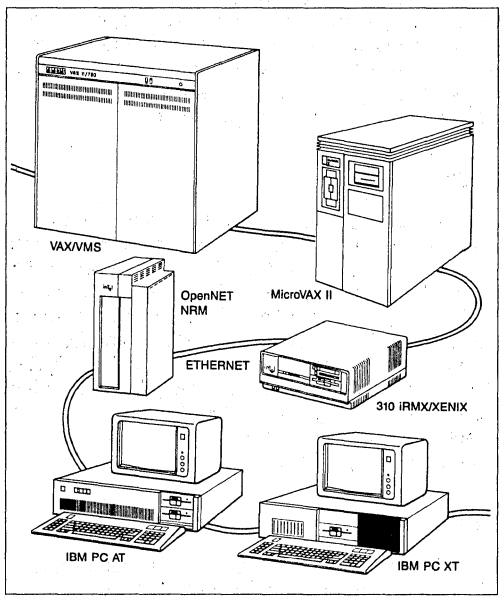
"NDS-II/Series-IV/OpenNET Toolbox" (122336)

ORDERING INFORMATION

NDS2 TLB	NDS-II/Series-IV/OpenNET Toolbox

VAX/VMS* NETWORKING SOFTWARE
Member of the OpenNET™ Product Family

As a member of Intel's OpenNET™ family of network software, VAX/VMS* Networking software (VMSNET) lets you connect a VAX or MicroVAX II* system to other OpenNET systems. This includes the IBM PC AT, PC XT, Intel's OpenNET NRM (Network Resource Manager), NDS-II NRM (with the OpenNET upgrade kit installed), iRMX®, and XENIX* systems. VMSNET enables a (Micro)VAX system to be configured as a Server System on the OpenNET network, thus allowing any OpenNET Consumer workstation (iRMX, XENIX, MS-DOS) to transparently access files residing at remote (Micro)VAX systems. In addition, VMSNET supports bidirectional file transfer initiated from a (Micro)VAX to all other OpenNET servers.



Product Highlights

- Connects a VAX and MicroVAXII to the OpenNET Network
- Interoperation between VAX/VMS and MS-DOS, iRMX, XENIX, and iNDX systems over a Local Area Network (LAN)
- Conforms to the ISO-OSI networking standards
- Adheres to ISO 8073 Transport and Ethernet/IEEE 802.3 Standard Communication Protocols
- Uses 80186/82586 Processor-based Unibus and Qbus Network Controller Boards
- All data stored at the (Micro)VAX is visible to, and can be transparently accessed by, all consumer workstations on the OpenNET network
- Enables high speed file transfer/file copy between the (Micro)VAX and OpenNET workstations
- Compatible with DECnet*

OpenNET Overview

Intel's OpenNET product family incorporates a set of system and component level LAN products covering all seven layers of the ISO (International Standards Organization) Open Systems Interconnect (OSI) model, and the protocols on which they are based. OpenNET protocols are, whenever possible, established industry standards for each function. Therefore, OpenNET network products can interconnect and interoperate not only with each other, but with the other vendors' ISO-OSI based LANs. An OpenNET network provides a high level of interoperability between heterogenous systems: MS-DOS, VMS, iNDX, XENIX, and iRMX operating system versions are available. Thus, users can tailor their networks to meet their specific needs by incorporating any combination of these diverse systems.

*XENIX is a trademark of Microsoft Corporation. VAX/VMS, MicroVAX II, DECnet are trademarks of Digital Equipment Corp.

ORDER NUMBER 280329-001

Physical Description

The VAX/VMS Networking Software package consists of the appropriate network controller board and the software necessary for the (Micro)VAX to communicate over the OpenNET network. The following sections describe the hardware and software components of VMSNET.

VMSNET Hardware

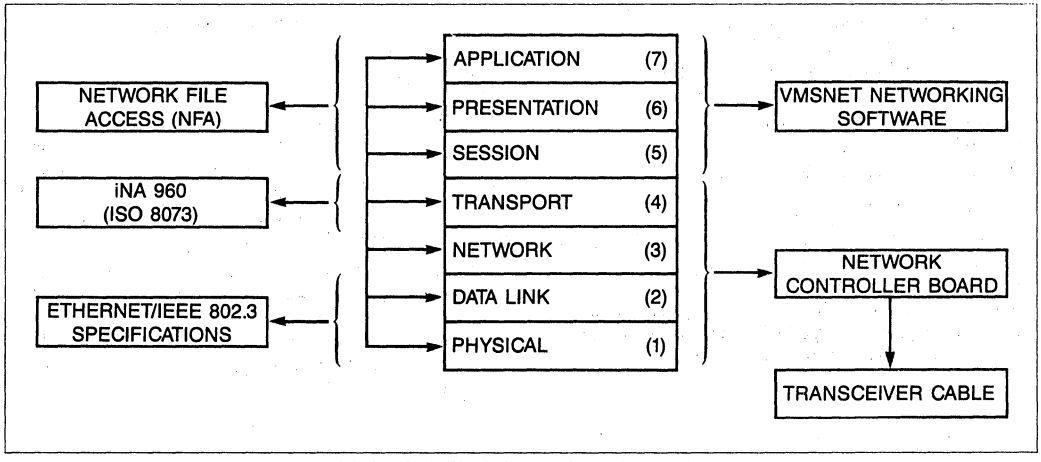
VMSNET comes with one of two types of Ethernet controller boards: a Unibus* board for the high-end VAX or a Qbus board for the MicroVAXII system. Both boards implement the industry standard ISO 8073 transport protocol and Ethernet/IEEE 802.3 physical data link technology. Both boards are high performance, intelligent communications controllers featuring onboard, dedicated Intel 80186/82586 processors which support layers 1 through 4 of the ISO OSI Reference Model. Thus, the Unibus and Qbus* boards perform the CPU tasks associated with lower layer LAN communications protocols, thereby freeing the (Micro)VAX host CPU to concentrate on applications requirements.

Power-up, self-test diagnostics are resident on both the Unibus and Qbus controller. Extended host resident diagnostics are also provided which can be loaded onto the boards to aid in problem resolution. In addition, appropriate internal cables, and chassis mounting hardware are included.

VMSNET Software

The software is supplied on either a 9 track magnetic tape (for high-end VAXs) or on both a TK50 cartridge tape and RX50 5¼-inch disk (for MicroVAXIIs). The following software components are included as part of the VAX/VMS networking software:

- A specially configured version of iNA 960 transport layer software which operates on the network controller boards
- A VMS interface driver which enables VMS programs to access the network controller board
- An implementation of the Network File Access (NFA) protocols (jointly developed by Intel, IBM, and Microsoft) which enables (Micro)VAX users to interoperate with other nodes on the OpenNET network



ISO-OSI VAX/VMS OpenNET Implementation

OpenNET, iRMX are trademarks of Intel Corporation.
* Unibus and Qbus are trademarks of Digital Equipment Corporation.

Functional Description

Transparent File Access

VMSNET provides transparent remote file access capability to the (Micro)VAX through a file server module. The server receives, interprets and executes the command acting as a user to its local file system. Consequently, a PC, iRMX, or XENIX user can work with data files and resources residing at the VAX as if they were resident on his/her system.

File Transfer

VMSNET also provides a set of file transfer utilities that allow (Micro)VAX users with the ability to transfer files that reside on other OpenNET server nodes to the (Micro)VAX or vice-versa. These utilities include copying files, deleting files, listing directories, and a help facility.

DECnet Access

VMSNET will allow consumer access to a file residing on DECnet nodes. The only protocol restriction is that the server will not allow file locking or compatibility mode opens on DECnet file access. The consumer may use logical names to define DECnet pathnames. For example, if "dev" is defined in login.com with an equivalence string of "isodev" user mypasswork"::dra 1[user]", the consumer can use "dev" as the first pathname component; the server will automatically use DECnet for the file access:

- net use vms //vms/user mypasswork
- lc //vms/dev
- cp //vms/dev/test.obj/usr/bin

Network Management

A set of network management utilities provide (Micro)VAX users with information and statistics of VMSNET along with the capability to control the execution of the VMSNET server and file transfer utility. To invoke the network utility, the user simply needs to type "NET" in response to the DCL (Digital Command Language) prompt.

Host Requirements

- VAX 750, 780, 782, 785
- VAX 8xxx family
- MicroVAXII
- (Micro)VMS operating system, version 4.2 or later

Physical Characteristics

Software

1. 9 track 1600 bpi magnetic tape
   or
2. TK50 cartridge tape and RX50 5¼-inch disk

Power Requirements

Unibus controller:	+5 vdc (±5%) at 4.5 amps typical, 6 amps maximum −15 vdc (±10%) at .5 amps, 3 amp surge
Qbus controller:	+5 vdc (±5%) at 6 amps typical +12 vdc (±10%) at .5 amps, 3 amp surge

Environmental Characteristics

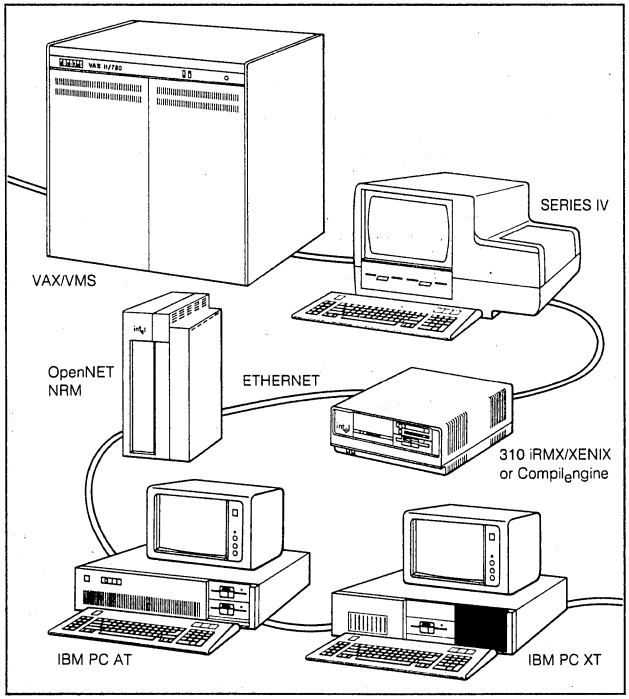
Operating Temperature:	0° to 50°C (32° to 122°F)
Operating Humidity:	Maximum of 90% relative humidity, non-condensing
	Forced air cooling

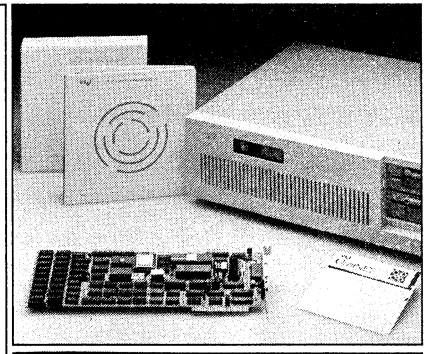
Ordering Information

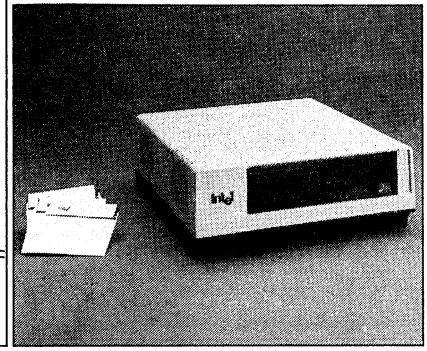
VMSNET	VAX/VMS Networking Software for installation on a high end VAX: consists of a Unibus network controller board with 256KB RAM, a 5 ft. and 10 ft. flat transceiver cables, software on a 9 track 1600 bpi magnetic tape, and an installation and user's guide.
MVMSNET	VAX/VMS Networking Software for installation on a MicroVAXII: consists of a Qbus network controller board with 256KB RAM, an 18 inch flat transceiver cable, software on both TK50 cartridge tape and RX50 5¼ inch disk, and an installation and user's guide.

NETWORKING FOR THE DEVELOPMENT ENVIRONMENT

- **OpenNET™ Network Resource Manager (NRM) provides shared file storage for all workstations**
- **OpenNET PC Link connects personal computers to the network**
- **Compilengine offloads compiles from any system on the network**
- **VAX Link for VAX/VMS* network communication**
- **NDS-II NRM OpenNET Upgrade**
- **Ethernet communication speeds**
- **Conforms to industry communication standards (ISO/IEEE)**







The total network development solution based on standards

Intel's open development networking encompasses the needs for existing as well as new Intel OpenNET™ development users. In the lab, Intel protects your investment by allowing you to interconnect existing Intel development workstations and other industry-standard hosts, such as the VAX/VMS* and the IBM PC. This network integrates OpenNET, Intel's open systems strategy for local area networks (LANs). It also ties the development lab, factory and office into a coherent environment.

The OpenNET family implements standards at each level of the International Standards Organization's seven-layer Open Systems Interconnect (OSI) model. For the lab, this includes a range of special networking services to provide your development lab with the power and flexibility needed to solve today's and tomorrow's problems.

A file server tailored to lab requirements

The OpenNET Network Resource Manager file server manages all network workstation requests for central resources, including file access, print spooling, tape back-up, remote job execution queue management, program management and network maintenance functions. The NRM, unlike many office file servers, features a full-featured, protected, hierarchial file system. The NRM supports transparent access to this file system from any OpenNET consumer (e.g., MS-NET, XENIX*, iRMX™ 86) as well as from Intel's Intellec Model 800, Series II, III and IV Systems.

Two OpenNET models are available: the MAXI, with a 140MB Winchester and a 60MB tape storage, and the MINI, with a 40MB Winchester only. Both are 8 MHz, 80286-based super-microcomputers with 1MB of zero wait-state RAM. And, both are optimized for file access using techniques such as caching of most recently used tracks, very fast disks, fast disk-seeking algorithms and communications boards with their own dedicated microprocessors.

Program Management Tools (PMTs) decrease the time spent tracking program/module changes and manually generating programs, giving software engineers more time for design, development and testing.

A remote job execution facility provides automatic network load balancing.

Transform your PC from an individual workstation to team member

The OpenNET PC Link enables users to connect their IBM PC XT/AT or compatibles to the OpenNET Network and to transparently access and share files and printers on an OpenNET NRM, NDS-II NRM, iRMX and XENIX-based file servers. OpenNET PC Link features an 80186/82586 microprocessor-based Ethernet/IEEE 802.3 expansion board, Microsoft networking software (MS-NET) and iNA960 transport software (ISO8073-compatible).

Compilengine: a shared network resource

Compilengine is a shared, networked system optimized to offload compile and link/locate jobs from any workstation or VAX on the network. It is an 80286-based supermicro-computer that compiles faster than any workstation and requires no terminal to operate. Moreover, because it supports two partitions, it can be used as a software workstation at the same time it is being used as a shared resource.

This product can be connected to an NDS-II NRM or OpenNET NRM.

Workstation requirements

WORKSTATION	SOFTWARE REQUIREMENTS	HARDWARE REQUIREMENTS
NDS-II workstations		
Compilengine	—	—
Model 800	—	PIMDX 455
Series II/III	—	PIMDX 455
Series IV	—	PIMDX 456
Cluster Chassis	NDS2TLB	PIMDX 455
VAX	IMDX 392; [VMS V4.2]	[DEUNA* Board]
OpenNET workstations		
PC DOS	DOS V3.1	PCLNK
System 310 XENIX	XNX-NET R1.0	iSXM 552
System 310 RMX 86	RMX-NET R1.0	iSXM 552

[] Available from DEC

NOTE: Interconnecting hardware (cables, transceivers) requirements are not included in the above chart.

Ordering Information

iMDX 460-140T	OpenNET NRM (MAXI model).
iMDX 460-40	OpenNET NRM (MINI model).
iMDX 555	NDS-II NRM OpenNET Upgrade Kit.
iMDX 485CE	Compilengine.
NDS2TLB	Network Software Toolbox.
iSYP 312	Floor stand which encloses either the OpenNET NRM or the SYS 311 peripheral expansion box.

NDS-II/VAX Link

This Ethernet-based link between an OpenNET NRM or an NDS-II NRM and a DEC* VAX/VMS microcomputer allows VAX users to copy files from the VAX to the NRM for debugging, in-circuit emulation and testing. With the remote job execution feature, VAX users can send CPU-intensive jobs to idle workstations (such as the Compil$_e$ngine) for execution. Conversely, NRM users can send jobs for remote execution on the VAX.

NDS-II NRM OpenNET™ Upgrade

This product allows your NDS-II NRM to double as an OpenNET file server for PC, XENIX and iRMX workstations; files on the NRM may be transparently accessed by any workstation on the network.

Peripheral Expansion Option

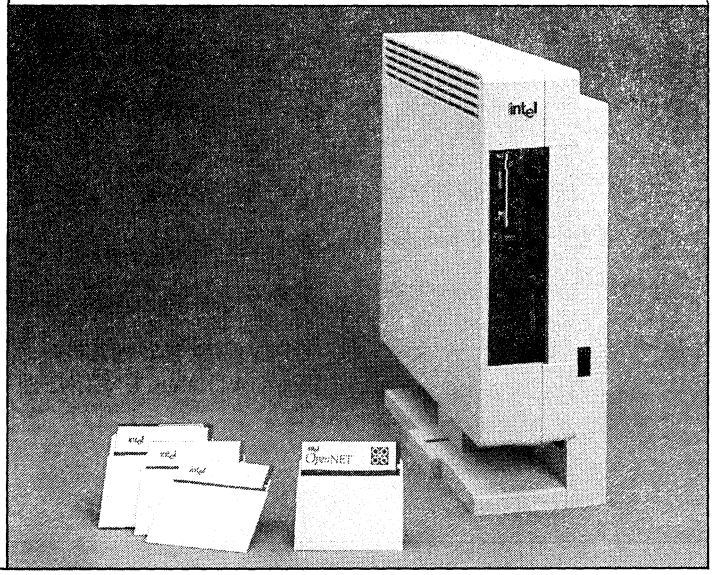
The OpenNET NRM supports 40 or 140MB of Winchester storage on a single disk drive. Mass storage can be expanded to 460MB on the MINI NRM and 560MB on the MAXI NRM, using the 311 peripheral system.

Workstation Kits

PIMDX 455	NDS-II Workstation Upgrade Kit for any Series II/85, Series III, or Model 800 to connect to the OpenNET NRM or NDS-II NRM.
PIMDX 456	NDS-II Workstation Upgrade Kit for the Series IV.
PIMDX 581	ISIS Cluster Board Package.
IMDX 392	VAX Link R2.1 for VAX/VMS connection to the NRM.
PCLNK	OpenNET PC Link hardware and software kit to connect the PC XT, PC AT, and compatible systems to the NRM via the OpenNET network; requires DOS 3.1 or higher.
RMXNETKITWRI	iRMX Networking Software for a 286/310 system running the iRMX 86 operating system to connect to the NRM via the OpenNET network.
SXM 552S	Ethernet-based Single Board Network Communication Engine for 310 systems.
XNXNETNRIKIT	OpenNET-XenixNET Networking Kit. Includes iNA 961, SXM 552 and XenixNET pass-through networking software.

Interconnecting Hardware

PIMDX 457/458	Transceiver cables (10/50 meters) (two are required for an OpenNET NRM; one is required for a Compil$_e$ngine).
PMDX 3015	Transceiver for Ethernet coaxial cables (at least two are required unless an Intellink is used).
iDCM 911-1	Intellink module (the OpenNET NRM uses two ports).
PIMDX 3016-1/ 3016-2	Ethernet coaxial cable (25/50 meters).



*VAX/VMS, DEC & DEUNA are trademarks of Digital Equipment Corp.
XENIX is a trademark of Microsoft Corp.

APPLICATION NOTE

AP-240

October 1986

Using Archive To Efficiently Control a Network

SRIVATS SAMPATH
DSO APPLICATIONS ENGINEERING

Order Number: 231476-001

INTRODUCTION

The onset of large scale software projects has generated additional needs in all levels of the development environment. The need for a sophisticated source and version control system and efficient disk management becomes particularly important as the project team grows. This need is more pronounced at the software management level, where operating the project on schedule is of prime importance. Efficient disk management includes keeping the disk free of redundant files and keeping copies of older versions somewhere other than the disk itself. In other words "ARCHIVING" all previous versions onto another storage media that is inexpensive, reliable and transportable is key. One storage media that meets all these requirements is the NDS II tape sub-system which forms an integral part of the development environment. Moreover, the actual archive process should be easy to use, preferably automatic and should not be a drain on resources. The ability to manage mass storage devices efficiently translates into a substantial increase in productivity for everyone. Intel realizes this need and has developed a solution that is tailored towards helping the NDS-II system manager efficiently control the development project. We introduced the TAPE SUB-SYSTEM on our Network to provide an inexpensive, reliable and transportable media, and a utility called ARCHIVE to make actual disk management both user friendly and automatic.

ARCHIVE

The ARCHIVE utility performs file backup and restoration by copying files and directories to magnetic tape or other secondary storage devices. This utility is executed at an NRM console, and with its powerful set of options, it positions itself as an invaluable tool for efficient disk management.

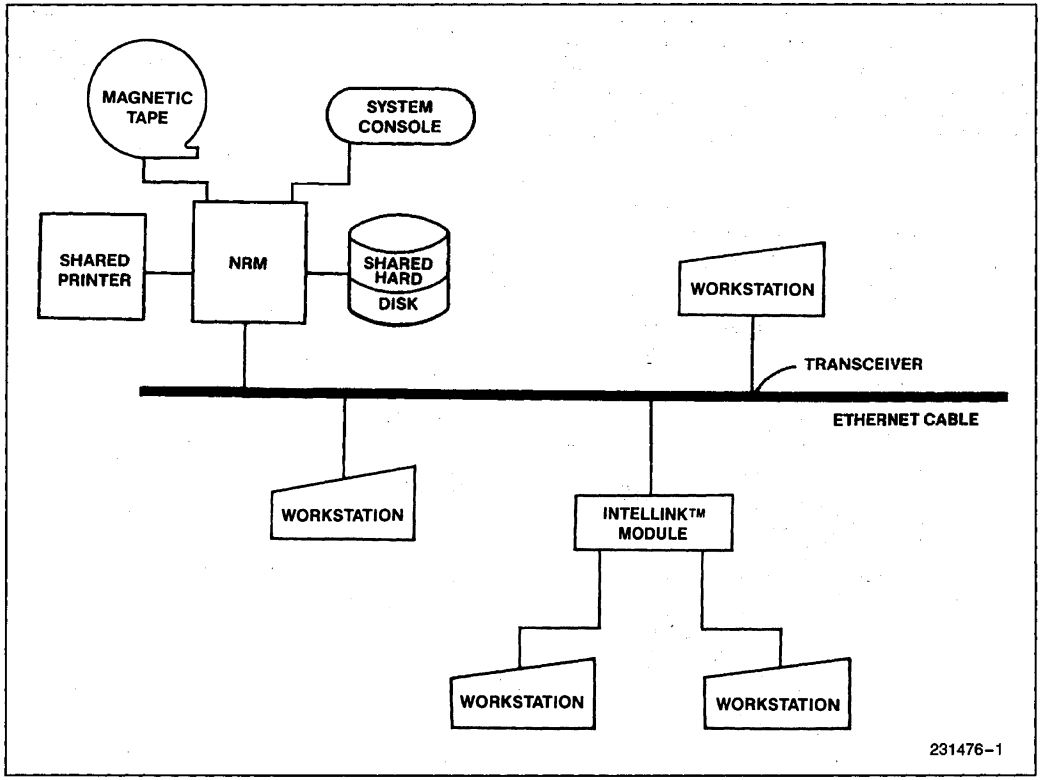


Figure 1. The Network and Its Components

WHY USE A TAPE?

Magnetic tape is regarded as the most useful storage device in the computer industry. In spite of its sequential structure, tape answers a number of requirements that can not be met by any other conventional mass storage devices. The most strong argument in favor of tape is its portability - the ability to transport tape with the minimum overhead and damage during transit makes it an extremely attractive media. Moreover, storing tapes is much more organized and efficient than storing diskettes. All these arguments lead to a single conclusion: "The Magnetic Tape should form an integeral part of any development environment". The ARCHIVE utility and the tape drive together form the foundation for effective disk management, bringing about a more productive environment for any development project.

Additionally, the tape is a safety net for one of those rare disk crashes. Having backups on tape will reduce the amount of data loss in the event of a fatal disk crash. Additionally, completed projects can be saved on ape to provide more disk work space. Having these projects on tape minimizes the effort in reloading all the data if major bugs are found, or if an update is involved. Multi-project sites can benefit from the fact that tape allows easy transporting of data.

Tape backup can be classified into incremental backup and tape streamer. Tape streamer allows volume copies, with every record on the disk copied onto tape. It is a mass data transfer from one storage device to another. This brings about a lot of overheads when only some parts of the disk need to be backed up or restored.

Incremental tape backup treats the tape as a random access device similar to a disk. Files can be added, appended or deleted, just as in a disk. This feature allows selective backup onto tape; thus eliminating the need for a mass copy operation when only a few files need to be archived or restored.

The tape drive on the NRM is an incremental backup device and ARCHIVE has been designed to use this to the fullest extent.

HOW DOES ARCHIVE HELP?

ARCHIVE has been designed to let the NDS II system manager operate the development project at maximum efficency. Using its various options, the system manager can selectively archive files and directories onto tape, employing various qualifiers such as date accessed, created, modified, before, since, on, etc.. These qualifiers will be discussed in a later chapter (Invocation and Syntax) with examples. The options are essential for NDS-II system managers to perform selective archives of files and directories.

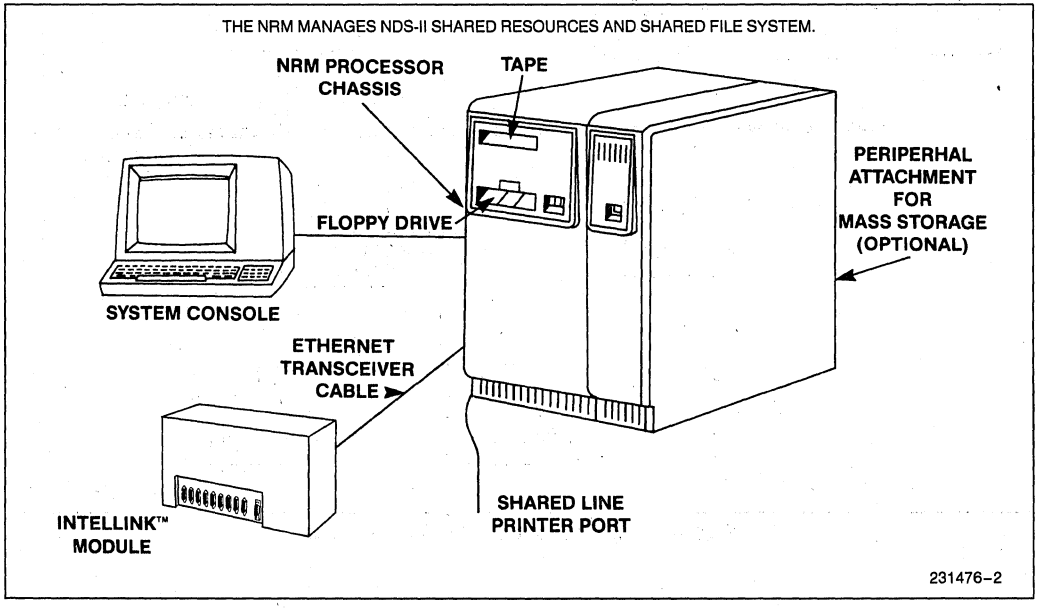


Figure 2. Network Resource Manager (NRM)

ARCHIVE can also be used in a submit or a batch file. This saves the NDS II system manager from having to type in the whole command syntax every time an ARCHIVE must be performed adding another step toward improved productivity.

Utilities like SLEEP, wakes up the system at a specified date or time, goes a long way in automating ARCHIVE. A submit file is invoked at the NRM that wakes up the system at a specified time (preferably near midnight when system load is low), archives all qualified files onto tape, then 'goes back to sleep' again. This is an important factor eliminating the need for operator intervention at any time and automating the entire process.

ARCHIVE frequency depends on the particular application and system load. It is recommended that ARCHIVE be performed at least once a week. However, in large project implementations (i.e 6 or more design engineers involved in generating or modifying more than 100K of code), ARCHIVE should be performed automatically each night. This ensures that even if a disk crash occurs, data loss is restricted to a single day's work.

All the features incorporated in ARCHIVE make it an attractive solution for effective version control and disk management. It is a productivity tool that no system manager should do without.

DATA LAYOUT ON TAPE

Tape is a sequential structured media, with all files and directories sequentially stored, but it maintains the hierarchic file structure of a disk. Every time an ARCHIVE is performed, a LOGICAL VOLUME is created, containing any number of files, from NULL to a set of files residing on a device. Each LOGICAL VOLUME has a VOLUME NUMBER and a HEADER associated with it. VOLUME NUMBERS start with 1 upwards. The HEADER is the source path name. For example:

```
ARCHIVE /WD0/USERS.DIR TO CT0
```

creates the first record onto tape and gives it the VOLUME NUMBER 1 and HEADER /WD0/USERS.DIR. Files and sub-directories under USERS.DIR will be copied in a TOP DOWN order. The same rules apply to all the subsequent sub-directories. The data layout on tape is shown in Figure 3.

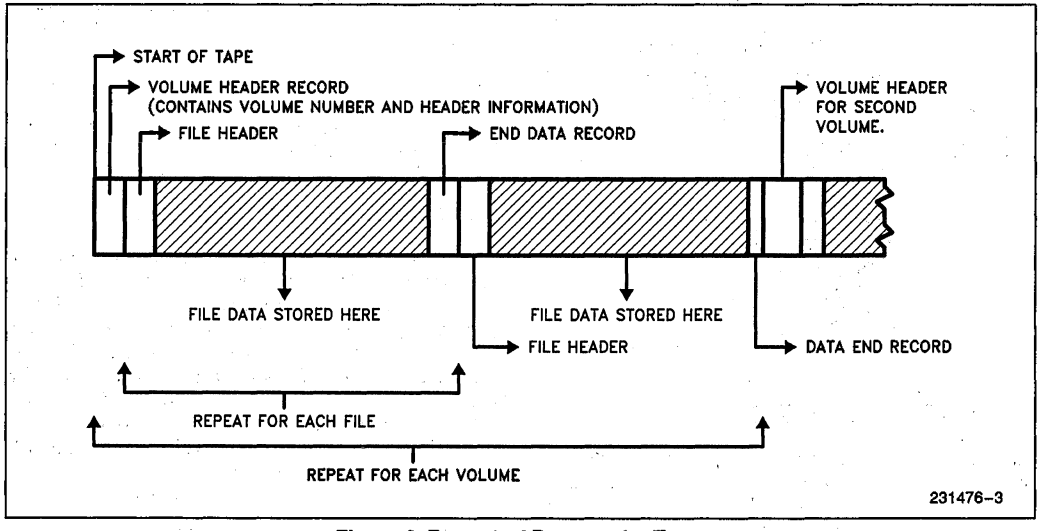


Figure 3. Format of Data on the Tape

INVOCATION AND SYNTAX

ARCHIVE, with its powerful set of options, gives the user flexibility in effectively managing the disk. The syntax of ARCHIVE is given below. During operation at the NRM, the system syntax builder prompts the user for options so none of the options have to be memorized.

The ARCHIVE syntax consists of a set of qualifiers and a set of switches. QUALIFIERS are options that qualify a file or directory for copying, allowing the user to selectively choose files and directories for archiving. SWITCHES are sets of controls that enable the user to actually control the I/O operation.

SYNTAX:

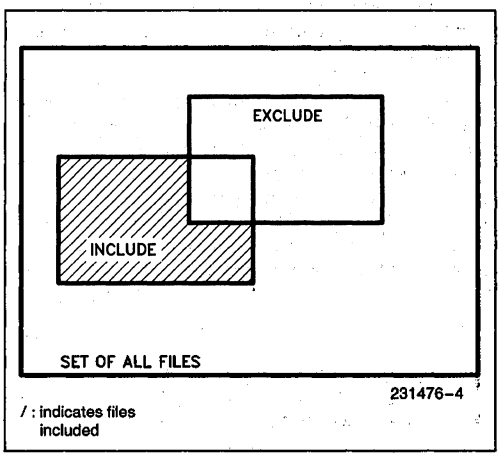
```
ARCHIVE source TO destination
  < O P T I O N S >

OPTIONS ARE:
1. QUALIFIERS:
INCLUDE,     EXCLUDE (files that were . . . )
          ACCESSED/CREATED/MODIFIED
               BEFORE / SINCE / ON
                  TODAY / date
                     hour

             DIRECTORY (directory name, . . . )
               OWNEDBY (Owner name, . . . )
                 FILE (path-name, . . . )
                   AND/OR . . .
2. SWITCHES: APPEND
             DELETE
             ERASE
             LOG log-file-name
             NAME physical volume name
             NOUPDATE
             QUERY
             UPDATE
             VOLUME (logical volume number )
```

ARCHIVE QUALIFIERS

Qualifiers enable the user restrict the number of files to be archived. and discriminate against any file by time stamps (time created, modified, etc), the owner, file names and even by parent directory. These qualifiers have no limit to their length or order of appearance, an may be specified using the keywords INCLUDE/EXCLUDE.

Include/Exclude

INCLUDE specifies the files that are to be included in the command, while EXCLUDE lists the file that can not be archived if the qualifying condition is met. EXCLUDE has precedence over INCLUDE; therefore, when both keys are used (INCLUDE files, EXCLUDE files) the following set of files will be archived:



The following is the list of all acceptable keywords for INCLUDE/EXCLUDE:

1. ACCESSED/CREATED/MODIFIED

These switches compare the time specified in the time qualifier to the last time the file was accessed, or the time it was created, or the time it was last modified. If this time agrees with the time qualifier condition, then the file is qualified. The time qualifier is required for ACCESSED and CREATED and is optional for MODIFIED. If the time is not specified with the MODIFIED switch, a default value of SINCE LAST-ARCHIVE-DATE will be used. This default value will qualify all the files which:

a. Were modified since last ARCHIVE.

b. Were created since last ARCHIVE.

c. Were created or modified prior to last ARCHIVE but, through use of qualifiers, they were somehow EXCLUDED from being archived earlier.

Time Qualifiers:

The time qualifiers allow the user to specify an instant in time which is used in a comparison with the time a file was last accessed, created, or last modified to qualify the file for archiving:

— BEFORE Allows specifying a file accessed, created, or modified BEFORE a specific date.

— SINCE Allows specifying a file accessed, created, or modified SINCE a specific date.

— ON Allows specifying a file accessed, created, or modified ON a specific date within a 24 hour period.

— date/TODAY

For neither BEFORE, SINCE, and ON a does default date value exist. Date could be specified in two forms, either by using TODAY switch, which would read the current date from the system, or by actually specifying the date in the form of mm/dd/yy. An optional time of the day (in hours) in hh:mm:ss form with a default value of 00:00:00 can be used.

— hour

Time of the day can be specified in hh or hh:mm or hh:mm:ss. The hour qualifier is to be in parenthesis.

Examples:

```
ARCHIVE /WD0 TO CT0 INCLUDE CREATED BEFORE 12/21/84
     ;would archive all files created before DEC 21.

     ;a time default of 00:00:00 would be used.
ARCHIVE /WD0 TO CT0 EXCLUDE MODIFIED SINCE 10/10/83 ( 10:11:22 )
     ;archives all files, exclude those which were modified
     ;since 10:11:22 on October 10th, 1983.

ARCHIVE /WD0 TO CT0 INCLUDE ACCESSED ON TODAY , EXCLUDE & CREATED BEFORE
      10/26/83 AND MODIFIED SINCE 10/24/83 ( 10:11:12 )
     ;archives all thefiles which were accessed today, and
     ;exclude those which were created before October 26th
     ;and were somehow modified since 11 minutes and 12 seconds
     ;after 10 AM on October 24th.

ARCHIVE CT0 TO /WD1/DIR1 INCLUDE ACCESSED ON TODAY , EXCLUDE & CREATED
 BEFORE 10/24/83 AND MODIFIED SINCE 10/26/83 ( 10:11:12 )
     ;all files on the tape which were last accessed today
     ;( exclude archive access itself ) will copied to the
     ;/WD1/DIR1 directory. Files which were CREATED before
     ;October 24,83 and were MODIFIED since October 26, 83
     ;will be excluded.
```

2. DIRECTORY/FILES/OWNEDBY

Allows the user to specify qualifiers other than time such as owner of files, directories, etc.

— DIRECTORY Allows the user to specify particular directories to be included or excluded in ARCHIVE. The directory could either be the full path name of the directory or partial name from where source-name left off. DIRECTORY does not accept wildcard characters. However, logical names are allowed.

— FILE Allows the user to specify particular files to be included or excluded in archive. The file name, in order to be recognized, should only be the filename, not a path name. Wildcard characters are accepted.

— OWNEDBY Allows archive select files on the basis of owner's name.

3. AND/OR

AND/OR allows the extension of the qualifying conditions within a qualifying set. AND/OR can not be intermixed within a qualifier set defined by one INCLUDE or EXCLUDE.

4. COMMA

Comma is the separator (delimiter) between INCLUDE and EXCLUDE. In English, it makes sense to use AND in between INCLUDE and EXCLUDE. However, in ARCHIVE, you can not use anything other than Comma to separate INCLUDE/EXCLUDE. Example:

```
ARCHIVE /WD0 TO CT0 INCLUDE ACCESSED
ON TODAY , EXCLUDE & CREATED BEFORE
10/26/83 AND MODIFIED SINCE 10/24/83
( 10:11:12 )
```

ARCHIVE SWITCHES

SWITCHES are controls that ARCHIVE gives the user to selectively copy files and directories to/from tape or any other storage device. We will discuss each of these switches in depth to highlight the versatality of ARCHIVE.

1.0 Append, Volume

Every time ARCHIVE is issued onto tape, a Logical Volume is created. This logical volume can consist of one file or as many as all files residing on a particular device. Unless otherwise specified ARCHIVE always starts from the beginning of a tape. The tape is rewound and the header information is written followed by the actual copy operation which copies all qualified files from the beginning of tape. Using the Append switch allows the user to have more than one volume or a related group of files on a single tape .Now instead of starting from the beginning of a tape ARCHIVE searches through the tape for the volume name specified. Default for Append is the last volume on tape.

ARCHIVE will always overwrite an existing volume if Append switch is not specified. Append to an empty tape is not valid as ARCHIVE will not know what to Append the new record to.

Recommendation: The tape should be dismounted only after ARCHIVE signs back on with the message 'ARCHIVE COMPLETE'. Use the ERASE option when writing to a new tape.

Examples: (WITH A NEW TAPE)

```
ARCHIVE   /WINIO/USERS.DIR TO TAPE0 APPEND
               ; This is an ERROR. No previous volume on tape
               ; to append new volume to
ARCHIVE   /WINIO/USERS.DIR TO TAPE0
               ; This will create header for Volume #1 and then
               ; copies all the USERS.DIR files and sub-directories
               ; to the tape.
ARCHIVE   /WINIO/SYSTEM.DIR TO TAPE0 APPEND
               ; This creates a new volume on the tape (Volume # 2)
               ; and adds all SYSTEM.DIR files and sub-directories
               ; to the tape at Volume #2
ARCHIVE /WINIO/ISIS.SYS/FILES TO TAPE0 APPEND VOLUME 3
               ; This skips to the third volume on tape ,writes
               ; the header for Volume #3 and then copies all
               ; files and sub-directories to Volume #3
```

2.0 Delete

This switch instructs ARCHIVE to delete all qualified files on the disk after they have been copied onto tape or disk. It is very useful when backups of older versions are performed. Once the archive process has been completed, all the old files are deleted from the source disk giving the user a better control over managing disk files. This is a disk only option.

Recommendation:

It is recommended that the user archive to tape first, using a LOG option and ascertaining that the files exist on tape. Then, he repeats the ARCHIVE to :BB: with the delete switch on to delete all the qualified files from disk. This will eliminate any possibility of deleting files without first archiving them.

3.0 Erase

This option causes the tape to be erased before any write operation is performed. ERASE goes over all tracks on the tape and erases everything written on it. ERASE and APPEND cannot be used simultaneously, since one erases the tape and the other tries to append to non-existent volumes. ERASE is a tape otion.

Recommendation:

Use the ERASE switch when archiving onto tape the first time. Use Append for subsequent logical volumes.

4.0 Log file—name

The LOG option will redirect all console messages to a specified LOG file. Errors generated because of LOG file existence will not abort ARCHIVE. (i.e. if a log file already exists it will be automatically overwritten by ARCHIVE)

Recommendation:

It is good practice to redirect console output to a LOG file when a sufficiently large ARCHIVE is being performed, keep a record of all succesful archives. This LOG file should be listed and stored along with the tape.

5.0 Name physical_volume_name

The first time an ARCHIVE is issued to a tape, using the NAME option will associate the physical_volume_name with that tape. This option ensures that the right tape is used when reading from or writing to the tape. When the NAME switch is specified the name on tape will be compared against the name on the ARCHIVE command line. ARCHIVE will not continue if names do not match. The default physical_volume_name is ARCHIVE, meaning that if a NAME option was not specified during the first write operation to tape, it will be named ARCHIVE.

Recommendation:

The use of logical sounding names for the physical—volume—name of the tape is good practice. This helps in fast identification of the tape being used. Names like PROJECT1 and VERSION1.0 are good names while THIS.IS.IT and LATEST are not. The physical—volume—name should not be more than 14 characters long.

6.0 Noupdate

When archiving information from any source to a hard disk, if an existing file is encountered, NOUPDATE instructs ARCHIVE not to copy over the existing files. Thus if a file is on the disk and there is a matching file name on that tape, archive from the tape to the disk will not update the contents of the file when the NOUPDATE switch is used. This option is a default switch in submit files. If neither UPDATE nor NOUPDATE is used, the user will be queried whether the existing file should be deleted.

Recommendation:

The specified default for ARCHIVE in a submit file is NOUPDATE. However, the default for ARCHIVE in a submit file is similar to the QUERY command. If files being restored already exist, ARCHIVE will prompt the user for deletion. It is recommended that UPDATE or NOUPDATE option be specified within a submit file.

7.0 Query

This causes ARCHIVE to prompt the user for every data and directory file in the source directory, then waits for confirmation. When the user is prompted regarding a directory file, and the user chooses not to copy that directory file, none of the files and sub-directories in that directory can be archived. The default is no QUERY. QUERY used in conjuction with NOUPDATE prompts the user for every qualified file in the source directory. When confirmed that the file exists in the destination directory, the user will be informed that the file exists at the destination directory, but the file will not be copied over. QUERY used in conjunction with UPDATE prompts the user for each file in the source directory. Once confirmed, it will copy the qualified files to the destination regardless of their existence.

```
ARCHIVE   /WDO/USERS.DIR TO CTO NAME TAPE1
               ;archives every file and directory in USERS.DIR
               ;onto the tape. The tape will be named TAPE1
               ;from now on. If an attempt is made to access
               ;or write more files onto the tape with the
               ;NAME switch on, TAPE1 is the only name that will
               ;be accepted by ARCHIVE.
ARCHIVE   /WDO/USERS.DIR/MINE.DIR TO CTO NAME TAPE1 APPEND
               ;would append MINE.DIR to the tape.
ARCHIVE   /WDO/USERS.DIR/YOURS.DIR TO CTO APPEND
               ;would still work fine and appends the
               ;new directory YOURS.DIR to the tape.
ARCHIVE   /WDO/USERS.DIR/WHOSE.DIR TO CTO NAME TAPE0
               ;would be rejected with the message:
               ;RIGHT VOLUME EXPECTED.......
```

8.0 Update

This switch is the exact opposite of the NOUPDATE switch. If UPDATE is specified, all the qualified files on the tape will be copied to the destination directory, despite the previously existing files in the destination directory.

Example:

Assume that files F1 and F2 are in /W1/D1 and directory file D2 is in /W1. Also assume F2 exists at /W2/D1.

```
> ARCHIVE /W1 TO /W2 QUERY <CR>
  iNDX-N11 (V2.8) ARCHIVE, V2.8
  10/26/84 11:12:33 DIRECTORY = /W1
  COPY /W1/D1 TO /W2/D1 ? Y <CR>

  DIRECTORY = /W1/D1
  COPY /W1/D1/F1 TO /W2/D1/F1 ? Y ©
  COPIED /W1/D1/F1 TO /W2/D1/F1
  COPY /W1/D1/F2 TO /W2/D1/F2 ? Y <CR>
  File Already Exists
  Pathname = /W2/D1/F2
  Delete Existing File ? Y <CR>
  COPIED /W1/D1/F2 TO /W2/D1/F2
  COPY /W1/D2 TO /W2/D2 ? N <CR>

  ARCHIVE COMPLETE

> ARCHIVE /W1 TO /W2 QUERY NOUPDATE ©
  iNDX-N11 (V2.8) ARCHIVE, V2.8
  10/26/84 11:12:33

  DIRECTORY = /W1
  COPY /W1/D1 TO /W2/D1 ? Y <CR>
  DIRECTORY = /W1/D1
  COPY /W1/D1/F1 TO /W2/D1/F1 ? Y <CR>
  COPIED /W1/D1/F1 TO /W2/D1/F1
  COPY /W1/D1/F2 TO /W2/D1/F2 ? Y <CR>
  File Already Exists Pathname
  = /W2/D1/F2
  COPY /W1/D2 TO /W2/D2 ? N <CR>

  ARCHIVE COMPLETE

> ARCHIVE /W1 TO /W2 QUERY UPDATE <CR>
  iNDX-N11 (V2.8) ARCHIVE, V2.8
  10/26/84 11:12:33
  DIRECTORY = /W1
  COPY /W1/D1 TO /W2/D1 ? Y <CR>
  DIRECTORY = /W1/D1
  COPY /W1/D1/F1 TO /W2/D1/F1 ? Y <CR>
  COPIED /W1/D1/F1 TO /W2/D1/F1
  COPY /W1/D1/F2 TO /W2/D1/F2 ? Y <CR>
  COPIED /W1/D1/F2 TO /W2/D1/F2
  COPY /W1/D2 TO /W2/D2 ? N <CR>

  ARCHIVE COMPLETE
```

9.0 Volume

The first time an ARCHIVE command is issued, one logical volume will be created on the tape. Subsequent ARCHIVE's to the tape using the APPEND switch create additional logical volumes on the tape. For instance, one ARCHIVE without APPEND and three more ARCHIVE's with APPEND create four logical volumes on the tape. If the user is restoring information from the tape, not specifying volume number restores all the logical volumes on the tape. Specifying a non-existent number causes an error message and aborts the command. A valid volume number searches that particular logical volume for the qualified files and restores only files from that specific logical volume.

Example:

```
ARCHIVE    /WDO/USERS.DIR TO CTO
                ;erases the tape and copies USERS.DIR to
                ;the tape as logical volume 1.
then
ARCHIVE    /WDO/MISC.DIR TO CTO APPEND VOLUME 3
                ;causes an error message, because logical
                ;volume number 2 is not created yet.
then
ARCHIVE    /WDO/MISC.DIR TO CTO APPEND or
ARCHIVE /WDO/MISC.DIR TO CTO APPEND VOLUME 2
                ;creates the second logical volume and
                ;copies all the files from MISC.DIR to it.
then
ARCHIVE    CTO TO /WDO/DIR1 VOLUME 3
                ;generates an error message because
                ;logical volume 3 does not exist.
then
ARCHIVE    CTO TO /WDO/DIR1 VOLUME 2
                ;copies to DIR1 all the data and directory
                ;files which were under /WDO/MISC.DIR and
                ;were archived to the tape. In a sense, the
                ;subtree starting from /WDO/MISC.DIR will
                ;be added to DIR1.
```

DATA RESTORATION FROM TAPE

ARCHIVE allows data restoration from tape onto disk, facilitating easy recovery from a disk crash without a significant loss of data. Reopening a project simply involves reloading all data archived onto tape. This also simplifies multi-site projects, where data can be transported and reloaded from one site to another.

In order to restore data from a tape, the user can use the device name CT0 and restore all information from tape to disk. Or the user can specify a pathname to a directory on tape and restore only that directory and associate files and sub-directories. Finally one can specify a particular VOLUME and restore information stored under that volume. Examples:

Assuming that /WD0/USERS.DIR/TEMP.DIR is empty and the tape has three records (i.e. LOGICAL VOLUMES)

APPEND and ERASE are switches that can be used only when archiving onto tape. DELETE, NOUPDATE and UPDATE switches can only be used with a disk.

```
Record #1 (Volume Number 1)
Header /WD1/USERS.DIR/TEMP1.DIR
FILES and DIRECTORIES
     /WD1/USERS.DIR/TEMP1.DIR/FILE1
     /WD1/USERS.DIR/TEMP1.DIR/FILE2
     /WD1/USERS.DIR/TEMP1.DIR/FILE.DIR/FILE3
     /WD1/USERS.DIR/TEMP1.DIR/FILE.DIR/FILE4
     /WD1/USERS.DIR/TEMP1.DIR/FILE.DIR/PASCAL.DIR/FILE5

RECORD #2 (Volume Number 2)
Header /WD0/MISC.DIR/TEMP2.DIR/TEMP3.DIR/ F1
FILES and DIRECTORIES
     /WD0/MISC.DIR/TEMP2.DIR/TEMP3.DIR/FILE1
     /WD0/MISC.DIR/TEMP2.DIR/TEMP3.DIR/FILE2

-- ARCHIVE CT0 TO /WD0/USERS.DIR/TEMP.DIR
```

would copy all files from tape onto disk.

```
-- ARCHIVE CT0 TO /WD0/USERS.DIR/TEMP.DIR VOLUME 2
```

would copy all files in VOLUME 2 to disk.

APPENDIX A

SLEEP:

SLEEP is a utility, available in the Network/Series IV toolbox, that executes at an NRM or SERIES IV, allowing the user to delay the execution of certain programs until a certain time. This can be included in a submit file and made to execute continuously. The sample/submit file looks like this:

```
Repeat

Sleep til 23:30:00

ARCHIVE /WDO/USERS.DIR TO CTO INCLUDE ACCESSED ON TODAY APPEND

End
```

This submit file will run forever at the NRM console and will wake up at midnight do all the archives, then go back to sleep again. Since sleep runs on the foreground at the NRM, a Cntr-C has to be performed if the user must utilize the NRM terminal for some other purpose.

This is a very useful utility in conjunction with ARCHIVE as it makes the whole process automatic and eliminates the need for operator intervention.

ACKNOWLEDGMENTS:

I would like to take this opportunity and thank Bahram Saghari in DSO Software Support for his contribution towards this Application Note. All examples on ARCHIVE were supplied by his group.

APPLICATION NOTE

AP-242

October 1985

Additional Printer Support for the NDS-II System

CHRIS FEETHAM
DSO APPLICATIONS ENGINEERING

Order Number: 231478-001

INTRODUCTION

Using printers for hard copy of data has long been necessary in most computer systems. Software engineers use printers primarily for software program listings, but increasingly, letter quality printers are being used to generate memos, reports, and other business documents, rather than queueing them up at the secretary's typewriter. Additionally, with the cost of computer terminals and network connections declining, it is becoming rare for the business professional not to have immediate or direct access to a terminal with some type of word processor available. The ability to send hard copy directly to a printer rather than waiting for a typist to re-type the input is a productive benefit for everyone.

THE NDS-II NETWORK

With Intel's advanced Network Development System II (NDS-II), development systems are connected into a network using Ethernet. Additionally, each development system has the ability to host several ISIS Clusters that use low cost serial lines to support the terminals. The complete product line is described in the NDS-II System description (refer to Appendix D for complete details).

With low cost terminals available to everyone, including engineers, managers, and secretaries, files and data can be shared and manipulated directly on the network, reducing the many intermediate steps required in producing a final document. The addition of CPM/80 coupled with the industry standard Wordstar word processing package, available for the NDS-II system (refer to Appendix D for details), further increases secretarial efficiency.

Engineers, managers and secretaries all benefit from the advanced editors and tools provided with Intel's systems. Getting the output to a printer is the next step in the process, and is the subject of this application note.

GETTING THE DATA PRINTED

Virtually every computer sold today, from the most inexpensive PC to the largest mainframe, has serial and/or parallel ports for connections to printers and other devices. Intel's development systems are no exception, providing hardware ports for both serial and parallel printer types.

Intel's operating systems supplied with the NDS-II network and development mainframes, INDX and ISIS respectively, provide software "devices" which the user can copy files to. The software device designations are :LP: for the parallel line printer, and :TO: for the serial device. However, varying types of serial printers and their associated protocols render the simple "Copy file to :TO:" inadequate. Additionally, printers are somewhat expensive and noisy. The desired method of operation is to provide one or two printers accessible by a group of people, located in a separate room away from the immediate working area.

This application note shows how Intel's NDS-II network, combined with ISIS Clusters and terminals provide a solution for the desired method of operation. The NDS-II's INDX operating system provides a print spooler that allows users to copy files to a central spool printer (:SP:). Files copied from the remote stations (ISIS Clusters , Series-II/III and Series-IV development systems) are then copied to a parallel line printer connected to the NDS-II.

This print spooling feature is not a new concept for computers, and is only one of many excellent features of the NDS-II system. Many users would like to support additional printers on the network, both parallel and serial, but the NDS-II's built in spooler does not provide for this.

SOLUTION–Prince

Prince is a versatile spooling program designed for use with Intel's Series-II, Series-III, and Series-IV development systems, either in standalone or network mode, and for ISIS Clusters operating with an NDS-II network. Using a dedicated ISIS Cluster is perhaps the most effective and efficient method of operation. The ISIS Cluster solution provides for the cheapest and most automatic operation, which is detailed in Appendix C.

HOW IT WORKS

Prince is an ISIS-based program operating in the 8085 environment of the Development System or ISIS Cluster. After extensive initialization, Prince continually polls the directory that is ASSIGNed to :F8:, and any files in this directory are PRINTED, then DELETED. As this is an ISIS based program, files to be printed must conform to the ISIS file name format:a maximum of six characters, plus an optional three character extension, separated by a period.

:F8: can be assigned to a directory created on a Series-IV for standalone operation, or to a directory on the Network Resource Manager. If the Network Resource Manager is used, and the NRM has no parallel printer attached, you may assign :F8: to /(root)/SPOOL, the main print spooler directory. A workstation could then copy directly to :SP: instead of :F8:. This saves each workstation from having to assign :F8: to a specific directory.

Prince has been designed for optimal use of network resources, and provides additional capabilities and flexibility above and beyond the automatic print spooler provided with the NDS-II. Prince also provides useful capabilities for Series-IV system operating in standalone mode.

Other applications might include operation of a parallel printer at a development system host for ISIS Cluster users, or even communication interface that automatically copies files from one system or network to another system connected via a serial or parallel line.

Upon invocation, Prince automatically checks its environment to determine the type of system it is loaded on. Valid systems are Series-II, Series-III, Series-IV, and the ISIS Cluster. Prince then sets up the appropriate serial channel for output, unless output has been directed elsewhere. For the Series-II and Series-III, this is serial channel 1. The Series-IV uses serial channel 2, and the ISIS Cluster uses the on-board serial channel normally used for the console.

Series-IV systems can use serial printers, but the control interface for the serial device, specifically the XON/XOFF (cntl-s / cntl-q) protocol, is currently not provided with a simple copy to the system serial file (designated :TO:). Prince solves this problem by providing the XON/XOFF protocol, and optionally checks for a hardware printer ready signal if desired, by selectively monitoring Data Set Ready (DSR) on the serial line.

The Intel development systems set the serial channel used for the serial device (:TO:) to a specific file transfer rate, better known as baud rate. Prince can selectively output serial data at user specified baud rates of 110, 300, 600, 1200, 2400, 4800, 9600, and 19200. This allows faster devices and devices that can "buffer up" data to take advantage of the full capabilities of the serial line, while the controls mentioned previously (XON/XOFF and DSR) provide the desired control protocol to run the serial devices and the development systems at their fastest rate.

For management tracking and control, Prince keeps a log of all activity, including error messages, initialization defaults, and information about each file printed. File PRINT.LOG is created in the directory assigned to :F9:, and contains relevant information about the files being printed: the file name, time that the file was printed, owner of the file, and the number of bytes actually printed. The log information can be re-directed to another file, including the console, line printer, or disk file. If the log file specified is a disk file, it can be viewed, copied, or deleted at any time. If the log file is deleted, Prince creates the log file again, using the original log file name, at the next file detected for printing.

Prince allows re-direction of the output to a file rather than the printer connected to the serial line. Spooling to a parallel line printer is accomplished by specifying :LP: as the output path. The output re-direction can also go to a disk file, or any other valid ISIS output file name except :TO:. If a disk file is specified for output, it can be viewed, copied, or deleted at any time. Files being copied to the output file are added to the end of the file. For orderly printing, Prince automatically outputs a form feed before printing each file.

This version is initialized for use with a Diablo 630 serial interface and supports the XON/XOFF protocol at 2400 baud. These parameters may be changed by command line controls.

The ISIS.INI, or submit file that invokes this program, must contain a directory assignment to :F8:, for the files to be printed, and also an assignment to :F9: for the log file, unless it has been re-directed.

The default log file name, if none other is specified, is :F9:PRINT.LOG. Any file specified for the optional re-direction of the log file and/or the output path must be a valid ISIS output file name (refer to the NDS-II ISIS III User's Guide #121765-004 for a definition of valid ISIS output filenames).

INVOCATION AND CONTROL OPTIONS

Invocation of Prince is best accommodated in a command file, or SUBMIT file. For Intel systems, use of a user 'init' file is recommended, and essential for automatic use with an ISIS Cluster. User Init files are automatically submitted for execution upon LOGON to the system. This file contains assignments, and the command line that starts Prince.

Control options are all single letter characters, followed immediately by an "=" sign, then the actual option. Controls can be entered in any order, upper or lower case, can be separated by spaces or commas, but must contain no imbedded spaces. If Output is redirected to a file, as opposed to the default serial channel, then DSR and Baudrate controls have no effect, and the serial channel is not initialized.

Control Description and Examples

```
Controls:                    Control Description:
L=logfile              ; Valid ISIS filename - log file re-direction
                       ; :F9:PRINT.LOG is the default
P=output$file          ; Valid ISIS filename - output re-direction
                       ; can be :LP: for the local line printer, etc.
D=T                    ; DSR control. Any character other than 'T' will
                       ; not set the DSR control - pin 6 on the RS-232
                       ; line is monitored for printer ready. No DSR is
                       ; the default.
B=baudrate             ; valid number. Only the first two characters
   110                 ; are checked to determine uniqueness.
   300                 ; Any following characters are ignored.
   600
   1200
   2400
   4800
   9600
   19200
```

Examples:

1. To set log file to console out and output to line printer:

   :F9:PRINCE l=:co: p=:lp:

2. To set baudrate to 9600 and initialize DSR control (defaults to :F9:print.log):

   :F9:PRINCE b=9600 d=t

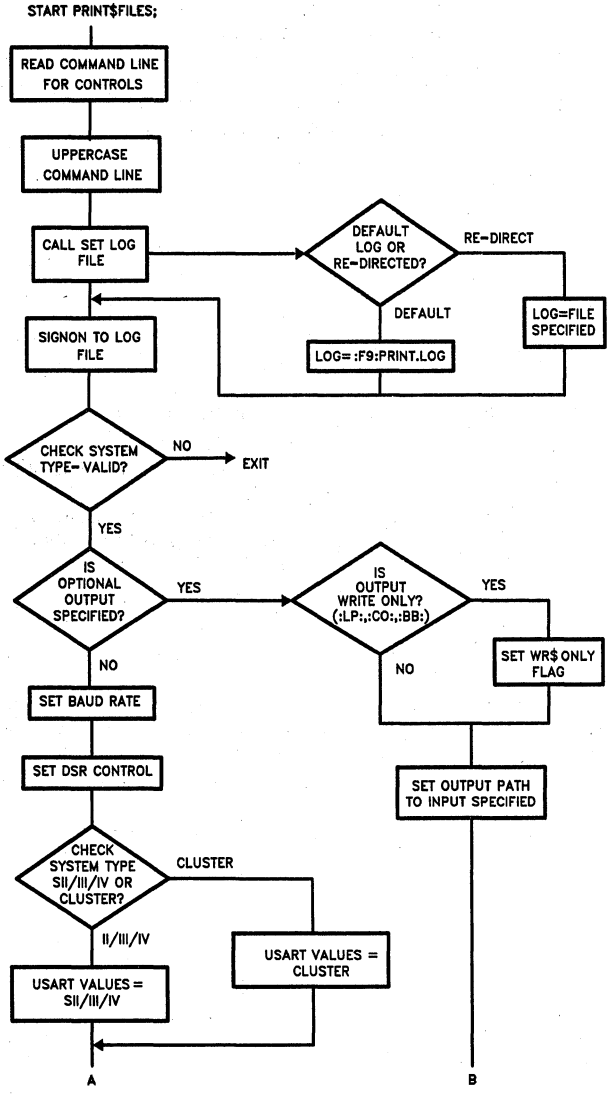
```
Example ISIS.INI:                ; ISIS.INI file for Series-II/III
                                 ; and ISIS Cluster
ASSIGN 8 to /w/prntspool.dir     ; copy files to be printed to :F8:
ASSIGN 9 to /w/printlog.dir      ; :f9:also contains the program
ISIS                             ; Invoke ISIS-IV - for Series-IV
:F9:PRINCE                       ; Invoke print spooler
```

CONCLUSION

Prince is a versatile utility that enhances the operation of standalone Series-IV systems or NDS-II networks. Prince is available separately from Intel's INSITE Library, (order PRINTS, Insite order code BG61) and is also available along with many other useful tools in Intel's NDS-II Software Tool Box.

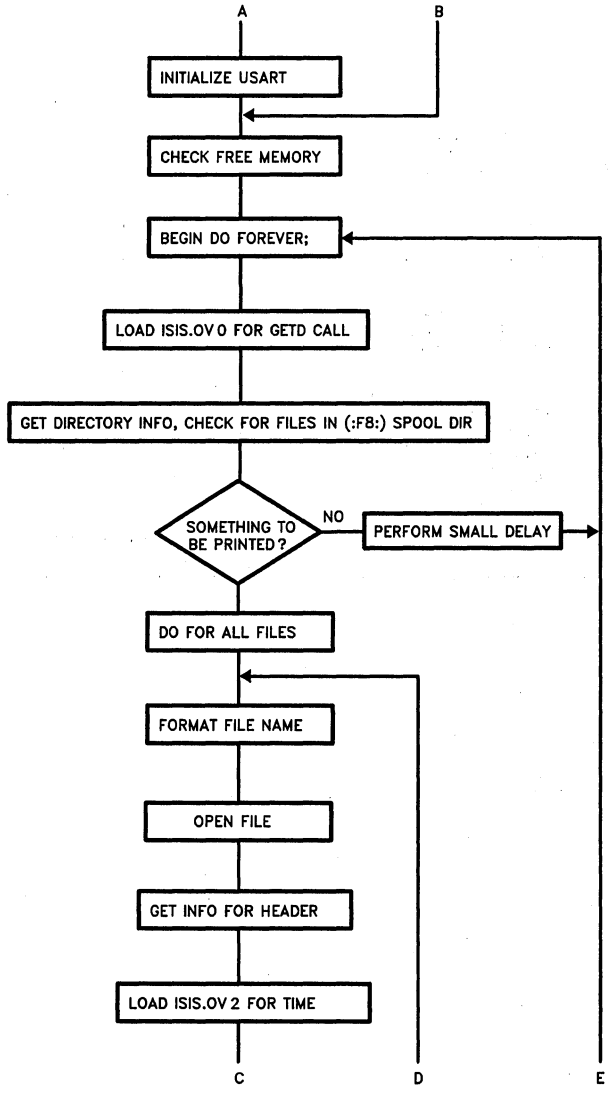
APPENDIX A

PROGRAM FLOW CHART



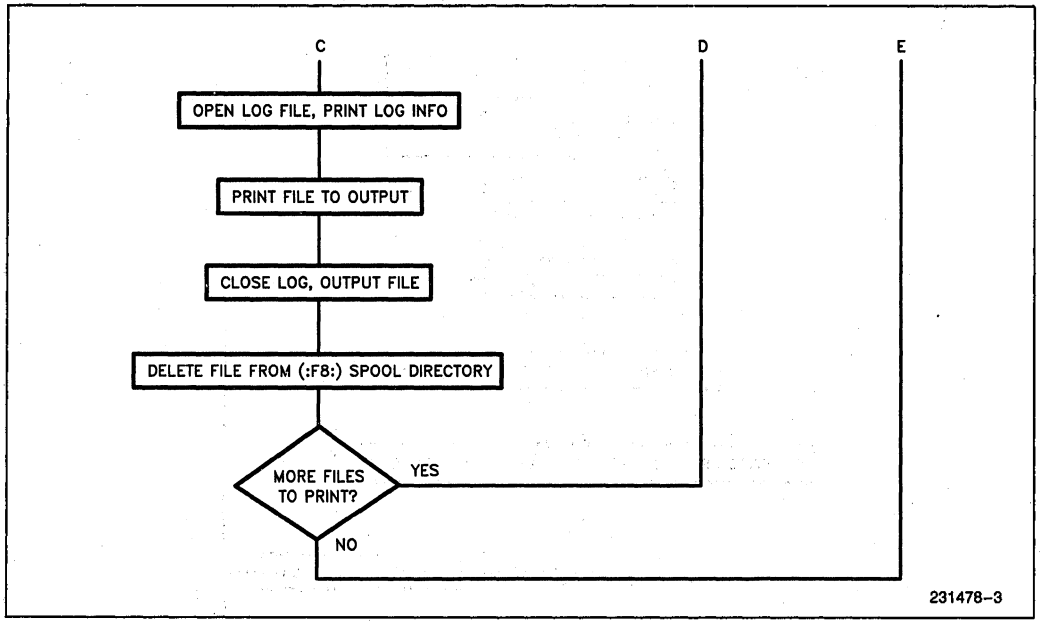
231478-1

PROGRAM FLOW CHART (Continued)



231478-2

PROGRAM FLOW CHART (Continued)



APPENDIX B
PROGRAM LISTING

```
PL/M-80 COMPILER    PRINT FILES              PAGE 1
ISIS-II PL/M-80 V4.0 COMPILATION OF MODULE PRINTFILES
COMPILER INVOKED BY:    :f1:plm80 :F3:prince.p80 PAGEWIDTH(80)
                $TITLE ('PRICE') PAGEWIDTH(80)
  1             Prince: do;
     $nolist include(:f3:procs.p80)
     $list
    /*************************** Program Start ************************/
    /*
     Read input line and set log file. Signon to log file, then determine
     system type. Check for optional baud rate control, and DTR/DSR control.
     Then set up the 8251 USART per the system type.
                */
450  1          call read (1,.input$buffer,128,.in$buffer$actual,.status);
451  1          do i = 0 to in$buffer$actual-1; /* UPPER CASE the input buffer.*/
452  2            input$buffer(i) = set$upper$case(input$buffer(i));
453  2            end;
454  1          call set$log$file;             /* Set up the log file. */
455  1          call print$message(0);         /* Signon to log file. */
456  1          if (high$byte<1) or (high$byte>5) then do;/* Exit if invalid */
458  2             call print$message(11);                 /* system type. */
459  2             end;
                /* Check if Line Printer specified in command.*/
460  1          call set$device;
461  1          if lp$flag <> true then do;/* If not line printer, set USART. */
463  2               call set$baud;         /* Set baud rate. */
464  2               call set$dsr;          /* Check for DTR/DSR control. */
465  2               if system$is$SII or system$is$SIV then do;
467  3                    serial$output=.s$serial$output; /* Use SeriesII/IV */
468  3                    wait$for$printer=.s$wait$for$printer; /* serial chn. */
469  3                    end;
470  2               call initialize$usart;
471  2               end;
                /* How much free memory below the Overlay base? */
472  1          limit = 0E87FH - .memory;
473  1          do forever;
                /* Any files to be printed ? */
```

```
474  2              call load(.('ISIS.0V0 '), 0, 0, .entry, .status);
475  2              call check$status(1);
476  2              start = 0;
477  2              call getd(8, .start, 1000, .actual, .dir$dump, .status);
478  2              call check$status(2);
479  2              if actual <> 0 then do index = 0 to actual - 1;
                 /* Something to be printed */
                 /* Format the filename */
481  3                    j = 4;
482  3                    do i = 0 to 5;
483  4                         if dir$dump(index).filename(i) <> 0 then do;
485  5                              filename(j) = dir$dump(index).filename(i);
486  5                              j = j + 1;
487  5                              end;
488  4                         end;
489  3                    if dir$dump(index).filename(6) <> 0 then do;
491  4                         filename(j) = '.';
492  4                         j = j + 1;
493  4                         do i = 6 to 8;
494  5                              if dir$dump(index).filename(i) <> 0 then do;
496  6                                   filename(j) =dir$dump(index).filename(i);
497  6                                   j = j + 1;
498  6                                   end;
499  5                              end;
500  4                         end;
501  3                    filename(j) = ' ';
                 /* Filename formatted, get the file */
502  3              do while status <> e$file$open;
503  4                   call open(.aftn, .filename, read$only, no$line$edit,
.status);
504  4                   end;
                 /* Get information for the header */
505  3                   file$table.aftn = aftn;
506  3                   call spath(.file$name, .file$table.device$number, .status);
507  3                   call check$status(4);
508  3                   call load(.('ISIS.0V1 '), 0, 0, .entry, .status);
509  3                   call check$status(3);
510  3                   call filinf(.file$table, 1, .file$info, .status);
511  3                   call check$status(6);
                 /* Load ISIS.0V2 to get the TIME! */
512  3                   call load(.('ISIS.0V2 '), 0, 0, .entry, .status);
513  3                   call check$status(5);
PL/M-80 COMPILER          PRINT FILES          PAGE 3
                 $eject
/* Print the header - form feed to printer, header to log file or :co:*/
514  3              call print(.(FF), 1);
515  3              call open$file$safely (.aftnl,.logfile,wr$only$log);
516  3              call write(aftnl,.header$1, length(header$1),.status);
517  3              call write(aftnl,.filename(4), (j-4),.status);
518  3              call write(aftnl,.header$2, length(header$2),.status);
519  3              call move(4, .zero$time, .dt.system$time);
520  3              call de$time(.dt.system$time, .status);
```

```
521  3          call write(aftnl,.dt.time(0), 8,.status);
522  3          call write(aftnl,.(' on '), 4,.status);
523  3          call write(aftnl,.dt.date(0), 8,.status);
524  3          call write(aftnl,.header$3, length(header$3),.status);
525  3          call write(aftnl,.fileinfo.owner(1), fileinfo.owner(0),.status);
526  3          call write(aftnl, .(cr,lf),2,.status);
527  3          call close (aftnl,.status);
                /* Print the file */
528  3          file$bytes = 1;
529  3          do while file$bytes <> 0;
530  4               call read(aftn, .memory, limit, .file$bytes, .status);
531  4               call check$status(8);
532  4               if memory(0) = FF then memory(0) = 0;
534  4               call print(.memory, file$bytes);
535  4               end;
                /* File has been printed */
536  3               call close(aftn, .status);
537  3               call check$status(9);
538  3               call open$file$safely (.aftnl,.logfile,wr$only$log);
539  3               call write(aftnl,.header$4,length(header$4),.status);
540  3               call print$size (file$info.len$hi,file$info.len$lo);
541  3               call write(aftnl, .(cr,lf),2,.status);
542  3               call close(aftnl, .status);
543  3               call delete(.filename .status);
544  3               call check$status(10);
545  3               end; /* Look for next file */
                /* No files to be printed , Wait a minute or so */
546  2          else do i = 0 to 60;
547  3               do j = 0 to 500;
548  4                    call time(10);
549  4                    end;
550  3               end;
551  2          end; /* of Do forever */
552  1     end Prince;
```

APPENDIX C
ISIS CLUSTER BOARD PREPARATION

Used with an ISIS Cluster, Prince can be driven from the ISIS Cluster board's serial channel, which is normally used for a terminal. With the addition of the special SERVER PROM for the ISIS Cluster, the Prince program can be automatically invoked and begin copying files from a network spool directory to a serial printer or other serial device, immediatley upon power-up. In this mode of operation, there is no console connected to the ISIS Cluster. Instead, the serial printer or other serial device is connected to the console port, and the SERVER PROM installed on the Cluster board automatically logs the Cluster board onto the NDS-II network, then submits the ISIS.INI command file. This command file contains the necessary assignments, as well as the Prince program invocation.

To prepare the NDS-II to support the Prince spooler, a username and home directory for the Prince program and SERVER prom must exist. To provide trouble free spooling, the username for the SERVER prom should be declared as a Superuser. This way, file access rights need not be set each time a file must be spooled, printed, and deleted.

The SERVER PROM image is included with the Prince program. The SERVER PROM image can be modified to change the username or password. Bytes 0FF0 to 0FFC (inclusive) are reserved for the SERVER username, password, and string terminator (00H). (Note that these are PROM addresses - this PROM image is moved to a different location in RAM on initialization.)

byte 0FFDH: PROM checksum

byte 0FFEH: resreved

byte 0FFFH: system ID (05 for Cluster - DO NOT CHANGE!)

The following strings are stored in the PROM image:

username: SERVER0<CR>

password: ©

checksum: 082H

If you change the LOGON name and/or password, remember to change the checksum, which is stored in byte 0FFDH. NOTE:The checksum is actually the two's compliment of the checksum calculated by the boot code. Thus, if you change the username to SERVER2 from SERVER1 (increment byte 0FF6H), you must decrement byte 0FFDH. Changing the PROM image can easily be accomplished using Intel's IPPS software, which is supplied with the iUP-200 PROM programmer.

CLUSTER BOARD PREPARATION - PROM BURNING

1. The PROM image is written in 286 format. Remember to initialize the iPPS properly.
2. Read in PROM image, modify LOGON strings, modify checksum, and burn a 2732 or 2732A.
3. Remove the old monitor prom from the Cluster board (A25) and place in the next-door socket (A37) for safe keeping (may be needed by CE).
4. Install new SERVER prom in A25.
5. Install a jumper between pins 67 and 68. This ties Clear-to-Send/ Request-to-Send together on-board. Prince uses XON/XOFF or XON/XOFF and DSR for control, so CTS/RTS is not required between devices.
6. If the printer to be used operates with the hardware DTR/DSR protocol, configure the serial cable such that the printer ready line comes in on pin 6 (DSR) of the serial cable to the Cluster board.
7. Refer to the ISIS Cluster installation manual for further Cluster installation instructions.
8. Set up ISIS.INI file to make assignments and invoke PRINCE, plug it all in and go.

FF0

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
53	45	52	56	45	52	30	0D	0D	00	spare			82	00	05	HEX
S	E	R	V	E	R	0	CR	CR								ASCII

APPENDIX D RELATED PUBLICATIONS

1. ISIS Users Guide 9800306
2. Network Development System II iMDX 450 210937-004
3. CP/M-80 on the NDS-II – Application NoteAP 253
4. ISIS Cluster Installation instructions .....

APPENDIX E ERROR MESSAGES

Prince returns messages and error conditions if certain external conditions prevent normal functioning of the spooler. All messages are directed to the log file, unless a fatal ISIS error occurs, preventing Prince from handling the error. ISIS will trap the fatal error and re-boot itself. Some error conditions that are not fatal ISIS errors are considered fatal by Prince, and after logging the error message in the log file, Prince will exit. The 18 messages given by Prince are as follows:

1. Serial printer driver xxx'

   Non-fatal message - The normal sign-on message at Prince invocation.

2. 'ISIS.OV0 not present on system disk'

   Fatal error, Prince will exit. ISIS overlay 0 must be present on :F0:for Prince to function properly.

3. 'GETD system call failed'

   Non-fatal - Prince uses this system call to determine the presence of files to be printed in directory :F8:. Possible causes for failure:a damaged or incorrect ISIS.OV0, file access rights, etc.

4. 'ISIS.OV1 not present on system disk'

   Non-fatal - Prince must load ISIS.OV1 to support the file$info system call. ISIS.OV1 is not on :F0:, or access rights are insufficient.

5. 'SPATH system call failed'

   Non-fatal - SPATH returns information about the file to be printed for log file status of the file.

6. 'ISIS.OV2 not present on system disk'

   Non-fatal - ISIS.OV2 is used to provide time and date information that is placed in the log file for all files printed, and all messages.

7. 'FILINF system call failed'

   Non-fatal - The file$info call returns file information to be used in the log file, such as the owner of the file, etc. If this call fails, meaningful file information will be absent from the log.

8. 'Could not open the file to be printed'

   Non-fatal - Most common causes of this malfunction are insufficient access rights to the file, or an invalid ISIS file name. Rename the file name, or give access rights to the Prince user.

9. 'Could not read the print file'

   Non-fatal - This error will occur only during the printing of the file, before the print has completed, but after the first successful open.

10. 'Could not close the print file'

    Non-fatal - Prince could not close the file just printed.

11. 'Could not delete the print file'

    Non-fatal - Prince attempts to delete the print file after printing. Most common cause of this error is insufficient (delete) access rights. Give Delete Access rights to the Prince user.

12. 'Unknown System Type - not supported'

    Fatal error, Prince will exit. Printfiles checks byte 0FFFFH to discern system type. Valid types are: 01 = Series-II 02 = Series-IV 05 = ISIS Cluster

13. 'BAUD control defaulted to 2400 baud'

    Non-fatal message - An attempt was made to set a different baud rate per the baud rate control (B=number) and number was invalid. Valid numbers are:110, 300, 600, 1200, 2400, 4800, 9600, 19200.

14. 'LOG control defaulted to :F9:print.log'

    Non-fatal message - An attempt was made to redirect the log file, but the log file name was greater than 14 characters. Prince does a gross check on the pathname specified to assure a correct ISIS file name.

15. 'DTR/DSR control activated'

    Non-fatal message - The DSR control is active.

16. 'DTR/DSR control not activated'

    Non-fatal message - An attempt was made to set DSR other than true - DSR not true is the default.

17. 'OUTPUT file defaulted to :F9:print.out'

    Non-fatal message - An attempt was made to re-direct the output file, but the file name was greater than 14 characters. Prince does a gross check on the pathname specified to assure a valid ISIS file name.

18. 'Could not write OUTPUT file'

    Fatal Error, Prince will exit. Prince could not write to the output file specified, so spooling is suspended.

intel®

APPLICATION NOTE

AP-244

October 1985

Distributed Job Control the Key to Increased Network Productivity

SRIVATS SAMPATH
DSO APPLICATIONS ENGINEERING

Order Number: 231480-001

INTRODUCTION

Large software projects and shorter production schedules generate the need for a more flexible and productive development environment, which allows users full access to all available resources.

Recognizing this need, Intel designed the Distributed Job Control (DJC) Facility into the NDS-II system. DJC allows currently idle networked development systems to be supplied to the network as public resources. This is essentially a remote job execution unit to which jobs can be sent by other users on the network. Remote job execution offers higher throughput and increased efficiency, since more than one computer on the network can be controlled and used by a single user. This ability to manipulate idle systems on a network and convert them into productive systems for other users directly translates into increased project productivity.

DJC consists of a set of system utilites that enable the NDS-II system manager to more efficiently run the network. When all idle systems on the network are allocated to other active users, the throughput and efficiency of the network dramatically increases. The Network Resource Manager (NRM) is the nerve center for the distributed job control system (DJC). All jobs are scheduled and queued by the NRM. The NRM also coordinates job cancellation and maintains a system log of job queue activity. DJC, with its powerful set of options, positions itself as an invaluable tool for increased network productivity.

WHY DISTRIBUTED JOB CONTROL?

The need for distributed job control (remote job execution) is apparent in a networked environment where a number of teams are working on different projects. With DJC, all idle systems can be channeled towards the particular time-critical project. As a result, the engineers have control over more than one system and increase their efficiency and productivity.

Figure 1 shows a typical NDS-II system. This network includes the NRM configured with two 84 MB Winchester drives, a 600-LPM line printer, three Series IV Microcomputer Development Systems (one of which has four cluster boards), two Series IIs with one cluster board each, and an assortment of ICETM and I²ICETM modules. Although the development systems are functionally similar, they are logically different as viewed by the NRM. Figure 2 illustrates the difference. Two teams are working on this network. Team 1 is an 8-bit development team, and Team 2 is a 16-bit development team. Both teams are meeting tight deadlines and need all the system time that they can get. One engineer working on the 8086 project is on vacation, as a result, one Series IV is underused. The other two Series IVs do not have anything running in their background. On an average of the 14 computers available to this network, (the Series IVs being counted as two each with foreground/background capabilities), only 10 are being used. The percentage use rate is only 60 percent when it should be close to 100 percent. Percentage use rate can be defined as:

(Total Number of Nonidle Systems/Total Number of Systems)*100

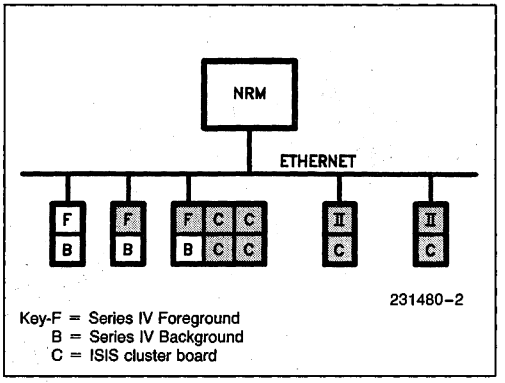


Figure 2. Non-Idle Computers are Shown Shaded

Meanwhile, the 8-bit team is trying to meet a very tight schedule and needs all the system time possible. This team requires a dedicated compile engine that will free its systems for interactive work, such as debugging and editing. DJC can help the 8-bit team by converting the idle machine and the backgrounds of the other two Series IV systems to productive work doers. This enables Team 1 to have all their compiles remotely executed while they concentrate on editing and debugging other modules. These remote execution units can serve both teams, since the Series IV can operate in both 8-bit and 16-bit modes. This results in definite increase in overall team and network productivity.

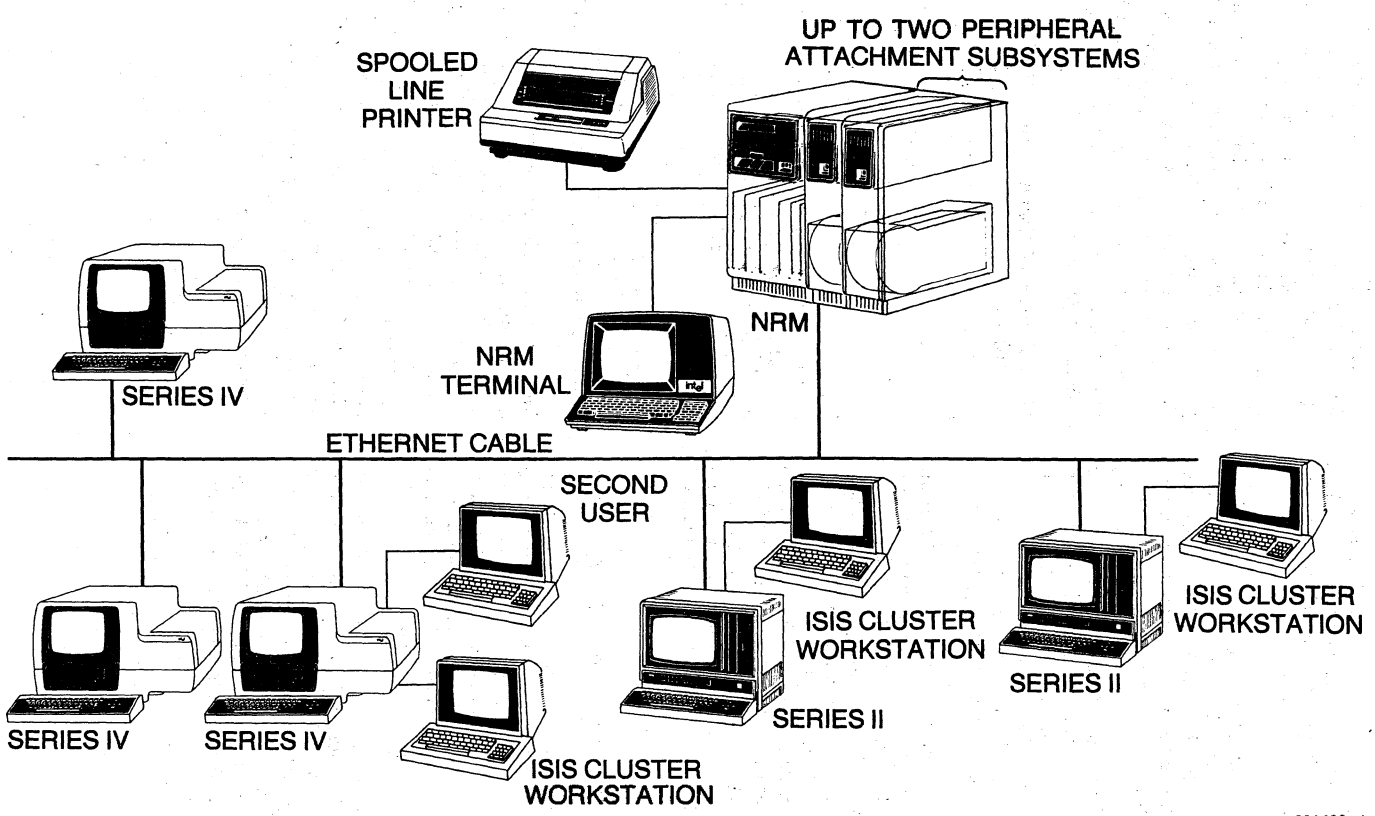


Figure 1. A Typical NDS-II Network

A CLOSER LOOK AT DISTRIBUTED JOB CONTROL

The DJC system recognizes that the network has three types of stations: the NRM, private workstations, and import workstations. The NRM is the nerve center of the DJC system, and it maintains all the state information of remote jobs and status of workstations. A private workstation is one that can send jobs to the NRM and have them executed at other workstations on the network. However, it does not accept jobs from the NRM. These are clasified as work generators. Examples of work generators are Model-800, Series II, Series III, Series IV and ISIS clusters.

An import workstation that can accept jobs from the NRM is called a work doer. Examples of work doers are Model-800, Series II, Series III, Series IV and ISIS clusters. Work generators and work doers use the same hardware. The software executing at the workstation determines if it is a generator or doer. The mix of generators/doers may be flexibly altered through the day/week/project to best suit the user's needs. Normally, when a workstation is first powered up or reset, it configures itself as a private workstation. (The workstation can also be configured to power up as an import station. See Appendix A).

A private workstation can be turned into a work doer for the network withth e IMPORT command. The import command informs the NRM that the private workstation is now capable of doing some type, or types, of jobs. A station remains an import station until the keys CONTROL and C are pressed from its keyboard. If an import station is executing a remote job when the CONTROL and C keys are pressed, it continues executing the job until the job is finished. Only then, does it return to private workstation mode. A Series IV stationthat supports both foreground and background partitions can import into either foreground, background, or both. Thus, a physical station can appear as two import stations to the DJC system.

DJC UTILITIES

Understanding Job Queues

Since the network consists of heterogeneous workstations, some type of mechanism is needed to match the job with the type of workstation it can execute on. This generated the concept of a job queue. A job queue can be envisioned as a waiting place for all work doers and a depository for workgener ators. Job queues are created and deleted using the QUEUE utility, and their statis is monitored using the SYSTAT utility. Each network can have up to 10 queues. The system does not support any predefined queues. While any name may be used, descriptive names based on the work doer's capabilities are recommended. 8-bit.q, 16-bit.q, and print.q are good names, while ONE.queue and compile.queue are not.

The system does not guarantee that a job is sent to a queue capable of doing the job. A Series II or ISIS cluster is only capable of doing 8-bit work. Therefore, the work generator is responsible for ensuring that the work doer chosen can execute the job. Multiple work generators can export to the same queue, and more than one work doer may import from a queue to get jobs done quicker.

Queues are maintained using files at the NRM. The DJC system uses these queue files to maintain job and queue status. These files are shared files and should not be tampered with.

Remote Job Execution

A job is scheduled for remote execution using the EXPORT command. An in depth discussion on the syntax and use of job queues is discussed in Chapter 5 (DJC Utilities). The export command must specify a job queue capable of executing the job. Export checks if the queue exists and displays an exception message if the job is not queued. It also warns the user if there are no importers (work doers) currently serving the chosen queue. The job will wait indefinitely at the job queue until a work doer is assigned to import from this queue.

The exported job may have to wait for some time before it can be executed, since other jobs arrived at the queue earlier might not have been executed. The queue is a first-in-first-out (FIFO) file.

In this way, the DJC system keeps a track of all jobs at the queue and executes them on a FIFO basis.

Example:

```
QUEUE NAME: 16BIT.Q              QUEUE NAME: 8BIT.Q

JOB NAME        STATUS           JOB NAME        STATUS

FOUR.CSD        WAITING          PRINT.CSD       WAITING
THREE.CSD       WAITING          LOCATE.CSD      EXECUTING
TWO.CSD         EXECUTING        LINK.CSD        DONE
ONE.CSD         DONE             COMPILE.CSD     DONE
```

When it successfully finds an importer for the specified job queue, the NRM will send the job over to that station. At the station, an implicit logon takes place using initialization options that are identical to a normal user logon. For example, the station will take the user's INIT.CSD file and execute it first and then execute the job. The environment set up at the import station is exactly as that at normal logon. The import station, a"reincarnation" of the user who exported the job, has access to all of files the user has. It looks just as if a user is inputting information at the import station. The only difference is that, in this case, input is from a file specified by the user exporting the job.

DJC Commands

DJC system on the NDS-II system has a number of commands that help theuser in effectively configuring an efficient remote job execution system. These are:QUEUE, IMPORT, EXPORT, CANCEL, and SYSTAT.

Each of these commands perform unique and important tasks to make the network distributed job control system a very productive and efficient solution.

QUEUE

QUEUE is a command for managing and displaying job queues at the NRM. The QUEUE command displays the name, number of jobs outstanding and the number of servers. After this information is displayed, the user is prompted to:

ADD DELETE LIST EXIT

ADD	option creates new queues. Up to 10 queues can exist at the NRM
DELETE	option deletes a queue
LIST	redisplays previously displayed information
EXIT	terminates QUEUE

For example:

>QUEUE <cr> will bring up the following display

```
NAME OF QUEUE       # OF SERVERS       # OF WAITING JOBS
16BIT.Q             2                  1
8BIT.Q              1                  0
PRINT.Q             1                  2
```

Anyone can delete these queues. Since there is no protection offered, the use of the QUEUE command should be restricted to a SUPERUSER. This may be accomplished simply by removing world access rights on QUEUE.86. However, a queue that has jobs waiting cannot be deleted; in this example, only 8BIT.Q can be deleted.

IMPORT

The syntax for the IMPORT command is the following:

```
IMPORT FROM queue ,queue ..... TO
BACKGROUND
```

where

queue	is a character string up to 14 characters long, which names the queues from which the import station can execute jobs. Up to five queues may be specified in the command line.
TO BACKGROUND	is an option that will execute the job in a background mode. This is a Series IV option only.

The IMPORT command declares the given workstation to be a public resource on the network, converting it from a work generator into a work doer. This public resource can now receive jobs from the various queues in the NRM. If the user enters the name of a queue that does not exist at the NRM, an exception message will be displayed. The queues are searched for jobs according to the order in which they were listed in the command line (left to right). If jobs are available on any queues, the import station starts processing them. The importing station starts by performing an implicit logon for the user, whose job is first at the head of the queue. Then it processes the commands within the command file. At the end of the command file, the importing station logs off the user and looks for jobs from the queues to process (left to right).

For example, the import station is configured to execute jobs from 16bit.q, 8bit.q, and iNDXutility.q. Initially, there is only one job on 8bit.q, so execution of it commences at the import station. During theexecution of this job, three more jobs arrive at 8bit.q and one at 16bit.q. The job on 16bit.q will be the next to execute, since the command line in import mode is always scanned left to right.

All output messages from the remote job, displayed on the screen of the import station, may be put in a log file if the LOG option is specified with the EXPORT command. When a station is in import mode, no local processing is possible. To reconvert the import station back to a private station, the user must enter CONTROL-C by pressing both the CONTROL and C keys.

EXPORT

The syntax for the EXPORT command is the following:

```
EXPORT pathname [parameters] TO queue
[{LOG/NOLOG}]
```

where

pathname	is a valid pathname for a command file
parameters	is a list of up to 10 parameters
queue	is the queue to which the job is to be sent
LOG, NOLOG	specifies whether a log is to be kept of all console activity on a mass storage device.

The EXPORT command allows a command file composed at one workstation to be executed on another workstation. The command file must be on a public volume, so that the import workstation can access it. An example of a public volume is a volume resident at the NRM and not a local mass storage device. If the queue does not exist at the NRM, an exception message is displayed and the job does not get queued. LOG, NOLOG determines whether a log file is to be maintained of all console activity, at the import station during the execution of that particular job.

The optional parameters specified in the command line are actual parameters to be substituted for the formal parameters embedded within the command file. In the example below, %0 will be replaced by the name of the source file specified in the command line. This way, one compile command file can handle programs with different names.

In this example, the command file links, and binds a "C" program.

```
Listing for:COMPILE.CSD
cc86      %0.c      debug
link86    %0.obj,                    &
          C/sqmain.obj,        &
          C/sclib.lib,         &
          C/small.lib,         &
          C/87null.lib         &
          to %0.86                   &
          bind                       &
          ss(stack(+800h),memory(+1200h))
>EXPORT COMPILE(/C--SOURCE--DIR/ISTIME) TO 16BIT.Q LOG
>EXPORT JOB NUMBER :0027H
```

This will export the job to the specified queue (in this case 16BIT.Q), print an export job number, and return control to the user, so that he or she may continue with productive work. Meanwhile, the import station acting as a server for 16BIT.Q will log on as the user, process his or her initialization file, and process the command file COMPILE.CSD. After all commands inCOMPILE.CS D have been processed, the import station goes back into waiting mode and waits for other jobs to be sent from the NRM.

CANCEL

```
CANCEL [BACKGROUND/REMOTE] queue
 {(job name) (# job number)}
```

where

queue — is the queue where the job has been queued for execution

job name — is the final component name of the remote job to be cancelled

(in the previous case, the job name will be COMPILE)

job number — is the assigned value of the remote job (this can be displayed by the SYSTAT command discussed next).

The CANCEL command is used to cancel a background or remote job. If the user wants to abort a remote job, the job name and job number must be entered. If the job name is selected and multiple instances of the job name are in the queue, the first one encountered is deleted (this may not be the first one queued). To avoid this, the unique name job number may be used,

Example:

```
> CANCEL REMOTE 16BIT.Q (COMPILE) will
result in
iNDX-W41 (V2.8) CANCEL VERSION V2.8
""COMPILE'' CANCELLED
```

The job name can be substituted with the job number. In this case, it will be 0027H (see example under EXPORT). Once the job is cancelled, the import station will execute the next job in the queue it is serving. If no jobs exist in the queue, it will go into a waiting mode for the next job.

SYSTAT

The syntax for the SYSTAT command is the following:

```
SYSTAT [{QUEUE/MY JOB } (queuename
[,....])] TO PATHNAME [EXPAND] [ALL]
```

where

queuename(s) — designates the name(s) of the queue(s) for which jobs are to be listed

pathname — designates the file where the information is listed

QUEUE — displays information for all queues, or for only those queues explicitly listed after the queue specifier. If this option is specified, the queuenames must be separated by commas.

MYJOB — parallels the queue option but lists information about jobs belonging onlyEXPAND specifies that complete information is displayed for each job. If expand is not specified, condensed information will be displayed for each job.

EXPAND — specifies that complete information is displayed for each job. If expand is not specified, condensed information will be displayed for each job.

ALL — displays appropriate information for all jobs in the specifiedqueue(s). If ALL is not specified, information is displayed only for waiting or executing jobs.

The SYSTAT command is used to display information about the DJC subsystem to the user. There are many options which are best discussed by examples.

Examples:

```
<SYSTAT <cr>
SYSTAT VERSION V2.8
QUEUE           # OF JOBS # OF IMPORT
NAME            WAITING   STATIONS
16BIT.Q         0         1
8BIT.Q          0         1
iNDXUTILITY.Q   1         0
```

This command displays the status of all queues and information on the number of jobs waiting and number of import stations serving any queue. No detailed information of actual job status is shown here.

```
<SYSTAT QUEUE
SYSTAT VERSION V2.8

JOB STATUS FOR: 16BIT.Q

JOB NAME  JOB #     OWNER     DATE      TIME        STATUS

No jobs are waiting or executing in this queue.

JOB STATUS FOR: 8BIT.Q

JOB NAME  JOB #     OWNER     DATE      TIME        STATUS

No jobs are waiting or executing in this queue.

JOB STATUS FOR: iNDXUTILITY.Q

JOB NAME  JOB #     OWNER     DATE      TIME        STATUS

PRINTFILE #0028     JOHN      11/30/84  16:20:22    WAITING
```

This command lists by queue all jobs waiting in a queue. This helps in quickly determining the status of jobs in a queue.

```
<SYSTAT QUEUE ALL
 SYSTAT VERSION V2.8

JOB STATUS FOR: 16BIT.Q

JOB NAME  JOB #     OWNER     DATE      TIME        STATUS

COMPILE   #1003     SRIVAT    11/30/84  12:12:30    DONE
COMPILE   #1002     SRIVAT    11/30/84  12:05:19    DONE
COMPILE   #1001     SRIVAT    11/30/84  11:30:20    DONE

JOB STATUS FOR: 8BIT.Q

JOB NAME  JOB #     OWNER     DATE      TIME        STATUS

COMP      #2008     WAYNE     11/28/84  18:12:30    DONE
COMP      #2007     WAYNE     11/28/84  15:10:20    DONE
LINK      #2006     NORI      11/27/84  10:10:23    DONE

JOB STATUS FOR: iNDXUTILITY.Q

JOB NAME  JOB #     OWNER     DATE      TIME        STATUS

PRINT     #2002     JOHN      11/30/84  18:10:20    WAITING
PRINT     #2001     SRIVAT    11/29/84  12:10:22    DONE
PRINT     #2000     WAYNE     11/29/84  10:10:10    DONE
```

This command lists the status of all jobs done or waiting in the queue since the queue was created. This is useful to the system administrator to study queue use.

This command lists the status of all of the jobs that users have submitted. Queue files are circular files 256 jobs long. For example, SYSTAT will display the last 255 jobs done or waiting. If the number of jobs exceeds 256, the first entries (jobs) into the queue file are deleted to make room for the new entries. The expand option, which displays all these jobs, is useful for system administration purposes. Information containing average wait time for each job, the average length of a job, may be obtained. The system administrator may use this information to install another work doer on a particular job queue, thereby optimizing the system for his or her particular environment. This queue can be deleted and then recreated once this information is recorded to clear this log of queue activity.

RECOMMENDATIONS FOR AN EFFICIENT DJC SYSTEM

The following discussion outlines recommendations for a useful DJC system for a network. A number of considerations should be made before your DJC system is implemented on the network.

A minimum of three queues should exist at the NRM:one queue for 8-bit work, one for 16-bit work, and the other an indxutility queue. Normally, one server is enough to serve these queues. However, if the load on any particular application increases, having a dedicated server for that queue will be more efficient.

In the example following, it is assumed that the high 16-bit workload requires a dedicated server for the 16-bit work being done on the network. Therefore, a dedicated server for 16BIT.Q has been generated using the IMPORT command. The other server imports from all three queues. Private workstations can also be converted into import stations whenever they are not being used. The background of one of the private workstations should come up in automatic import mode on powerup. This is discussed in Appendix D.

APPENDIX A
Looping in Export Files

Often, a job needs to be run continuously to do a predetermined task like checking mail. The versatility of the DJC system allows the user to do this in just one submit file. For example, an import station can export a job to itself or any other server on the network.

Example:

```
Mail Box(%0)
Save 1 msg.file
EXIT
checkexist msg.file
if %status ≠ 0 then
     report YOU HAVE MAIL IN BOX %0
end
export mailcheck (%0) to indxutility.q
nolog
end
```

This is an example of an export file that constantly checks for mail in a user's box. If a mail message exists, a message is sent to the user. Checkexist is a program that looks for a specified file and sees the value of %status to 1 if the file exists and 0 otherwise. Report is a utility that sends a message to the user's console. These utilities are explained in depth in the Application Note AP-245:"Using Command Files to speed program development."

The submit file is exported using the command:

```
EXPORT MAILCHECK(SRIVAT) TO
iNDXUTILITY.Q
```

The import station will execute this command file and later reexport the job back to the queue. This job will be put at the end of the job queue behind all others waiting at this queue. It will not totally dominate the job queue. The only way to stop MAILCHECK once it is running is to use the CANCEL command. There is no limit to the number of times an export job can be looped.

Conditional exports can also be done from within an exported job. The IF, THEN, ELSE constructs of command files are used. The above example is just one of the different ways DJC can be used. This feature is very useful if some remote job has to be done continuously.

APPENDIX B
REPORT.86

Since all exported jobs are remotely executed, the only method of monitoring their status is by using the SYSTAT utility. The need for a more interactive status reporter becomes more pronounced. REPORT.86 has been designed to answer this need. REPORT is a utility that should be included in all export files. The syntax for REPORT is the following:

```
REPORT <any message>
```

The following command file example shows how REPORT is used:

```
cc86      %0.c debug                        ; Compile the program
if %status <> 0 than                        ; If error in compile
     REPORT Error in compile of %0.c        ; Send message to user
else                                        ; and exit.
     REPORT Successful compile. Proceeding with LINK
          link86 %0.obj, &
          l/sqmain.obj, &
          l/sclib.lib, &
          l/small.lib, &
          l/87null.lib &
          to %0.86 &
          bind &
          ss(stack(+800h),memory(+2800h))
          if %status = 0 then                          ; Check for error in link
               REPORT Successful Link. End of Job.     ; If no error inform user
          else
               REPORT Error while linking.....         ; If error inform user and
          end                                          ; and exit.
```

REPORT.86 writes the message specified into the user's home directory in a file called REPORT.DAT. All the messages get appended on to this file. The ISIS and iNDX command line interpreters (CLI) have been extended to check for the existence of the file REPORT.DAT in the user's home directory. If the file exists, the contents of the file are displayed on the user's screen. The CLI then deletes this file. This gives the user the ability to constantly monitor the execution of a remote job. In the above example, if there was an error in compilation of the program, REPORT will write the message "Error in Compile of filesheck.c" and the remote job will terminate. This message will then come up on the user's terminal anywhere on the network, and the user can take corrective action. All messages are held until the user returns to the command level. They are not displayed instantaneously in the middle of an AEDIT session, for example.

The REPORT function used throughout the submit file will keep the user constantly informed on the success of all required operations. This results in greater productivity, since the user does not have to wait until the whole submit file is over and then examine the log file. The extensive use of the variable %STATUS in this submit file requires explanation. All Intel utilities, such as PL/M86, C86, and LINK86, exit with a UDI call DQ$EXIT(0) if the operation is successful and DQ$EXIT(n) if the operation was not successful (N is any number). This value passed into the DQ$EXIT call is stored in a variable called STATUS. This variable can be accessed from any submit file. Conditional operations can be done by accessing this variable.

APPENDIX C
CHECKTIME.C

Often, a program must be executed at a particular time. CHECKTIME.86 is a utility that allows a program to be executed at a particular time from within a submit file. The concept of STATUS and looping in submit files are used here again. This program obtains from the user a particular time, which can be set to be less or greater than system time. When the defined condition is satisfied, the program will exit with a return code of 1. Otherwise, it will exit with a return code of 0. For example, a match condition will exit with DQ$EXIT(0). This return code is passed on to the %STATUS variable that can be accessed by a submit file.

This program has been designed for doing jobs at a particular time of day in an export file.

The syntax for CHECKTIME.86 is the following:

```
CHECKTIME greater   22:23:45  or
CHECKTIME greater   22:23      or
CHECKTIME greater   22         or
CHECKTIME g         22:23:45  or
CHECKTIME g         22:23      or
CHECKTIME g         22
```

This will return with a return code of 1 if the system time is greater than the time specified and 0 for all other cases.

```
CHECKTIME less      22:23:45
CHECKTIME less      22:23
CHECKTIME less      22
CHECKTIME l         22:23:45
CHECKTIME l         22:23
CHECKTIME l         22
```

This will return with a return code of 1 if the system time is less than the time specified and 0 for all other cases.

For example, backup needs to be done only at a particular time, preferably during the night, when the system load is lighter. This can be done in an export file using the CHECKTIME.86 utility.

```
File: BACKUP.CSD

CHECKTIME g 22:59:00
if %STATUS = 1 then
     TREE BACKUP /APS--WO/USER.DIR/
SRIVAT.DIR/* to /APS1/SRIVAT.DIR
else
EXPORT BACKUP to iNDXUTILITY.Q
end
```

In the above submit file, CHECKTIME compares the given time with the system time. If a match is found, it will exit with STATUS set to 1; otherwise it will exit with STATUS set to 0. If STATUS is set to 0, a match has not been found and the submit file will export itself to the queue. In this way, the jobs get stacked up on the queue. When the CHECKTIME condition does get satisfied, the export file will back up all files in the volume

APS—WO/USER.DIR/SRIVAT.DIR to the /APS1/SRIVAT.DIR.

APPENDIX D
Configuring a Station to Come Up as an IMPORT Station

A workstation (Series IV) can be configured to power up as an import station through the SYSGEN command at the NRM. SYSGEN, restricted only to the SUPERUSER, will not allow any other user to modify the system configuration. Invoke SYSGEN by typing:

```
SYSGEN
```

SYSGEN will then clear the screen and display all the workstations on the network and their Ethernet addresses. Select the soft key labelled "Options". Next, select the node that has to come up as an import station. SYSGEN will then display another screen with one of the options being:

(7) Automatic Import to Partition 1 Partition 2

Select the partition needed to to come up in import mode. Both partitions can be selected. SYSGEN will next ask for queue names that will serve that import station. List the queues (maximum of 10) and then exit from SYSGEN. Reset the network and the Series IV will come up as an import station on powerup. To terminate import mode, do a Control-C at the import station keyboard by pressing the Control and C keys simultaneously.

intel®

APPLICATION NOTE

AP-246

October 1985

Setting Up an Efficient Hierarchical File System

WAYNE ROSEN
DSO APPLICATIONS ENGINEERING

Order Number: 231482-001

INTRODUCTION

Software development has become a team activity.

— Team members need an efficient file management scheme.

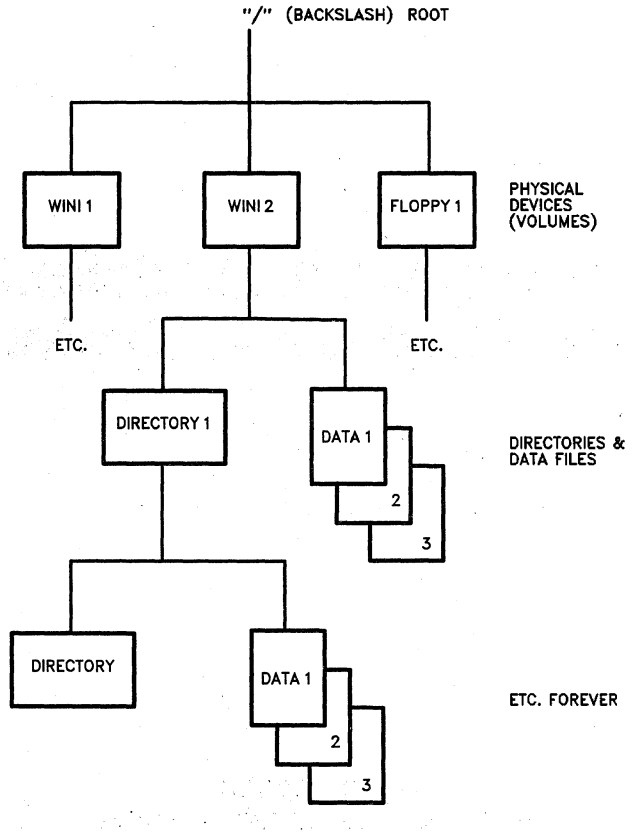
— Team members need to share common databases, but need to be protected against unauthorized file access.

Intel provides a superior hierarchical file system for file management, protection, and sharing in a totally controlled environment.

This Application Note is directed to the NDS-II or Series IV SUPERUSER who is setting up the system's hierarchical file system (HFS) and software development environment. We will be using some hypothetical products to illustrate our recommendations for this HFS.

Intel's NDS-II Network Resource Manager (NRM) and Series IV, running the iNDX operating system, encourage a logically constructed HFS. However, unless set up in a well-structured manner, an HFS can cause many problems. As software tasks grow larger and more complex, a properly structured file system will speed overall system development.

An HFS, (a tree-type file system opposed to a flat file system), promotes system protection and project partitioning and allows users to quickly find needed files. Figure 1 shows a stylized HFS.



NOTE:
1. A "volume" is a mountable, physical device. The system maintains device names for these, which should not be confused with the names you give them. For example:
Device name WFO is the 85 MB Winchester drive controlled by the disk controller board's first driver circuit.
Device name WF1 is the 84 MB Winchester drive controlled by the disk controller board's second driver circuit.

Figure 1. A Stylized HFS

ADVANTAGES OF AN HFS

The following are properties of Intel's HFS (See Figure 1).

- All files have a unique pathname starting from the root.
- Each physical device represents a directory at the root.
- Directories may contain data files or more directories.

Where the root (represented by "/") is the symbolic connection point for all physical volumes of an HFS.

To determine the physical volumes available to you as a user, enter the command "DIR /". For example:

```
DIR /
iNDX-W41 (V2.8) DIR V2.8
DIRECTORY OF /

FILE_NAME LOCATION  ACCESSIBILITY

SYS         remote    ;an NDS-II device
APS0        remote    ;     ""
APS1        remote    ;     ""
WLR.BACKUP  local     ;a local device on
                       my Series IV
```

A Logical Place to Put "Things"

A big advantage to using an HFS is the ability to group files according to user-defined relationships. Let's illustrate this important feature with a story.

We live in a disorganized universe. The laws of entropy tend to maintain and promote this disorganized state. Human beings fight the forces of entropy and try to maintain order in the small niche they carved out for themselves.

In our small corner of the universe, some people like "a place for everything and everything in its place;" that is, they expend some energy organizing their life and surroundings, while others do not bother. Joe Slobotnick (a very bad NDS-II manager) leans towards maximum entropy. Joe does not bother to expend the minimal energy necessary to maintain order on his system.

Joe is the only one in the world who might know where something is kept on the system. Occasionally, even Joe forgets where something needed is stored ("I swear I put that file in the TEMP3 directory along with the other prototypes") At this point, a mad, random-access search begins. This frantic search follows no known rules (like a binary or Shell sort), and no maximum search time can be calculated to tell Joe how long the search will take. Thus, the frantic search may take considerable time, may not be worth the effort, and may have disastrous side effects (like never finding the object of the search).

How much simpler it is for Joe, and for anyone working on his system, if a logical structure is imposed on the system. More importantly, how much simpler for all if a minimal effort is exerted to maintain this logical order.

Protection

Another advantage to using INTEL's HFS properly is file protection. iNDX provides the capability to protect critical files not only from malicious tampering, but from accidental changes (accidents do happen!). For example, all users (even SUPERUSER) should be able to use a compiler; but they should not be able to change or delete it.

Every file in INTEL's HFS has an "owner" associated with it. This owner is someone the SUPERUSER has defined as a system user (see the USERDEF utility). This owner controls access rights to his or her files by:

- Setting the individual access rights
- Setting the world's (the rest of the users on the system) access rights.

Superuser (including those people with secondary Superuser rights) can override the built-in protections and do anything to your files. This is a good reason to restrict the use of Superuser authority to the absolute minimum.

The Software Version Control System (SVCS), Intel's database manager, maintains another level of protection over that provided by the HFS. Features of this utility are discussed in Application Note AP 162 - a PMT tutorial.

Your Home Directory

You will want to keep your personal files in a protected directory that you own. This directory should be your home directory defined at USERDEF time.

When you log on, the iNDX operating system will automatically assign the logical names ' ' (the NULL logical name) and WORK: to your home directory.

```
LNAME Path
iNDX-W41 (V2.8) LNAME V2.8
LOGICAL      NAME PATHNAME
" "          /volume/USER.DIR/WAYNE.DIR
:WORK:       /volume/USER.DIR/WAYNE.DIR
```

Utilities will default their operations to the NULL logical name if no directory is specified, that is, the DIR command will give a directory listing of my home directory. PLM86 SOMEFILE.PLM will look for the file SOMEFILE.PLM in my home directory. In addition, the NULL logical name is the starting point to easily reference subdirectories located in your home directory. For example:

```
DIR MEMOS.DIR                ;MEMOS.DIR is
a sub-directory
iNDX-W41 (V2.8) DIR V2.8  ;in my home
directory
DIRECTORY OF /volume/USER.DIR/
 WAYNE.DIR/MEMOS.DIR -full pathname
FILE_NAME    FILE_NAME    FILE_NAME
MANPOW.D14   MAILD.323    UPGRAD.D12
CONF.305     SUNEWS.127   HFS.612
VACATION.N14
```

The NULL logical name can be redefined, but we do not recommend it.

The :WORK: logical name is used by various utilities (including translators) for workspace. We do recommend redefining this logical name. For example, if you are logged onto a network and have an Series IV with a Winchester (a fast device), defining :WORK: to some working directory on your local Winchester will speed you own processing and will reduce overall network Ethernet traffic. In fact, any temporary files created by you should be dispatched to this :WORK: directory. At the end of the day, you can clean up your workspace by simply deleting this working directory.

As you accumulate additional files in your home directory, you should break off related files into subdirectories, such as:

```
MEMOS.DIR          ;all memos
   KEYRESULTS.DIR;those memos that are
key results
```

In fact, we recommend that, other than needed initialization or configuration files, only put other directories in your home directory. Figure 2 is a sample home directory.

Under iNDX, the maximum directory name is 14 alphanumerics. Periods as readability delimeters count as one of the 14 characters.

Under ISIS-III(N), the maximum directory name is 6 dot 3. That is, six alphanumerics, a period and three alphanumerics for the optional extension. Also, it is convenient to name memos in the following form:

```
name.date_code
```

Where date_code = Mdd (three alphanumerics)

M = month code
(1-9 for Jan through Sep, 0 for Oct, N for Nov, D for Dec)
dd = day of the month (01 - 31)

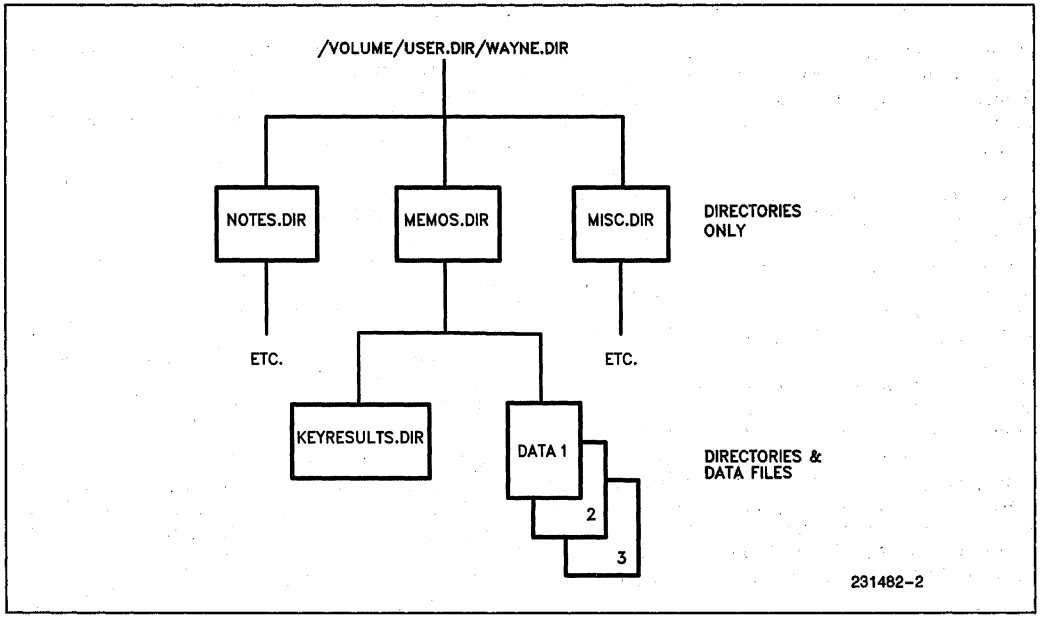


Figure 2. A Sample Home Directory

One Place For Tools

Tools (compilers, linkers, editors, etc.) should be kept in only one location. The world and owner should have DISPLAY ACCESS rights only.

This centralized tool directory is very convenient. Since there is only one copy of each of the tools, the system manager can guarantee that:

- Everyone is using the same version of the tools
- Tool updates need be made in one place only
- When a system generation is done, it can be proven that every module was generated using the same tools.

This last point is very important when it comes to system validation and certification. The Department of Defense (DOD), the Federal Aviation Administration (FAA), and others require such version control guarantees.

What takes place inside the computer when a command is invoked? For example:

```
PLM86 some.file Debug
```

Based on user and system-defined search rules (discussed later in this note), the operating system will begin searching specific directories for the PL/M86 compiler. If the compiler is not found in the first directory, the operating system will then search subsequent directories.

How does the operating system know if a file is in a directory? The operating system performs a linear search through the file entries until a match is found. Files marked as "deleted" and subdirectories count as entries too. The average "match time" is:

(½) × (total # of file entries) × (time to perform the match function)

To optimize system performance, you will want to:

- Make certain that the most frequently used utilities are located in the first directory searched
- Order the utilities in this first directory to minimize "match time" for very frequently used utilities,, such as AEDIT, DIR, and COPY (put them into the tools directory first)
- Minimize the total number of files in a directory (especially the tools directory). The maximum number of files that iNDX will allow in a directory is 1,024.

The Problem With Dir

Doing a directory (DIR) listing has to be one of the most frequently used commands of any computer system. However, doing a DIR on an unsorted directory that contains 756 files not only takes a lot of time, but limits the probability of locating all desired files. Your brain and eyes have a difficult time scanning pages of scrolling directory listings. Approximately 75 files (one page of a three-column DIR listing) is the recommended maximum amount of files to be scanned at one time.

How do you pick out subdirectories in a DIR listing? Unless you know the names of those subdirectories, an expanded DIR is the only way to find those directories. Expanded DIRs take a long time and degrade overall network performance. The answer? Where possible, suffix all directories with .DIR. Under iNDX, you have up to 14 characters to specify a directory name. This way, you will be able to find your subdirectories with a standard DIR listing. Or, you can search for occurrences of .DIR only. For example:

```
DIR directory_name FOR *.DIR
```

However, you might prefer having 756 files (or more) in your directories. When you look for a particular file(s), you will use wildcard characters and match for a particular pattern. Unfortunately, this also takes time. Then, there is the problem of possibly missing a needed file that does not quite match the search pattern (or getting other extraneous files that do match the pattern).

The bottom line is this: If you have a system that supports an HFS, use it wisely! And, be sure to:

GROUP RELATED FILES UNDER A
M E A N I N G F U L L Y
NAMED DIRECTORY !!

A SAMPLE PROJECT

The following sample project will help illustrate how to set up an NDS-II hierarchical file system. There are many projects being developed on our NDS-II network. The one project our group is working on is:

ROBOTWELDER a dual 186-based project

Our development environment has the following components:

- One NDS-II
- built-in tape cartridge (for back-up)
  - — One 35 MB Winchester (what we originally ordered)
  - — One 84 MB Winchester (we bought this unit when we needed more disk space)
- Two Series IVs
  - — One flippy/winny
  - — One flippy/flippy

- Two Series IIs (our original boxes, pre-network)
- One Series III
- Four ISIS cluster stations
- Intel in-circuit emulators as needed.

A MODEL HFS

Software is divided into three worlds:

*** program equivalent ***

TOOLS: editors, compilers, linkers, etc. (code)

USERS: you, me, and our projects (data)

SYSTEM: network operations (operating system)

In general, the tools operate on output from the users,

Users' files = Tools (users' files)

The system software is responsible for the operation of the computer. The system software manages the tools, the users' files, and itself.

Network operation = System (Tools, users' files, system)

Under iNDX, the system software is responsible for file protection, distributed job control, resource sharing, electronic mail, etc.

Using The Winchesters

Since we have this particular Winchester configuration (see "Sample Project"), we will use the 35 MB Winchester as the boot and system device. In fact, we will make this disk "read only". All of our tools (which have read permission only) will reside on this disk. We get a performance benefit by making this disk read only. A disk write takes approximately seven times longer than a disk read (reduced head thrashing). In our particular configuration, we gain added performance, since there are separate disk controllers for the 35 MB and 84 MB Winchesters (each type of controller can support up to four disks).

Since this 35 MB disk has our tools, contains our system software, and is the boot disk, we will name it something meaningful. For example:

```
/SYS
```

not simply /W or /WO.

Later, we will show why limiting this name to three alphanumerics is useful. An early hint:14 characters is the maximum that the SEARCH CUSP presently accepts.

* Wordstar is a trademark of Micropro.

** Multiplan is a registered trademark of Microsoft Corp.

You should always have at least 2 MB of spare room on the boot disk. iNDX creates many temporary files, some quite large, and puts them on this boot disk. For example, the SPOOL directory is the temporary holding space for print jobs. If you have sent 500k worth or listings (all at once) to be printed, you will need at least 500k free on the boot disk.

Tools

Let us take an in-depth look at these software tools. As far as our NDS-II and workstations are concerned, these tools are divided between:

8-bit tools

— These tools run on the 8085 microprocessor.

— The Series II, cluster board, and Model-800 can run ONLY these tools.

— The hosted 8-bit operating systems are ISIS and CP/M.

16-bit tools

— These tools run on the 8088/8086 microprocessors.

— The Series III and Series IV run these tools (in addition to being able to run all the 8 bit tools).

— The hosted 16-bit operating systems are iNDX and ISIS RUN.

As far as CP/M is concerned, Intel's development tools do not run under CP/M. CP/M is useful if you wish to include others into the development process for example, the professional using Wordstar* and the financial planner using Multiplan.** CP/M running on a workstation on the network is discussed in depth DSO Application Note AP-253:Adding Value to Intel's NDS-11 Development System Network with Network CP/M-80.

It is a misconception to believe that tools running on a 8-bit machine can only generate objects that an 8-bit microprocessor can use. Intel supplies a 8-bit PL/M compiler (i.e., runs under ISIS) that generates object code for an 8088/8086 microprocessor.

Due to the inertia of history or tradition, the 8-bit world is called:

```
ISIS.SYS (it would have been nice to
call it 8BIT.DIR).
```

However, we can call our 16-bit world:

```
16BIT.DIR.
```

16BIT.DIR

It is useful and very convenient to subdivide ISIS.SYS and 16BIT.DIR into logical groups. Under ISIS, we have great flexibility to do this. A Series IV can specify one additional Search path right now.

We would like to digress a bit and talk about the iNDX SEARCH CUSP. Under ISIS, when a CUSP is invoked or referenced by just its name, the command line interpreter (CLI) looks for the CUSP in the default directory, :F0:. For example:

```
  DIR (this is the same as typing
:F0:DIR)
```

Similarly, under iNDX, when a CUSP is invoked or referenced by just its name, the CLI:

- First looks in the system volume directory (in our example, this is called /SYS).
- If not found, the CLI then looks in the directory specified by the NULL logical name (''). The NULL Logical name is defaulted to your home directory and should not be changed.

It is convenient to have additional search paths. The current SEARCH CUSP gives us one more, which we can use to point to 16BIT.DIR:

```
SEARCH /SYS/16BIT.DIR
```

Thus, for our Series IVs, our search paths are:

```
 /SYS/16BIT.DIR
 /SYS                              ;boot
device
 /WORK/USER.DIR/home_directory     ;the
NULL logical name
```

The CLI will first search /SYS/16BIT.DIR, then the boot device, and finally our home directory. A CUSP found in any of our search directories with an entry in the menu compiler, will be able to:

— Access its syntax builder

— Complete command lines (FILL ON)

— Display HELP messages for any portion of the command line.

Why did we limit the system volume name to three alphanumerics? Currently, the search pathname, /SYS/16BIT.DIR, cannot be longer than 14 alphanumerics (including backslash delimeters). Our way to get around this limitation (if you have a longer volume name) is to use an LNAME.

```
  LNAME define 16BIT.DIR for
/long_volume_name/16BIT.DIR,
```

But then:

— You use one more LNAME

— This LNAME cannot be removed or redefined

— All users have to set up this LNAME.

ISIS.SYS

It seems that everyone sets up his or her own virtual floppy assignments in individual ISIS.INI files. We recommend that the group adopt a common standard and stick with it. We suggest the following:

```
                              *** 8-bit workstation ***
ASSIGN     :F0:to /SYS/ISIS.SYS
ASSIGN     :F1:to :F9:today's_project.DIR (your working directory)
ASSIGN     :F2:to /work_volume/PROJECT.DIR/xxx.DIR/DATABASE.DIR/MODULE.dir
     ;xxx is the project you're working on
     ;module is the particular piece of the project you're working on at the
      moment (if a large project)
ASSIGN     :F3:to /SYS/ISIS.SYS/LIB.DIR
     ;libraries, system $INCLUDE files
ASSIGN     :F4:to /SYS/ICE.DIR/which_ICE_you're_using.DIR
     .
     .
     .
ASSIGN :F9:to /work_volume/USER.DIR/home_directory.DIR
```

NEVER, NEVER re-ASSIGN :F9:, your home directory.

16-BIT WORKSTATION (SERIES III)

This is a Series III in RUN mode. In addition to the assignments above, add the following:

```
ASSIGN    :F7:to /SYS/16BIT.DIR/LIB.DIR      ;16-bit libraries
ASSIGN    :F8:to /SYS/16BIT.DIR              ;16-bit CUSPS
```

DIRECTORY STRUCTURE FOR TOOLS

We will put all interactive software tools under one of the following directories:

```
/SYS/ISIS.SYS, or
/SYS/16BIT.DIR
```

based on whether the tools are hosted on an 8-bit or a 16-bit processor.

Since commonly used $INCLUDE files (.H files for you "C" people) and libraries are nonexecutable and are usually brought in during a SUBMIT file, we can put them into subdirectories:

```
/SYS/ISIS.SYS/LIB.DIR, and
/SYS/16BIT.DIR/LIB.DIR
```

Project-related $INCLUDE files should be stored in the project database.

There are other files that have nothing to do with the host processor, such as configuration files that set up a terminal for an editor or PSCOPE or configure a terminal for a second user (Series IV). These we will put under:

```
/SYS/CONFIG.DIR
```

Our target processor (the processor(s) in our product) has nothing to do with the host processor that our development system is running. For this reason and for modularity and partitioning purposes, we have elected to break out all of the ICE emulator software and lump it under a separate directory:

```
/SYS/ICE.DIR
```

For convenience, certain CUSPs should be kept in the root directory of the boot device. We suggest that all you need to leave behind are:

```
LOGON    ;Never remove from the root
(see Appendix A)
DIR.86
LOGOFF.86
```

You may be wondering why we chose to remove as many CUSPs out of the root as possible. The root directory of the boot device is already cluttered with many system files; the total number can be substantial. (See Appendix 1 for a list of these system files.)

Figure 3 contains our suggested directory structure for tools.

```
/SYS                                    ;volume name
        /16BIT.DIR                      ;16-bit tools
                /LIB.DIR                ;libraries
        /ISIS.SYS                       ;8-bit tools
                /LIB.DIR                ; libraries
        /CONFIG.DIR                     ;configuration files
                                        ; for various terminals
                /AEDIT.MAC.DIR          ; AEDIT
                /STTY/CFG.DIR           ; Series IV users
        /ICE.DIR                        ;in-circuit emulators
                /ICE51.DIR
                /ICE.86.DIR
                /I2ICE.DIR
                    .
                    .
                    .
```

Figure 3. Directory Structure for Tools.

The first thing your INIT.CSD (Series IV LOGON initialization file) should do is:

```
SEARCH /SYS/16BIT.DIR
```

THE CASE OF THE MISSING CUSPS

You have looked everywhere on your disk, but you simply cannot find the EXPORT.86 file. Has someone deleted it? No! This iNDX CUSP, and other "hidden" CUSPs listed below, are built directly into the iNDX CLI.

***COMMAND FILE PROCESSING

BATCH	command file editor with syntax help
SUBMIT	executes the command file instantaneously
EXPORT	executes the command file at another time and place
BACKGROUND	executes the command file in my background now

***COMMAND FILE CONTROLS

COUNT	allows multiple executions of commands
REPEAT	allows multiple executions of commands
UNTIL	used by COUNT and REPEAT
WHILE	used by COUNT and REPEAT
ENDJOB	terminates this command file now
OPEN	opens a parameter file
READ	gets a parameter from a file
SET	allows (re-)definition of CLI variables
IF	conditional execution
ORIF	conditional execution
ELSE	conditional execution

***MISCELLANEOUS CUSPS

SEARCH	enables or lists CLI search paths
FILL	enables disables CLI command completion
LOG	saves all console output to a file
END	noop command for ISIS compatability
RUN	noop command for ISIS compatability
VIEW	scan a file (AEDIT-like interface)

The command file controls (IF, UNTIL, etc) are very useful for controlling command file (SUBMIT) execution. (Refer to the Application Note AP 245 for further discussion.)

Users

The next part of our software world is for the users. The first part of this world contains all our personal, home directories. The second part of this world contains all the project files.

The following example shows setting up this directory. We are using our 84 MB Winchester as our work disk and, therefore, are naming it WORK1.

PEOPLE

```
/WORK1                                  ;volume name
     /USER.DIR                          ;main directory
               /SUPERUSER.DIR           ;with release 2.8 of iNDX, the
                                         superuser
                                        ;gets his or her own home directory
               /WAYNE.DIR
                    /MEMO.DIR                ;as an example
                         /KEYRESULTS.DIR      as an example
                         /APNOTES.DIR         ;as an example
                    /BRIAN.DIR
                    /CHRIS.DIR
                    /DEBBIE.DIR
                         .
                         .
                         .
```

Figure 4. Setting Up Home Directories

USER.DIR contains all the home directories for everyone using the system. These directories should be assigned at USERDEF time;

```
USERDEF define WAYNE id 20000 DIR
/WORK1/USER.DIR/WAYNE.DIR
```

The names of the home directories should be the usernames (the name asked for at logon time) plus the suffix .DIR.

NOTE:
For normal operation, I will logon as WAYNE. When superuser priviledges are required, I can logon as SUPERWAYNE (previously defined as a secondary superuser). If you reserve logging on as SUPERUSER for the times you need to do a USERDEF, you can let the system protect you as it was designed to do.

PROJECTS

The next major directory is for our projects. Our group is working on ROBOTWELDER. Other groups are also using the NDS-II. Lump their project directories under PROJECT.DIR, too.

```
/WORK1                                  ;volume name
     /PROJECT.DIR
          /ROBOTWELDER.DIR              ;a main project
               /DATABASE.DIR
                    /SYSTEM.DIR         ;overall system database
                     Only put SVCS-type files in this directory.
                    /ASM.DIR
                     Only put SVCS-type files in this directory.
                    /DISPLAY.DIR
                     Only put SVCS-type files in this directory.
                    /database4.DIR
                     Only put SVCS type files in this directory.
                         .
                         .
                         .
          /yet--another--project.DIR
               /DATABASE.DIR
                    /SYSTEM.DIR         ;overall system database
                     Only put SVCS-type files in this directory.
                    /database.2.DIR
                     Only put SVCS-type files in this directory.
                         .
                         .
                         .
```

Figure 5. Project Directories

Put all files associated with your projects into a protected SVCS database. These file include source and objects, $INCLUDE files, MAKE files, and documents.

Break the database into many databases, each supporting a particular function or system block. In our example for the ROBOTWELDER project, one major system building block is called DISPLAY. We lump all files connected to DISPLAY into a subdirectory. DISPLAY is just one basic function of our slick new microprocessor-based ROBOTWELDER machine.

Each database should contain no more than 50-related modules to reduce database contention. The system database contains the files associated with overall project maintenance and organization. These files include block diagrams, documents and memos, timetables, and system integration test procedures.

You should not keep .LST files in a database or even on the Winchesters. They take up alot of space, and can always be regenerated when needed.

NOTE:
Only people using the network should have user names. Do not set up a user name, for example, called ROBOTWELDER.

We believe that if you do set up a project user name, such as ROBOTWELDER, with people logging on with this user name, you will lose control over your sources. It will be difficult to know who made changes on files. (Refer to the Applications Note AP 162 for further discussion.)

APPENDIX A

SYSTEM FILES

Appendix A contains a list of description of those files that the iNDX operating system maintains in the boot disk. Files beginning with r?DUP are duplicate files maintained by the operating system in case a disk "glitches." The original files are kept near the physical front of the disk, and the duplicates are kept near the back of the disk.

If the operating system detects that an original file is bad, a warning message will be printed and the duplicate files will be used. At this time, save all your files onto another device and reFORMAT the suspect disk. Otherwise, you might lose everything on your disk.

CAUTION: unless you really know what you are doing:

LEAVE ALL THE FILES LISTED BELOW ALONE!!!

DJC Queues

1. Information about DJC queues. These file can grow to contain a maximum of 256 job entries.

```
16BIT.Q          ;an import queue we created for 16-bit jobs
 8BIT.Q          ;an import queue we created for 8-bit jobs
   etc
```

HINT: Give your queues meaningful names, not like:

FOOWAFFLE or BOZO.

2. DJC header file.

```
DJC--CHK--PT     ;contains names of queues, which stations service
                 ;which queues, and the protocol version number
```

Series IV Temporary Files

1. SUBMIT jobs.

```
88--CMD--18          ;as an example
88--CMD--19          ;the naming format is 83--CMD--xy
     .
     .
     88--CMD--Y9
     etc.

     and

88--STACK--18   ;as an example
88--STACK--19
     .
     .
88--STACK--Y9
     etc
```

2. LNAMES (logical names the Series IV people are using)

```
88--LNAME--1
     .
     .
88--LNAME--J
     etc
```

Series IV and NRM CLI File

```
CLI_HELP                    ;large Series IV text file
PRM_HELP                    ;large NRM help text file

CLI_SYN_TBL                 ;used by the Series IV menu compiler
PRM_SYN_TBL                 ;used by the NRM menu compiler

HCLI                        ;Series IV error messages
PRM_HCLI                    ;NRM CLI error message
```

Series IV and NRM Logon Files

***Series IV

```
LOGON                       ;logon CUSP
                            ;delete this and no one can logon to Series IV
LOGON_HELP                  ;online HELP text
LOGON.SYN                   ;logon menu line
```

***NRM

```
PRM_LOGON                   ;logon CUSP
                            ;delete this and no one can log on at NRM)
PLOGON_HELP                 ;online HELP text
PLOGON.SYN                  ;logon menu line
```

Electronic Mail

1. MAIL.DIR is a directory. In this directory, electronic mail will set up individual mailbox directories.

```
MAIL.DIR
```

As an example:

```
WAYNE                       ;all of Wayne's messages will be put in this
                            ;directory
BRIAN                       ;Brian's mailbox (MAIL uses USERDEF usernames)
```

iNDX Operating System

```
OS88.RESIDENT               ;NRM operating system, iNDX.G11 (no overlays)
OS88.OVERLAY                ;empty (length 0)
```

Communication Software

```
SYSTEM                      ;a directory that contains the following files
    CONFIG                  ;SYSGEN info (including Ethernet addresses)
    MUSER.INFO               terminal configuration info
    COMMIDOS                ;communication info
    COMM3.X02               ;Ethernet communication software
    COMM3.X03               ;Ethernet communication software
    INDX.W31                ;OS downloaded to a SeriesIV/3
    INDX.W41                ;OS downloaded to a Series IV/4
```

Disk Maintenance

```
VERIFYFIX                   ;a directory used by the VERIFY CUSP
r?BADBLOCKMAP               ;which parts of the disk not to use
r?DUPBADBLOCK               ; duplicate
r?SPACEMAP                  ;which parts of the disk are used
r?DUPSPACEMAP               ; duplicate
r?FNODEMAP                  ;which directory entries are used
r?DUPFNODEMAP               ; duplicate
r?DUPFNODE                  ;contains all file information (redundant)
                            ; Note:an iNDX disk is RMX-86-compatible
r?DUPBLOCKZERO              ;a copy of the first block on the boot disk
r?VOLUMELABEL               ;name of the disk
```

System Files

```
UDF                         ;user-definition file (NAMES + PASSWORDS)
BAK.UDF                     ;a duplicate copy you should make every time
                            ; you do a USERDEF
HOME                        ;user names and home directories
BAK.HOME                    ;a duplicate copy you should make every time
                            ; you do a USERDEF
PUBLIC.UDF                  ;public versions of UDF NAMES and home directories
BAK.PUBLIC.UDF              ;a duplicate copy you should make every time
                            ; you do a USERDEF
SPOOL                       ;the directory behind :SP:device
r?ACCOUNTING                ;not currently used
r?ISOLABEL                  ;standard ISO label
r?RESERVED1                 ;reserved for future use
r?RESERVED2                 ;reserved for future use
```

APPENDIX B

Acronyms and Definitions

iNDX (i)ntel (N)etwork (D)istributed e(X)executive

Intel's proprietary 16-bit operating system that runs on the NRM and the Series IV.

ISIS (i)ntel's (S)ystems (I)mplementation (S)upervisor

Intel's proprietary 8-bit operating system that runs on the Model-800, Series II, Series III, Series IV and cluster stations.

RMX (R)eal time (M)ultitasking (E)xecutive

Intel's real-time proprietary system (8-bit and 16-bit versions).

CLI (C)ommand (L)ine (I)nterpreter

CUSP (C)onnonly (U)sed (S)ystem (P)rogram

NDS-II (N)etwork (D)evelopment (S)ystem, version II

NRM (N)etwork (R)esource (M)anager

PL/M (P)rogramming (L)anguage for (M)icroprocessors

Intel's system's implementation language.

PMTs (P)rogram (M)anagement (T)ools

SVCS (S)oftware (V)ersion (C)ontrol (S)ystem, one of the PMTs

An automated means of tracking changes to program source code, maintaining variants of the source and objects modules for a program, and recording access to the source and object modules in a multiprogrammer environment.

MAKE not an acronym, one of the PMTs

A program designed to generate a submit file that can be used to construct the most current version of the requested software.

intel®

APPLICATION NOTE

AP-247

November 1986

Adding Capability to the NDS-II System with Cluster Boards

CHRIS FEETHAM
DSO APPLICATIONS ENGINEERING

Order Number: 231483-001

INTRODUCTION

The ISIS cluster board (iMDX 581) was introduced into the NDS-II product line to reduce dramatically the cost of a personal workstation. It achieved this goal and gave the network numerous expansion opportunities. All of the applications discussed in this note are available through the NDS-II toolbox.

ADDING ADDITIONAL USERS

The cluster board is a single MultibusR board with an 8085-2 processor, 64 K of RAM, an RS232 serial port, and other supporting circuitry. Figure 1 shows a block diagram, and a complete circuit diagram is included in Appendix A. A cluster board may be installed into any master slot of a network Model 800 or Series II, III, or IV development system to support an additional network user via a dumb terminal. This low-cost method of adding extra users to the network served as the primary motivator for the development of the cluster board.

With the exception of Multibus slot, some power, and access to the host's Ethernet controller board, the cluster board uses none of its host development system's resources. The cluster board does not slow the host, which generally has no knowledge of its presence in the system. A host may support multiple cluster boards. Figure 2 shows the maximum number that may be added to each host system.

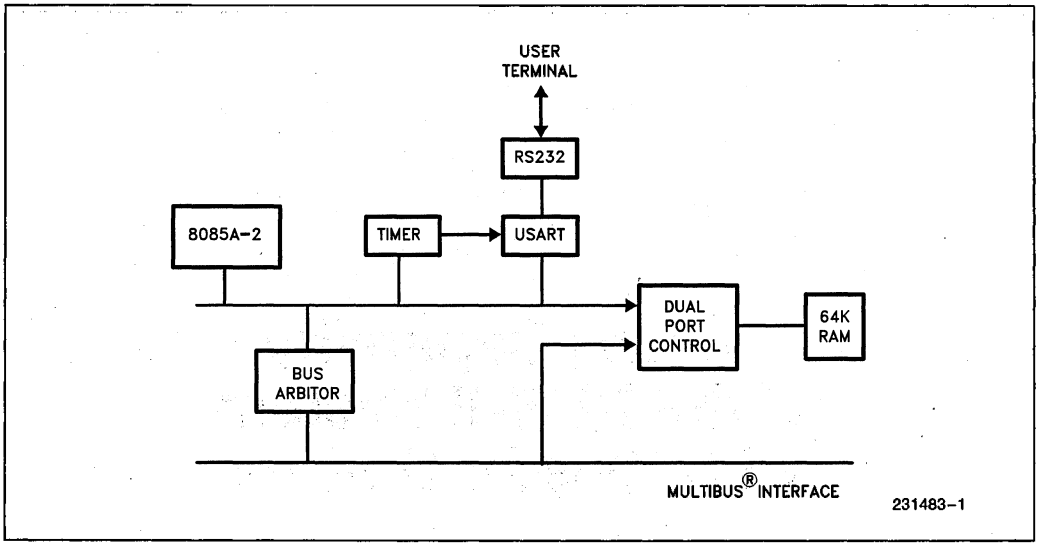


Figure 1. Cluster Board Block Diagram

During initialization of the host system, an operating system is loaded from the network resource manager (NRM) into the RAM of the cluster board. While ISIS operating system was chosen to ensure compatability with previous development environments, CP/M-80 may also be used (see AP 253). During operation, ISIS accesses data files and programs from the protected hierarchical file system of the NRM using the Ethernet controller boards. Access to local host devices, such as floppy disks or Winchester disks, is not permitted.

In normal use, a dumb CRT would be connected to the RS232 port of the cluster board. The user would then have access to all of the 8-bit network tools, including full-screen editors, program management tools, and electronic mail. While some 8-bit compilers are also available, the cluster board is generally used for interactive work supportng the engineer (or the support staff). Access to 16-bit advanced tools is available via the Export facility of the networks' distributed job-control system, where the cluster user may generate a job using

local tools and then request its execution on a more capable system upon the same network. This productive shared-tool environment is described further in AP 244.

It is not mandatory to install a dumb CRT. In fact, any RS232 device will suffice. The possibilities are endless, since RS232 is one of the few standards in the electronics industry today. Although this article will discuss various applications, the solution is general in nature, and anysystem with an RS232 interface could be connected to the cluster board.

REMOTE NETWORK OPERATION

Figure 3 shows the connection of an Intel iPDSTM portable development system. The iPDS system is especially suited to 8-bit microprocessor applications development. It has many tools for individual development but does not include advanced network tools, such as electronic mail or program management. In this application, the iPDS system is at a remote site, and a modem link connects the iPDS system to the NDS-II network.

A dumb terminal emulator program called REMOTE has been written for the iPDS system. This program, as part of the network toolbox, includes autodialing a Hayes smart modem. While running in terminal emulation mode, the iPDS can access all facilities of the network, including electronic mail and distributed job control facilities. REMOTE also includes a file-transfer protocol that enables data transfer between the iPDS system and the NRM.

If the iPDS system is at a service location, you need a diagnostic program from the NRM. Or, the iPDS could have data gathered from a remote site to be analysed back at base. The possibilities are endless.

System	Maximum Clusters
Model 800	2
Series II	3
Series III	1
Series IV	3
Expansion Chassis	4

Figure 2. Adding Cluster Boards to Host Systems

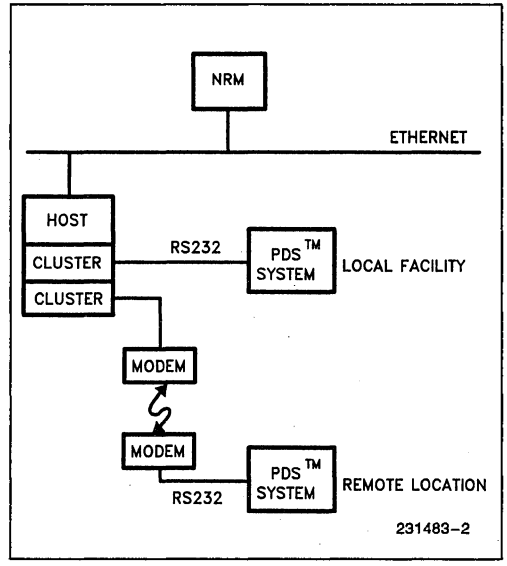


Figure 3. Attaching the iPDS™ System to the Network via an ISIS Cluster Board

ADDING AN ADDITIONAL PRINTER

An additional printer is often required on an NDS-II system. Letter quality printers are popular and their RS232 connection makes them a natural for connection to the cluster board. One problem - how does an output device such as a printer LOGON to the network and initiate file transfer from file to paper.

Server is a slight modification of the standard cluster PROM - it includes a PROM based console to solve the initialization problem. After power-up the LOGON program calls the console input routine to input the user name and password - within server a user name and password is supplied from PROM (Refer to the AP-242 — Additional printer support for the NDS II — for more complete information.)

Once logged on the system executes an initialization file ISIS.INI from the users home directory. In this server example a program that never exits will be chosen - PRINCE, a versatile serial printer driver, is such a program. Following initial drive assignments PRINCE polls a directory looking for files, once a file is identified it is copied to the serial printer and then deleted - simple but most effective.

AUTOBOOT CLUSTER BOARD

BOOTUP is an extensively modified cluster PROM. Rather than rely upon a host system to provide its operating system BOOTUP allows a cluster board to load its own ISIS operating system from the network. Following power-up BOOTUP initializes an SBC550 Ethernet controller and then logs on to the NRM under a predefined name of ISIS. Once logged on BOOTUP loads its operating system from the network. Before passing control to the user BOOTUP seeks out and initializes any other cluster boards also installed within the same chassis.

BOOTUP provides the network user with a low cost method of adding software developers - take any iSBC chassis, add an Ethernet Comm set and a cluster board containing the BOOTUP PROM and the system is complete. Up to seven additional cluster boards may be added to provide a very low cost eight-user environment as shown in Figure 4. BOOTUP also supports the server concept. The BOOTUP PROM is provided with the Network/Series IV Toolbox product.

CONCLUSION

I hope I have explained some of the versatility of the ISIS cluster board. Think of it as a universal interface board between the complex multi-protocol world of Ethernet and the straight forward start-data-stop world of RS232. I am sure this will prompt many new applications for the product - feel free to experiment and benefit from your findings.

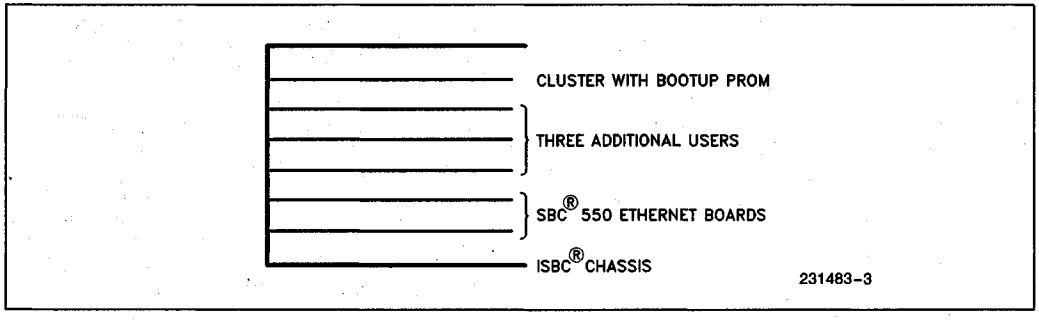
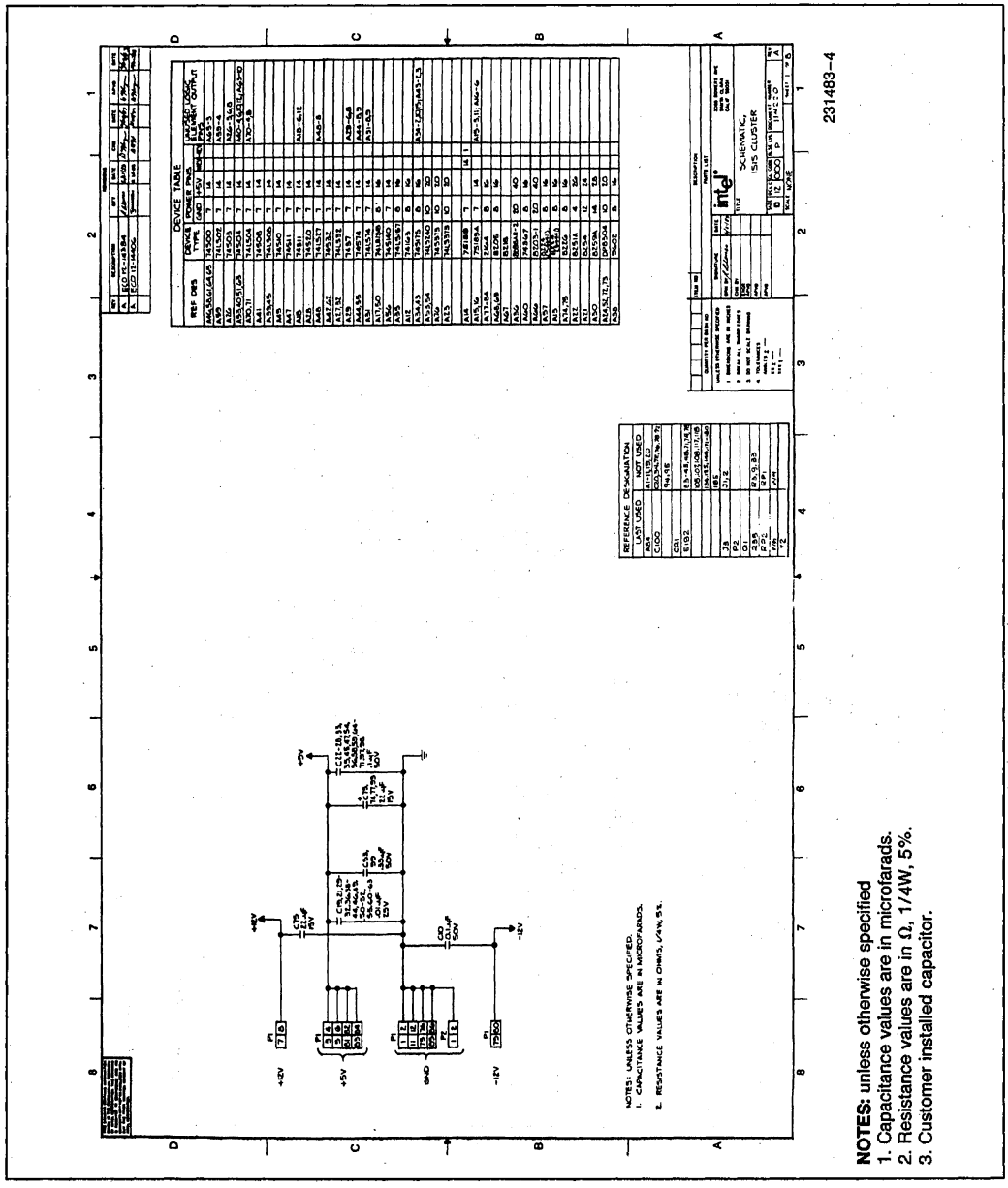


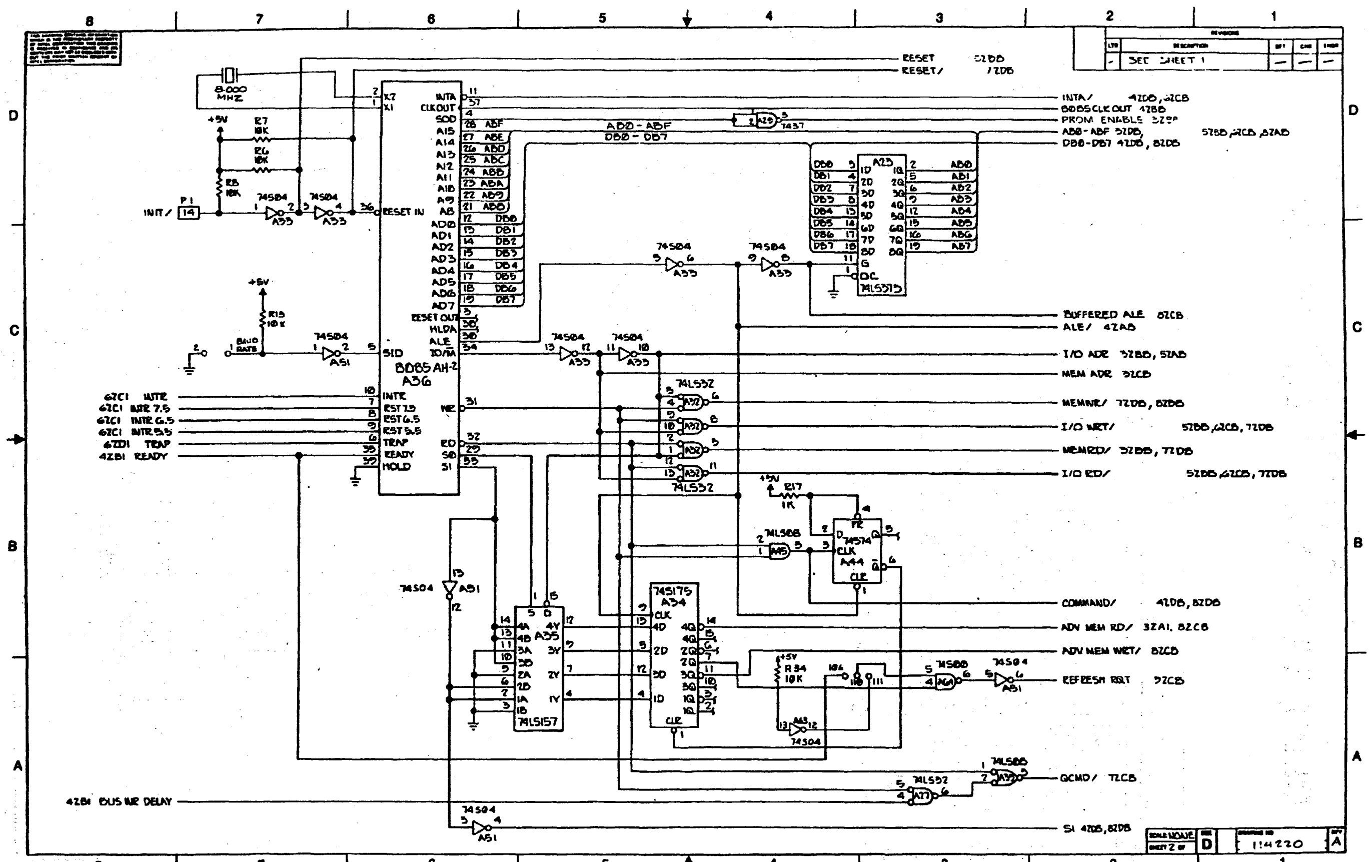
Figure 4. BOOTUP Allows a Low Cost iSBC® Chassis to Act as a Host for Software Developers

APPENDIX A
CLUSTER BOARD ADDS CAPABILITY

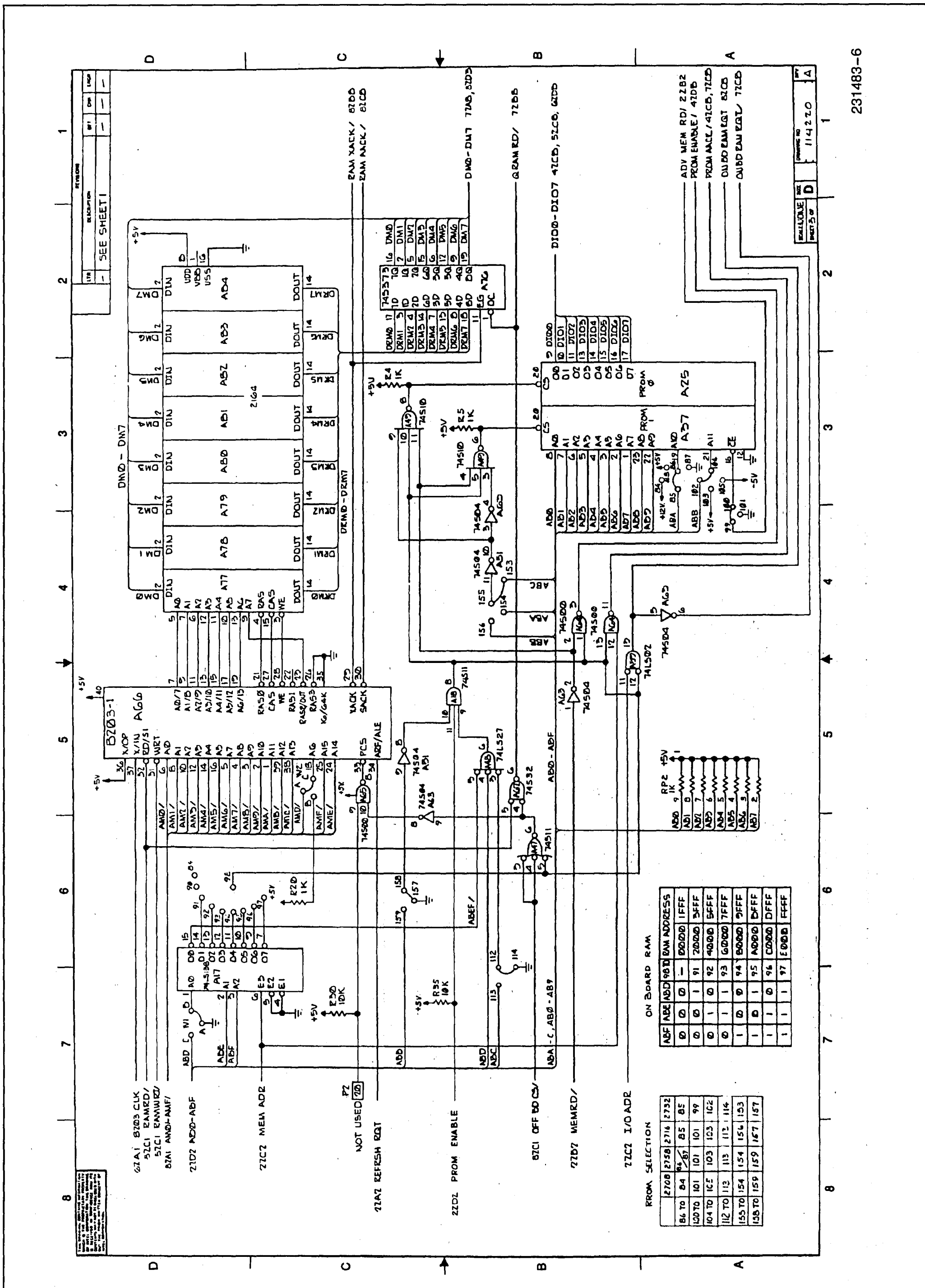


231483-4

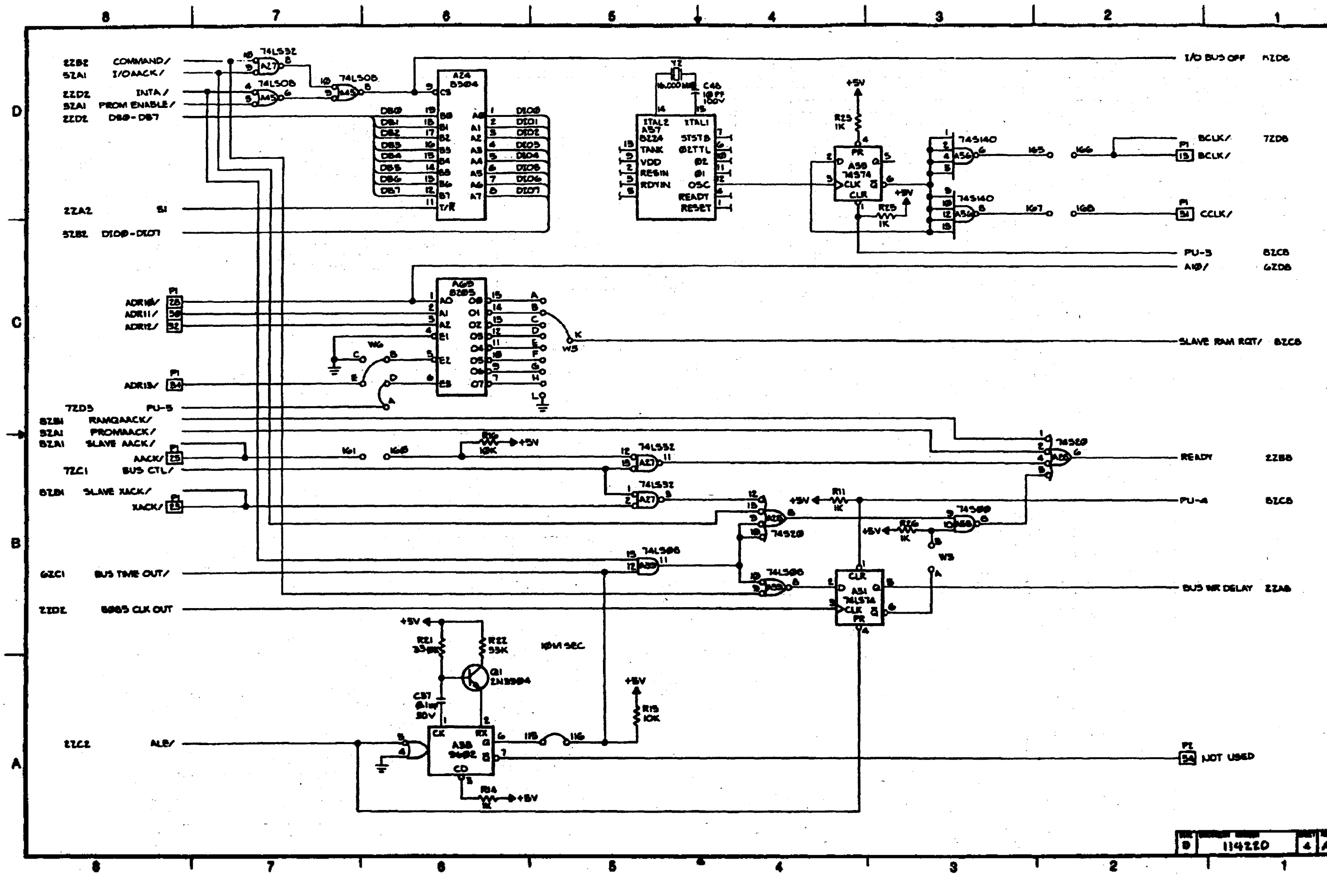
NOTES: unless otherwise specified
1. Capacitance values are in microfarads.
2. Resistance values are in Ω, 1/4W, 5%.
3. Customer installed capacitor.


RESET 5ZDB
RESET/ 7ZDB
8.000 MHZ
INTA/ 4ZDB, 6ZCB
8085CLKOUT 4ZBB
PROM ENABLE 5Z?
AB8-ABF 5ZDB, 5ZBB, 6ZCB, 8ZAB
DB0-DB7 4ZDB, 8ZDB
AB8-ABF
DB0-DB7
INIT/
8085 AH-2
A36
74LS373
BUFFERED ALE 8ZCB
ALE/ 4ZAB
I/O ADR 5ZBB, 5ZAB
MEM ADR 3ZCB
MEMWR/ 7ZDB, 8ZDB
I/O WRT/ 5ZBB, 6ZCB, 7ZDB
MEMRD/ 5ZBB, 7ZDB
I/O RD/ 5ZBB, 6ZCB, 7ZDB
6ZC1 INTR
6ZC1 INTR 7.5
6ZC1 INTR 6.5
6ZC1 INTR 5.5
6ZD1 TRAP
4ZB1 READY
74LS32
74LS74
A44
74LS08
COMMAND/ 4ZDB, 8ZDB
ADV MEM RD/ 3ZA1, 8ZCB
ADV MEM WRT/ 8ZCB
REFRESH RQT 7ZCB
74S175
A34
74LS157
A35
QCMD/ 7ZCB
4ZB1 BUS WR DELAY
S1 4ZDB, 8ZDB
SEE SHEET 1
114220

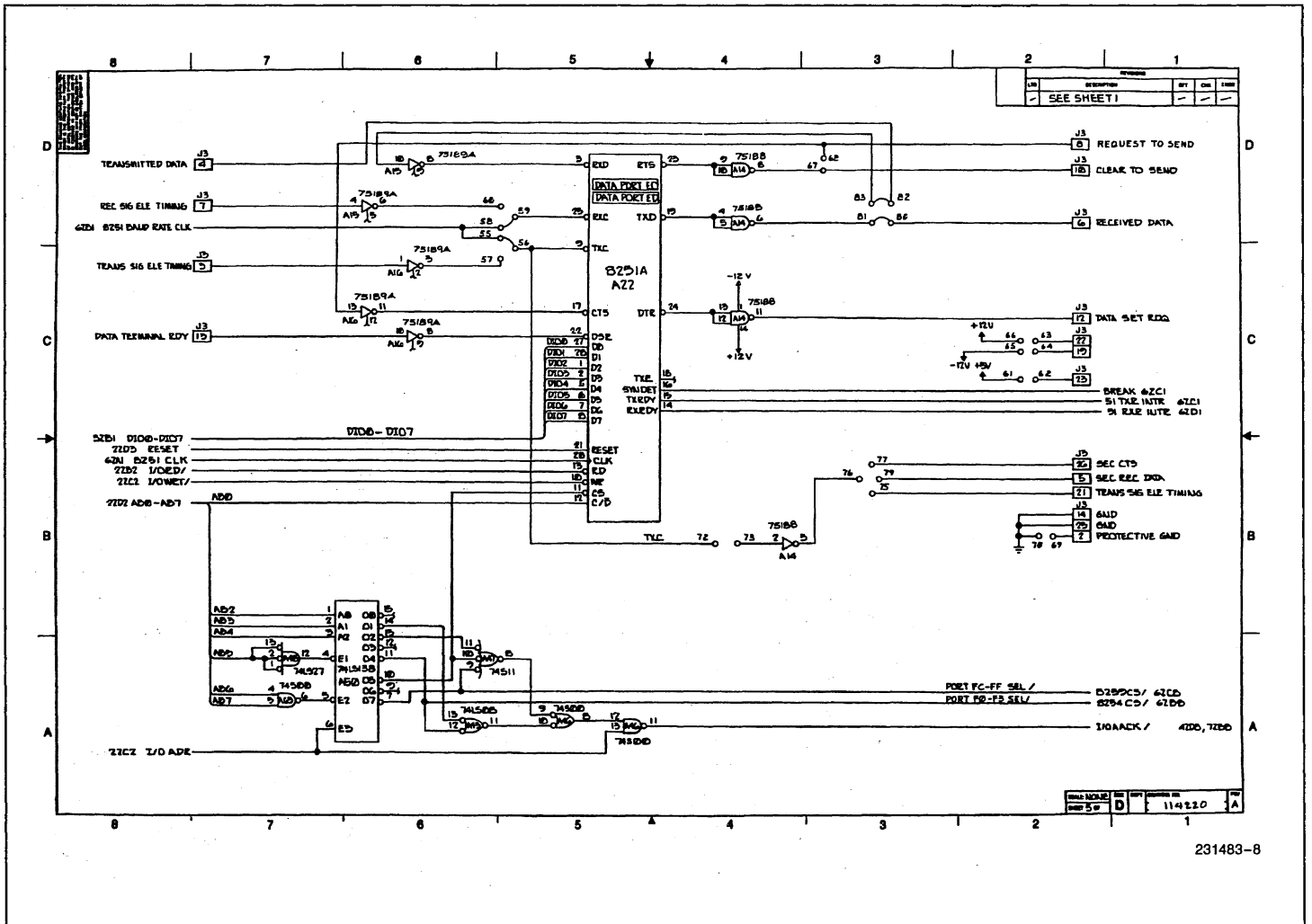
231483-5

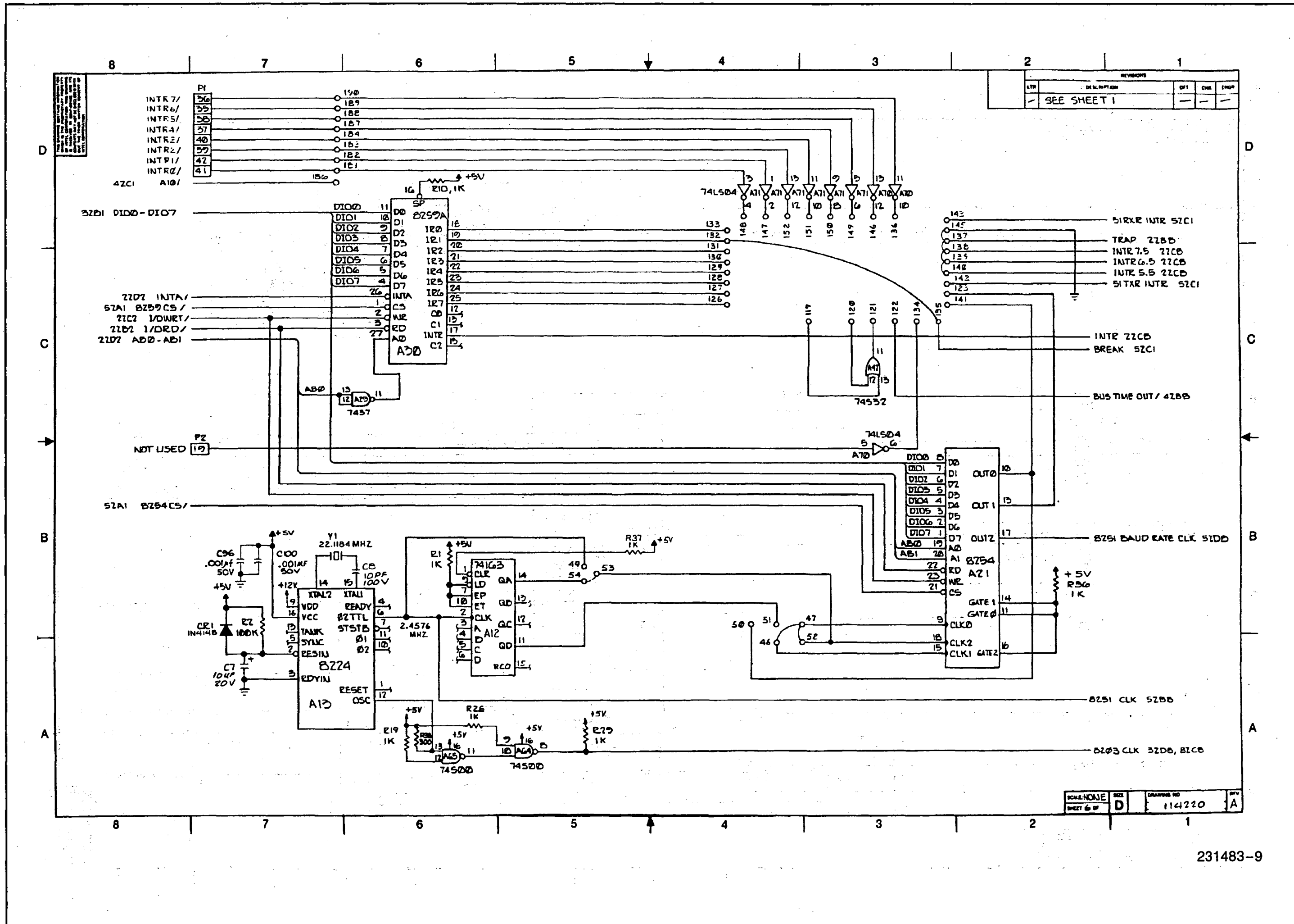

8203-1 A66
2164
PROM 0 A25
PROM 1 A27
74S373 A76
DIN0 - DM7
DIDO - DIO7 4ZCB, 5ZCB, 6ZDD
QRAMRD/ 7ZBB
RAM XACK/ 8ZDD
RAM AACK/ 8ZCD
PROM SELECTION
ON BOARD RAM
NOT USED
PROM ENABLE
REFRESH RQT
MEM ADR
I/O ADR
SEE SHEET 1

231483-6

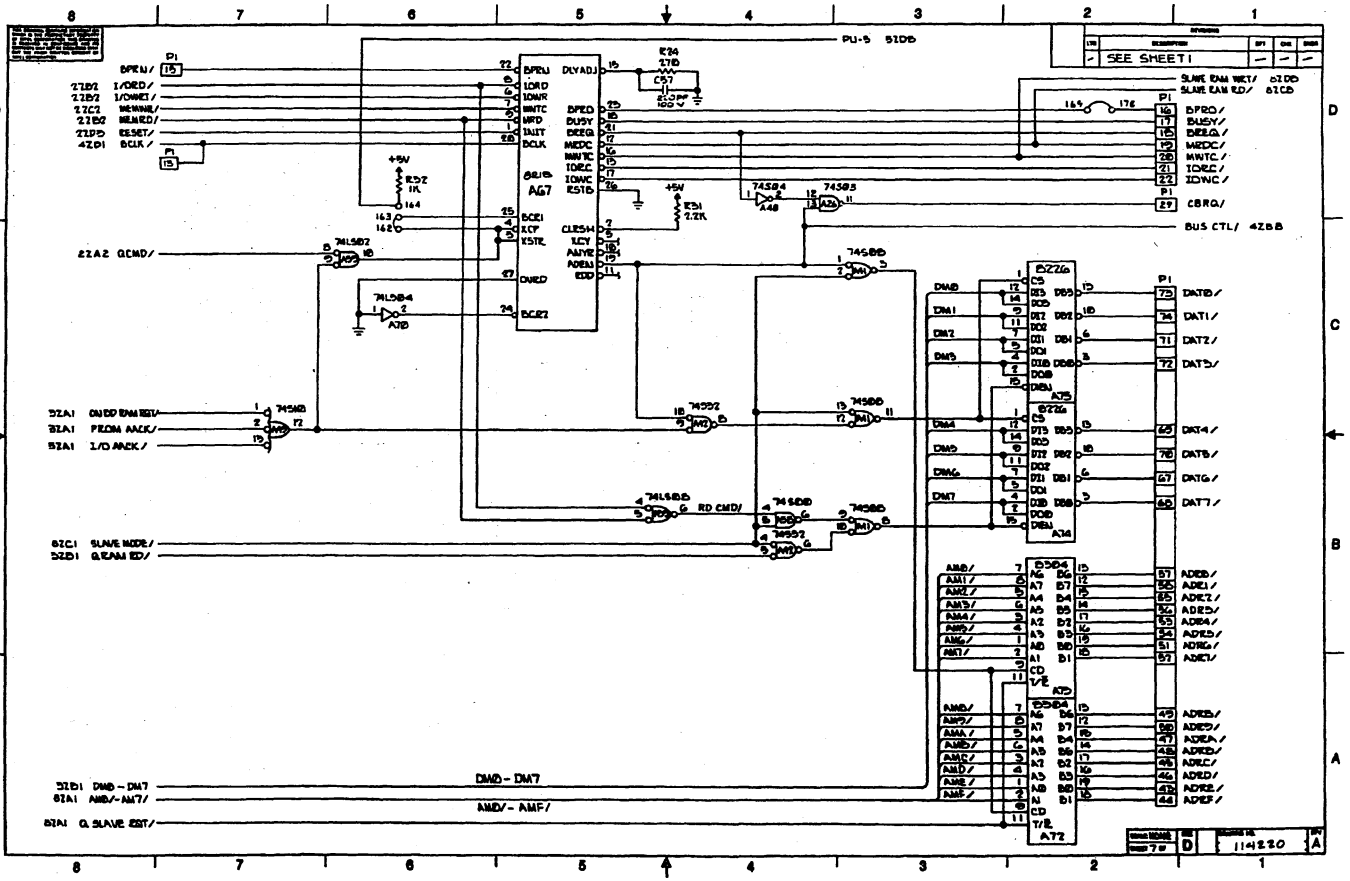

COMMAND/
I/OAACK/
INTA/
PROM ENABLE/
DB0-DB7
DIO0-DIO7
I/O BUS OFF
BCLK/
CCLK/
PU-3
ADR10/
ADR11/
ADR12/
ADR13/
SLAVE RAM RQT/
RAMOAACK/
PROMAACK/
SLAVE AACK/
AACK/
BUS CTL/
SLAVE XACK/
XACK/
READY
PU-4
BUS TIME OUT/
8085 CLK OUT
BUS WR DELAY
ALE/
NOT USED
114220

231483-7

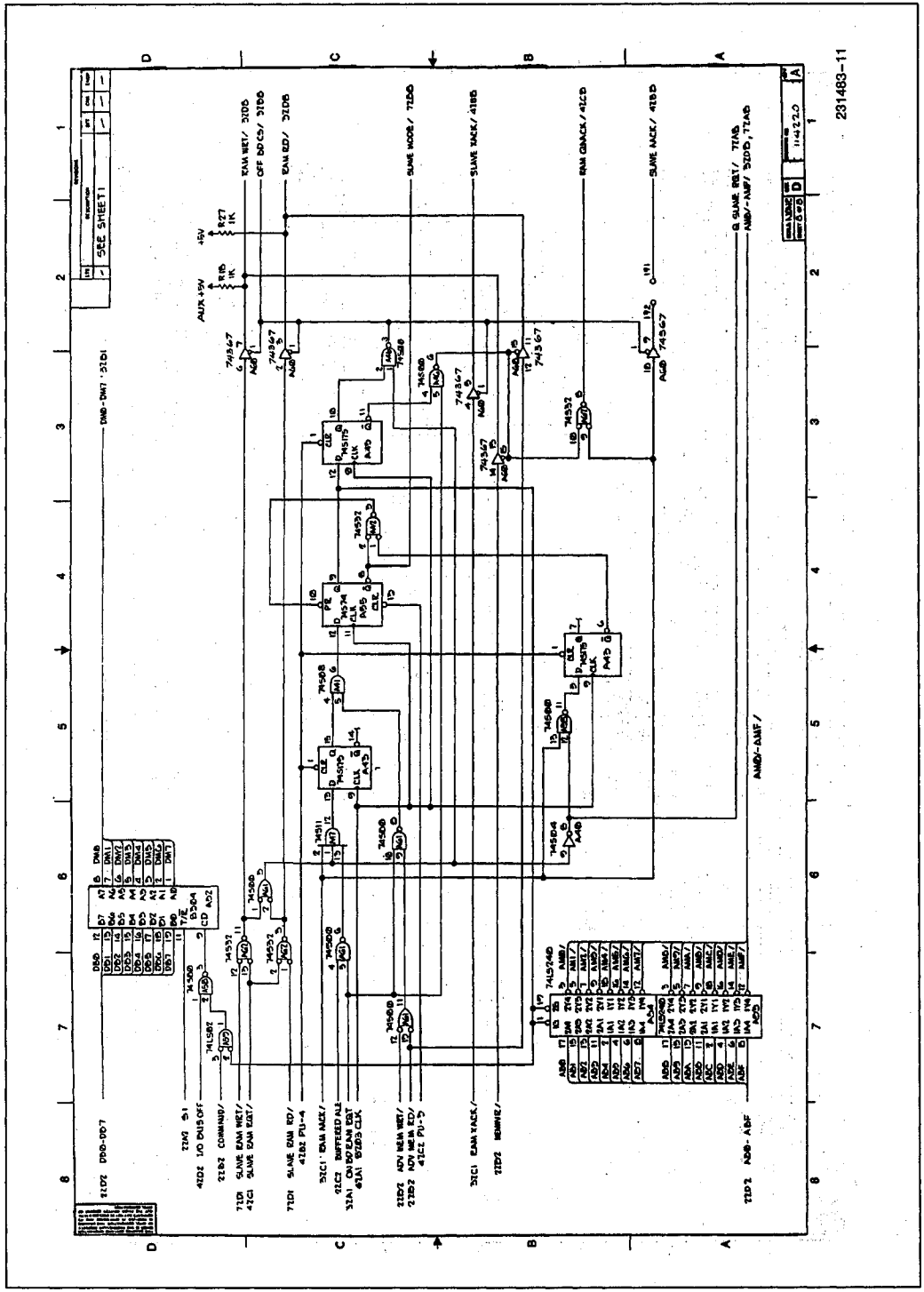

SEE SHEET1
TRANSMITTED DATA
REC SIG ELE TIMING
6ZDI 8251 BAUD RATE CLK
TRANS SIG ELE TIMING
DATA TERMINAL RDY
REQUEST TO SEND
CLEAR TO SEND
RECEIVED DATA
DATA SET RDY
BREAK 6ZC1
SI TXE INTR 6ZC1
SI RXE INTR 6ZD1
SEC CTS
SEC REC DATA
TRANS SIG ELE TIMING
GND
GND
PROTECTIVE GND
8251A
AZZ
RESET
DATA PORT EC
DATA PORT ED
PORT FC-FF SEL/
PORT F0-F3 SEL/
I/OACK/
231483-8


SEE SHEET 1
8259A
A30
8254
A21
8224
A13
74163
A12
74LS04
74S32
7437
74S00
22.1184 MHZ
2.4576 MHZ
INTR 22C8
BREAK 52C1
BUS TIME OUT/ 42B8
8251 BAUD RATE CLK 51D8
8251 CLK 52B8
8203 CLK 32D8, 82C8
NOT USED
114220

231483-9


SEE SHEET1
BUS CTL/ 4ZBB
DMA0
DMA1
DMA2
DMA3
DMA4
DMA5
DMA6
DMA7
DMA0 - DMA7
AMA8/ - AMAF/
RD CMD/

231483-10


SEE SHEET1
231483-11

intel®

APPLICATION NOTE

AP-278

October 1986

Integrating the PC AT Into the Intel Development Environment

SRIVATS SAMPATH
DSO APPLICATIONS

Order Number: 280272-001

INTRODUCTION

In recent years the Personal Computer has become a popular vehicle for delivering computing power to the engineers. IBM's latest offering the Personal Computer AT (Advanced Technology) incorporating INTEL's 80286 16-bit microprocessor, brings about a high level of technical sophistication into a personal computer. The power, speed and memory addressability of the 80286 microprocessor is now available to the user to do tasks which at one time could be run only on a mini or mainframe.

Intel has recognized this growing trend and has introduced translators, debuggers and networking for the PC AT. The same tools that have in the past, been offered only on Intel's proprietary development systems are now available on the PC AT under PC-DOS 3.0 or greater. Language translators are available for the complete spectrum of Intel microcontrollers (MCS®-51 and MCS-96 families) and microprocessors (8086, 80186, 80286). 80386 tools will be introduced early 1987. This is the first time powerful software debuggers like PSCOPE and TSCOPE, hardware debuggers like I²ICE™ have been made available on a personal computer. Intel already supports a broad range of workstations which may be networked to form a productive network. This application note discusses the multiple methods in which the PC AT may be integrated into this development environment.

All software discussed in this application note is available (except where listed) in the Network Toolbox, Part No NDS2TLB 2.0, as discussed in Appendix D.

THE INTEL DEVELOPMENT ENVIRONMENT

The Intel development environment consists of iPDS™ Personal Development System, Series-II's, Series-III's, Series-IV's all of which can be operated standalone or networked through a file sever using the NRM (Network Resource Manager). Intel also supports industry standard hosts such as the DEC VAX and now we include the PC AT as a supported host. Figure 1 shows all combinations of the Intel development environment while Figure 2 illustrates the possible inclusion points of the personal computer.

This appliction note assumes that the reader is familiar with the current Intel Development Environmanet. For more information on the Intel Development Envionment please refer to:

1. AP Note Number AP-244 *DJC A Key To Increased Network Productivity*
2. AP Note Number AP-245 *Creating an Efficient HFS*

These application notes are also available in the 1986 DSO Handbook, Order Number 210940.

This application note deals with integrating the PC AT into an existing Intel development environment. The enclosed details will allow the reader to choose the method that best suits the project. This applications note will discuss three methods of data transfer—serial interconnect, media transfer and networking. Of these, serial transfer is the most universal, media transfer is the most straightforward and Ethernet transfer is the most efficient.

SERIAL INTERCONNECTS

The simplest method of integrating the PC AT into the Intel development environment is through serial interconnects. This method is inexpensive albeit slow. Serial interconnects also allows the user the flexibility to use modems to interconnect with systems which are not in the same location. It allows for both terminal emulation and file transfer at speeds of up to 9600 baud. Serial connections have always been a popular method of linking different computers. Unfortunately this has resulted in a variety of serial communication software being developed that are incompatible over different operating systems. Intel recognized this incompatibility and decided to advocate serial communication software that was compatible over a range of operating systems. The KERMIT file transfer protocol developed by Columbia University, addressed all the needs of serial interconnects, and resolved most inadequacies in previously available serial software. It also solved the multiple operating system incompatibility issue. KERMIT, is available for all hosts shown in Figure 1. Note however that all KERMIT implementations are not equal. The specification details a minimal set of and also specifies numerous additional features which may be added. Currently, the ISIS implementation on the iPDS system, Series II, III and Series IV is a minimal set while the VAX and PC implementations are both extensive. The XENIX and iRMX versions are good and being improved. KERMIT is public domain software and cannot be charged for. Versions for the Intel hosts are available from Insite for a small disk copying fee (see Appendix C).

The following paragraphs discuss the different methods of integrating the PC AT using the KERMIT file transfer protocol. Intel, with help from a number of customers presently using our systems, has developed KERMIT software for the following systems:

- iPDS—Serial port
- Series-II or III—Serial port
- Series-IV—Serial port 2
- ISIS cluster board—Directly into the ISIS cluster board

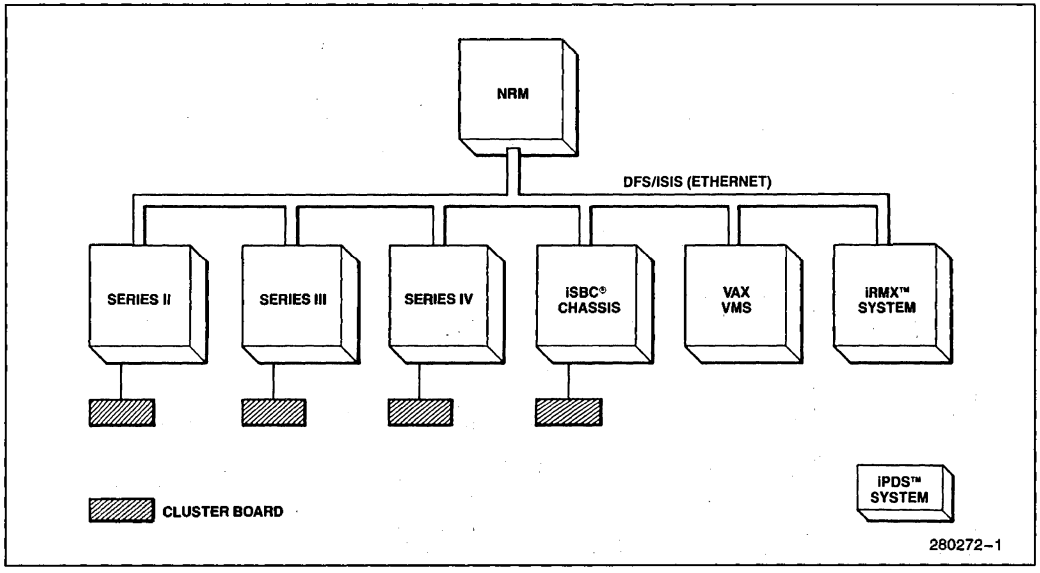


Figure 1. The Intel Development Environment

KERMIT

KERMIT is a file transfer and terminal emulation protocol developed by Columbia University in 1981. Since then KERMIT has been ported to over 30 different systems and is on its way in becoming an industry standard protocol. The KERMIT protocol is designed around character oriented transmission over serial lines. The design allows for peculiarities in transmission medium and requirements of different operating environments. The KERMIT protocol incorporated features and ideas from protocols like DIALNET, DECNET and APPANET. Currently KERMIT has been implemented in over 26 systems. A detailed discussion on the KERMIT protocol is covered in Appendix A.

The following list shows the different systems and operating systems that support the KERMIT protocol.

System	O/S
Series-II, III, IV, iPDS	ISIS
IBM 370 Series	VM/CMS, MVS/TSO, MTS
CDC Cyber	NOS
DEC VAX-11/7XX	VSM, UNIX
PC	MS-DOS, PC-DOS
Apollo	Aegis
PRIME	PRIMOS
HP 3000, 1000	
Apple 11 6502	Apple DOS

KERMIT is a two ended protocol. It needs the remote system to have KERMIT running on it too, to do file transfers. The KERMIT executing on the PC is MS-KERMIT and the one on the Series-II, III, IV, iPDS is the ISIS-KERMIT. The following chapters will detail how this serial interconnect is established.

KERMIT can communicate over either port on the PC AT. Switching between the PC ports can help the PC user communicate with two different systems alternatively.

MS-KERMIT

MS-KERMIT is a program that implements the KERMIT file transfer protocol for the IBM PC AT and several other machines using the same processor family (Intel 8088 or 8086) and operating systems family (PC-DOS or MS-DOS 2.0 or greater).

MS-KERMIT has an extensive command set. A brief summary is shown in Figure 4 wih a more detailed explanation in Appendix A.

ISIS KERMIT

ISIS KERMIT is a minimal KERMIT implementation. This is also available in Insite as described in Appendix C.

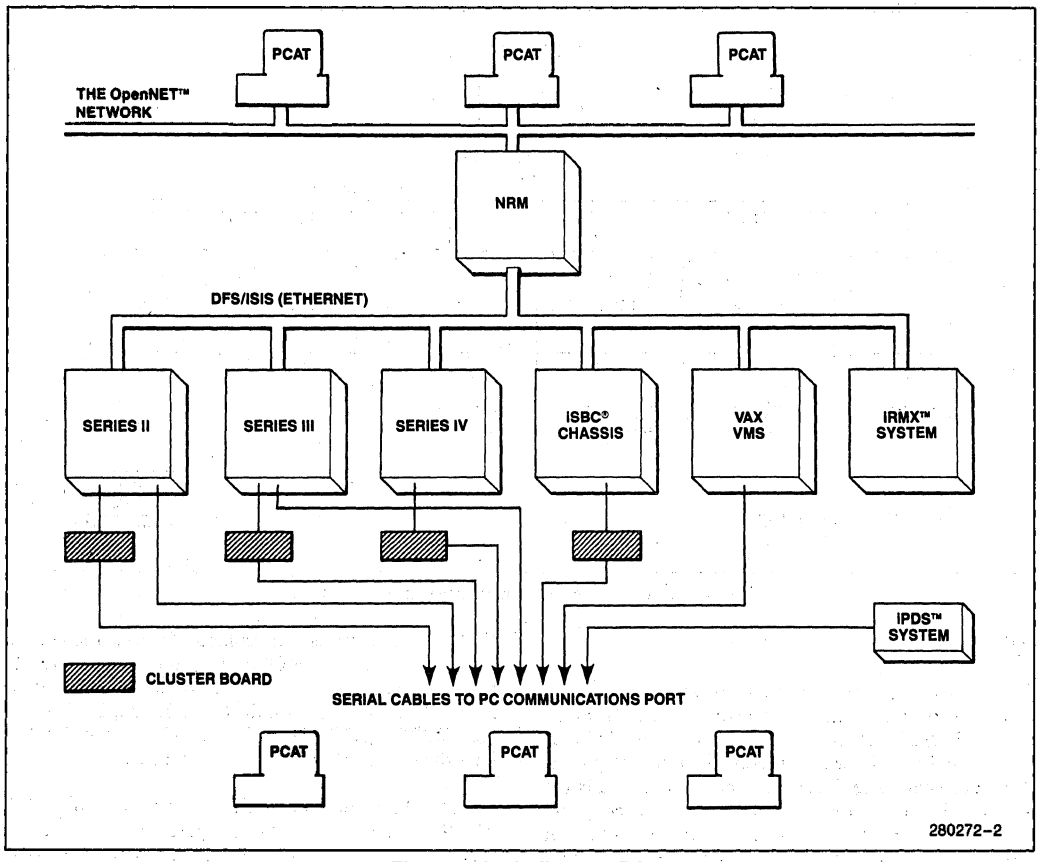


Figure 2. Including the PC

It operates under the ISIS operating system. The basic command set supported by ISIS KERMIT are:

CONNECT—enters terminal mode for communication with host.

DEBUG—toggles debug mode on/off. Prints messages during transfers. Normally used only during troubleshooting.

EXIT—Return back to ISIS.

SEND filename—specifies the file to be transferred to host. May use the ISIS :fn: drive designation to open file on any logical drive in the system. That drive designator will be stripped from the name before it is sent to the host.

RECEIVE [n]—After commanding the host to send a particular file, or a group of files (wildcards can be used on hosts if they're smart enough), press 'HOME' or 'control]' to drop back to ISIS-KERMIT and enter the 'RECEIVE' command. If the command is followed by a number (0–9) the files(s) will be sent to that logical drive. For example, 'REC 4' will cause the filename(s) to be prefixed by :f4: when opened. The number of drives varies, depending on which system is used.

Since there are only 5 commands, a single letter is all that is required to use them.

'r 3' is euivalent to 'RECEIVE 3'

KERMIT is invoked as follows:

KERMIT [baud-rate] [port number]

The default baud rate is 2400, Others available are 300, 1200, 9600.

The port number selection is effective only on Series II. The iPDS system has only one port, and the Series IV must use to port 2, since it is global in multi-user mode.

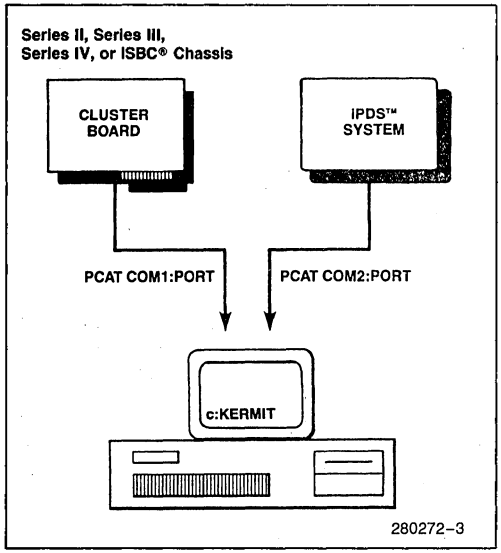


Figure 3. How to Connect the PC to a Remote Host Using KERMIT

A TYPICAL KERMIT SESSION

With the availability of 8051 and 8044 languages on DOS, an existing user may need to move existing software from the iPDS system to the PC AT. The following paragraphs serve both as an example, and as a method to help the iPDS user set up his serial interconnects to do the migration. The various steps, which are explained in detail, include setting up the iPDS system for serial communications, setting up the PC and the actual terminal emulation and file transfer sequence. The steps illustrate the ease with which this migration is brought about.

Command	Explanation
CONNECT	To connect as a remote terminal to a remote system.
DELETE	Delete local files
LOCAL	Prefix for local file management commands
RECEIVE	Receive files from remote system
SEND	Send files to remote system
QUIT	Quit from MS-KERMIT
RUN	Execute a MS-DOS program
SET	Set parameters like baud rate, serial channel
SHOW	Display all parameters
EXI	Exit from MS-KERMIT
DIRECTORY	Directory of local PC

Figure 4. KERMIT Command Set

STEP 1.

Install ISIS KERMIT on a iPDS diskette, and create a CSD file ABOOT.CSD that looks like this:

```
SERIAL A    B = 9600
  ;Set the serial port in ASYNC mode at 9600
ASSIGN :CO:TO :SO:
  ;Redirect console out to the serial port
ASSIGN :CI:TO :SI:
  ;Redirect console in to the serial port
```

This step sets up the iPDS for serial communication by initializing the serial port to communicate asynchronously at 9600 baud. The console I/O redirection is done to enable KERMIT-MS to control the iPDS. Placing these commands in the ABOOT.CSD file help bring up the iPDS system in the right mode whenever it is reset.

STEP 2.

Install DOS KERMIT on the PC AT and invoke it by typing KERMIT from the command line.

```
C: \> KERMIT

IBM-PC KERMIT-MS VER 2.26
TYPE? FOR HELP
KEMIT-MS > SET BAUD 9600
KERMIT-MS >connect
```

Once a successful connection has been made to the iPDS the PC AT terminal will display the iPDS prompt.

```
AO>
```

Now the user can do any operation like DIR, ASSIGN, etc., on the iPDS system from the PC keyboard.

STEP 3.

```
FOR FILE TRANSFER.
```

Invoke the iPDS KERMIT by typing in KERMIT

```
AO>KERMIT 9600 1 ;9600= baud rate and
1=port
```

The ISIS KERMIT prompt will appear ISIS-KERMIT>

For receiving files type in

```
ISIS-KERMIT>RECEIVE :FO:
```

Exit back to the PC by typing in CNTRL] C at the same time. The user is now back to the KERMIT-MS> prompt. Now type in

```
KERMIT-MS>SEND EXAMPLE.BAT
```

A Screen comes up showing data transfer status and on successful completion on file transfer will give back the prompt. More information on setting the number of retries on error packets and timeouts are explained in Appendix A.

```
KERMIT-MS>
```

MEDIA TRANSFER UTILITIES

Diskettes constitute the main source for data and information storage. Most software is kept on diskettes for ease of storage, transportability, and safekeeping. Migrating from one host system to another involves transferring this data on to the new host system media. Media transfer is useful only if both hosts are at the same site and support a compatible peripheral device. If both these conditions are met, media transfer provides a fast convenient way for casual data transfer.

Handling diskettes and interacting with two host computer systems is error-prone and inconvenient. I recommended the other two methods of file transfer in a production environment where the two computers may converse without operator intervention. One major problem with media transfer is the lack of industry standards. There is an 8 inch single density standard (IBM 3740) but not for other densities nor for 5¼ inch media. To solve this problem, special host dependent utility programs must be written to permit the reading of another systems diskettes. This was addressed for the PC by developing a set of utilities that allow file transfer from 5¼ inch and 8 inch. This gives the user the ability to move between the Series-IV environment and the PC-DOS environment with the least overhead and loss of productivity. A key factor in projects these days.

MSCOPY is program that manipulates a MS-DOC disk on a Series IV or NRM. It also helps the PC AT user access the NRM print spooler. The user can copy software both to and from a Series-IV. MSCOPY expects the MS-DOS diskette in FL0. While running MSCOPY do NOT change the disk as MSCOPY keeps the Disk allocation table in memory and will not reread them from a new disk but will write out the old table and directory.

MSCOPY supports 48 or 96 TPI disks, 8 or 9 sectors per track, 1 or 2 heads, MS-DOS vers. 1, 2 or 3. It does not support 1.2 Mb high density diskettes.

It has two modes of operation, interactive and non-interactive. In the non-interactive mode you may enter only one command and it must deal only with the MS-DOS root directory. To use the non-interactive type the command on the invocation line.

In the interactive mode, (entered by invoking MSCOPY with no parameters) MSCOPY will prompt you for a command. Currently there are seven legal commands.

The seven commands are:

READ msfile indxfile—Copies msfile to indxfile. msfile must be in the current directory. indxfile can be any valid iNDX pathname up to 40 characters long.

WRITE indxfile msfile—Copies indxfile to msfile. msfile will be added to the current directory. indxfile can be any valid iNDX pathname up to 40 characters long.

CD msdir—Changes the current directory to the directory msdir. This command will only go up or down the tree one node at a time. To go back one level say "CD". To go deeper say "CD name" where name is a dir entry in the current directory. Typing in "CD \" will jump to the root directory.

DELETE msfile—Removes msfile from the current directory and reclaims the space it occupied.

RELAB label name—Will change or add the volume name of the MS-DOS disk. Label name may be up to eleven characters long.

DIR—Will display the current directory.

EXIT—To return to iNDX.

8" ISIS Media

Flagstaff Engineering in Phoenix, Arizona have a set of tools that allow direct transfer from 8" ISIS (SS/SD) media to PC's. The tool set consists of a add-on board for the PC, an 8" drive and the driver software to do the required transfer. File transfer is bidirectional. The program ISS8TO5 copies files from 8 inch media to PC, and ISS5TO8 copies the other way. An example is shown in Figure 5.

Please note that Intel does not sell, support or warrant reliability of this product. Intel's evaluation sample has proved reliable and Flagstaff technical support has been good.

For more information please contact:

Flagstaff Engineering
Box 1970
Flagstaff, AZ 86001

```
>ISS8TO5
COPY INTEL ISIS DISKETTE FILE TO IBM PC-DOS FILE PROGRAM
COPYRIGHT FLAGSTAFF ENGINEERING 10/17/83

THIS PROGRAM WILL COPY A FILE FROM A 8" ISIS SINGLE DENSITY DISKETTE TO AN IBM
PC-DOS DATA FILE. THE FILES MUST BE CREATED USING ISIS-II OR RMX/80 SYSTEMS.

INSERT 8" ISIS DISKETTE--ENTER DRIVE(1/2) WHEN READY.?
FILE DIRECTORY FOR DISKETTE 164539001
01-ISIS    .DIR(025)  02-ISIS    .MAP(002)  03-ISIS    .TO(023)
04-ICE51   .   (253)  05-ICE51   .0V0(020)  06-ICE51   .0V1(011)
07-ICE51   .0V3(027)  08-ICE51   .0V4(008)  09-ICE51   .0V5(036)
10-ICE51   .0VE(082)  11-ICE51   .0VH(498)  12-ICE51   .0VS(049)

ENTER ISIS FILE NUMBER (1-96/99=ALL)--PRESS ENTER IF NONE?
DO YOU WANT TO COPY FROM ANOTHER ISIS DISKETTE (N/Y)?
```

Figure 5

NETWORKING THE PC AT WITH THE OpenNET™ SYSTEM

OpenNET™ Architecture

OpenNET is Intel's Local Area Network architecture. OpenNET conforms to the Open Systems Interconnect (OSI) model defined by the International Standards Organization (ISO). The major objective of ISO is to create an open systems networking environment where any vendor's computer system can be connected to any network and freely share data with the network.

The OSI ISO architecture is based on a seven layer model (see Figure 6). The seven layers isolate independent functions so that the network can better make use of new software and hardware without adversely affecting the other layers. The upper three layers (5 through 7) provide interoperation functions, while the lower four layers (1 through 4) provide interconnect functions and the bottom two layers (1 through 2) are concerned with the transmission through physical medium.

OpenNET™ Family

As part of Intel's Open Development Environment (ODE), OpenNET supports a number of industry standard hosts and operating systems. To date, OpenNET runs on the IBM PC family with PC-DOS 3.1 or greater, iRMX, XENIX and iNDX as shown in Figure 7.

Since all of the above mentioned systems conform to the ISO seven layer model, they can all interoperate and interconnect over the same network.

OpenNET™ PC Link: Overview

Intel's PC connection on the OpenNET system, named OpenNET PC Link, consists of an add-in controller board for the PC, XT or AT (Layers 1–2), the iNA960 ISO transport software (layers 3–4) and the MS-NET software (layers 5–7).

PC AND THE NRM FILE SERVER ON THE OpenNET™ SYSTEM

The remainder of this application note discusses the use of a PC on the OpenNET system with Intel's Nework Resource Manager (NRM) as the OpenNET file server.

The current NDS-II NRM can be converted into an OpenNET file server by installing the iSXM™ 552 board and iNDX R3.0 or greater software (the board and the software have been kitted into the "NDS-II OpenNET Upgrade Kit", Part # "iMDX555"). New users have the choice of a Mini OpenNET NRM with a 40 Mb disk or a Maxi OpenNET NRM with a 140 Mb disk (upgradeable to 4 140 Mb disks) and a 60 Mb tape.

DETAILED EXPLANATION OF CONNECTING PC TO THE OpenNET™ SYSTEM

The OpenNET system uses concepts such as SERVERS and CONSUMERS which allow a building block approach to creating a network that can be tailored to your particular specification. The following chapters will discuss indepth the various concepts of the OpenNET system and on how to implement PC's and iNDX systems on an OpenNET network.

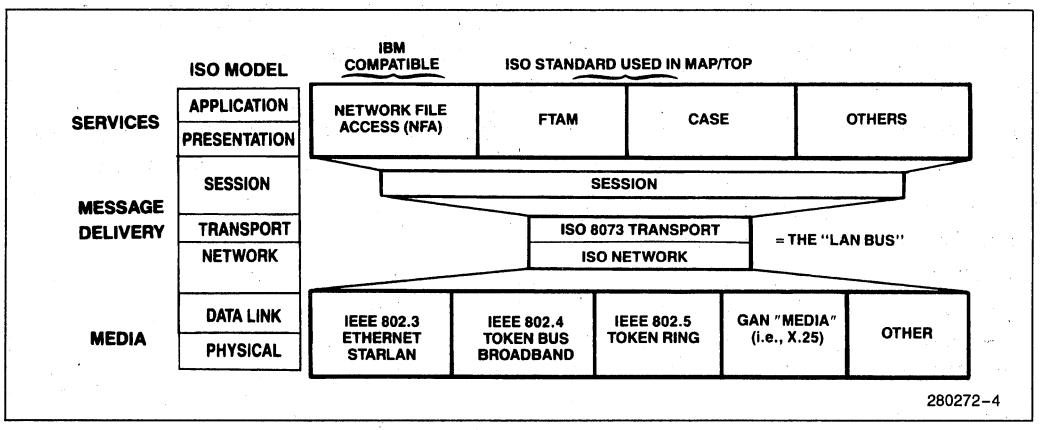


Figure 6

A server is defined as a system on which network resources like files, directories and a printer are kept. A Server usually has a number of hard disks. It is called a "SERVER" because it serves the other systems on the network when they request for files and printer service. There may be a number of servers on the network. The NRM with the iSXM 552 board and iNDX 3.0 installed in it acts as a SERVER for the other systems on the OpenNET network. XENIX and iRMX system can also be servers.

Computers that are linked to a server and use it as a resource for files and printer service are called CONSUMERS. CONSUMERS can also operate independent of a SERVER. An example of a CONSUMER on the OpenNET network is the PC. It can operate independently as a workstation and also uses the NRM for file services. XENIX and iRMX are capable of operating as consumers too.

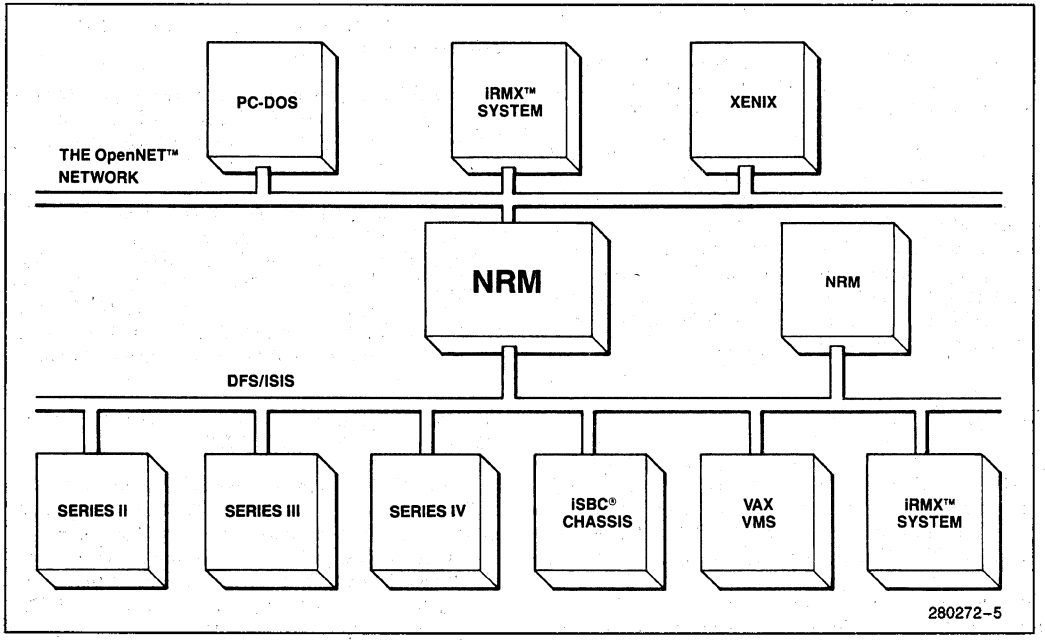


Figure 7

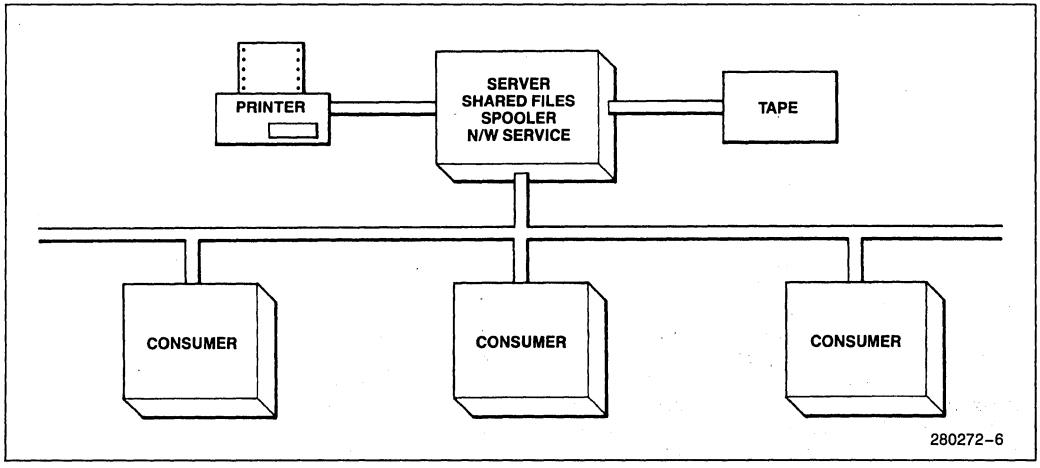


Figure 8. Server and Consumer

A list of all servers and consumers on an OpenNET system is stored in a database file called the NETADDR file. This is discussed in Appendix C.

Figure 8 illustrates how the server and consumers interact on a network.

A SAMPLE OpenNET™ SESSION FROM THE PC TO THE NRM

The following paragraphs will describe how the PC user can access files at the NRM. But before going into the details a few hints on making the process automatic and easy.

Since the OpenNET system uses the concept of virtual drives, it will be beneficial to have as many virtual drives as possible. Refer to the Virtual Drives section under the Chapter "Connecting PC's to OpenNET". DOS 3.1 has a default number of virtual drives 5 (A: through E:), however for OpenNET more drives may be needed. This can be achieved through modifying the CONFIG.SYS file in the root directory of the PC. Edit this file to include the command:

```
lastdrive = z  ;increase the number of
               virtual drives to 26.
```

A typical CONFIG.SYS file is shown in Figure 9.

On boot up, DOS 3.1 will read this file and automatically configure the PC as specified.

Now start the PC as a consumer by entering:

```
C:NET START RDR <this PC name>
```

This is specified in the NETADDR file discussed in Appendix C.

This command loads the PC-LINK communication software onto the PC-Link board, and sets up the environment for communicating with any server. This sets up the session layer on the controller board.

If all the steps went through successfully, the network software will be loaded into the PC-Link board and will sign on with:

```
*******************************************
*                                         *
*            OpenNET™ PC Link             *
*                                         *
*     Copyright 1985, Intel Corporation   *
*                                         *
*******************************************

C:>
```

Now connect to the NRM using the NET Use command. The syntax for this command is:

```
C:> NET USE <virtual drive>\\<server name>
 \username password
```

Example:

```
C:> NET USE J:\\APPS--NRM1/GUEST WELCOME
```

The above command creates a virtual drive J: which points to the home directory of the user GUEST with a password WELCOME at APPS—NRM1. This drive J: is like any PC drive except that it points to a directory

```
lastdrive = z           ;Set the number of virtual drives to 26
device = \sys\ansi.sys  ;Set the terminal characteristics to ANSI
files = 20              ;Number of files open at one single time
buffers = 20            ;Number of buffers for file I/O
break on                ;set break key on
```

Figure 9

at a remote NRM. The user can do any DOS function like copy, dir, etc. even execute DOS applications programs that are stored at the NRM in this directory. Just by having this capacity the PC becomes a very powerful and flexible workstation. By sitting at one PC the user can have access to several file servers.

The command will come back with a message "command successfully completed" if it was successful, otherwise it will wait about 4 minutes before timing out and giving back the DOS prompt.

The user can now use this virtual drive just as if it was any other PC drive. For example a DIR command on drive J: will look like this.

```
C:>dir J:

Volume in drive J has no label
Directory of J:/

INIT        BAK          83   9-06-85    9:23a
INIT        CSD         290   9-06-85    9:23A
CDISK       CPM      401408   9-06-85    9:24a
NNMAC       MAC          74   9-06-85    9:24A
HILIB            <DIR>       9-06-85    9:26a
DJC              <DIR>       9-06-85    9:26a
DATABASE    DIR  <DIR>       9-06-85    9:55a
KERMIT      DIR  <DIR>       9-06-85    9:57a
IMPORT      DIR  <DIR>       9-06-85   10:15a
OPENNET     DIR  <DIR>       9-06-85   10:28a
     10 File(s)   30642176 bytes free
```

This is the home directory of the user GUEST at the NRM. Users familiar with the Intel development environment, will notice that the OpenNET network translated the output of the DIR command at the NRM to DOS format. The user can change directories at this drive, invoke DOS applications from this drive, if they are stored at the NRM, store data files on this drive. The underlying OpenNET protocol is transparent so all existing DOS applications programs can access this drive just as if it was stored locally.

Any time a user wants to fiind out which of his virtual drives are connected to which servers he enters the NET USE command.

```
C:> NET USE
Local Network
Status Device Name
---------------------------
E:    \\APPS_NRM1\GUEST
F:    \\APPS_NRM2\GUEST
G:    \\APPS_NRM1\GUEST
Command completed successfully.
```

More information on the NET USE commands are given in the OpenNET PC Link manuals.

DISTRIBUTED JOB CONTROL SYSTEM FOR THE PC

The Distributed Job Control system on the NDS-II allows currently idle networked development systems to be supplied to the network as public resources. This is a remote job execution unit to which jobs can be sent by other users on the network. Remote job execution offers higher throughput and increased efficiency, as more than one computer can be controlled by a single user. For more information on DJC, refer to Application Note 244, "DJC A key to increased network productivity". It is recommended that this Application Note be read since a number of concepts explained in the following paragraphs assumes that the user has knowledge of the DJC system at the NRM.

The Network Resource Manager (NRM) is the nerve center of the DJC system. All jobs are scheduled and queued by the NRM. Traditionally the PC user had to wait for his compiles to be finished locally before doing anything else on the PC. With the introduction of the OpenNET network, a mechanism has been designed to let the resources of DJC at the NRM be made available to the PC user. This feature enables the PC user to edit files at he PC and send the compiles over to the NRM. An efficient tracking system has also been designed to help keep the user informed at all times on the status of his job. With the introduction of the 286/310 iNDX based Compile Engine shown in Figure 10, the

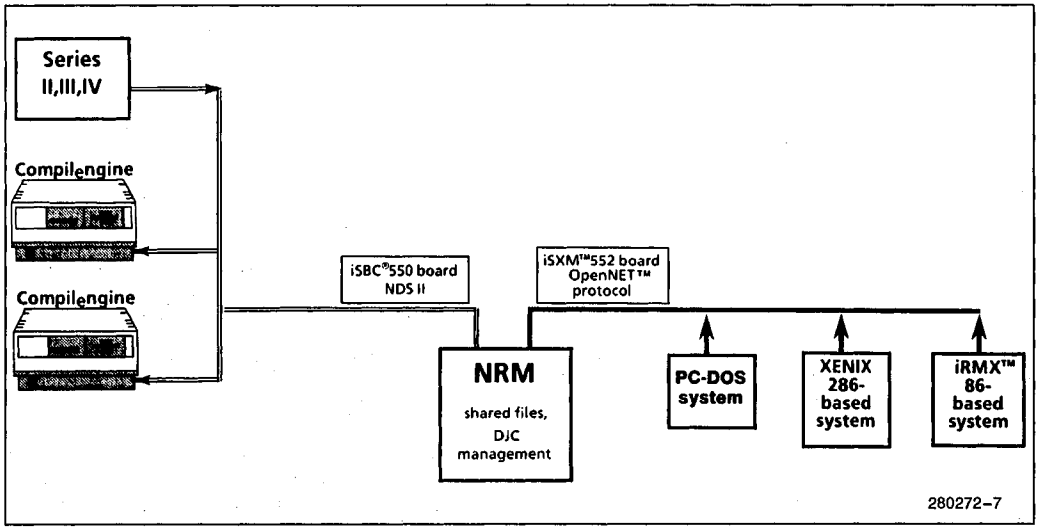


Figure 10. PC's, Compilengine, DJC and the OpenNET™ Systems

```
Reexporter          ;invoke the REEXPORTER utility
Export Reexport to iNDXUTILITY.Q nolog ;now export this job again
```

Figure 11. Contents of REEXPORT.CSD

throughput on such exported jobs increases dramatically. This directly translates into increased productivity and efficiency for the PC user.

The PC-Export package consists of a utility that runs at an iNDX station on the DFS side of the NRM (i.e., NRM itself, Series-IV or 286/310 compile Engine). This package called REEXPORTER.86 checks through a specified directory of all users on the network and if any jobs from the PC exist it will export it to the appropriate queue at the NRM and then will delete the file. Each OpenNET PC user who needs access to the DJC manager at the NRM must create a directory NRMDJC.DIR under his/her HOME directory. It is advisable to limit the above operation to only those users who need to access the DJC system at the NRM. This will help the REEXPORTER utility in having a faster turnaround rate, by not checking redundant directories.

The Superuser then has to create an export file REEXPORT.CSD shown in Figure 11.

It is assumed that the NRM has three queues, 8bit.q (for 8 bit jobs), 16bit.q (for 16 bit jobs) and iNDXUTILITY.Q (for utilities other than compiles, limited to doing system administrative jobs). For more information please read Application Note #244 "DJC a Key to increased network productivity". The Superuser must bring up a Series-IV workstation, or a 286/310 Compile Engine in Import mode, importing from all the three queues. This is shown in Figure 12.

```
>Import from 8bit.q, 16bit.q, indxutility.q
```

Figure 12

The command now puts the workstation as a network resource that all user can acess.

Now export REEXPORT.CSD to iNDXUTILITY.Q

```
>Export REEXPORT.CSD to iNDXUTILITY.Q Nolog
```

Exporting from the PC

To export jobs from the PC, the user must first connect to the NRM using the NET USE command explained in previous chapters. One of the design considerations was to make this utility as easy to use as possible. The following paragraphs illustrate how this has been brought about.

Consider this example file that does a compile and link, COMPILE.CSD. A requisite is that the file must have a .CSD extension, as it is this extension that informs the NRM that it is a command file.

```
;queue = "16bit.q"
lname define l for/wini0/libs.dir
lname define p for /wini0/strng.dir
plm86 example.p86 debug optimize (2)
if % status = 0
 link86 example.obj,l/compac.lib,&
   p/hstrng.lib,l/osxcom.lib &
   to example.86 bind
 end
```

The first line in the command file is a comment, which also indicates the queue where this job is to be executed. As far as the DJC manager is concerned, this is just a comment. But the REEXPORTER utility uses this field to find out the queue name. This comment field must exist within the first 128 bytes of the command file. An absence of this field will result in the job not being sent to any queue. The queue name must be enclosed within double quotes.

Since there is no way by which the NRM can access files stored on the PC, all source, libraries and objects must reside on the network. Note the two commands after the comment which set up the logical names for directories used in the job. Doing this will ensure that the right libraries are used.

Now all the PC user has to do is to copy this file to the directory NRMDJC.DIR under their home directory at the NRM. The REEXPORTER utility does the rest. For example if virtual drive G: has been connected to the NRM.

```
C: \Copy COMPILE.CSD G: \NRMDJC.DIR
```

Once the job has been reexported it will be deleted from the directory. This helps the user in determining if the job got exported or not. The REEXPORTER utility also creates a log file in the NRMDIC.DIR directory for each job exported. This allows the user to find out if his job was successful or not. The COMPILE.CSD file will be replaced by a COMPILE.LOG file once the job has been completed. The COMPILE.LOG file is a LOG file of all the operations by the job.

The REEXPORTER utility was designed and implemented to allow the OpenNET PC user access to the powerful Distributed Job Control mechanism at the NRM. The utility has the capability to determine all the users on the network, work out their home directories and look for jobs sent from PC's. On finding a job, the utility determines the appropriate queue and reexports the job to that queue. The status of each job is displayed on the screen at all times. Status information includes username, jobname, queue and the status of the exported job. The use of this utility is restricted to the Superuser. A normal user invoking REEXPORTER.86 will generate an insufficient access rights exception.

REEXPORT.CSD is a batch file configured as a job that runs forever. It first uses REEXPORTER to check for jobs in the directory NRMDJC.DIR of all users and exports them to appropriate queues. It then reexports itself to the same queue. Due to the way the DJC mechanism is structured, the import station will start executing all the jobs found by the REEXPORTER utility, and on completing all of them will execute the REEXPORT.CSD job once again to look for more work to do. The cycle keeps repeating forever.

Referring back to the IMPORT command in Figure 12, highest priority is given to 8bit.q and lowest to iNDXUTILITY.Q. This way the system manager makes sure that all jobs waiting in the first two queues are executed before the REEXPORT.CSD job is started again. This helps the NRM in controlling the queues, and not overloading them at one single time. A point to note at this time is that the REEXPORTER utility is capable of exporting up to 23 jobs a minute.

The REEXPORT utility combined with OpenNET networking opens out a completely new environment for the PC AT user. An environment where compiles, links and locates can be done remotely. The PC user has at his/her disposal the power of the iNDX Distributed Job Control subsystem. This help in bringing about an increase in productivity that normally could not have been achieved without Intel's OpenNET network. The PC AT user can spend more time on interactive work such as program generation or debugging, while the compiles are being done elsewhere. This feature set should be used by developers doing system designs in todays world where "Time to Market" is key.

Summary

This application note has discussed in detail all the different methods by which the PC AT can be integrated into the Intel Development Environment, from serial interfaces to networking. These different methods can be intermixed to suit your needs. Serial interfaces and disk transfers support the low end needs while OpenNET brings about a powerful new environment to the PC AT. This coupled with the REEXPORTER utility, can help increase the productivity of the PC AT user dramatically.

APPENDIX A
THE KERMIT PROTOCOL

THE KERMIT PROTOCOL

The KERMIT protocol is designed around character oriented transmission over serial lines, and incorporates features from decnet, arpanet, dialnet etc.

File transfer takes place over transactions. A transaction is an exchange of packets. A successful transaction is done when one system sends a packet and the remote system acknowledges it.

Transmission begins with a send_init packet(s) and ends with a break_transmission (b) or error (e) packet. All communication is done through packets, even if no data is being sent.

The Kermit packet is built around the following format.

mark	length	seq	type	data	check

All the fields are ASCII characters.

Mark

This is the synchronization character that marks the beginning of a packet. This is normally a cntrl-a and can be redefined.

Length

The number of ASCII characters within the packet following this field.

Seq

The packet sequence number, ranging from 0 to 63. Sequence numbers wrap around to 0 after each group of 64.

Type

The packet type is represented in a single ASCII character. The different packet types are:

```
D data packet
Y acknowledge
N negative acknowledge
S send initiate
B break transmission
F file header
Z end of file
E error
```

Data

The contents of this packet.

Check

A checksum on the characters between, but not including the mark and check. The check for each packet is computed by each host and must be equal for the packet to be accepted.

KERMIT File Transfer Sequence

File transfer is initiated by the sender sending a send_initiate packet, where parameters like packet length, time out limits are specified.

The receiver than sends an ack (y) with its own parameters in the data field.

The sender then transmits a file—header packet which contains the filename in the data field. The receiver then sends an ack.

The sender then sends the contents of the file in data packets (d), any data that is not in the printable range is prefixed and replaced by a printable equivalent. Each d packet has to be acknowledged before the next one is sent.

After all the file data has been sent the sender then sends an eof packet. The receiver acks it.

End_of_transmission packet (b). The receiver acks it and the transaction is over.

KERMIT-MS Commands

CONNECT

The CONNECT command connects the PC as a terminal to the remote system. KERMIT-MS uses either communications port 1 or 2 and uses full duplex and no parity. These can be changed using the SET command. To get back to KERMIT from terminal emulation type in the escape character followed by the letter C. The escape character by default is CENTRL-]. This can be modified by the SET ESCAPE commnd. SET BAUD changes the baud rate, SET PORT changes the serial port. For example:

```
C:\KERMIT <cr>
KERMIT-MS>SET PROT 1    ;select port 1
KERMIT-MS>SET BAUD 9600
KERMIT-MS>C             ;connect
```

SEND

The SEND command causes a file or a group of files to be sent from the local PC to the KERMIT on the remote system. The remote KERMIT must be running in either server or interactive mode; in the latter case the user must have already given a RECEIVE command and escaped back to the PC.

SEND filespec1 [filespec2]

If filespec1 contains a wildcard then all matching files will be sent in the same order that the DOS would show them on a directory listing. If filespec1 contains a single file, the user may direct KERMIT-MS to send that file with a different name.

SEND KERMIT.ASM TEST.ASM would send the file KERMIT.ASM as TEST.ASM

or

SEND *.ASM will send all files with the extension .ASM to the remote system.

Once the SEND command has been invoked the name of each file will be displayed, packets transferred, retries and other counts will be displayed along with other informational messages. If file transfer is successful a "COMPLETE" message will be displayed else an error message will be displayed. When the specified operation has been done the program will sound a beep.

Several single character commands can be given while a file transfer is in progress.

CNTRL-X	Stop sending the current file and go on the next one.
CNTRL-Z	Abort file transfer.
CNTRL-C	Return to KERMIT-MS.
CNTRL-E	Send an ERROR packet to the remote server in an attempt to bring it back to server or interactive mode.

RECEIVE

The RECEIVE command tells KERMIT-MS to receive a file or a group of files from the remote KERMIT. KERMIT-MS simply waits for the file or files to arrive. The user should have already issued a SEND command at the remote KERMIT and escaped back to the PC before issuing the RECEIVE command.

Syntax:

RECEIVE filespec

If the optional filespec is provided the incoming file is stored under that name. The filespec may include a device designator or may consist only of a device designator. For example:

RECEIVE TEST.ONE	;will name the incoming file as TEST.ONE
RECEIVE A:TEST.ONE	;will name the incoming file as TEST.ONE and store it in drive A:
RECEIVE A:	;will store all incoming files in Drive A

If an incoming file does not arrive in its entirety, KERMIT-MS will normally discard it. This may be changed by the SET INCOMPLETE KEEP command, which will keep as much of the file that arrived successfully.

If the incoming file has the same name as a file that already exists and "WARNING" is set ON, KERMIT-MS will change the incoming file name and inform the user of the new name. If WARNING has been SET OFF using the SET WARNING OFF command, files with the same name as incoming files will not survive.

SET

The SET command allows the user to modify various parameters for file transfer and terminal emulation. These parameters can be displayed with the SHOW command.

APPENDIX B
SETTING UP OpenNET™ CONNECTIONS BETWEEN PC & NRM

```
APPS__NRM1      :address = 0x80000a0100000000aa00003510000000
APPS__NRM2      :address = 0x80000a0100000000aa000034de000000
APPS__XENIX__1  :address = 0x80000a0100000000aa00002de2000000
DIAG__NRM       :address = 0x80000a0100000000aa00000c0f000000
MFNG__RMX__1    :address = 0x80000a0100000000aa000006e4000000
MKTG__NRM       :address = 0x80000a0100000000aa00000304000000
APPS__PCAT__SS  :address = 0x00010a0100000000dd00002584000000
```

Figure 13

OpenNET™ CONCEPTS

To enable the PC to talk DFS-OpenNET protocol, the user must install a PC-Link card in the IBM PC, and the networking software PC-LINK supplied by Intel. The prerequisite is that the PC should have PC-DOS Version 3.1 or greater. Follow the installation instructions given in the PC-Link manual. It is advisable to create a directory on the PC called COM (short for Communication S/W) and install all the PC-LINK software in this directory.

Once the hardware and software have been installed on the system the user can now use the PC as a consumer off of the NRM. Before going into a discussion on the actual use, a number of concepts have to be explained, to give the reader a better understanding of how PC's work when networked to the NRM using the OpenNET protocols.

The NETADDR File

Each computer on the network is assigned a name, to identify it. These computers can be named anything, but it is preferable to have one standard naming convention. This makes it easier for the users to connect up to the server of choice. The NETADDR file is a text file that contains the names of these servers and their addresses on the network. This is extensively used by the PC-LINK software to connect to the desired server. This gives the user the flexibility to connect to any server just by giving its name, and the PC-LINK software automatically directs the connection to that server. A sample NETADDR file is shown in Figure 13.

The NETADDR file above has the lists of 6 servers on the OpenNET network. Note the naming convention. The first four letters in the name indicate the department where the server is stationed. For example APPs is short for Applications Engineering, DIAG for Diagnostics Engineering, MFNG Manufacturing etc. The rest of the name indicates the type of server (NRM, XENIX or RMX). This naming convention helps in connecting up to the different servers without any confusion. This file is created by each individual PC consumer using any standard text editor (one which does not put control characters in the file). One of the entries in the NETADDR file is the name of the users PC where this file resides. This is important as the PC-LINK software cannot be invoked without this information. The name and ethernet address of the consumer station is necessary for the NRM OpenNET file server to send message packets back to the correct originating consumer (i.e., the NRM should know which consumer station has requested for a particular process for sending back the response to it).

The rest of the numbers following each computer name are the Port address and the iSXM 552 Ethernet address of the server. An example is shown in Figure 14.

The Ethernet address of the iSXM 552 is obtained from the NRM on boot up. Refer to the Chapter "Installing the iSXM 552 in the NRM" for further details.

The MSNET.INI File

The MSNET.INI file is the PC-LINK initialization file for setting up the help files and loading the PC-Link

APPS	is the department where the server is located
NRM2	is the NRM name (2 is used to differentiate it from the other NRM)
80	is the port address (always 80)
00aa000034de	is the ethernet address of the iSXM552 board at NRM2.

Figure 14

board with the communication software. Any time a PC station is brought up as a consumer this file is parsed and executed. The file may need a little modification if the communication software is located in some other directory. The sample MSNET.INI file is used as an example, see Figure 15.

Virtual Drives

PC-LINK uses the concept of virtual drives to connect to the NRM file server through the OpenNET network. When a connection to a NRM server is made the PC-LINK software creates a virtual drive, which is identical to the PC drives. The only difference is that it does not physically exist on the PC. The user treats this as any other drive on the PC. The number of virtual drives that can be used is limited to the English alphabet (26). The user can connect different virtual drives to different directories at the NRM file server and all these drives can be used just as if they existed on the PC.

Configuring PC Link

PC Link as shipped uses default settings for its memory window, number of connections, interrupt levels and number of servers that it can access. The following paragraphs will discuss various configuration parameters that will allow the user to configure PC link to suit his environment.

CONFIGURING THE BASE FOR THE PC LINK BOARD

All communication with the PC link board takes place via a dual-port memory, which is a 32K window that may start on any 64K window within the first megabyte of addressable memory. The default base is 0A000h. In some cases the window might clash with an existing board or application. This default can be changed by jumpers on the PC link board and by making modifications to the MSNET.INI file. Figure 16 indicates all possible bases and associated jumpers.

After the jumpers have been selected, the MSNET.INI file has to be changed to inform the PC link software about the new window being used. Referring to Figure 15:

```
start redirector $1
start rdr $1
 \command.com/c type \com\pclink.msg
 chknet
 xport/sys:at/base:d
   /file:c:\com\ubcode.mem
 session \com\netaddr
 redir
 setname $1
 \command.com/c psclose
```

The string of commands following the start redirector $1 or start rdr $1 indicate the sequence of events in

```
use $*
   use $* /*

print $*
   printq $*

name
   setname

start redirector $1
start rdr $1
   \command.com /c type \com\pclink.msg
   chknet
   xport/sys:at /base:d
    /file:c:\com\ubcode.mem\nvcs:5
   session \com\netaddr
   redir
   setname $1
   \command.com /c psclose
```

Figure 15. (Sample MSNET.INI file)

loading the PC link board and starting the network. XPORT is the network program that loads the PC link board and the driver. One of the options that can be specified in the xport commands is the base option. In the previous example the base was set to D. This base should reflect the base at which the PC link board is strapped.

Configuring Connection Limits

PC link allows the user to have simultaneous connections to a number of file servers which is configurable.

The PC link defaults allow the user to connect to two file servers simultaneously and have up to 5 active virtual circuits. The MSNET.INI file supplied with PC link has to be modified for users who need access to more than two file servers and more simultaneous connections.

For example:

A user needs connections to three different servers. The first two NET USE commands will come back successfully, however the third command will come back immediately with a message "Connection Refused". This is due to the fact that the PC link software uses a default value of 2 for the number of servers it can simultaneously access.

Starting Address	E5	E8	E11	E14
00000	E4	E7	E10	E13
10000	E4	E7	E10	E15
20000	E4	E7	E10	E13
30000	E4	E7	E10	E15
40000	E4	E9	E10	E13
50000	E4	E9	E10	E15
60000	E4	E9	E12	E13
70000	E4	E9	E12	E15
80000	E6	E7	E10	E13
90000	E6	E7	E10	E15
A0000 (DEFAULT)	E6	E7	E12	E13
B0000	E6	E7	E12	E15
C0000	E6	E9	E10	E13
D0000	E6	E9	E10	E15
E0000	E6	E9	E12	E13
F0000	E6	E9	E12	E15

Figure 16

```
C:\ NET USE H: \\APPS_NRM\JOHN PASSME
Command completed successfully
C:\ NET USE I: \\DEMO_NRM\JOHN PASSME
Command completed successfully
C:\ NET USE LPT1: \\DIAG_NRM\JOHN PASSME
Command Refused
```

In the above case the user tried to access more than one server and since the default was set at 2, the PC link network management facility came back with an error. To change this default value, edit the MSNET.INI file and modify the REDIR specification to read:

```
REDIR /S:5  ;connection available for up to
5 servers
```

Reboot the PC and PC link will now allow the user to connect to up to 5 servers simultaneously.

Another variable that can be configured is the number of active virtual circuits at the PC. As a default PC link will allow up to 5 net uses (Logons) to a server. Again when the default limit is exceeded the system will come back with a "CONNECTION REFUSED" message. For example consider the following NET USES:

```
C:\ NET USE F: \\APPS_NRM\SRIVATS PASSME
Command completed successfully
C:\ NET USE G: \\APPS_NRM\AP1 PASSAP1
Command completed successfully
C:\ NET USE H: \\APPS_NRM\SY1 PASSSY1
Command completed successfully
C:\ NET USE I: \\APPS_NRM\SY0 PASSSY0
Command completed successfully
C:\ NET USE J: \\APPS_NRM\AP0 PASSAP0
Command completed successfully
C:\ NET USE K: \\APPS_NRM\JOHN PASSME
Connection Refused.
```

In the above example the number of simultaneous connections was set at a default of 5. Modify the REDIR command to include the following option.

```
REDIR /S:5/L:10  ;connection available for up to 5
                 servers and 10 simultaneous con-
                 nections.
```

Reboot the system and PC link will allow the user to connect up to 5 servers with up to 10 simultaneous connections.

APPENDIX C
INSITE LIBRARY PROGRAM

MS-KERMIT and ISIS KERMIT are available from:

Intel Corporation
2402 West Beardsley Road
Phoenix, Arizona 85027

ATTN: Insite User's Program Library

Telephone: (602) 869-3805

This is a public domain software and is available for a nominal charge of $25 each.

APPENDIX D
NDS-II/SERIES IV TOOLBOX V2.0

NDS-II/SERIES-IV TOOLBOX V2.0

The NDS-II/Series-IV Toolbox V2.0 is a set of 7 diskettes with useful programs for NDS-II/Series-IV and OpenNET users. The programs used from this toolbox were MSCOPY.86, ReExporter, NET CONNECT. It contains many other useful utilities. An index listing the various programs in the Toolbox are listed below.

NDS-II/Series IV Toolbox 2.0

intel

ARTICLE REPRINT

AR-204

Technical articles

Smart link comes to the rescue of software-development managers

Resource-management hardware and software join existing development systems into an Ethernet-based network that eases software creation and control

by James P. Schwabe, *Intel Corp., Santa Clara, Calif.*

☐ A strong lifeline in a sea of complexity, the new NDS II network development system will help manage the writing of complex software for tomorrow's powerful microsystems. It builds on existing Intellec development systems and the specifications of the Ethernet protocol to create a local network for distributed software development.

Considerable intelligence is contained within the NDS II system, linking programmers' work stations and managing the interactive flow of software development that results. Communications control, via Ethernet or an even simpler alternative, is split between the central manager and the work stations.

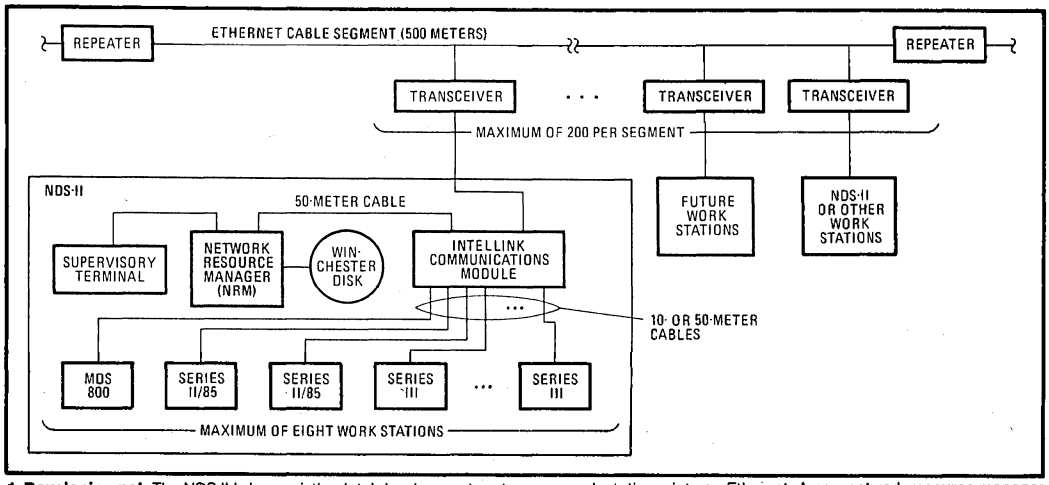
At the heart of the system is the network resource manager, which both controls the net of work stations and lets the user configure it to suit the development task under way. The NRM will also manage a powerful system memory of Winchester-technology disk drives.

The manager itself is an example of the boons of well-thought-out and complex software, for it contains powerful system tools. Among these features are a hierarchical file structure that is also distributed and a file-protection setup that offers the maximum flexibility in access to files while guaranteeing their integrity. Important program-management tools include a routine that oversees the rewriting of software during development and another that automates the generation of a complete program from the most current modules.

The NDS II is the second step in the evolution of Intel's network architecture, iLNA [*Electronics*, Aug. 25, 1981, p. 120]. It connects Intellec development systems together so they can share large-capacity Winchester disk drives and a line printer located at the NRM. It will also serve as the basis for a whole new line of modular development system tools such as remote emulators, logic analyzers, and more.

Both the NRM and each work station can be connected directly to the Ethernet coaxial cable by a transceiver or by the Intellink communications module (Fig. 1). By itself, the Intellink module provides nine ports for interconnection, creating a local network of nine systems (eight work stations and one NRM). To another controller, the Intellink represents a segment of Ethernet cable that has nine transceivers already in place and working.

For networks with a radius of 50 meters or less, Intellink is a simple, low-cost alternative to installing Ethernet cabling and transceivers. Any work station can



1. Developing net. The NDS II brings existing Intel development systems, or work stations, into an Ethernet. A new network resource manager and the Intellink communications manager make management of distributed software development possible.

be installed by simply plugging a 50-m transceiver cable directly into the Intellink—a 5-second operation.

For expansion beyond nine systems or to a distance greater than a 50-m radius, the Intellink provides a built-in port for connecting the local cluster to Ethernet cable by means of a transceiver. Connection to the Ethernet allows communication with other work stations, NDS II networks, or other Ethernet-compatible devices that use the iLNA network architecture.

No matter which physical setup is chosen, each work station has independent access to, and can be directly accessed from, the Ethernet and the NDS II network. Each has a unique work-station identifier, distinguishing it from every other terminal in the world and ensuring correct communication between stations on the various local networks.

For multiple-net environments, each network can have a unique network identifier to allow their coexistence on one Ethernet. In a single net, the network identifier is not used, but its assignment ensures an orderly progression to a multi-net environment.

All current Intellec development systems can be upgraded to NDS II work stations. An upgrade consists of a communication-controller board set, software, and either 10- or 50-m cables.

The communication controller, a two-board set that plugs into any Intel Multibus chassis, provides many of the data- and physical-link functions of the six-layer standard reference model for open-systems interconnection (Fig. 2). The data-link functions performed are framing, link management, and error detection. Physical-link functions include preamble generation and decoding and bit encoding and decoding.

One board contains a 5-megahertz 8086 microprocessor with local random-access and read-only memory and interval timers, as well as direct-memory-access channels for sending and receiving data at 10 megabits per second. The second board contains bit-serial send-and-receive logic, packet address-recognition logic, and error-detection logic. The boards ensure that bad packets resulting from a collision are ignored.

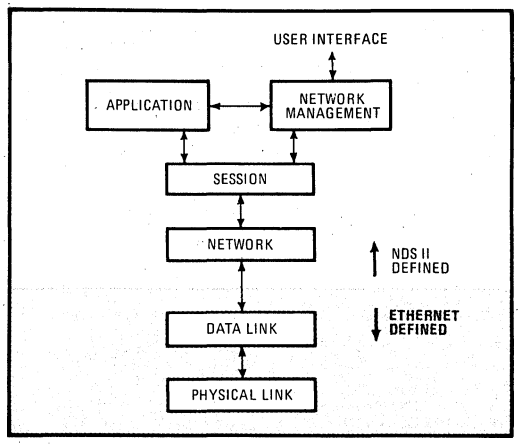
The NRM coordinates all the work stations' activities and manages file access to the shared disks. Initially, it will support one 8-inch 35-megabyte Winchester disk subsystem, as well as Intel cartridge-module disks. Multiple-disk support is in the wings, along with a larger 84-megabyte disk. It will be possible to attach six disks to one NRM, providing more than enough on-line shared storage for large program development and archiving. In addition, each work station can contain 2.5 megabytes of floppy-disk storage as a local resource.

Control contingent

The NRM (Fig. 3) comprises 13 Multibus slots, power supply, 8086-based system-processor board, input/output board based on the 8088 and 8089, 512-K-byte memory board with error checking and correction, two communication boards, and one 5¼-in. floppy-disk drive. The cabinet also has space for a cartridge-tape unit, expected to be delivered in mid-1982, which will give full intelligent archival backup for the Winchester disks housed in the attached cabinet.

To protect the integrity of the network, access to the NRM is restricted: a special supervisory terminal connected to the unit's serial port provides an interface with its commands and utilities. These facilities include system generation, intelligent archiving, and normal network maintenance such as the creation of any necessary user identifications.

The most important utility for system configuration is called Sysgen, an interactive routine designed to assist the supervisor, or project manager, in creating the NRM operating system. Sysgen makes it possible to create, modify, or delete system parameters, peripheral-devices configuration, and network configuration. It allows the project manager to tailor the network configuration on the fly in order to fit the changing needs of microprocessor development projects.



2. New layers. To the hardware layers of Ethernet, NDS II adds software layers that permit up to eight users to work together. The network layer need not be present if NDS II is not linked to the Ethernet, simplifying the operating system.

From the work-station perspective, the NRM is a remote file system. Each station functions as a stand-alone development system for all tasks not requiring NRM resources. When access to these resources is required, the user simply logs onto the network. The work station's resident operating system formats the appropriate file request, which the NRM processes interactively with other stations' demands.

The NRM operating system is multitasking, allowing a work station to access a file on the shared disk while other stations concurrently access other disk files. The interleaving of disk accesses, as well as the high-speed packet transmissions on the Ethernet, enables each work station to share equally the large file store—its being accessed by one user does not prevent other work stations from gaining access.

In an eight-station environment, the performance degradation due to network contention and the NRM operating system will be no more than 10%. This performance is one of the major reasons why distributed development systems provide a more cost-effective method for microprocessor development than time-shared systems; the former are much less susceptible to saturation under concurrent loading than are the latter.

Managing the work

To ensure efficient software development, high performance must be combined with tools to manage software complexity. For example, large software projects are often broken down into small tasks, and efficient file sharing becomes essential to project coordination. The shared-file system on NDS II is built on the RMX-86 volume-based hierarchy in which each user directory represents a node on a hierarchy of directories, commonly referred to as a hierarchial file system (Fig. 4).

Hierarchical file systems can contain a multitude of directories and data files. At the apex is the root volume, a conceptual file from which all directories emanate. The root volume contains all the volumes of the directories. Each volume can contain as many directories or files as available disk space will allow, and any directory may contain other directory files or data files. Each file (directory or data) can be traced through the hierarchy by its own path name. The NDS II hierarchical file system goes one step further by extending from the NRM to include the directories at the user's work station. When the user logs off the network, the only directories available are those on the work-station disks. When the user logs on, he or she gains access to the NRM system directories.

Thus each programmer has access to a common data base without the confusion of sifting through one massive directory. What's more, the structure keeps other users' files out of the way. In addition, it permits logically separate types of software within a user's directory. A programmer can create subdirectories to separate source files from object files, from backup files, and so on.

As a project's size increases, the number of directories and the complexity of path names in the system also increases. To simplify the task of accessing any particular directory, the user can assign a less cumbersome name—what amounts to a macroinstruction. Then, the user simply types in this macroname. Maximum flexibility is maintained, as each programmer can assign macronames to any directory.

An added benefit from macroname assignment is device transparency: the user concerns himself only with directories, irrespective of physical location. Physical devices are fixed in size and location, as opposed to directories, which can be adjusted to organize the contents in an optimal fashion.

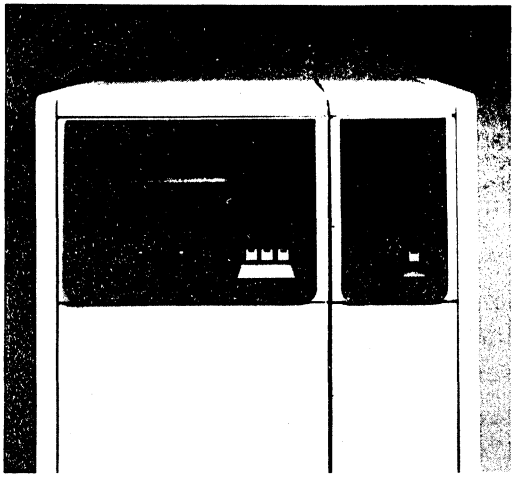
File protection

Before accessing the network, each user must be identified to the NRM through a log-on procedure. This setup establishes a unique user identification that is subsequently used to control access to files and directories in the hierarchical file system. Each directory and data file has specific "owner" and "world" access rights, which protect against accidental modification or deletion.

A file has three possible access rights for both the owner and the world: read, write, and delete. A directory also has three similar access rights for both the owner and the world: list a directory, add a directory entry, and delete a directory entry.

The access rights in file systems improve coordination during software development by allowing complete modules that have been tested and debugged in a user's work space to be converted into read status for the world. Then these modules can be integrated and tested with other independently developed software modules. Thus modules declared as read-only are guaranteed to be the most current debugged versions, and a common data base of completed modules is ensured.

Extended to multiple-project environments, the file system can provide logically separate work spaces for each project group. Specific directories can be set aside for complete modules for various projects. Each user can develop portions of the program in a private work space with guaranteed file protection and can use the public files (or directories) for integration and testing of the



3. Manager. The network resource manager (NRM) in the cabinet's left side governs access to the 35-megabyte Winchester drive on the right. Access to network-managing software is gained only through a supervisory terminal attached directly to the NRM.

module under development. Commonly used utilities and compilers can be accessible in a specific directory as public files (read-only for world access) to eliminate the necessity of redundant files at each work station. As a result, all programmers can proceed without fear of inadvertent modification of private files either by others or by themselves.

As well as managing communications between shared disks and work stations, the NRM maximizes the use of all network resources with distributed job control. DJC allows the user of any work station to export a batch job to the NRM for remote execution.

To accomplish this, the NRM classifies each work station into one of two groups—private and public. It keeps track of the public work stations and uses them to execute the queue of batch-type jobs. A user can declare any work station as public: available for use by the NRM for remote execution. Also, a programmer can send a job to a specific queue at the NRM by using the export command. The NRM executes the job on a public work station and return the results to the user directory.

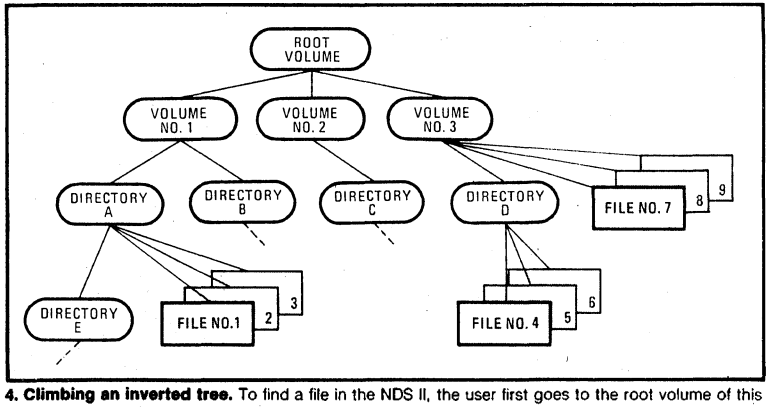
With DJC, the resources of the entire network can be shared to maximum advantage. A typical project involves program-module editing and debugging at Intellec series II or model 800 work stations, while a 8086-based Intellec series III unit can provide a host execution environment to compile completed modules quickly. DJC allows the user to export the compilation process to the high-performance series III work station, then return immediately to other tasks while the NRM oversees the compilation. At any time, the users can check on job status or queue status by typing a command from their work stations.

New work stations

Currently, Intellec development systems provide a single-task environment and therefore can be declared public to the NRM as users finish on-line work. Later this year, Intel will introduce high-performance work stations with foreground-background capability to allow a user to run a job in the foreground while making the background public so that jobs exported by other programmers can be executed through DJC. Foreground-background capability with DJC will effectively double the usefulness of the work station and substantially cut the cost of development time.

In-house benchmark tests indicate that the performance of each work station connected to the NRM is much improved. For example, a compilation executed with all file requests from the NRM hard disk is twice as fast as requesting files from the work station's floppy disk. Each station enjoys hard-disk performance during compilation, assembly, and any file manipulation—at a fraction of the cost of a dedicated disk system.

User's tools also speed program development, as well as make management easier. The most important programmer tools on NDS II are SVCS (software-version control system) and MAKE, an automatic software-generation tool. They provide a superset of the functions offered by the SVCS and MAKE found in the Unix programmers workbench.



4. Climbing an inverted tree. To find a file in the NDS II, the user first goes to the root volume of this hierarchical file structure. From that volume, he or she can go to the project volume assigned by the project manager and access other directories or files that have been declared accessible.

SVCS controls and documents changes to software products, handling both source and object files. It contains facilities for storing and retrieving different versions of a given program module, for controlling update privileges, and for recording who made what changes, when, and why.

Documentation of module status and of the levels, or versions, involved is the key factor determining the success of program development by group effort. Valu-

MAKEing it easy to revise programs

NDS-II's MAKE facility is a development tool for both generation and documentation of a software system. Suppose, for example, a software system called PGM.86 consists of three separate programs linked together, and, for simplicity, that each program consists of only one compiled source file, rather than a subsystem of multiple files. This relationship forms a dependency that would be graphed by the user as in the figure below.

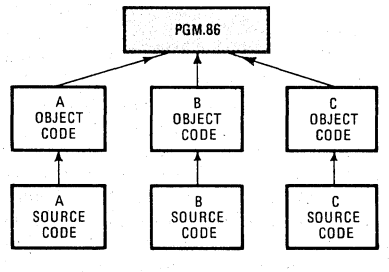
With the MAKE facility, a user can create an automated-generation procedure for the system PGM.86 that checks the currency of each subprogram. A MAKE command file that does so is illustrated in the accompanying table.

When the command file is invoked, the commands it contains are executed in top-down fashion. In step 1 of the table, the facility first checks if the PGM.86 is older (represented by the greater-than sign) than any of its dependent object-code modules. The facility checks and compares the date and time stamp of each module with that of PGM.86. Date and time stamps are updated automatically whenever a file is modified.

If any of the object modules are newer versions, then MAKE is instructed to link together the latest versions of the object modules to form the latest version of the software system. Before executing the link routine, the MAKE facility must first check to see if any of the object files are older than the related source files given in the dependency graph, as shown in steps 2, 3, and 4.

The MAKE facility goes through each step and executes the specified task only if the specified condition is true. Once the dependency graph is created, the MAKE facility can quickly and automatically generate the latest version of a software system under development even when source files change frequently.

The MAKE facility removes much of the guesswork surrounding software-system generation by ensuring the latest versions of source code is incorporated into the final software system. The dependency graph in its current form can also be printed by NDS II to document the software-system construction without having to keep an out-of-date sketch taped to the laboratory wall.



MAKE PROGRAM FOR PGM.86

Steps	Statements
1	IF PGM.86 > A.OBJ, B.OBJ, C.OBJ THEN RUN LINK86 A.OBJ, B.OBJ, C.OBJ TO PGM.86 END
2	IF A.OBJ > A.SRC THEN RUN PLM86 A.SRC END
3	IF B.OBJ > B.SRC THEN RUN PLM86 B.SRC END
4	IF C.OBJ > C.SRC THEN RUN ASM86 C.SRC END

able development time can be lost trying to work someone else's modified modules if documentation specifying what, where, when, and why changes were made is not available. In fact, as programs become more complicated, even the module writer may not exactly remember the history of the module.

Automatic documentation

SVCS provides a tool for automatic documentation of these facts. When a new module is created, it is set to level 1. All subsequent versions of the module are maintained with in a single file. Changes to the module are stored as "deltas" to the original. SVCS automatically records what changes were made and when they were made, and it requires the modifier to specify a reason for the change. The project manager may create a software checkpoint at any time by declaring the module as the next release level; subsequent deltas will then be applied to only this new release level.

Other capabilities in SVCS also increase project control. Restrictions may be placed on who is allowed to make changes to which modules and at which levels. An identification facility is also included, allowing the system to stamp modules containing object code with version information. From this information alone, a user can determine the level of source code used to generate the object module and thereby determine exactly which level of software is current and which level is being executed. To aid support groups in future maintenance of the program, any level of a software system can be regenerated from the original modules.

The second important program management tool on NDS-II is called MAKE, (see "MAKEing it easy to revise programs," above). When MAKE is invoked, a software system is automatically generated from the most current version of specific modules delineated by a dependency graph. MAKE ensures that the software generation is current and correct, while recompiling only program modules that need to be updated. To coincide with the concept of modular program development, any component of a MAKE could invoke another MAKE to generate a lower-level component such as a library. □

intel

ARTICLE
REPRINT

AR-425

IEEE SPECTRUM March 1986

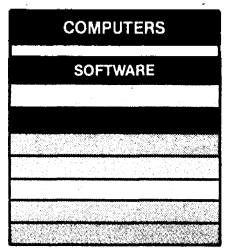
Helping Computers Communicate

JOHN VOELCKER

Order Number: 280271-001

Helping computers communicate

The Open Systems Interconnection model promises compatibility for a variety of computer systems, although not all its functions are yet defined



Computers made by different companies ordinarily do not "talk" to one another. This aloofness sometimes applies even to different types of computers made by the same company. And when it comes to computerized systems, like machine tools and automatic teller machines, the communications problems can be nightmarish. Aside from expensive customized adapters and software links, compatibility remains an elusive target of the computer industry.

Even when two computers or computerized systems can be made to talk to each other, problems may arise in getting networks—whether telephone, satellite, or microwave—to handle the conversation. Will universal compatibility among computers, computerized equipment, and communications networks ever become reality?

Many industry leaders believe that the Open Systems Interconnection (OSI) model is the key to making users' dreams come true. The set of OSI standards being developed by the International Organization for Standardization (ISO) in Geneva, Switzerland, is a framework for defining the communications process between systems. It includes a Reference Model, with seven layers that define the functions involved in communicating; and definitions of the services required to perform these functions.

To implement the OSI model, the ISO also describes protocols —specifications for how information is coded and passed between parties in a communication. Only protocols can actually be implemented; both the Reference Model and the service definitions are merely structures for discussing the functions involved in communications between dissimilar equipment.

Standards are emerging gradually

Computers, computerized equipment, and communications networks are all covered by OSI. This has created considerable confusion among users, because systems that manufacturers claim "conform to OSI" may not necessarily be compatible with other systems for which the claim is made. What the label means is that the equipment uses some of the OSI standards—a subset of those that have been defined so far. Testing for OSI conformance is just beginning.

Many of the protocols to implement OSI are now complete, and some manufacturers offer products to implement various of these standards. The ISO will continue to expand the functions covered by OSI as new communications network architectures and technologies emerge. But the revisions are being made, the architects say, so that older equipment will not be rendered obsolete.

Some companies—General Motors and the Boeing Co., for example—already specify the use of OSI protocols in certain computer networks. IBM is examining how it can make its own computers and network standards communicate with OSI equipment. Digital Equipment Corp. has announced that within three years, it will replace proprietary protocols in its DECnet network with OSI protocols. A number of suppliers have banded together in a new group, the Corporation for Open Systems, to promote acceptance and use of OSI protocols. In short, the outlook for OSI is promising.

No one contends, of course, that all computerized equipment should be covered by OSI. There seems little need, for example, to allow the microprocessor in a new refrigerator to communicate with the international banking industry's funds-transfer network. But in many industries, the ability to interconnect many different computer systems and communications networks could radically improve the way business is done.

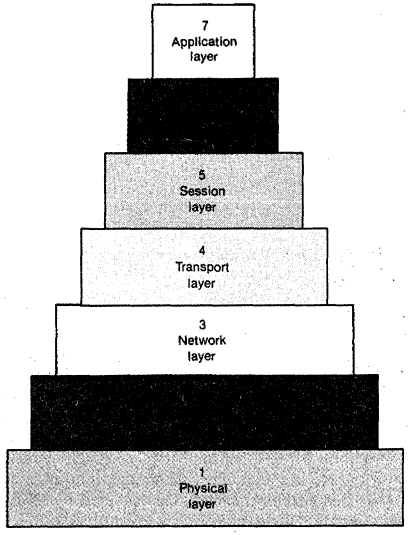
Frustrations abound

Consider the plight of a design engineer who must use a newly installed computer-aided engineering (CAE) system to analyze the deformation of a cylindrical strut with a load applied. The strut was designed on an older computer-aided design (CAD) system made by another company; the system uses a different graphic descriptor language to represent cylinders than the CAE system does. The design engineer must enter the description of

John Voelcker Associate Editor

Defining terms

Application process: a part within an open system that processes information and uses OSI communication services to communicate with other application processes in other open systems.
Channel: the part of a communications system that connects a message source to a message link; a path for electrical transmission between two or more points.
Conversation: an interactive exchange of information between two systems or systems users.
Function: an action performed to further the communications process by parts of a communications system.
Gateway: devices or systems to connect different network architectures having different protocols by providing protocol translation. Gateways may use all seven layers of the OSI model, but must include Layers 1 through 4.
Layer: a set of network-related services within an OSI networking architecture.
Open system: a computer processor or set of connected processors, for which standards are published, that allows an application running in the system to communicate with other applications in the same or other systems.
Packet-switching: the use of software to route messages dynamically from source to destination within a communications network.
Protocol: a specification for coding messages exchanged between two communications processes.
Service: a function offered by some part of an open system to communicating application processes.



[1] The OSI Reference Model breaks the process of communicating into an orderly sequence of seven layers.

the strut into the CAE system to analyze it. But if he makes any specification changes, he will have to enter them into the old CAD system, which will ultimately generate the drawings to produce the part.

These drawings will be sent to the machine shop, where a prototype part will be produced on a digitally controlled lathe. Once again, the part description must be entered into a computer terminal—that of the machine tool—before the next step in the production process can be completed. If all of these machines could communicate, the engineer could change the specification and transmit the design automatically to the machine tool.

The systems designer who must connect automated office equipment from several manufacturers faces a similarly challenging task. While compatible personal computers can be connected to one of the many local-area networks, other types of computers are not so easily attached.

An IBM mainframe used for accounting and corporate record-keeping, for example, cannot easily exchange information with the company's personal computers. Even the physical media for data storage differ—large tape drives for the mainframe, 5¼-inch floppy diskettes for the personal computer.

To "hardwire" the personal computers to the mainframe would require, among other accommodations, the ability to translate every file from one character set to another. The same obstacle applies to most minicomputers, which might be used for other applications like inventory control in a small warehouse.

This lack of compatibility has remained essentially unchanged for at least 20 years. Families of computers or computerized equipment from a single manufacturer can be connected by proprietary communications protocols, but this ties users to a particular supplier's equipment, locking out competing manufacturers.

OSI to the rescue

The OSI model offers a way to establish unity in the fragmented computer and communications fields. It provides a framework for connecting open systems, allowing any supplier to construct a system that communicates with another made by a different company.

Richard desJardins, chairman of the ISO subcommittee responsible for OSI, notes, "The OSI Reference Model simply describes the many functions involved in a communication between two computers or systems, and the terms used to define those functions."

Implementations of these functions consist of software written to span the gap between the application process, which starts the communication—a program in an automated teller machine, say, that responds to a customer's balance request—and the physical medium over which the communication travels—the bank's private telephone lines, in this case. Often this software is embedded in special-purpose circuitry that is included in computers or other communications equipment.

The physical medium is simply the "channel" over which the message is sent. It includes not only the wires in a telephone system, but also transmitting and receiving stations for satellite and microwave communications, as well as local-area networks.

In each layer of the Reference Model, major functions have been defined. International standards define the services and the protocols to implement them. The OSI Reference Model, defined by ISO 7498, is complete and was adopted by the ISO in 1984 [Fig. 1].

The bottom layer of the Reference Model, Layer 1, is called the Physical Layer. It includes the functions to activate, maintain, and deactivate the physical connection. It defines both the functional and procedural characteristics of the interface to the physical circuit; the electrical and mechanical specifications are considered to be part of the medium itself.

Layer 2, the Data Link Layer, covers synchronization and error control for the information transmitted over the physical link, regardless of the content. This can be thought of as "point-to-point error checking."

Layer 3 is the Network Layer. Its functions include routing communications through network resources to the system where the communicating application resides; segmentation and reassembly of data units; and some error correction.

The Network Layer acts as the network controller by deciding where to route data—either out along a physical network path or up to an application process. Data routed between networks or from node to node within a network requires only the functions of Layers 1 to 3 [Fig. 2]. The network node is called a relay system.

End-to-end reliability ensured

The Transport Layer, Layer 4, includes such functions as multiplexing a number of independent message streams over a single connection when desired, and segmenting data into appropriately sized units for efficient handling by the Network Layer. Through these functions, it compensates for differences in the network services that have been provided. It also provides end-to-end control of data reliability, regardless of the type or quality of the network used.

The functions of Layer 5, the Session Layer, are to manage and synchronize conversations between two application processes. Data streams, for example, are marked and resynchronized to ensure that dialogues are not cut off prematurely. The layer provides two main styles of dialogue: two-way alternating (half-duplex), in which two parties alternate in sending messages to each other; and two-way simultaneous (full-duplex), in which two parties may send and receive at the same time.

The Session Layer's control functions are analogous to the use of control language to run a computer system. While Layer 5 selects the type of service, the Network Layer chooses appropriate facilities and the Data Link Layer formats the messages.

Layer 6, the Presentation Layer, ensures that information is delivered in a form that the receiving system can understand and use. The format and language (syntax) of messages can be determined by the communicating parties; the functions of the Presentation Layer translate if required. The meaning (semantics) of the message is preserved. If, for example, one application process

transmitted a file in ASCII code, while another used IBM's EBCDIC, the two sides would negotiate which encoding to use and which side would perform translation.

The top of the Reference Model, Layer 7, is the Application Layer. To support distributed applications, its functions manipulate information. It provides resource management for file transfer, virtual file and virtual terminal emulation, distributed processing, and other functions. It is the layer that will contain the most functionality, and it is certainly the one in which the widest variety of work is being done at present.

Viewed as a system, the layers of the Reference Model can be broken into two groups. The bottom three layers—Physical, Data Link, and Network—cover the components of the network used to transmit the message. The top three layers, however, generally reflect the characteristics of the communicating end systems. Their functions take place without regard for the physical medium actually used, whether it is a satellite, an X.25 network, or a local-area network (LAN). Only the two parties to a communication invoke the functions of the Session, Presentation, and Application layers. The Transport Layer acts as the liaison between the end system and the network.

Freedom of services provided

At each layer of the Reference Model, there are services to carry out the functions. For instance, a service such as requesting initialization of a conversation is needed to initiate the control function for a conversation between two end systems.

The services defined for each layer are performed by building on services provided by the layer directly underneath. Conversely, the services at each layer are called upon by those at the next higher layer. Thus when an application process initiates a communication, it passes its message down through each layer. The functions of each layer add value by providing services that enable the communication to be completed.

One shining feature of OSI is that these service definitions are independent. In other words, any service can be implemented regardless of the methods used to implement services in the layers above and below it. Error checking, for instance, may be provided in different systems by dissimilar devices or by unique software. As long as the device or software provides the service defined by OSI, using an approved protocol, it will perform the same function for the end user.

Specifying independent services allowed protocols for several layers of the model to be developed in parallel, before Subcommittee 21 of ISO's Technical Committee 97 had defined the entire set of services. In theory, it also allows individual service definitions to be modified without disturbing other layers in the model. However, few users are likely to implement protocols in such a way that their interfaces correspond to all the service boundaries. Instead, for instance, a single ROM chip might provide the functions of two or three layers together.

The ISO specifies protocols for each service definition within the layers of the model. These are descriptions of the bit coding formats in which specific information is passed between processes, as well as the procedures to interpret it. Protocols operate between "peer entities"—the parts of a system providing services for a given layer—in the different end systems. Thus information about Network Layer protocols in a message sent by one system is used only by the Network Layer in the receiving system.

A number of protocols may implement a given service, and more than one service may be provided by each layer of the Reference Model. In this respect, OSI can be viewed as a collection of worldwide engineering design activities, with overall coordination provided by the ISO Reference Model. Thousands of engineers from hundreds of organizations worldwide participate, including all major computer and network manufacturers.

The service definitions and protocols are in various states of development for each layer [Fig. 3]. Service definitions have been completed for Layers 2, 3, and 4, and work on other layers is well underway. Protocols for some layers are already international standards, including Network, Transport, and Session layers. The working group's goal is to complete the initial set of protocols for the Application and Presentation layers of the Reference Model by the end of 1986.

Some manufacturers have already moved to implement OSI protocols that have been completed. General Motors, for example, is using one set of OSI protocols in its Manufacturing Automation Protocol (MAP). Boeing is proposing a similar set for its Technical and Office Protocols (TOP). [The MAP and TOP standards will be covered in the April issue of *IEEE Spectrum*.]

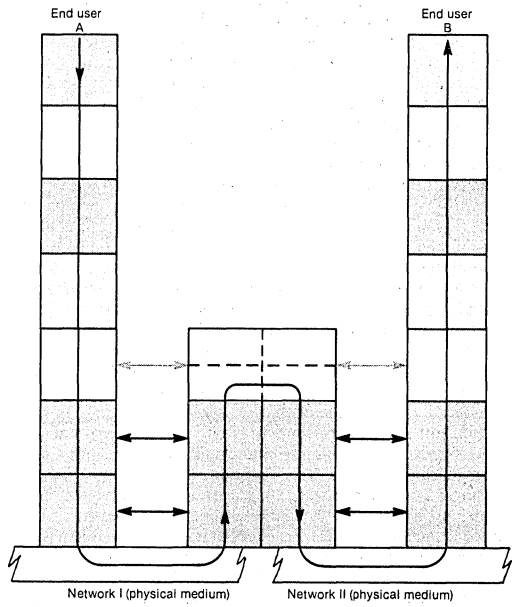
An application using a different set of OSI protocols, however, may or may not be able to communicate with MAP and TOP. OSI does not allow all computers and communications networks to communicate automatically and at will. Rather, OSI users will form "communities of interest" to limit the options. They will define a set of services to be provided by communicating machines in their particular industry and then implement those services in a handful of protocol options for each level.

Lower layers based on existing standards

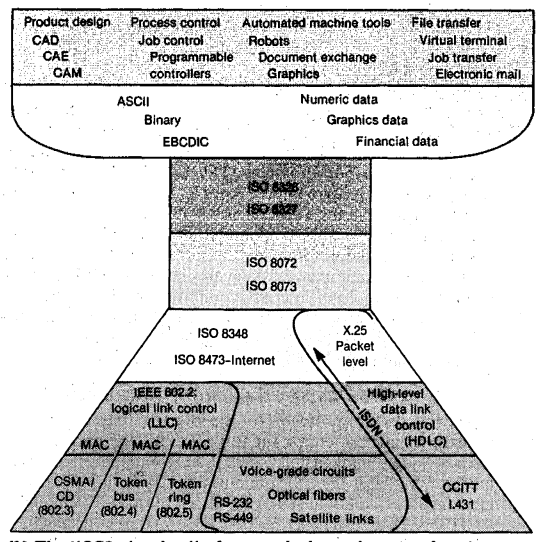
The initial set of protocols for the lower OSI layers are based on existing international standards and thus the protocols are already widely implemented. There are a number of physical medium standards for OSI communication over short distances, including the traditional analog RS-232C, the more recent digital CCITT X.21, and the IEEE 802 LAN standards (ISO 8802.3, 8802.4, and 8802.5). [For a comparison of the IEEE LANs with the OSI model, see "Lining up against the layers," p. 68.]

For longer-distance communication among OSI applications, the physical interface for the Integrated Services Digital Network (ISDN) may become the dominant standard [see "A universal plug already developed," p. 70].

The set of protocols that provides the services of Layer 2 over X.25 networks is called the High-level Data Link Control. Sev-



[2] A message may pass through many relay systems on its way between application processes. In an OSI application, the path taken is invisible to the end users. Only a relay system "knows" what route a message is using.



[3] The "OSI wineglass" of protocols shows the many functions covered by upper layers and the multiple options for physical media at the lower layers. The Session and Transport Layers, however, have fewer alternatives and are now international standards.

eral subsets have been defined; work is proceeding on others. The first to be defined was the CCITT X.25 Link Access Procedure B, for balanced connection-oriented communication—that is, a one-to-one link over a dedicated circuit between two parties.

Next to come was an option allowing multilink communication, or splitting a single communication among several physical channels. And most recently a new subset provides multiplexing functions—allowing several communications to use a single physical channel. Three types of service are provided by the High-level Data Link Control: connection-oriented, connectionless, and single-frame transmission.

Connection-oriented service requires a connection to be established between the two end systems before the communication is transmitted. The connection can be either physical—a set of wires—or "virtual"—preplanned routes over which packets will travel. A good analogy here is a telephone call; a line is established and dedicated to a particular conversation before the two parties begin talking.

Connectionless service involves communication in which each data unit, or packet, travels independently. The path may be established in advance—as on certain LANs—or as the message arrives at each network junction. A good analogy for this type of service is mailing a letter, since it will travel to its destination independently of any others sent to the same address, regardless of whether the same route is used.

Single-frame transmission sends only one frame of data at a time. An example of this is a remote sensor that transmits a signal to a guard station if it detects motion.

Layer 2 protocols may break a stream of data up into frames, which are transmitted sequentially, and may require a frame acknowledgment signal from the receiving system. If so, the frame is retransmitted if the signal is not received. The Data Link Layer may also provide flow control—monitoring the rate of frame transfer—so that systems can exchange data at different speeds.

How to connect networks

As for Layer 3, the Network Layer, its service definition includes network connection, data transfer, reset, and connection-release functions. Expedited data and receipt-confirmation services are optional and are specified when a network connection is established. The receipt-confirmation service supports conformance to the CCITT X.25 standard. An addendum to ISO 8348 is now being developed to add connectionless network service—the simple transfer of a data unit.

The Network Layer must provide for many network types, only some of which have been fully identified. Each type or family of protocols within the layer has a unique identifier, so the protocol can be identified and changed during the transmission of a message. All protocols in the 1984 revision of X.25 are accommodated without change.

The service definition (ISO 8072) and protocol specification (ISO 8073) are now approved for Layer 4, the Transport Layer. ISO 8073 specifies several classes of protocol for connection-oriented communication, with a wide range of functionality—from the simple (Class 0), for use with highly reliable X.25 networks, to high-quality service (Class 4), with error detection and recovery for possibly unreliable networks.

Specifically, Class 4 service ensures that data is not lost, duplicated, or corrupted in transit and that it arrives at its destination in the right order. The Transport Layer can also provide end-to-end error checking between communicating parties, or it may rely on the quality of service provided by the Network Layer. Work is now underway on Transport Layer service definition and protocol specification for connectionless data transmission.

The service definition (ISO 8326) and protocol specification (ISO 8327) are also approved for Layer 5, the Session Layer. While this layer is full of options among the facilities available to the users, initial implementations will contain two subsets of service definitions: the session kernel, for establishing and releasing a session; and the basic combined subset, which adds token management—a request for use of resources—to the kernel.

For Layer 6, the Presentation Layer, an Abstract Syntax Notation 1 developed by CCITT has been adopted as ISO 8824 to provide rules for defining and recording the meaning, or semantic content, of messages. Associated with this are a basic encoding rule (ISO 8825), as well as custom encodings registered with ISO, to turn such notations into actual messages for transfer.

Layer 7, the Application Layer, is the only one that provides services directly to the application process. It does so by drawing on the services of all six layers below it. Conceptually, the Application Layer is broken down into three parts: a user element, common-application service elements (CASEs), and specific-application service elements (SASEs).

The user element represents functions specific to the application process that needs to communicate. It selects among the services offered by the rest of the layer, including the CASEs and SASEs, on behalf of the application program.

The CASEs are general-use capabilities needed by nearly all applications. Included among CASE functions are commitment, concurrency, and recovery for distributed processing.

The SASEs include file transfer, access, and management; job transfer and manipulation, for distributed batch jobs; message handling facilities; virtual terminal systems, which allow remote systems to communicate as terminals; and directory services.

Serving 'communities of interest'

These functions serve specific industries, known as communities of interest. The financial services and banking industry is one example of a broad community of interest; another is the users of automated industrial equipment. Each group has unique needs and requires Application Layer services specific to the industry.

For example, industrial automation applications may not have a high volume of on-line inquiry. But because factory communication must occur in real time—as opposed to sending messages in batches—the maximum permissible waiting time between

commands sent to the machine tools must be very low. In this environment, real-time "foreground" protocols will be carefully designed for high performance. For "background" processing—analyzing production data, for instance—the job transfer and manipulation function of the SASE might be used.

The message from the Application Process, plus information added by each layer below, forms the frame that is sent out over the network [Fig. 4]. At each layer, header control information is appended to the data unit received from the layer above. This information identifies the protocol options used and gives other data about the message and its routing. At the receiving end, the header information is removed and processed by each layer. Then the remaining data unit is passed up to the next layer, where a similar operation takes place.

A good analogy to show how the functions of the OSI model operate is the production and transmission of a simple business letter. It parallels the OSI process, using only the language of business communications; no computer terminology is needed [Fig. 5].

The thank-you letter

Imagine that the president of a West German company has agreed to buy 50 tons of wheat from a firm in Wichita, Kan. Because he got a good price, he asks the public relations manager to send a thank-you note to the sales director of the Wichita firm.

The West German executive represents the application process that initiates a communication. He deals in terms of the meaning of the communication, or the semantics; he merely tells the PR manager to send a thank-you note. The PR manager actually gets the machinery going. He is the Specific Application Service Element of the Application Layer, calling on the services of the layers below him to meet his needs in transmitting the message.

The West German PR manager dictates the note onto a cassette tape and gives it to his secretary—who acts as the Presentation Layer. She translates the message into English and types it as a formal business letter. In OSI terms, she has prepared the Transfer Syntax—a string of data in a language common to the sender and the receiver—in this case, English.

After typing it, the secretary hands the letter to her administrative assistant—the Session Layer. He records the letter in the German company's file on Wichita Wheat Co., ensuring that the right person has been addressed, with the correct title and spelling, exact office number, and other details. This checking allows both ends of the communication to organize and synchronize their dialogue, by noting where the message goes and when it was sent. If there is back-and-forth exchange of information, the Session Layer will manage the dialogue.

The next layer—Transport—is provided by the manager of shipping and receiving. His job is to negotiate the quality of service available from the Network Layer, approve the connection, and provide receipt and delivery. He is really guaranteeing end-to-end transmission. If something untoward happens during transmission, he will recover by sending another copy of the letter—hence he always copies a letter before sending it.

After copying the letter, he assigns a sequence number (in this case, "1 of 1"). Then he passes the shipment—tagged with both destination address and phone sequence number—to a shipping clerk. He tells the clerk to establish a route over which the note will be sent to Kansas. The Network Layer (the shipping clerk) will select the routing and advise the Transport Layer (the transport manager) of it.

The shipping clerk calls his counterpart in the German company's New York City office. He learns that the company's internal mail service can take the shipment to the New York office, and Federal Express will deliver it to Wichita the next day. Note that OSI applies to communications over private networks (the company's internal mail operation) and public networks (Federal Express).

He attaches a routing slip and puts the letter with others into a mail cart labeled "New York." Then he sends the cart to the mailroom, which serves as the Data Link Layer.

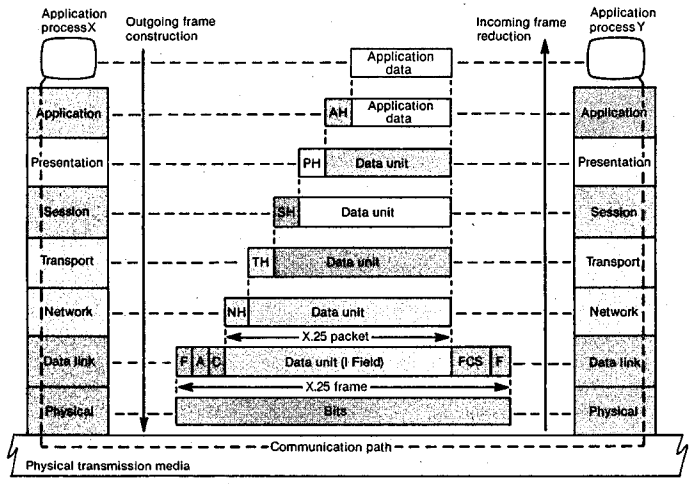
The mailroom workers also make copies of everything they receive, bag the mail, and weigh it on a very accurate scale. They note the destination and weight of each mailbag on a tag attached to the bag. Then they move the bag to the loading dock—the Physical Layer, or the interface to the physical medium (the trucks, trains, and airplanes to take it to the United States).

The workers on the dock call the trucks and load the mailbags onto them when they arrive. At this point, the "bits" have left the machine and are in transit on the medium—the communication has been sent on its journey.

When the mailbag arrives in New York City, the workers on the New York loading dock—the Physical Layer—pass the mailbag to the workers in their mailroom—the Data Link Layer. This mailroom has a scale identical to the one in West Germany, which can detect the loss of even one letter from the mailbag. If the weight of the bag does not match that on the label, the whole shipment is rejected and the mailroom in Germany is notified to send replacement copies of all the letters, using the duplicates they have kept.

This task represents "frame check sequences" performed by the Data Link Layer. In this case, the weight of the letters matches exactly, so the New York mailroom sends word back to Germany that the mailbag is OK. Then the shipment goes to the routing clerk in New York—the Network Layer—who opens the mailbag and sorts the mail.

Mail for employees in the New York office gets passed along to the transport manager—the Transport Layer—for processing up



[4] A message passed from Application Process "X" down through the layers to an X.25 network acquires header information from the functions of each layer. The receiving Application Process "Y" does not see this, however; its Application Layer passes along only the message sent by "X." The message is stripped of all its headers and frame information by the layers below the application process. The bitstream actually sent over the network is an X.25 data frame.

in the organization. Other mail remains at the Network Layer to be rerouted. The routing clerk recognizes the thank-you letter as one to be sent through Federal Express, so she tags it for Federal Express and sents it back to the mailroom.

The mailroom groups together (multiplexes) all mail for Federal Express delivery to the Wichita firm, as there are many letters concerning the grain deal. Again the contents are copied, weighed, sealed (in a Federal Express package), and tagged with a new shipment number and address. The bags go out onto the loading dock and away in the Federal Express trucks.

Assuming Federal Express and the Wichita firm use an OSI model, they will go through a similar process to route the package. In all the cases, only the lower three layers—Network, Data Link, and Physical—are involved when a message is routed via intermediate networks. The upper layers — Transport and above—are involved only at the origination and destination of a communication.

When the Federal Express package arrives in Wichita, the routing clerk passes it up to the transport manager, who checks the packing slip and telephones her counterpart in Germany to let him know that the letter has arrived in good order.

In this way the Transport Layer acknowledges "end to end" communications. All previous acknowledgments have been at the Data Link Layer, from one leg of a journey back to the previous leg. This final acknowledgment connects the end of the journey to the beginning, no matter what carriers—reliable or not—have been used in between.

Once the communication has been received and acknowledged by the Transport Layer, it is passed along to the Session Layer. A file clerk logs the letter in the file for the German wheat buyer and takes the letter to the Presentation Layer — the sales director's secretary. She reads the letter and determines that it is in English; no translation from German is necessary.

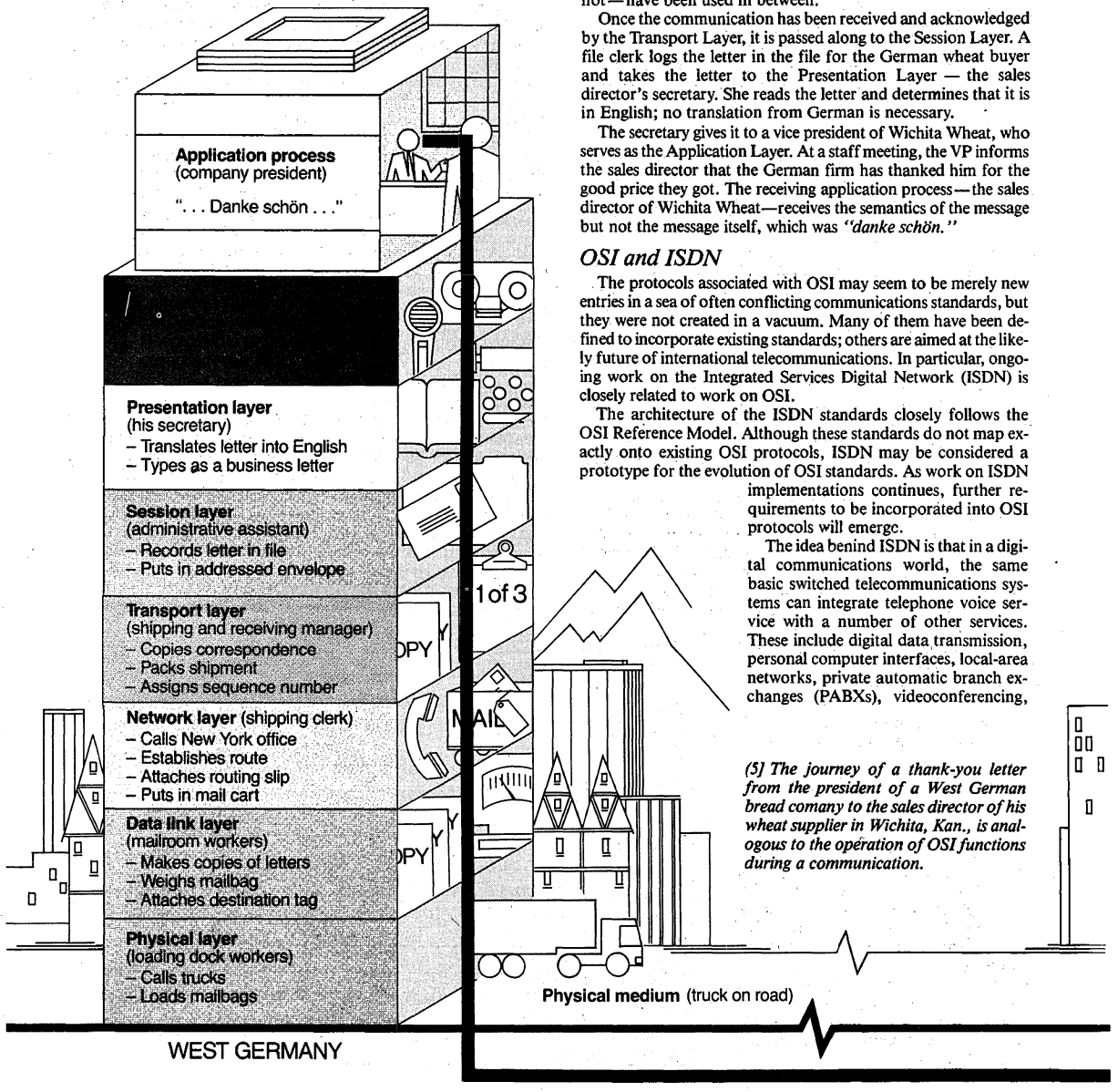
The secretary gives it to a vice president of Wichita Wheat, who serves as the Application Layer. At a staff meeting, the VP informs the sales director that the German firm has thanked him for the good price they got. The receiving application process—the sales director of Wichita Wheat—receives the semantics of the message but not the message itself, which was *"danke schön."*

OSI and ISDN

The protocols associated with OSI may seem to be merely new entries in a sea of often conflicting communications standards, but they were not created in a vacuum. Many of them have been defined to incorporate existing standards; others are aimed at the likely future of international telecommunications. In particular, ongoing work on the Integrated Services Digital Network (ISDN) is closely related to work on OSI.

The architecture of the ISDN standards closely follows the OSI Reference Model. Although these standards do not map exactly onto existing OSI protocols, ISDN may be considered a prototype for the evolution of OSI standards. As work on ISDN implementations continues, further requirements to be incorporated into OSI protocols will emerge.

The idea benind ISDN is that in a digital communications world, the same basic switched telecommunications systems can integrate telephone voice service with a number of other services. These include digital data transmission, personal computer interfaces, local-area networks, private automatic branch exchanges (PABXs), videoconferencing,



(5] The journey of a thank-you letter from the president of a West German bread comany to the sales director of his wheat supplier in Wichita, Kan., is analogous to the operation of OSI functions during a communication.

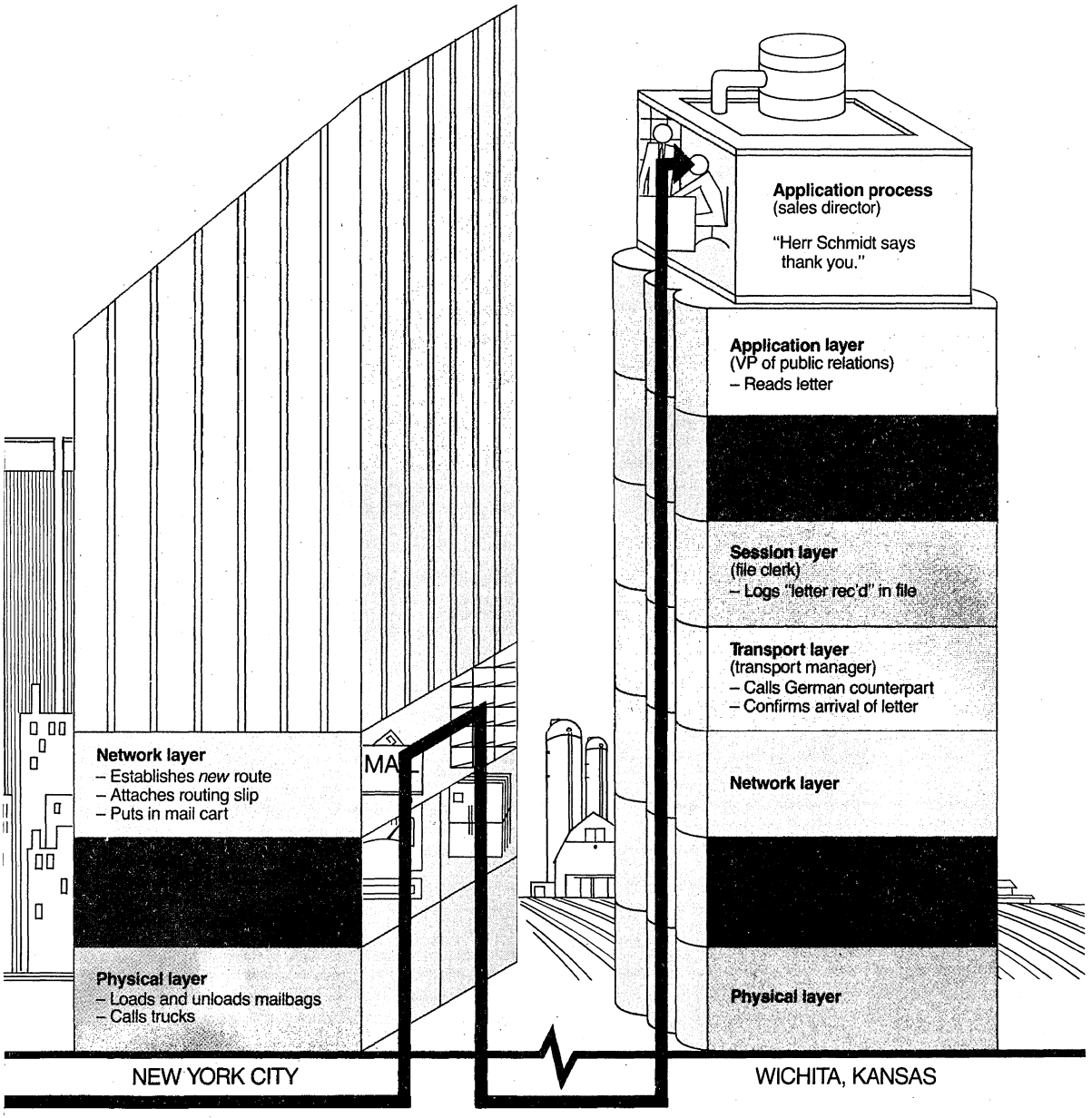
and joint-use remote applications, like automatic teller machines and self-service fuel pumps.

Only the lower three layers of OSI are applicable to initial ISDN work [Fig. 6]. The basic ISDN interface is composed of a 16-kilobit-per-second signaling channel (D-channel) plus two circuit-switched 64-kb/s digital channels (B-channels). Depending on the characteristics of connecting networks, ISDN offers access to the D-channel alone or in combination with one or both B-channels.

In the basic service, the ISDN Physical Layer operates with a bit stream of 192 kb/s and provides a multiplexing arrangement. Of this, 48 kb/s is control information that facilitates the multiplexing.

The signaling channel uses the Link Access Procedure D protocol for Data Link Layer services. It provides multiplexing for three functions: signaling information that controls switching connections on the B-channels; low-speed packet-switched services; and optional channels that can be used for sporadic low-bandwidth transmission, like burglar-alarm signaling.

The OSI Network Layer protocol for the D-channel is specified by CCITT recommendation Q.931. It provides the mechanism for making and breaking connections on the



Lining up against the layers

To understand the division of functions among the layers of the Open Systems Interconnection (OSI) model, it is helpful to compare it with some existing networks and communications standards. Perhaps the most widely known among data processing professionals is IBM's Systems Network Architecture (SNA), designed to provide a common architecture for communication within the company's diverse equipment.

The distribution of functionality among the layers of SNA varies significantly from that of OSI; the number of options is not as large, because SNA is intended for a limited set of machines operating in a known network environment.

The SNA transaction services layer has elements of both the application process and the OSI Application Layer. It provides services necessary for operation of the network—for instance, a program to agree on the number of sessions between network nodes. But it also includes such functions as electronic mail, defined under OSI as an application process.

The presentation services layer is comparable to the rest of the OSI Application Layer and to the entire Presentation Layer. It provides a high-level interface to application programs, taking high-level statements—such as SEND DATA—and translating them into lower-level service requests.

One "half-session" serves each user on either end of a communication, although more than one user may use a piece of equipment on the network. A half-session includes resources for data flow control — controlling requests and responses in a dialogue—and transmission control, managing buffer resources and expediting data. The services manager responds to requests from the presentation services layer to create, assign, and destroy conversations.

Beneath the half-session, the path control function is divided into three layers for IBM's "backbone" machines, predominantly mainframe computers. Peripheral equipment —personal computers, minicomputers, and terminals—uses a much simpler path control function without the sublayers.

Virtual route control provides flow control and error control to the logical—as opposed to the physical—network. Explicit route control establishes physical paths for connection. Transmission group control provides a multilink capability and ensures data delivery. And the SNA data link control provides reliable transfer of information between nodes, as does the OSI Data Link Layer. The physical layer is not explicitly defined in SNA, since the medium is known, but the OSI Physical Layer functions are implicit in the architecture.

The ARPAnet — developed for research purposes by the Department of Defense in the early 1970s — has a different distribution of functionality. Here too, the Application Layer provides the same user services as does the OSI equivalent. But there is no ARPAnet equivalent of the OSI Presentation and Session layers; either ASCII characters or data bits are passed through the network, and the communicating applications know what to send and expect.

The ARPAnet Service Layer provides OSI Transport Layer functions and a few from the Session Layer — namely the ability of the network itself to initiate a "graceful close" to a communication session if a user drops off the network.

ARPAnet's Internet Layer routes communications among networks, either to intermediate networks or to an application process if the message has reached its destination. Finally, the rest of OSI's Network Layer functions and the Data Link and Physical Layers are contained in ARPAnet's Network Layer. This layer addresses the real characteristics of the many actual networks across which the ARPAnet system operates. The functions are not strictly independent of the physical medium, however, as is the OSI Physical Layer.

The X.25 standard for packet-switched communication over national and international data networks defines one major way to implement network services.

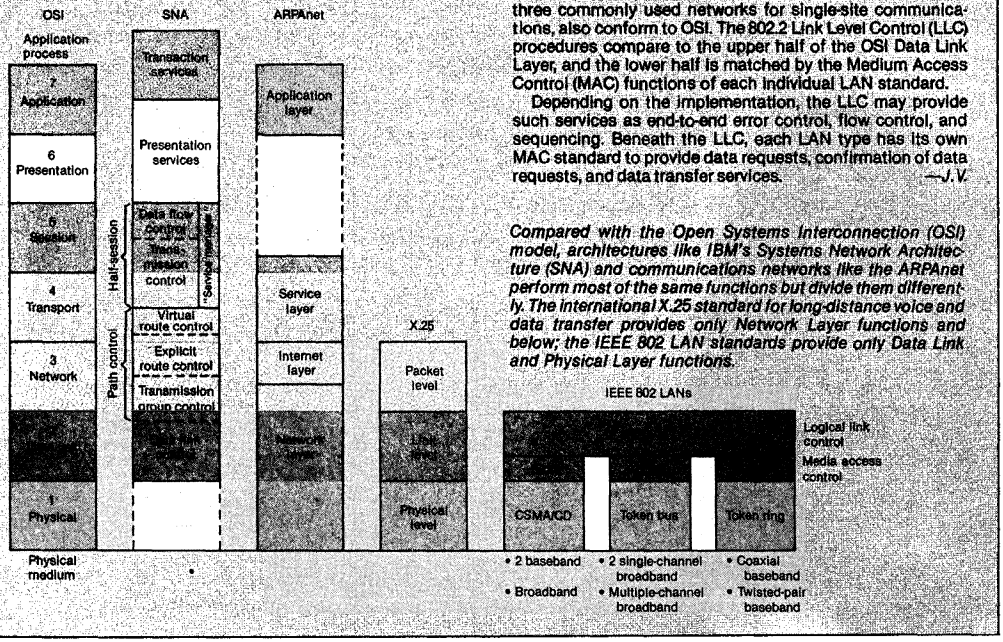
The functionality of the latest X.25 specification, issued in 1984, corresponds entirely to OSI. The X.25 Packet Level protocol, which routes and switches packets through network nodes, corresponds to the OSI Network Layer. The X.25 Link Level guarantees reliable data transfer across the physical link—as does the OSI Data Link Layer. And the Physical Level of X.25 includes the functional and procedural characteristics to activate, maintain, and deactivate the physical connection.

The IEEE 802 local-area networks (LANs), which define three commonly used networks for single-site communications, also conform to OSI. The 802.2 Link Level Control (LLC) procedures compare to the upper half of the OSI Data Link Layer, and the lower half is matched by the Medium Access Control (MAC) functions of each individual LAN standard.

Depending on the implementation, the LLC may provide such services as end-to-end error control, flow control, and sequencing. Beneath the LLC, each LAN type has its own MAC standard to provide data requests, confirmation of data requests, and data transfer services.

—*J. V.*

Compared with the Open Systems Interconnection (OSI) model, architectures like IBM's Systems Network Architecture (SNA) and communications networks like the ARPAnet perform most of the same functions but divide them differently. The international X.25 standard for long-distance voice and data transfer provides only Network Layer functions and below; the IEEE 802 LAN standards provide only Data Link and Physical Layer functions.



B-channels and for other ISDN control functions. For the packet-switching function, Layer 3 is the packet level of X.25. The protocols for optional functions will be defined by the CCITT at a later date or may be specified as a national option.

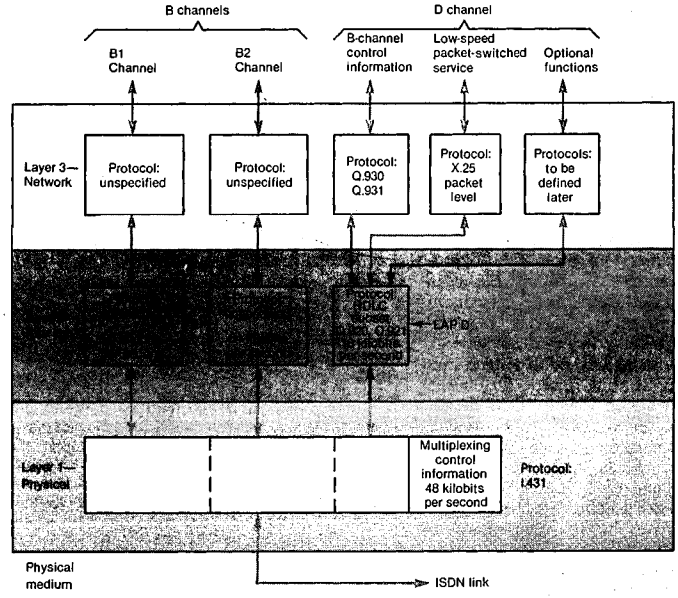
The data link and network protocols are unspecified for the B-channels; these channels provide a "transparent" facility that may use whatever protocols are appropriate for the application.

Above the Network Layer, ISDN protocols depend on the application being used. CCITT Recommendation I.212 covers the upper four layers of ISDN services, referring to them as Teleservices. Protocol recommendations for Layers 4 through 7 have been developed by CCITT.

But will it work?

Computer users—many stung in the past by false promises of compatibility—may be inclined to greet claims of compatibility with skepticism. If OSI is to catch on, there must be a way to verify that products conform to its definitions. Work on testing products for OSI conformance has just begun but is developing rapidly.

The main influence in the United States to date has been the National Bureau of Standards, headquartered in Gaithersburg, Md. A newly formed group of equipment manufacturers, the Corporation for Open Systems in Washington, D.C., is also likely to become an important factor in OSI testing.



[6] Initial ISDN work is concentrated in the Physical, Data Link, and Network layers of OSI. Above the network services, protocols for ISDN will depend on the application. For example, teletex terminal equipment interface specifications, character sets, and mixed-mode terminal capabilities are included in the OSI Application Layer protocols.

The NBS does not provide testing services to manufacturers; it simply develops methods and software to test conformance to various OSI protocols and sells them through the National Technical Information Service. Currently, software to test the complete set of protocols for the OSI Transport Layer and the upper—or Internet—portion of the Network Layer are available.

Under development at NBS are tests for the messaging, file transfer, and virtual-terminal protocols for the Application Layer; the agency is developing these test programs under a Department of Defense contract. Also being developed are programs to test the Physical Layer and the bottom half of the Data Link Layer for the IEEE token-bus local-area network standard.

The NBS hopes to bring on line shortly a service called Osinet, a nationwide network for manufacturers interested in OSI testing. There are three immediate goals, according to John F. Heafner, chief of the systems and network architecture division at the NBS: to promote the development and dissemination of testing systems; to allow vendor-to-vendor testing of products, or interoperability testing; and to offer demonstrations of OSI testing services and products.

A more distant goal is to tie together OSI testing centers around the world, Heafner said, noting: "We would like to be able to offer testing for worldwide product conformance, but only if we can be assured that there will be no trade barriers created to protect individual markets."

The goals of the Corporation for Open Systems are equally ambitious. Initiated by the Computer and Communications Industry Association (CCIA), the corporation will encourage the development of test capabilities that manufacturers can use during product development, to reassure customers that products being marketed conform to the appropriate standards.

The corporation, formed late last year by a group of 18 suppliers of computer and communications equipment, is not a standard-setting body, noted Jack Biddle, president of the CCIA. But it does intend to promote development of a universally accepted set of OSI protocols for individual applications, including such standard data processing functions as file transfer and management and electronic message handling.

The group does not plan to provide testing services, Biddle said, but instead will develop testing programs and services or subcontract this development to others. A possible provider of those services, he noted, is the Industrial Technology Institute of Ann Arbor, Mich. This not-for-profit organization is currently involved in testing compatibility with the MAP specification among suppliers of factory automation equipment.

Over the long term, the corporation hopes to convince executives in the computer and communications industries of the strategic importance of a single open network architecture. "There is a need for greater voluntary efforts in the standards community," Biddle said. "These activities are not yet accepted as an integral part of product planning strategies by many companies."

Biddle is confident that testing for OSI conformance will become vitally important to the world electronics market, but he expects "a long struggle to make it happen." Asked whether users were really demanding OSI-compatible equipment, he said: "You should have heard my phone ringing off the hook after the word got out about the corporation. They are frustrated, they don't like buying from just one vendor, and they want solutions."

IBM pursuing OSI

A big question is how OSI will affect IBM. For a quarter of a century, IBM has been the leader in the computer industry. Its 11-year-old Systems Network Architecture (SNA) is the most widely implemented communications architecture for mainframe computers, and IBM has often functioned as a de facto standard-setting body for computer networking.

At the start of work to define the OSI Reference Model in the late 1970s, IBM participated in standards meetings and technical sessions. "IBM contributed very significantly and very constructively," said Harold Folts, president of Omnicom Inc. in Vienna, Va., a telecommunications consulting and education concern.

"And there is no question that they are moving in the direction of OSI for the European market."

Last year Digital Equipment Corp. announced that it would gradually modify its Digital Network Architecture to conform to OSI protocols as they are developed. The 12 major European computer manufacturers have indicated that they too will adopt OSI protocols. To promote OSI, some European governments have introduced regulations requiring OSI compatibility in new data network installations. With a significant presence in Europe — as indeed it has anywhere — IBM has announced plans to support OSI there.

The company stated: "As standards for Layers 6 and 7 are agreed upon over the next two years, based on business considerations, IBM will develop products that will meet the requirements of both the customers and OSI standards." IBM said that "OSI will complement the well-proven SNA architecture" and that "OSI and SNA can supplement each other to provide a balanced solution for the management of networks and for the transfer of information between them."

IBM Europe offers OSI capability through Layer 5, which indicates that the company will offer OSI implementations in addition to its own SNA architecture. [For a comparison of SNA with the layers of OSI, see "Lining up against the layers," p. 68.] The company's center for research on OSI implementations is the IBM European Networking Center in Heidelberg, West Germany.

IBM's Open-systems Transport and Session Support software, first shipped last December, supports most functions of OSI Layers 4 and 5 on the IBM/370 mainframe. The company has also offered several products for Layers 1 to 3 of OSI, mainly interfaces for various equipment to connect to X.21 and X.25 communications networks. But the Open-systems Transport and Session Support software is IBM's first comprehensive offering for OSI connectability above the Network Layer.

Currently the company is testing the X.400 messaging standard, a set of CCITT recommendations developed within the OSI framework. IBM may attempt to provide a bridge between its own document architectures and X.400. There will ultimately be a host of applications to which IBM's massive array of equipment will have to be connected, including electronic mail, teletex, videotex, and other such European services.

Many observers feel that ultimately IBM will offer not only full OSI implementations but also gateways to allow OSI to interconnect with existing SNA networks. "There will be a migration of SNA to OSI standards, probably without a lot of flag waving," said Folts of Omnicom. "They will offer two standards to start with, then they will merge—they can afford to make major leaps without worrying about backward compatibility." IBM has also recently joined the Corporation for Open Systems.

The implications are profound. If IBM's equipment uses essentially the same communications protocols as those of its many competitors, the company will be forced to compete increasingly on the technological merits of its products and perhaps on price. In a sense IBM will lose some control over the direction of computer equipment and design that it has enjoyed — particularly in the United States — for the last quarter of a century.

In some ways the promise of OSI has been oversold. It is not a magic cure-all that will allow every variety of computer equipment to be plugged together as stereo components are.

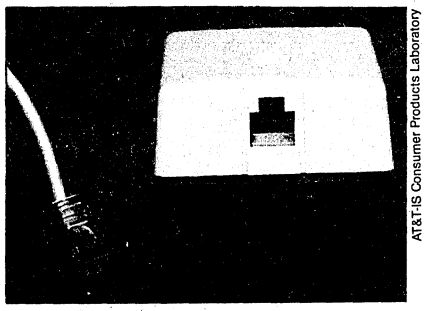
But OSI probably has a better chance than most of living up to its potential. For one thing, the group of potential users for OSI implementations spans many countries and diverse industries. Many suppliers will compete to supply conforming equipment.

OSI users can also decide which protocols are appropriate for their own needs. The best examples so far are the MAP and TOP standards, and there will be more as OSI gains public attention. The banking community, for instance, is working hard to apply OSI to electronic funds transfer and other services.

Finally, OSI leaves room for inevitable growth and change in a most elegant way. Standardizing protocols between functions — but not the design for implementing those functions — ensures compatibility between different systems while leaving room for innovative engineering.

One communications design engineer told *Spectrum* that "the only interesting question provoked by OSI is whether we end up with communications provided by the computer industry, or computers made by the communications industry." The answer may not be clear for decades. But OSI will provide a giant step toward the worldwide integration of computing and communication. From any perspective, Open Systems Interconnection promises to affect every part of both industries. It is, in the words of the same designer, "the only game in town."

A universal plug already developed



AT&T-IS Consumer Products Laboratory

The RJ-45 minimodular connector is likely to be approved as the universal interface for the Integrated Services Digital Network (ISDN), and thus ideally would be used for many Open Systems Interconnection (OSI) applications. Developed by AT&T Bell Laboratories, it is an eight-wire version of the familiar RJ-11 jack and plug widely used in U.S. telephone terminals and instruments (see photo).

The pins are arranged as follows: 1 and 2 are power sources, 3 and 6 transmit, 4 and 5 receive, and 7 and 8 are power sinks. Because the plug centers itself in the socket, the current four-pin plug would contact pins 3 to 6, allowing customer premises in the United States to be wired with the new socket and still remain compatible with existing telephone equipment.

While it may appear to be fragile, the plug has proved to be quite rugged, and it meets a number of criteria: it is small, keyed, self-orienting, and can be released without any tools. The connector set is now an International Organization for Standardization draft standard, and chances for final approval appear good.

—*J.V.*

To probe further

Over 20 articles in the *Proceedings of the IEEE* for December 1983 cover virtually all aspects of Open Systems Interconnection in detail. This issue can be ordered from the IEEE Service Center, 445 Hoes Lane, Piscataway, N.J. 08854.

An index of standards relating to OSI is available from Omnicom Inc., 501 Church St. NE, Suite 304, Vienna, Va. 22180. Proposed, draft, and approved ISO standards are available from the American National Standards Institute, 1430 Broadway, New York, N.Y. 10018.

The IEEE 802 LAN standard documents (802.2, 802.3, 802.4, and 802.5) are available from the IEEE Service Center. For further discussion of IEEE 802 LANs, see "Local area nets: a pair of standards," by Maris Graube in the June 1982 issue of *Spectrum*. For more details on ISDN, see "The innovation revolution awaits," by Paul Wallich and Glenn Zorpette, in the Nov. 1985 issue of *Spectrum*. Copies of both issues are available from the IEEE Service Center. ◆