

**THIS BOOK
CONTAINS
NUMEROUS PAGES
WITH THE ORIGINAL
PRINTING BEING
SKEWED
DIFFERANTLY FROM
THE TOP OF THE
PAGE TO THE
BOTTOM.**

**THIS IS AS RECEIVED
FROM THE
CUSTOMER.**

STANDARDIZATION OF BASIC FILE MANAGEMENT SERVICES
AND I/O DATA TRANSFERS FOR HETEROGENEOUS MINI
COMPUTER NETWORKS

by

DOLAN M. MCKELVY

B. S., UNITED STATES AIR FORCE ACADEMY, 1971

A MASTER'S REPORT

submitted in partial fulfillment of the
requirements for the degree

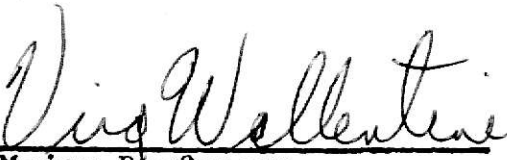
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1976

Approved by


Major Professor

LD
2668
R4
1976
M 319
C. 2
Document

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	iv
LIST OF FIGURES AND TABLES.....	v
1. INTRODUCTION.....	1
1.1 Report Objectives.....	1
1.2 Motivation/Significance.....	1
1.3 Approach To Problem.....	4
1.4 Report Structure.....	5
2. PROPOSED NETWORK STANDARD CONTROL BLOCKS.....	7
2.1 Conceptual Considerations In Control Block Design.	7
2.2 Network Control Block Descriptions.....	8
2.3 Transformation Table Analysis Using Examples.....	15
3. NETWORK CONVERSIONS AND ALGORITHMS FOR MACHINES.....	19
3.1 Network Conversion Flow Overview.....	19
3.2 English/PASCAL Definitions of Conversion Modules..	31
3.3 Examples of Complete Network Conversion Flow.....	35
4. DISCUSSION/CONCLUSIONS.....	44
4.1 Summation of Accomplishments.....	44
4.2 Problem Areas.....	46
4.3 Future Research And Approaches.....	48
REFERENCES.....	50
APPENDICES	
APPENDIX A--MACHINE TRANSFORMATIONS	
Data General Nova.....	52
Interdata 8/32.....	61
Interdata 7/16.....	69
APPENDIX B--MACHINE DEPENDENT PARAMETER BLOCKS	
Data General Nova.....	78
Interdata 8/32.....	80
Interdata 7/16.....	84

APPENDIX C--MACHINE/COMMAND DEPENDENT PARAMETERS

Data General Nova.....	89
Interdata 8/32.....	90
Interdata 7/16.....	90

APPENDIX D--MACHINE FILE STRUCTURE COMPARISONS.... 92

APPENDIX E--MACHINE DEPENDENT COMMAND HIPOS

SRMN-modules.....	94
DRNM-modules.....	104
DSMN-modules.....	116
SSNM-modules.....	122

ACKNOWLEDGEMENTS

I wish to express my sincere thanks and appreciation to my major advisor Dr. Virgil E. Wallentine for his continued guidance and direction. I also want to thank the other members of my committee, Dr. William J. Hankley, and Dr. Myron A. Calhoun, for their assistance during finalization of this report.

To my wonderful wife, Pat, and my children, Amy and Shawn, I owe a very special thanks for their understanding and support throughout this endeavor.

LIST OF FIGURES

Figure		Page
1	Local Operating System Implementation of MIMICS	3
2	Network Standard File Management Control Block	9
3	Network Standard I/O Control Block	12
4	Overview Flow of Network Conversion Process	22
5	Network Trap Handler	24
6	Network Destination Handler	25
7	Network Response Handler	27
8	Network Conversion of Successful FMS CLOSE	37
9	Network Conversion of Bad FMS Error Code	39
10	Network Conversion of Random I/O Request	41

LIST OF TABLES

Table		
1	NFMB Detailed Mnemonic Descriptions	10
2	NIOB Detailed Mnemonic Descriptions	13
3	Explanation of Mnemonics in Figure 4	23
4	Functional Responsibilities of Each Module	28

1. INTRODUCTION

1.1 REPORT OBJECTIVES

This report is an initial attempt to develop standard network control blocks for basic file management services and data transfer Inputs/Outputs (I/O) for use in mini-computer networks of heterogeneous machines with real-time operating systems. One control block has been designed to handle capabilities such as creating, opening, closing, and deleting files, and another control block designed to handle the file I/O for data transfers. Transformations between the network standard control blocks and the machine dependent parameter blocks are presented. A network conversion process is modularly described. Each module in the conversion process is defined in detail using current documentation techniques. Sample hand simulation traces are used to display the designs conceptual correctness.

1.2 MOTIVATION/SIGNIFICANCE

The increasing interest and potential in computer networks coupled with technical advancements and a large variety of marketed mini-computers have created a tremendous need for standardization of file management services and associated parameters. Standardization of file management services in itself is an overwhelming task in light of the variety of computers on the market. However, by limiting the scope of possible computer architectures to basically the sixteen bit ASCII oriented mini-computers with real-time operating systems, a practical standardization of parameters

for file management services and I/O can be realized.

This research is of particular importance at the present time because of a U.S. Army research grant now underway at KSU on Functionally Distributed Computer Systems(R6). The major research effort involves design and implementation of a prototype mini-computer network of heterogeneous machines which will support a distributed data base management system. One primary concern is designing software to control inter-machine and inter-process communication. Figure 1 shows a current design overview of the inter-process communication which will be implemented as a set of modules below the local operating system on each machine. The parameter variations in file management and I/O capabilities between heterogeneous machines in the network requires some type of conversion and standardization of parameter information. This conversion process and use of standard control blocks is shown between the network user task and the sending/receiving function of the I/O request to/from the network resource controller. It is hoped that the network standard control blocks and the conversion modules and submodules designed in this report will help serve to further define that portion of the inter-process communication dealing with conversion and network standardization of file management and I/O parameter information.

The network user task 1 in Figure 1 is not aware of the network and associated processing. All I/O or file management services (FMS) requests are trapped by the local operating system and the conversion process as defined in

**THIS BOOK
CONTAINS
NUMEROUS PAGES
WITH DIAGRAMS
THAT ARE CROOKED
COMPARED TO THE
REST OF THE
INFORMATION ON
THE PAGE.**

**THIS IS AS
RECEIVED FROM
CUSTOMER.**

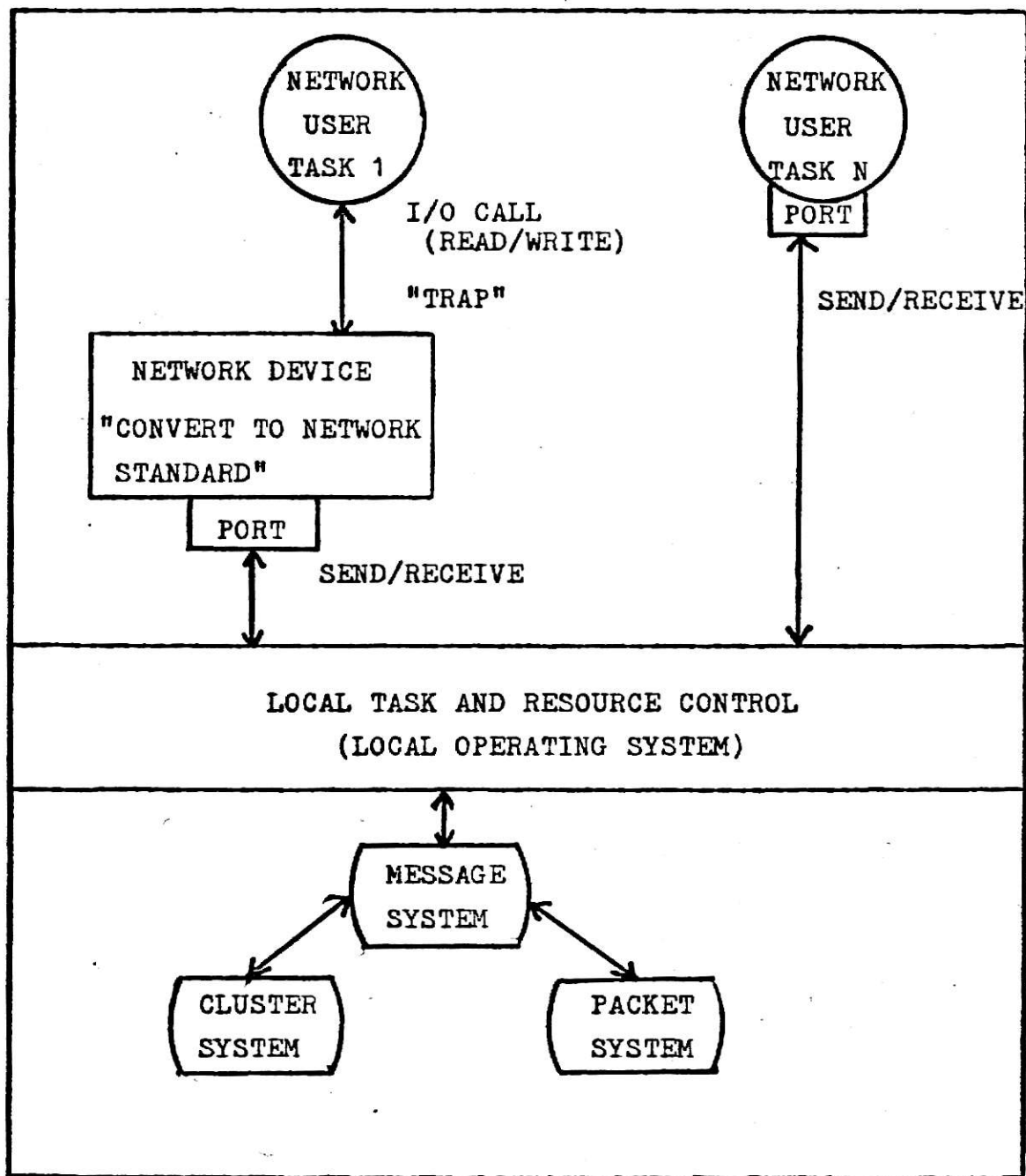
MIMICS

FIGURE 1:

Local Operating System Implementation of MIMICS

TASK 1---Unaware of the network.

TASK N---Aware of the Inter-Process Communications.

this report is performed. The network control block is then sent via the send/receive commands from a port through the network to the destination machine. The message system is responsible for describing the message and for sending variable length records throughout the network. The cluster system transfers accross the local network variable length records from memory to memory. The packet system breaks up messages and transports them one at a time to remote computers. Both the packet system and the cluster system are transparent to the software above them in the system. The network user in task N in Figure 1 is aware of the network and uses a network control language to eliminate the need for a conversion of the initial request into network standard form.

1.3 APPROACH TO PROBLEM

The first step taken in approaching this task was to investigate several representative mini computers (Data General Nova(R1), Interdata 8/32(R4), Interdata 7/16(R3)) and identify their file structures, file management services, I/O commands, associated parameters, characteristics, and limitations. This information was then integrated into network standard control blocks. Transformation tables were set up to handle parameter conversions between the network standard control blocks and machine dependent parameter blocks. While this detailed study was in progress, high-level modular design of the overall network conversion process was also being done. With high-level modules designed in a portable manner, only the logic for the lowest level command modules

required machine dependence.

Use of these standard network control blocks can be divided into two levels;

1. A low-level use for transformation to and from machine dependent parameter blocks so current programs not running on a network can be run without changes. This can be done by having all file operations which are trapped by the local operating system inspected by network software and handled internally if network action is required (See Figure 1).
2. A higher-level use, whereby a network control language is designed which uses the network standard directly and thus requires only transformations from the network standard to the destination machine. No machine to network conversions would be necessary except for status returns. This report is limited to providing only the lowest level type transformations; however, the network standard control blocks to be used by the higher level are developed.

1.4 REPORT STRUCTURE

This report is structured in the following top-down manner. We present the proposed network standard control block structures and define in detail all of their parameters. We then demonstrate their use through sample mappings between specific machines and the network standard control blocks. Using these transformations as a basic foundation, there is a discussion of overall network flow, describing where and

how each conversion is used. A hierarchical modular structure is developed to portray the necessary machine independent high-level modules and the machine dependent lowest level command submodules. English HIPO charts (Hierarchical Input, Processing, Outputs) are used to document each module. PASCAL type algorithms are used in the lowest level module HIPOs to explicitly define the conversion steps necessary. Example commands are then used to hand simulate the applicable algorithms for inter-machine mappings showing network flow and conceptual correctness. Lastly, there is a summarization of what has been accomplished thus far, a discussion of overall problems, suggested areas requiring future study, and possible approaches to any continued research.

The attached appendices contain the following information: Appendix A contains general characteristics of each machine, along with the actual transformations between each command and the network standard control block. Appendix B contains the machine dependent parameter blocks with the descriptions and settings for each parameter. Appendix C further defines each machines parameter block by command, showing what specific parameters are used for each command. Appendix D is a mapping between machine dependent file structures. Appendix E defines the lowest level command modules in HIPO form for the individual commands on each machine.

2. PROPOSED NETWORK STANDARD CONTROL BLOCKS

This section contains network standard control blocks to help standardize file management services and I/O capabilities. A general overview of concepts involved with designing the control blocks is presented to familiarize the reader with some of the design considerations. We then portray and briefly define each resulting network control block in Figures 2 and 3. Detailed descriptions of all associated parameters are given in Tables 1 and 2. Lastly, the transformations in appendix A are used as examples to show exactly how to interpret them.

2.1 CONCEPTUAL CONSIDERATIONS IN CONTROL BLOCK DESIGN

Our initial project definition called for a network control block to contain the intersection of compatible capabilities between machines in the network. However, a detailed study revealed very few straight forward compatibilities. The idea of emulating certain basic capabilities was discussed but later dismissed because of its restrictive nature. Rather than designing a standard control block with limited emulation capabilities, it was decided to design a general control block containing the parameters required for a union of all network machine capabilities. This would then provide for complete sharing and utilization between homogeneous machines in the network, while at the same time isolating distinct differences between architectures. Later, emulation could then provide additional capabilities on a needed basis for specific commands and machine types. In

addition to this major change in design philosophy, other information such as a machine type indicator and machine dependent status are retained in the control block to allow for more complete error analysis, possible recovery action, and increased processing speed.

2.2 NETWORK CONTROL BLOCK DESCRIPTIONS

As a result of extensive research, integration of similar parameters, and the conceptual philosophy used, the network standard control blocks described in Figures 2 and 3 were designed. They provide for a union of those capabilities presently existing on the three mini-computers studied. General mnemonic descriptions accompany each Figure, and detailed descriptions are given in Tables 1 and 2.

NFMB	NFMB-CODE (1)	NFMB-MD____ (1)			NFMB-STNT (1)
		AP 3	BF 2	FT 3	
	NFMB-STAT (1)	NFMB-LUNM (1)			NFMB-PKEY (2)
	NFMB-SZFL (1)	NFMB-SZNX (1)			NFMB-LRCL (2)
	NFMB-MTPS (1)	NFMB-MTPD (1)			NFMB-FDDV (3)
	NFMB-FDVL (4)	NFMB-FDDR (20)			NFMB-FDFN (10)
	NFMB-FDEX (3)	NFMB-FDVR (1)			

FIGURE 2:

Network Standard File Management Control Block (NFMB)

Mnemonics

NFMB-CODE Specifies desired File Management command.
 NFMB-MD____ Modifications for access, buffering, and file type.
 NFMB-STNT Network standard status response.
 NFMB-STAT Destination machine dependent status obtained.
 NFMB-LUNM Logical unit number assigned to file/device.
 NFMB-PKEY Read/Write protection keys.
 NFMB-SZFL File size parameter in bytes.
 NFMB-SZNX Index block size in bytes.
 NFMB-LRCL Logical record length in bytes.
 NFMB-MTPS Machine type of source machine.
 NFMB-MTPD Machine type of destination machine.
 NFMB-FDDV Device portion of file descriptor.
 NFMB-FDVL Volume portion of file descriptor.
 NFMB-FDDR Directory portion of file descriptor.
 NFMB-FDFN Filename portion of file descriptor.
 NFMB-FDEX Extension indicating content of file.
 NFMB-FDVR Version number of file described.

Table 1 (Page 1 of 2)

NFMB DETAILED MNEMONIC DESCRIPTIONS

NFMB-CODE	File management service code, 1--CREATE 2--OPEN/ASSIGN 4--CLOSE 8--DELETE
NFMB-MD	The modification argument, <u>AP</u> --Access Privileges, 0--Shared RD/WR 1--Shared RD, Exclusive WR 2--Shared RD only 3--Shared WR only 4--Exclusive RD/WR 5--Exclusive RD, Shared WR 6--Exclusive RD only 7--Exclusive WR only <u>BF</u> --Buffering Provided, 0--System supplied 1--User supplied <u>FT</u> --File Type, 0--Contiguous 1--Chained (sequential) 2--Indexed (random)
NFMB-STNT	The network standard status response, 0--Successful completion of request. 1--Illegal function/filename. 2--Illegal logical unit/channel. 3--Volume error. 4--File non-existent on volume. 5--File already exists. 6--Protection error. 7--Access conflict--request denied. 8--File descriptor error. 9--Logical unit number assignment error. 10--Size error. 11--Device error. 12--Other--Not a common network error. 13--NO-MAPPING condition at destination machine.
NFMB-STAT	Actual destination machine dependent error status.
NFMB-LUNM	Logical unit number associated with file/device.
NFMB-PKEY	Protection write and read keys used to protect access to the files, 00--General access allowed XX--Requires exact match
NFMB-SZFL	Size parameter specified in bytes. Contiguous file--number of bytes/file. Chained file-----number of bytes/block. Indexed file-----number of bytes/data block.

Table 1 (Page 2 of 2)

NFMB-SZNX	Index block size---number of bytes/index block.
NFMB-LRCL	The number of bytes/logical record, or certain machine dependent information.
NFMB-MTPS	Machine type of source machine. 1---Interdata 8/32 2---Interdata 7/16 3---Nova 4-256---Other architectures
NFMB-MTPD	Machine type of destination machine with the same code values as NFMB-MTPS.
NFMB-FDDV	The device name portion of the file descriptor.
NFMB-FDVL	The volume on which the desired file resides.
NFMB-FDDR	The directory for the desired file.
NFMB-FDFN	The filename string of the file descriptor.
NFMB-FDEX	The extension portion of the file descriptor, indicating the content of the file.
NFMB-FDVR	The file version number portion of the file descriptor.

<u>NIOB</u>	NIOB-CODE (1)	NIOB-FUNC (2)	NIOB-STNT (1)
	NIOB-STDD (1)	NIOB-STD I (1)	NIOB-LUNM (1)
	NIOB-SADR (4)	NIOB-EADR (4)	NIOB-RREC (4)
	NIOB-NMBT (4)	NIOB-MTPS (1)	NIOB-MTPD (1)

FIGURE 3:

Network Standard I/O Control Block (NIOB)

Mnemonics

NIOB-CODE	Specifies type of I/O request.
NIOB-FUNC	Describes I/O function and characteristics.
NIOB-STNT	Network standard status response.
NIOB-STDD	Device dependent status obtained at destination machine.
NIOB-STD I	Device independent status from destination machine.
NIOB-LUNM	Logical unit number designating file for I/O.
NIOB-SADR	Starting byte of the data buffer.
NIOB-EADR	Last byte of the data buffer.
NIOB-RREC	Random record number (0 relative).
NIOB-NMBT	Number of bytes in transfer.
NIOB-MTPS	Machine type designator for source machine.
NIOB-MTPD	Machine type designator for destination machine.

Table 2 (Page 1 of 2)

NIOB DETAILED MNEMONIC DESCRIPTIONS

NIOB-CODE	I/O operation code, 0--Data transfer ≠ 0--I/O command (REWIND, etc)
NIOB-FUNC	I/O function code, <u>BIT</u> 0 Function---0--Write 1--Read 1-3 Access Mode---0--Sequential, current pointer 1--Random, use record number 2--Line, ASCII line 3--Free Form, variable size 4--Direct Block, Contiguous BLK 4 Format Type---0--ASCII 1--Binary 5 Formatting ---0--Yes, by device/file, bit 4 1--No 6 I/O Wait---0--Proceed processing 1--Wait for I/O completion 7 I/O Connect---0--Wait for connect 1--Immediate reject if busy
NIOB-STNT	The network standard I/O status. 0--Successful I/O completion. 1--Illegal channel. 2--Illegal request. 3--File not opened yet. 4--Protection conflict. 5--End of medium. 6--End of file. 7--Out of space. 12-Other--Not a common network error. 13-NO-MAPPING condition at destination machine.
NIOB-STDD	Actual device dependent status obtained at the destination machine.
NIOB-STD I	Actual device independent status obtained at the destination machine.
NIOB-LUNM	Logical unit number designating file for I/O.
NIOB-SADR	Starting byte of the data buffer where data is to be put or taken from.
NIOB-EADR	Ending byte of the data buffer whose address is in NIOB-SADR.

Table 2

(Page 2 of 2)

NIOB-RREC	The random record number to be used for random I/O. It is zero relative.
NIOB-NMBT	The number of bytes to be transferred or already transferred.
NIOB-MTPS	The machine type of the source machine. 1---Interdata 8/32 2---Interdata 7/16 3---Nova 4-256---Other architectures
NIOB-MTPD	The machine type of the destination machine with the same types as NIOB-MTPS above.

2.3 TRANSFORMATION TABLE ANALYSIS USING EXAMPLES

In this section we present the mapping between each mini-computer's unique calling parameters and the generalized standard blocks. Appendix A contains these transformation mappings ordered by machine and command. To show exactly how these mappings work, we use the following examples to specify a mapping direction and use the transformations from Appendix A for the Nova OPEN and CLOSE commands, the Interdata 8/32 I/O command, and the Interdata 7/16 CREATE command to describe the mapping with the machine dependent parameter blocks in Appendix B.

EXAMPLE #1--A Nova OPEN command going from machine dependent to network standard.

The transformation table shows an OPEN is represented by NFMB-CODE = 2. The Nova has three separate OPEN commands each describing different access privileges. If .OPEN is used, then it is converted to the network standard by setting NFMB-MDAP = 0 to indicate an access privilege of shared reading and writing. If .EOPEN is used, the network standard for shared reading and exclusive writing is one, so NFMB-MDAP is set to one. For .ROPEN, NFMB-MDAP = 2 indicates shared reading only is requested. Since there are no other access privileges on the Nova, nothing maps into the additional network standard access privileges indicated by values from three to seven. The channel number specified in the OPEN command is placed directly in NFMB-LUNM. The filename and extension, which are pointed to by accumulator zero, are

placed into their perspective locations in the NFMB. The content of accumulator one, which would only be used by another Nova, is placed into NFMB-LRCL in case the destination machine is a Nova. Having completed these assignments to the NFMB, the conversion to network standard is completed for the Nova OPEN command, and the NFMB is ready for shipment to the destination machine.

EXAMPLE #2--An Interdata 8/32 I/O command from machine dependent to network standard.

According to the transformation table, when the Interdata 8/32 function code has bit zero = 0, then NIOB-CODE is set to zero to indicate a data transfer I/O. The other bit settings of the Interdata 8/32 function code are then placed into their appropriate counterparts in the network standard I/O block. Bits one through three of the NIOB-FUNC byte have additional settings into which the Interdata 8/32 cannot be mapped since it does not support line I/O, Free Form I/O, or Direct Block I/O. The logical unit number is placed directly into NIOB-LUNM and the starting and ending buffer locations are also loaded directly into the NIOB. If the I/O is random, then the random record number is placed in NIOB-RREC. In order to conform to other network standard parameter requirements, the number of bytes to be transferred is computed by subtracting the starting buffer address from the ending address and placing the result into NIOB-NMBT. Once these assignments are made, the NIOB is ready for shipment to the destination machine.

EXAMPLE #3--An Interdata 7/16 CREATE command from machine dependent to network standard.

The transformation table shows the Interdata 7/16 CREATE command has a value of 80, which is converted to a network standard code of one for a CREATE (NFMB-CODE = 1). The Interdata 7/16 has two distinct file types as described in Appendix D, contiguous files and indexed files. For contiguous files, MOD-FT = 0, the network standard is NFMB-MDFT = 0. For indexed files, MOD-FT = 2, NFMB-MDFT is set to two. The Interdata 7/16 requires user buffering on contiguous files (MOD-BF = 1) so this converts to NFMB-MDBF = 1. For indexed files, system buffering is provided (MOD-BF = 0), so NFMB-MDBF is set to zero. The protection keys are moved directly to NFMB-PKEY and the file descriptor components (Volume, Filename, and Extension) are stored directly into their NFMB counterparts. Contiguous files on the Interdata are created by specifying the number of 256 byte sectors to be allocated to the file, so the NFMB-SZFL is set by multiplying the number of sectors by 256 bytes to get the total number of bytes to be allocated. For indexed files, another size is specified for the index blocks and this is converted, as above, before storing in NFMB-SZNX. For contiguous files the Interdata 7/16 does not specify the logical record length, it allows for user buffering and variable length records, so NFMB-LRCL is set to a negative one to indicate it is not actively being used for this command. The LRECL is specified for indexed files so it is moved directly into NFMB-LRCL. These assignments show decision

processes are required during conversion since different variations are possible. But once the necessary parameters in the NFMB are set, the request is ready to go to the destination machine.

EXAMPLE #4--A CLOSE request from network standard to Nova machine dependent, and all associated status conversions.

The transformation table shows a network CLOSE is represented by NFMB-CODE = 4. To convert this to a Nova CLOSE, the Nova command .CLOSE channel number is generated and the channel number is set from NFMB-LUNM. Upon completion of the .CLOSE, status is returned by virtue of the return point and accumulator two. The transformation table indicates that a return to the second branch is a successful return, and that successful completion is indicated in the network by setting NFMB-STNT = 0. If return one is taken, then accumulator two contains the unsuccessful error code. This error code is converted and stored into NFMB-STNT. When the Nova receives a network standard status response from the destination machine, the transformation table shows that it is converted into the appropriate status if possible. If the network status is a twelve, or any unrecognizable status, then some Error-Action is required because the error is not recognizable by the Nova software.

3. NETWORK CONVERSIONS AND ALGORITHMS FOR MACHINES

We assume the reader has knowledge of the network standard control blocks and the associated machine transformations in Appendix A. This section contains the development of a high-level top-down modular design with associated algorithmic descriptions to perform these transformations on each machine. Each modular algorithm is defined in an English hierarchical description of inputs, processing required, and outputs (HIPOs). Each HIPO description isolating problem areas is followed by more detailed discussion. The lowest level module HIPO definitions in Appendix E are described with PASCAL-type algorithms and then used for example hand simulations. The examples use inter-machine file management commands requiring network transformations on both the source machine and the destination machine. Hand simulations of the overall flow through the network by using the PASCAL algorithms in the HIPOs for the modules involved indicates the conceptual correctness of the network standard blocks and the transformations.

3.1 NETWORK CONVERSION FLOW OVERVIEW

Figure 4 gives an overview of the flow between the source and destination machines in the network with a brief description of the network functional software involved in handling the network control blocks and associated conversions. Some of the terminology used is defined in Table 3. Each machine has some network interpreter system software such as system service calls, which analyzes certain local operating system traps to determine if the request requires

network action (The Network Trap Handler is shown in Figure 5). If network action is necessary, the machine dependent calling parameters are converted to network standard and sent to the destination machine via network send and receive linkages (R6). At the destination machine, network system software converts the standard block into the particular machine dependent calling parameters and performs the requested action if possible, (The Network Destination Handler is shown in Figure 6). When the action is completed (either successfully or unsuccessfully) the status is converted from machine dependency into network standard and sent back to the source machine via the network send and receive capabilities. The source machine has additional network system software which receives the response and converts the network standard status into machine dependent status. The Network Response Handler is shown in Figure 7. If the status response value is not valid on this machine and thus cannot be returned to the user, some desired Error-Action needs to be built-in to handle the situation. Otherwise, the recognizable status is passed back to the original requester with additional information (i.e. possibly a data buffer).

The English description of the algorithms for each machine researched is in HIPO documentation form, where the Inputs, Processing required, and Outputs are defined for each module depicted in the hierarchical diagrams presented in Figures 5 through 7. Table 4 describes each module's functional responsibilities within the network conversion process. The higher level modules are not machine depen-

dent so their HIPO definitions are in a more general English notation. However, the lower level module HIPOs define machine dependent inputs and outputs and describe the processing required in PASCAL type algorithms.

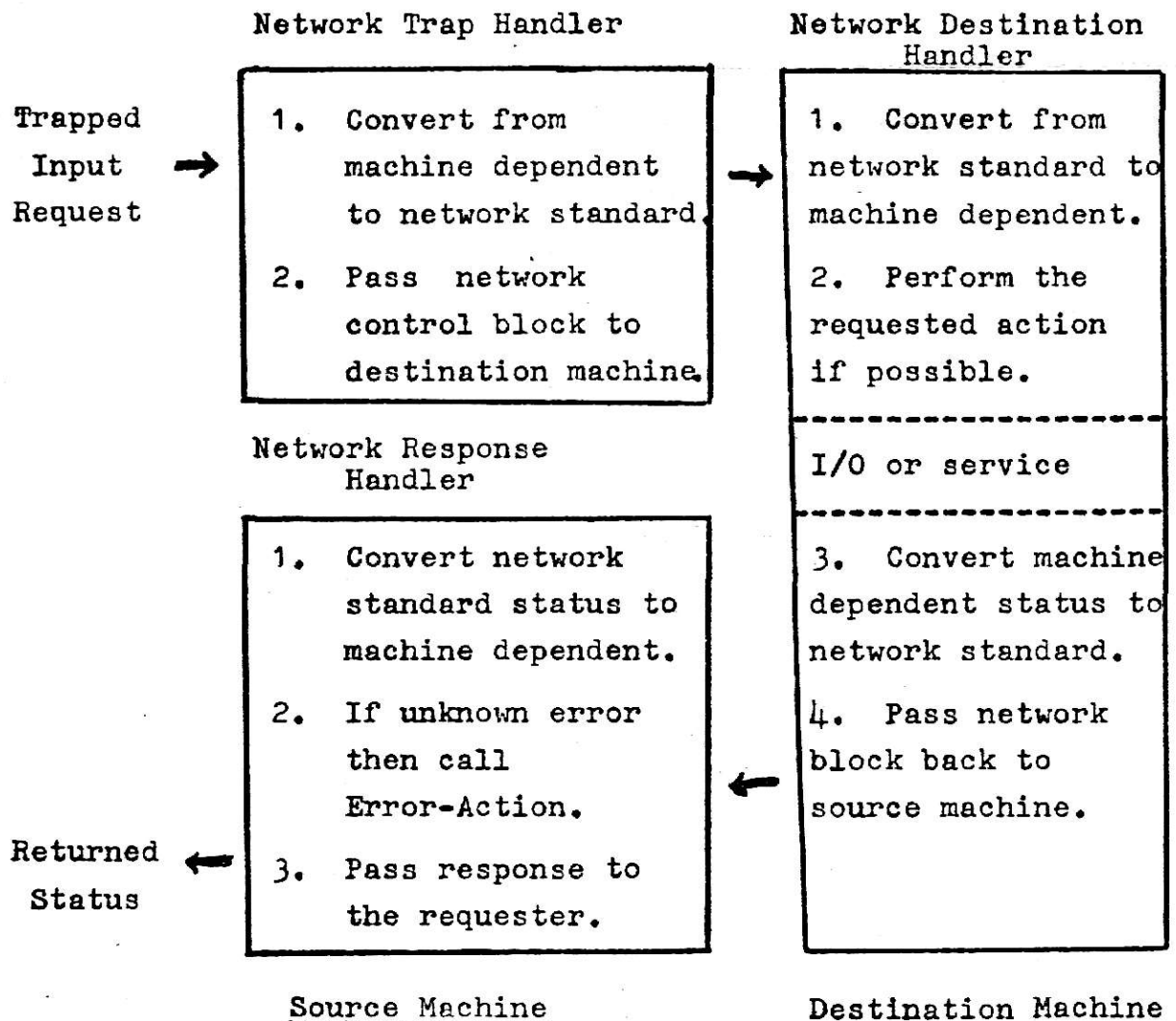
NETWORK SYSTEM SOFTWARE

FIGURE 4: Overview Flow of Network Conversion Process

Table 3

EXPLANATION OF MNEMONICS IN FIGURE 4

NTH	Network Trap Handler---Network system software which determines if trapped commands require network action and converts those that do to network standard control blocks before passing them on to the destination machine.
NDH	Network Destination Handler---Network system software which converts requested action from network standard into machine dependent arguments, executes the request if possible, converts the status from machine dependency to network standard and passes the information back to the source machine.
NRH	Network Response Handler---Network system software which converts the status from network standard into machine dependency and returns the results to the requesting user if it is recognizable. It performs an Error-Action module when the status is not machine dependent.
Source Machine	The machine on which the requested action is initiated.
Destination Machine	The machine on which the requested action is actually executed.
Error-Action	The module called by NRH to contain error handling logic for non-mappable capabilities or non-recognizable error codes. Due to the complexity of these situations and their close relation to implementation, run time characteristics, and actual user requirements, further study at a later date will be necessary.

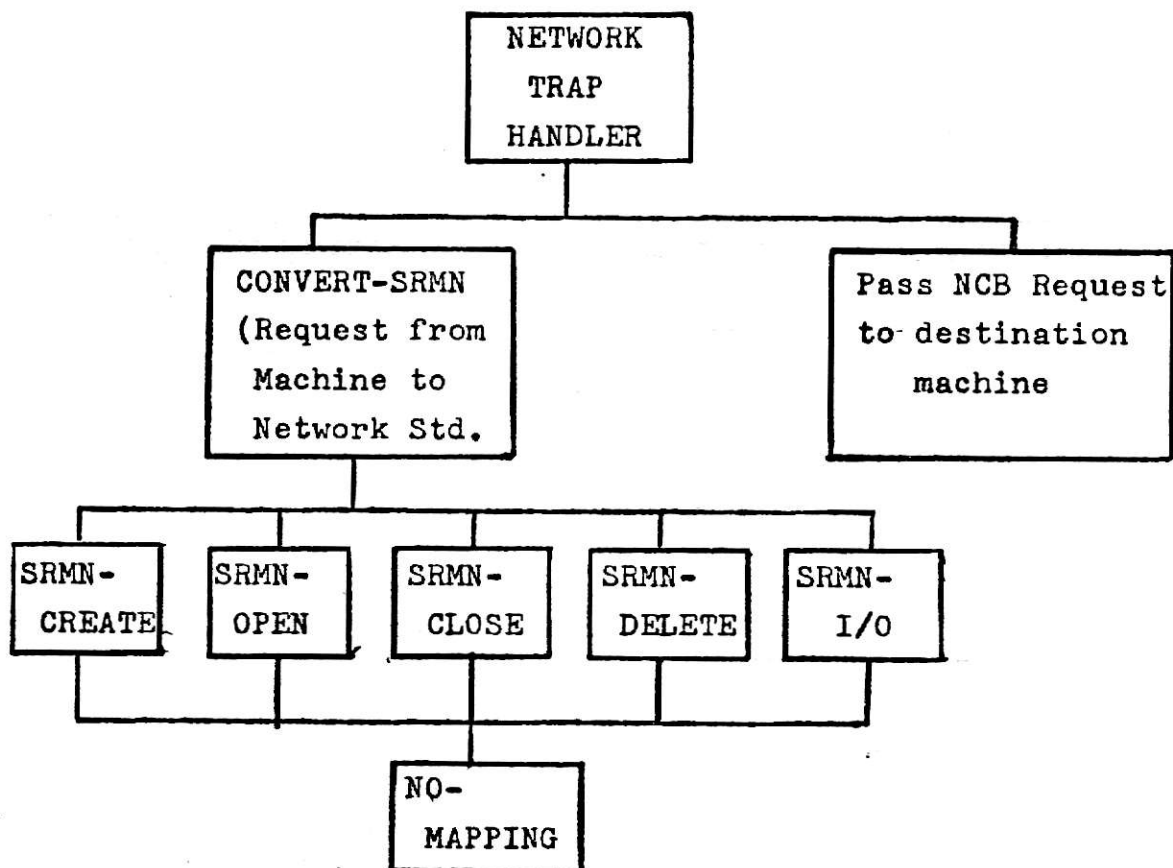


FIGURE 5: Network Trap Handler--Source Machine

Naming Convention: SRMN indicates the Source machine, a user Rquest, from Machine dependent to Network standard.

INPUTS

Machine dependent arguments + a possible data buffer.

PROCESSING REQUIRED

1. Determine if network action is required.
2. If so, convert from machine dependent to network standard by performing CONVERT-SRMN.
3. Pass the network standard request to destination machine.

OUTPUTS

Network standard control block + possible buffer to go to the destination machine.

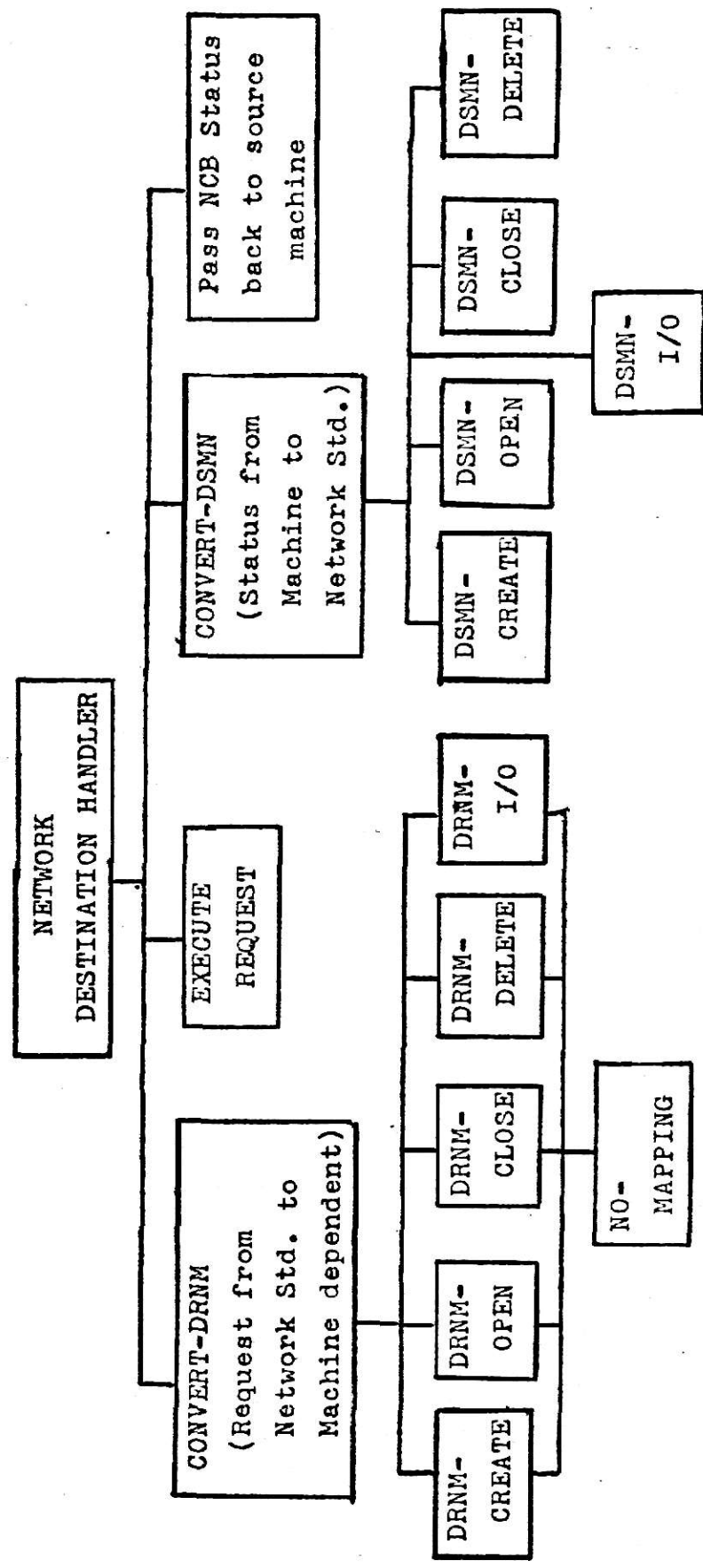


FIGURE 6: Network Destination Handler--Destination Machine

Naming Convention: DRNM indicates the Destination machine, user Request, Network standard to Machine dependent.
 DSMN indicates the Destination machine, Status response, Machine dependent to Network standard.

HIPO for the Network Destination Handler.

INPUTS

The network standard control block request + a possible data buffer from the source machine.

PROCESSING REQUIRED

1. Convert the request from network standard to machine dependent by performing CONVERT-DRNM.
2. If there is a NO-MAPPING condition then go to step 5.

OUTPUTS

Machine dependent arguments ready to be executed.

PROCESSING

3. Perform the requested action.

INPUTS

Machine dependent status arguments returned from execution.

PROCESSING REQUIRED

4. Convert the status from machine dependent to network standard to be passed back to the source machine.
5. Pass the network standard control block with the status back to the source machine. Also any data buffers.

OUTPUTS

A network standard control block with the status response for the requested action by the user on the source machine.

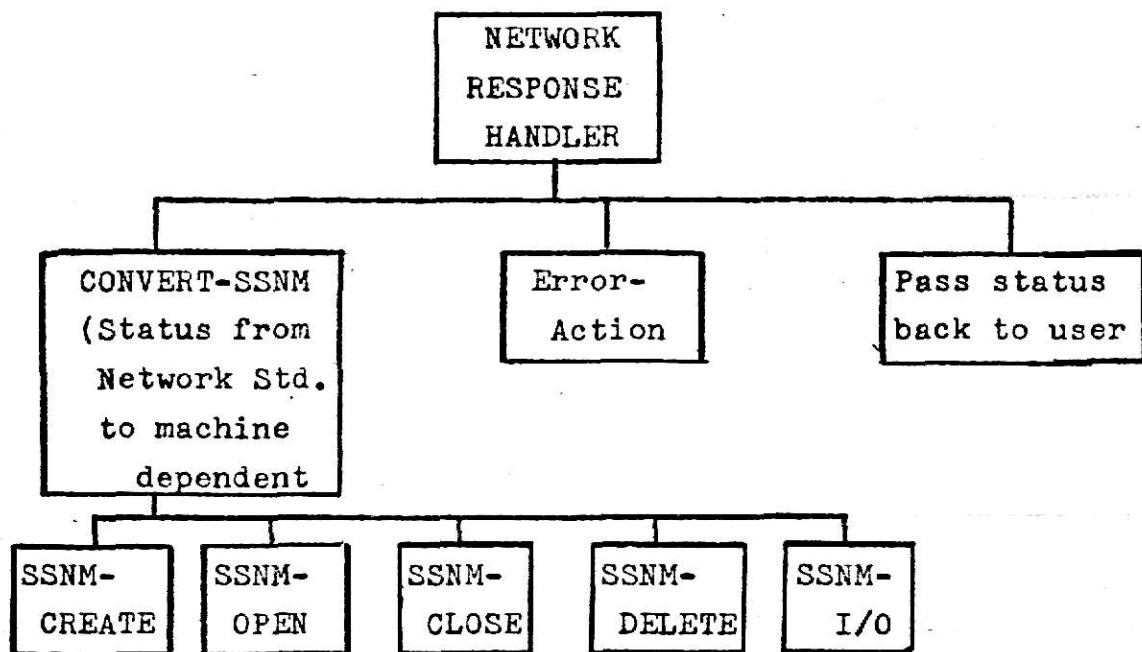


FIGURE 7: Network Response Handler---Source Machine

Naming Convention: SSNM indicates the Source machine, a Status response, from Network standard to Machine dependent.

INPUTS

Network standard control block with status + a possible buffer.

PROCESSING REQUIRED

1. Convert network standard status to machine dependent.
2. If unknown error, then take desired Error-Action.
3. Pass the machine dependent status back to the user.

OUTPUTS

The status + a possible data buffer, or some Error-Action.

FUNCTIONAL RESPONSIBILITIES OF EACH MODULE

- NO-MAPPING The module called by SRMN- and DRNM- modules to handle non-mappable conditions. This could cause status to be set to 13 and passed back so that Error-Action could handle the problem.
- CONVERT-SRMN Module called by the NTH to determine the specific command and convert its arguments from machine dependent to network standard.
- CONVERT-DRNM Module called by the NDH to determine the specific request command and convert its arguments from network standard to machine dependent.
- CONVERT-DSMN Module called by the NDH to determine the specific command and convert the status response from machine dependent to network standard.
- CONVERT-SSNM Module called by the NRH to determine the specific command and convert the network standard status into a machine dependent status if possible.
- SRMN-CREATE Module called by CONVERT-SRMN to convert a machine dependent CREATE command to a network standard CREATE command.
- SRMN-OPEN Module called by CONVERT-SRMN to convert a machine dependent OPEN command to a network standard OPEN command.
- SRMN-CLOSE Module called by CONVERT-SRMN to convert a machine dependent CLOSE command to a network standard CLOSE command.
- SRMN-DELETE Module called by CONVERT-SRMN to convert a machine dependent DELETE command to a network standard DELETE command.

Table 4 Page 2 of 3

SRMN-I/O	Module called by CONVERT-SRMN to convert a machine dependent I/O command to a network standard I/O command.
DRNM-CREATE	Module called by CONVERT-DRNM to convert a network standard CREATE command to a machine dependent CREATE command.
DRNM-OPEN	Module called by CONVERT-DRNM to convert a network standard OPEN command to a machine dependent OPEN command.
DRNM-CLOSE	Module called by CONVERT-DRNM to convert a network standard CLOSE command to a machine dependent CLOSE command.
DRNM-DELETE	Module called by CONVERT-DRNM to convert a network standard DELETE command to a machine dependent DELETE command.
DRNM-I/O	Module called by CONVERT-DRNM to convert a network standard I/O command to a machine dependent I/O command.
DSMN-CREATE	Module called by CONVERT-DSMN to convert a machine dependent CREATE status response to a network standard CREATE status.
DSMN-OPEN	Module called by CONVERT-DSMN to convert a machine dependent OPEN status response to a network standard OPEN status.
DSMN-CLOSE	Module called by CONVERT-DSMN to convert a machine dependent CLOSE status response to a network standard CLOSE status.
DSMN-DELETE	Module called by CONVERT-DSMN to convert a machine dependent DELETE status response to a network standard DELETE status.

DSMN-I/O	Module called by CONVERT-DSMN to convert a machine dependent I/O status response to a network standard I/O status.
SSNM-CREATE	Module called by CONVERT-SSNM to convert a network standard CREATE status response to a machine dependent CREATE status if possible.
SSNM-OPEN	Module called by CONVERT-SSNM to convert a network standard OPEN status response to a machine dependent OPEN status if possible.
SSNM-CLOSE	Module called by CONVERT-SSNM to convert a network standard CLOSE status response to a machine dependent CLOSE status if possible.
SSNM-DELETE	Module called by CONVERT-SSNM to convert a network standard DELETE status response to a machine dependent DELETE status if possible.
SSNM-I/O	Module called by CONVERT-SSNM to convert a network standard I/O status response to a machine dependent I/O status if possible.

3.2 ENGLISH/PASCAL DEFINITIONS OF CONVERSION MODULES

The HIPO definitions of the higher level network handlers accompanied Figures 5 through 7. The HIPO definitions for the CONVERT-XXXX modules will now be described, and their command submodules for each machine will be referred to in Appendix E.

The following HIPO definition is for the conversion module on the source machine which converts the request arguments from machine dependence to network standard within the Network Trap Handler module.

CONVERT-SRMN

INPUTS

Machine dependent arguments from the requesting user.

PROCESSING REQUIRED

1. Determine the exact command.
2. Perform the associated command transformation module.

OUTPUTS

Network standard control block for the users request, ready to be sent to the destination machine.

This module has command submodules which are used to perform the transformations in Appendix A from machine dependence to network standard before shipping to the destination machine. They are ordered by machine and command in Appendix E with all problems noted where applicable.

On the destination machine the conversion module used by the Network Destination Handler to convert the request from network standard to machine dependent for execution is defined by the following HIPO.

CONVERT-DRNM

INPUTS

A network standard control block request from source machine.

PROCESSING REQUIRED

1. Determine the exact command.
2. Perform the associated command transformation module.

OUTPUTS

A machine dependent argument list ready for execution.

The command submodules used to perform the transformations in Appendix A to go from network standard to machine dependency are defined in Appendix E and ordered by machine and command. Incompatible mappings are discussed in detail following each applicable module.

Upon completion of the requested action, the NDH then uses the following conversion module to convert the status response from machine dependent to network standard for shipping back to the source machine. CONVERT-DSMN is defined by the following HIPO.

CONVERT-DSMNINPUTS

Machine dependent arguments from the completed action---
the status and possible data buffer.

PROCESSING REQUIRED

1. Determine the exact command.
2. Perform the associated command transformation module.

OUTPUTS

Network standard control block containing the user's status
response ready to be sent back to the source machine.

The command submodules used to perform the status transformations in Appendix A from machine dependency to network standard are defined in Appendix E and ordered by machine and command. Problem areas are discussed where they are applicable.

Once the source machine receives the response from the destination machine, the NRH passes it to the CONVERT-SSNM module to convert the status from network standard to machine dependent so it can be passed back to the requesting user. The following HIPO defines CONVERT-SSNM.

CONVERT-SSNMINPUTS

A network standard control block containing the status response from the destination machine.

PROCESSING REQUIRED

1. Determine the exact command.
2. Perform the associated command transformation module.

OUTPUTS

A machine dependent status response ready to go to the user, or an indication that Error-Action is required.

The command submodules used to perform the status transformations in Appendix A from network standard to machine dependency are defined and ordered by machine and command in Appendix E.

3.3 EXAMPLES OF COMPLETE NETWORK CONVERSION FLOW

Now that all the modules have been defined, this section contains example hand simulations to show the conceptual correctness of the network conversion process, transformation tables, and network control blocks.

EXAMPLE #1--A network CLOSE request initiated on an Interdata 8/32 for a file on a Nova. The input parameters are CMD = 4, and LU = 10. Figure 8 pictorially represents steps 1-16 and Figure 9 represents steps 12'-16'.

1. The Interdata 8/32 is the source machine. When the command begins execution the NTH on the Interdata 8/32 looks at the machine dependent arguments and destination machine to determine if network action is required. Since the destination machine is not the source machine, network action is required and the CONVERT-SRMN module is performed.
2. The CONVERT-SRMN module uses the machine dependent arguments to determine which command submodule is to be called.
3. Since CMD = 4, the SRMN-CLOSE module is called and performs the following assignments. NFMB-STNT ← -1, NFMB-CODE ← 4, and NFMB-LUNM ← 10.
4. The NFMB is ready for shipment to the destination machine and is passed on to it by the NTH.
5. At the destination machine (a Nova) the NDH receives an input NFMB containing NFMB-CODE = 4, NFMB-STNT = -1, NFMB-LUNM = 10, and NFMB-MTPS = 1.
6. This information is passed to CONVERT-DRNM where the code is determined to be a CLOSE,,so DRNM-CLOSE is called.

7. The Nova DRNM-CLOSE module sets CMD to .CLOSE and the channel number to 10 (the value in NFMB-LUNM). The output ready to be executed is .CLOSE 10 as required for a Nova CLOSE.
8. The NDH then executes the request and upon completion the machine dependent arguments are an instruction counter (IC) to Return-1 or Return-2, and accumulator two containing an error code if the IC is set to Return-1.
9. The NDH then performs CONVERT-DSMN to convert the machine dependent status to network standard.
10. CONVERT-DSMN determines from the CMD = .CLOSE to perform the DSMN-CLOSE submodule for the status conversion.
11. DSMN-CLOSE uses the machine dependent status arguments to perform the necessary conversion to network standard.
12. Assuming the IC is set for Return-2 (successful), then NFMBV-STNT ← 0 and the module returns control back up to the NDH where the NFMB with NFMB-STNT = 0 is passed back to the Interdata 8/32 source machine. If IC is not set for Return-2, then use alternate 12'.
13. The NRH then accepts the NFMB and calls CONVERT-SSNM to convert the status from network standard to machine dependent.
14. CONVERT-SSNM sees that NFMB-CODE = 4 and performs SSNM-CLOSE to convert the CLOSE status as desired.
15. SSNM-CLOSE on the Interdata 8/32 first checks to see if the two machines are the same type. Since one is a Nova and the other an Interdata 8/32, the next check is to map the NFMB-STNT into the Interdata 8/32 dependent STAT argument.
16. STAT ← 0 and control is passed back up to the NRH which then passes the response back to the user who initially requested the CLOSE in the first place.

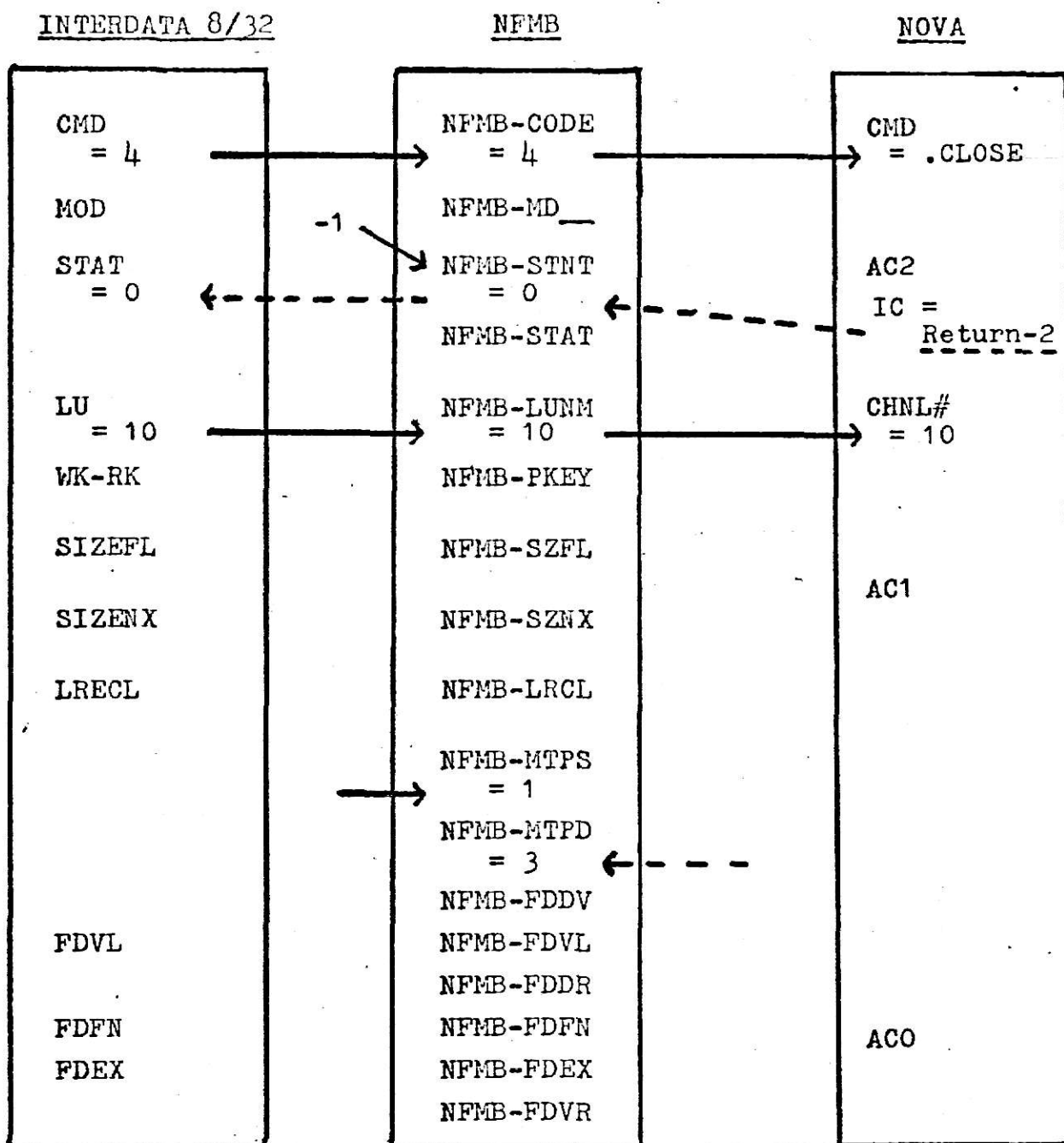


FIGURE 8:

Network Conversion of Successful FMS CLOSE

→ Indicates SRMN and DRNM conversions.

←--- Indicates DSMN and SSNM conversions.

- 12'. The alternate path (Figure 9) has IC = Return-1 and accumulator two contains an error code of 101 for a ten second time out. DSMN-CLOSE first checks to see if IC = Return-2. Since it does not, NFMB-STAT ← 101 and AC2 is compared with the two possible errors which map into network standard codes. Since AC2 does not contain either of these convertible values, NFMB-STNT is set to a twelve to indicate the error was non-network standard.
- 13'. The NFMB that is now passed back up to the NDH to be passed back to the source machine contains NFMB-CODE = 4, NFMB-LUNM = 10, NFMB-STNT = 12, NFMB-STAT = 101, and (NFMB-MTPS = 1, NFMB-MTPD = 3) which are always set based on the machine types of the source and destination machines.
- 14'. The NDH then passes this NFMB back to the Interdata 8/32 where the NRH receives it and passes it to CONVERT-SSNM.
- 15'. The CONVERT-SSNM module sees that NFMB-CODE = 4 and therefore performs SSNM-CLOSE to convert the network CLOSE status into machine dependency.
- 16'. SSNM-CLOSE first checks to see if the two machines have the same type. If so, then the NFMB-STAT actual status could be returned to the user. But if not, the next check is to map the NFMB-STNT into the Interdata 8/32 dependent STAT argument. Since NFMB-STNT = 12, it is mapped into the Error-Action module because the Interdata 8/32 cannot handle the original 101 status return occurring on the Nova. The action taken by the Error-Action module is undetermined at this time and requires further study in the future.

As a result of an unrecognizable error status, the parameter values and action taken at step 16' were quite different from step 16. Notice the semantic action differences and additional alterations for homogeneous types.

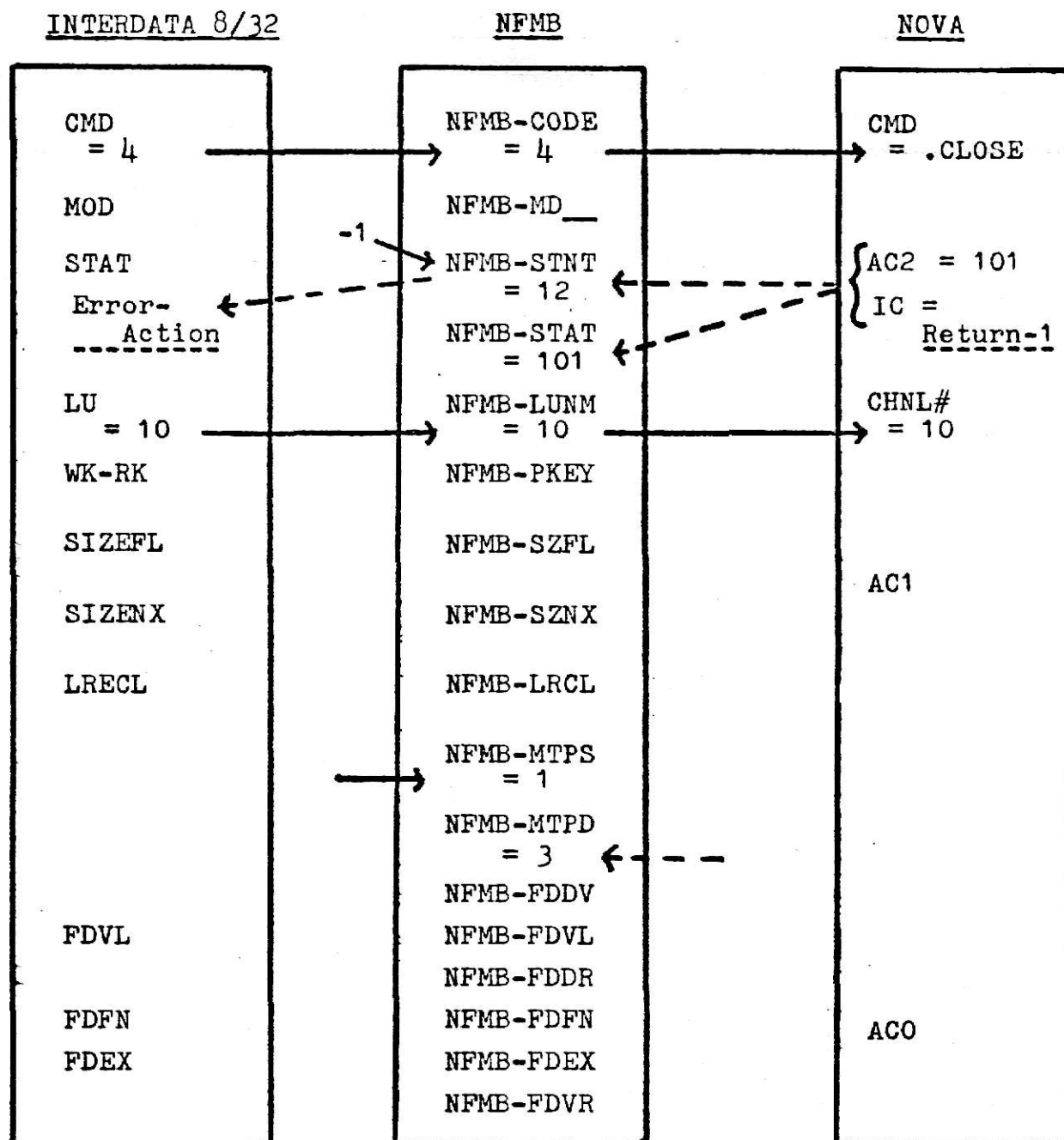


FIGURE 9:

Network Conversion of Bad FMS Error Code

→ Indicates SRMN and DRNM conversions.

←--- Indicates DSMN and SSNM conversions.

EXAMPLE #2--An Interdata 7/16 random write I/O request to be executed on the Nova. The input parameters are FC = 00111101, LU = 5, BUFSADR = 105, BUFEADR = 115, and RNRCDNUM = 3. Figure 10 pictorially represents the following conversion process.

1. The Interdata 7/16 is the source machine. When the command begins execution the NTH on the Interdata 7/16 looks at the machine dependent arguments and destination machine to determine if network action is required. Since the destination machine is not the source machine, network action is necessary and the CONVERT-SRMN module is performed.
2. The CONVERT-SRMN module uses the machine dependent service call type to determine which command submodule is to be called.
3. Since the request is an I/O, the SRMN-I/O module is called and performs the following assignments. NIOB-STNT \leftarrow -1, NIOB-CODE \leftarrow 0, NIOB-FUNC(0) \leftarrow 0, NIOB-FUNC(4) \leftarrow 1, NIOB-FUNC(6) \leftarrow 1, NIOB-FUNC(7) \leftarrow 0, NIOB-FUNC(1-3) \leftarrow 001, NIOB-FUNC(5) \leftarrow 1, NIOB-LUNM \leftarrow 5, NIOB-SADR \leftarrow 105, NIOB-EADR \leftarrow 115, NIOB-RREC \leftarrow 3, and NIOB-NMBT \leftarrow 115 - 105 = 10.
4. The NIOB is ready for shipment to the destination machine along with the associated data buffer. It is passed on to the destination machine by the NTH.
5. At the destination machine (a Nova) the NDH receives an input NIOB containing NIOB-CODE = 0, NIOB-LUNM = 5, NIOB-FUNC(0-7) = 00011110, NIOB-SADR = 105, NIOB-EADR = 115, NIOB-RREC = 3, NIOB-NMBT = 10, and NIOB-MTPS = 2.
6. This information is passed to CONVERT-DRNM where it is determined to be an I/O block, so DRNM-I/O is performed.

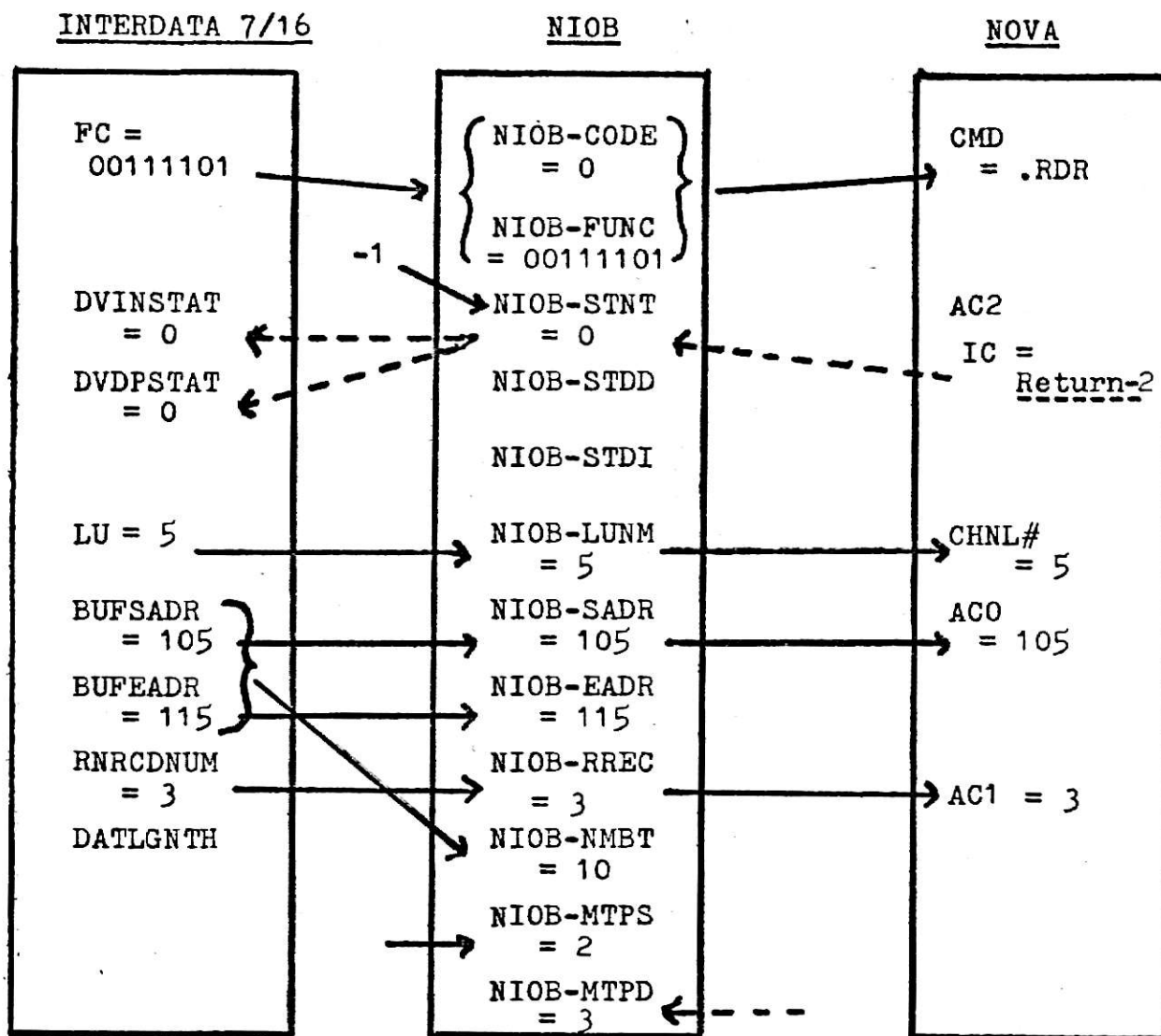


FIGURE 10:

Network Conversion of Random I/O Request

→ Indicates SRMN and DRNM conversions.

←--- Indicates DSMN and SSNM conversions.

7. The Nova DRNM-I/O module checks to see if bits 4-7 of NIOB-FUNC = 1110 to insure to Nova I/O assumptions are set correctly (actually the I/O module may do the I/O assuming the assumptions are true and not bother checking). It then compares bits 0-3 of NIOB-FUNC to determine what CMD should be set to. Since the value is 0001, CMD is set to .RDR for a random read. CHNL# is loaded with a 5 from NIOB-LUNM, accumulator zero is loaded with the buffer starting address (105), and since CMD = .RDR, the NIOB-NMBT value is checked to insure it is less than the assumed 128 bytes. Since only 10 bytes are to be transferred, accumulator one is loaded with the random record number stored in NIOB-RREC and the Nova is ready to execute the I/O requested from the Interdata 7/16. This example shows why the buffering, logical unit number, buffer address, and other information may not be needed or used at the destination machine.
8. The NDH then executes the request and upon completion the machine dependent arguments are an instruction counter (IC) to Return-1 or Return-2, and accumulator two containing an error code if the IC is set to Return-1.
9. The NDH then performs CONVERT-DSMN to convert the machine dependent status to network standard.
10. CONVERT-DEMNI determines DSMN-I/O is to be performed to convert the status response since it was an I/O request.
11. DSMN-I/O uses the machine dependent status arguments to perform the necessary conversion to network standard.
12. Assuming the IC is set for Return-2 (successful), then NIOB-STNT ← 0 and the module returns control back up to the NDH where the NIOB with NIOB-STNT = 0 is passed back to the Interdata 7/16 source machine. If IC is not set for Return-2, then an alternate course of action is taken almost identical to 12'-16' of example #1. The only difference is this uses the I/O conversion modules.

13. The NRH then accepts the NIOB and calls CONVERT-SSNM to convert the status from network standard to machine dependent.
14. CONVERT-SSNM sees that it is an I/O status response and performs SSNM-I/O to convert the status to machine dependence if possible.
15. SSNM-I/O on the Interdata 7/16 first checks to see if the two machines are the same type. Since one is a Nova and the other an Interdata 7/16, the next check is to see if it was a successful I/O. Since NIOB-STNT = 0 (successful), the device dependent and device independent status parameters are set to zero, as expected by the Interdata 7/16 on successful I/O completions.
16. With the Interdata 7/16 status set, the control is passed back up to the NRH which then passes the response back to the user who initially requested the I/O in the first place. If an unrecognizable error is returned, then some recovery is required so an Error-Action module is performed.

4. DISCUSSION/CONCLUSIONS

4.1 SUMMATION OF ACCOMPLISHMENTS

In this project, a major task evolved around defining the desired goal, and limiting the scope of possibilities. Initially, disk file I/O was to be standardized on the three mini-computers considered and the IBM-370(R5). However the complexities of a high-level file manager such as the one on the IBM-370 diversified the effort too much. Machine dependent file structures also became important factors along with their normal file management services. The scope of the problem was restricted in terms of architectures (mini-computers with real-time operating systems) and made more extensive in terms of I/O (now file management services were included).

Another major factor affecting design efforts involved an initial false dependence on actual network design communications and processing. It was only after considering the integration and standardization of file management and I/O command parameters external to other network design that actual progress became evident. Initial time spent in network design was useful and informative, but not mandatory for proposed achievements. Other involvements detracted and often confused design efforts towards attempted goals.

One of the major accomplishments resulting from this study was the decision to have the network standard be a union of network machine dependent capabilities rather than just an intersection of compatible capabilities. Very few capabilities are directly mappable as there are too many

internally assumed differences which require some kind of emulation before being standardized.

Having standardization incorporate the union of capabilities allows for greater resource and capability utilization throughout a network which has some homogeneous machines. Like machines can use their entire repertoire of file management and I/O commands. By developing a unionized standard, and noting heterogeneous mapping incompatibilities, the job of emulating additional capabilities between two machines becomes much easier. The exact incompatibilities are defined, the network standard is defined, and the desired capabilities are defined on the machines where they already exist. Any emulation can be done on an individual basis in a very modular way.

The initial ground work has been laid by using the Interdata 7/16, 8/32, and Nova computers. Detailed descriptions and documentation have been developed for rapid implementation when desired. Problem areas and machine dependent decisions which need to be made at implementation time are discussed where they occur. Overall modularity, straightforward design, and documentation techniques have been used to increase understandability, conceptual clarity, reliability, and implementability.

4.2 PROBLEM AREAS

When comparing the Nova and the Interdata 8/32 and 7/16, the Interdata machines file management and I/o capabilities are almost identical. Examples of a few differences are: work size, error codes, and the absence of chained files in the 7/16. But the major differences are between the Interdatas and the Nova. Areas such as protection, buffering techniques, when record size information is retained for a file, when it is specified at request time, access mode variations, fixed versus variable record sizes, and file descriptor information require the most attention. Most of these problems are discussed in the report where they are isolated, however some still unanswered questions need further explanation. The idea that all requests will be performed on the destination machine by network system software raises questions about how buffering will be done, whether logical unit numbers are necessary at the destination machine, and how will file assignment and access privileges be managed. Concurrency of I/O also needs to be taken into account. The I/O wait request information may not be used within the network on the destination machine. If the network is to assume some of these responsibilities, then the conversions on the destination machine may not be necessary for logical unit number or channel number, buffering technique, and I/O wait. If tables are kept by the system software to identify by whom and how files are being accessed, then possible emulation techniques will have to take this

into consideration.

This report is mainly concerned with providing the lowest level of interface between the network and existing programs not aware of the network. During local operating system trap operation, the request is examined and network action is taken when required. Since most application software uses higher level job control languages to create and delete files, the main concern for internal network conversion unknown to the users is on file I/O. Based on the completion of this study, it seems reasonable to minimize the use of file management conversions at request time by requiring the programmers to use a higher level language and specify network files in a network form. This would resolve many of the difficult file management mapping problems and require much less conversion overhead.

4.3 FUTURE RESEARCH AND APPROACHES

In addition to the mini-computers considered in this report, time was spent studying other mini-computers such as the PDP-11 (R2). Although lacking enough documentation to incorporate them in this study, certain concepts and knowledge were gained.

1. Not all mini-computers have a similar level of file management capabilities. The PDP-11, for example, is sophisticated, much like the IBM-370, and thus requires more detailed investigation of its full capabilities.
2. Although certain general information appears to be identical to standard concepts, there are almost always unique options to extend or change certain meanings.
3. As more mini-computer capabilities are added to the network standard, these concepts become better defined. Commonalities or norms start to appear and advantages and disadvantages become more evident. Comparisons between machines can be more detailed and possibly serve as a tool for standardization within the industry of certain basic capabilities and calling parameters.

In addition to these areas of concern, another area ignored in this report is the Error-Action module and exactly what it does. Again certain machine dependencies are involved, but certain network procedures might be necessary.

An individual's approach to these varied areas of extended research can have a tremendous impact on success and time involved. In order to incorporate additional machine architectures into the network, the following steps

should be a helpful guidance.

1. Make a detailed analysis of a machine's file structures, file management capabilities, I/O commands, and all of the necessary parameters associated with them.
2. Map compatible characteristics and capabilities into the network standard.
3. Incorporate the incompatible capabilities into the network standard to insure it is a union of heterogeneous machine capabilities.
4. Define in detail extensions to the network standard control blocks and their descriptions.

As more and more machines are added to the network, fewer and fewer additions should be necessary to the network standard control blocks. As additional network capabilities are desired on a given machine, the transformation tables and detailed descriptions should make modular implementation of emulation techniques straight forward.

REFERENCES

- R1 DATA GENERAL CORPORATION; RDOS Real Time Disk Operating System User's Manual, PUB. #093-000075-04, 1973.
- R2 DIGITAL EQUIPMENT CORPORATION; IAS/R SX-11 I/O Operations Reference Manual, PUB. #11-010RA-A-D, Maynard, Massachusetts, 1975.
- R3 INTERDATA INCORPORATED; OS/16 MT2 Programmer's Reference Manual, PUB. #29-429, Oceanport, New Jersey, 1975.
- R4 INTERDATA INCORPORATED; OS/32-MT Program Reference Manual, PUB. #29-390R04, Oceanport, New Jersey, 1976.
- R5 INTERNATIONAL BUSINESS MACHINES CORPORATION; IBM System 360 OS Data Management Macro Instructions, Form No. GC26-3794-1, San Jose, California, 1973.
- R6 KANSAS STATE UNIVERSITY; Progress Report on Functionally Distributed Computer Systems, Development: Software and Systems Structure, Army Research Office, Grant No. DAAD-29-76-G-D108, May 1976.

APPENDIX A

MACHINE TRANSFORMATIONS

NOVA GENERAL CHARACTERISTICS

1. All calling sequences have the following format.
 - .SYSTEM
 - .CMD
 - Return-1, Error
 - Return-2, Normal
 - A. Return-1 is the error return branch (in which case accumulator two contains the error status code).
 - B. Return-2 is the normal return for successful request.
2. There is no protection on the Nova from accessing files.
 - A. When converting Nova dependent arguments to network standard, NFMB-PKEY is set to zero for general permissions.
 - B. When converting network standard to Nova dependent arguments, NFMB-PKEY is ignored and access allowed.
3. Before creating, opening, or deleting files, the current device and directory needs to be initialized.
 - A. When converting Nova dependent arguments to network standard, the .INIT command trap action requires taking the string pointed to by ACO and storing the device mnemonic in NFMB-FDDV and the directory string in NFMB-FDDR.
 - B. When converting network standard to Nova dependency, the contents of NFMB-FDDV and NFMB-FDDR are pointed to by ACO.
4. Sequential files on the Nova have variable length records.
5. The Nova requires fixed length records in both random and contiguous files.

NOVA INITIALIZE TRANSFORMATIONS

Used for CREATE, DELETE, and OPEN modules.

NOVA PARAMETERSNFMB PARAMETERS

CMD = .INIT	←-----→	NFMB-CODE ≠ 4
ACO = Pointer to Device.Directory	←-----→	NFMB-FDDV
	←-----→	NFMB-FDDR
AC1 = 0 to make current directory	←-----	Assumed
AC1 = 1777 to have full device sweep	←---	Not used

STATUS RETURNS Possible on CREATE, DELETE, and OPEN.

NOVA STATUSNFMB-STNT =

No Error, Return-2	←-----→	0
<u>ERRORS, Return-1, AC2 =</u>		
1---Illegal Filename	←-----→	1
31--Improper unit	←-----→	3
36--Device not in system	←-----→	11
45--Device already initialized	←-----→	0
51--Out of Disk control blocks	-----→	12
52--Bad directory specifier	←-----→	8
53--Directory specifier unknown	←-----→	8
57--Link depth exceeded	-----→	12
74--Address outside space	-----→	12
101-Timed out after 10 seconds	-----→	12
102-No linking allowed	-----→	12
Requires network Error-Action	←-----	12 and all others.

NOVA CREATE TRANSFORMATIONS Page 1 of 2

NFMB-CODE = 1 Used for SRMN-CREATE and DRNM-CREATE.

NOVA PARAMETERSNFMB PARAMETERS

<u>CONTIGUOUS File</u>	←-----→	<u>CONTIGUOUS File</u>
CMD = .CCONT	←-----→	NFMB-MDFT = 0
ACO = Pointer to Filename.Extension	←--→	NFMB-FDFN
	←--→	NFMB-FDEX
AC1 = # of 512 byte Blocks/file	-----→	NFMB-SZFL ←--
		C(AC1) * 512.
AC1 ←-- NFMB-SZFL ÷ 512	←-----	NFMB-SZFL
System Buffered	←-----→	NFMB-MDBF = 0
**NO-MAPPING	←-----	NFMB-MDBF = 1
LRECL assumed 128 bytes	-----→	NFMB-LRCL ←-- 128
**NO-MAPPING	←-----	NFMB-LRCL > 128
Some space wasted	←-----	NFMB-LRCL < 128

<u>SEQUENTIAL File</u>	←-----→	<u>CHAINED File</u>
CMD = .CREAT	←-----→	NFMB-MDFT = 1
ACO = Pointer to Filename.Extension	←--→	NFMB-FDFN
	←--→	NFMB-FDEX
System Buffered	←-----→	NFMB-MDBF = 0
**NO-MAPPING	←-----	NFMB-MDBF = 1
Assumes 512 bytes per block	-----→	NFMB-SZFL ←-- 512
**NO-MAPPING	←-----	NFMB-SZFL > 512
Some space wasted	←-----	NFMB-SZFL < 512
No LRECL assumed or given	-----→	NFMB-LRCL ←-- -1
**NO-MAPPING	←-----	NFMB-LRCL ≠ -1

<u>RANDOM File</u>	←-----→	<u>INDEXED File</u>
CMD = .CRAND	←-----→	NFMB-MDFT = 2
ACO = Pointer to Filename.Extension	←--→	NFMB-FDFN
	←--→	NFMB-FDEX

NOVA CREATE TRANSFORMATIONS Page 2 of 2NOVA PARAMETERSNFMB PARAMETERS

System Buffered	←-----→	NFMB-MDBF = 0
**NO-MAPPING	←-----	NFMB-MDBF = 1
Assumes 512 bytes/block for FL	-----→	NFMB-SZFL <-- 512
**NO-MAPPING	←-----	NFMB-SZFL > 512
Some space wasted	←-----	NFMB-SZFL < 512
Assumes LRECL = 128 bytes	-----→	NFMB-LRCL <-- 128
**NO-MAPPING	←-----	NFMB-LRCL > 128
Some space wasted	←-----	NFMB-LRCL < 128

STATUS RETURNS Used for DSMN-CREATE and SSNM-CREATE.

NOVA STATUSNFMB-STNT =

No Error, Return-2	←-----→	0
<u>ERRORS, Return-1, AC2 =</u>		
1---Illegal Filename	←-----→	1
11--File already exists	←-----→	5
74--Address outside space	-----→	12
101-Time out, 10 seconds	-----→	12
Requires network Error-Action	←-----	12 and all others.

NOVA OPEN TRANSFORMATIONS

NFMB-CODE = 2 Used for SRMN-OPEN and DRNM-OPEN.

NOVA PARAMETERSNFMB PARAMETERS

CMD = .OPEN	←-----→	NFMB-MDAP = 0--Shared RD/WR
CMD = .EOPEN	←-----→	1--Shared RD, Exclusive WR
CMD = .ROPEN	←-----→	2--Shared RD only
**NO-MAPPING	←-----→	3-7
CHNL#	←-----→	NFMB-LUNM
ACO = Pointer to Filename.Extension	←--→	NFMB-FDFN
	←--→	NFMB-FDEX
AC1 = Channel Inhibit mask	-----→	NFMB-LRCL ←-- C(AC1)
AC1 ←-- NFMB-LRCL	←-----→	NFMB-MTPS = NFMB-MTPD

STATUS RETURNS Used for DSMN-OPEN and SSNM-OPEN.

NOVA STATUSNFMB-STNT =

No Error, Return-2	←-----→	0
<u>ERRORS, Return-1, AC2 =</u>		
0---Illegal channel number	←-----→	2
1---Illegal Filename	←-----→	1
12--File does not exist	←-----→	4
21--Channel in use	←-----→	9
31--Wrong unit selected	←-----→	3
53--Directory specifier unknown	←-----→	8
57--Link depth exceeded	←-----→	10
60--File in .EOPEN use	←-----→	7
66--Directory not initialized	←-----→	8
74--Address outside space	-----→	12
101-Timed out after 10 sec.	-----→	12
102-No linking allowed	-----→	12
Requires Network Error-Action	←-----→	12 and all others.

NOVA CLOSE TRANSFORMATIONS

NFMB-CODE = 4 Used for SRMN-CLOSE and DRNM-CLOSE.

NOVA PARAMETERSNFMB PARAMETERS

CMD = .CLOSE CHNL#

←-----→ NFMB-CODE = 4

CHNL#

←-----→ NFMB-LUNM

STATUS RETURNS Used for DSMN-CLOSE and SSNM-CLOSE.

NOVA STATUSNFMB-STNT =

No Error, Return-2

←-----→ 0

ERRORS, Return-1, AC2 =

0---Illegal channel number

←-----→ 2

15--Channel not in use

←-----→ 9

101-Timed out after 10 seconds

-----→ 12

Requires network Error-Action

←----- 12 and all others.

NOVA DELETE TRANSFORMATIONS

NFMB-CODE = 8 Used for SRMN-DELETE and DRNM-DELETE.

NOVA PARAMETERSNFMB PARAMETERS

CMD = .DELET	←-----→	NFMB-CODE = 8
ACO = Pointer to Filename.Extension	←--→	NFMB-FDFN
	←--→	NFMB-FDEX

STATUS RETURNS Used for DSMN-DELETE and SSNM-DELETE.

NOVA STATUSNFMB-STNT =

No Error, Return-1	←-----→	0
--------------------	---------	---

ERRORS, Return-2, AC2 =

1---Illegal Filename	←-----→	1
12--File does not exists	←-----→	4
13--Permanent file, cannot delete	←-----→	6
60--File in use	←-----→	7
74--Address outside space	-----→	12
100-Error in MAP.DR of non-master DV	----→	12
101-Timed out after 10 seconds	-----→	12
102-No linking allowed	-----→	12
Requires network Error-Action	←-----	12 and all others.

NOVA I/O TRANSFORMATIONS Page 1 of 2

NIOB-CODE = 0 Used for SRMN-I/O and DRNM-I/O.

NOVA PARAMETERS

NIOB PARAMETERS

CMD = .WRS CHNL#	←-----→	NIOB-FUNC(0-3)=0000
CMD = .RDS CHNL#	←-----→	NIOB-FUNC(0-3)=1000
Assumes binary data	-----→	NIOB-FUNC(4) ←-- 1
Assumes no format	-----→	NIOB-FUNC(5) ←-- 1
Assumes wait I/O	-----→	NIOB-FUNC(6) ←-- 1
Assumes wait connect	-----→	NIOB-FUNC(7) ←-- 0
**NO-MAPPING	←-----	NIOB-FUNC(4-7)≠1110
CHNL#	←-----→	NIOB-LUNM
ACO = Pointer to first byte in buffer	←--→	NIOB-SADR
No last byte address given	-----→	NIOB-EADR ←-- C(AC1) - C(ACO)
AC1 = # of bytes to RD/WR	←-----→	NIOB-NMBT ≠ -1
**NO-MAPPING	←-----	NIOB-NMBT = -1

CMD = .WRR	←-----→	NIOB-FUNC(0-3)=0001
CMD = .RDR	←-----→	NIOB-FUNC(0-3)=1001
Same set of assumptions as above	-----→	NIOB-FUNC(4-7)=1110
**NO-MAPPING	←-----	NIOB-FUNC(4-7)≠1110
CHNL#	←-----→	NIOB-LUNM
ACO = Pointer to first byte of buffer	←--→	NIOB-SADR
No last byte address given	-----→	NIOB-EADR ←-- 128 + C(ACO)
Assumes fixed length 128 byte records	---→	NIOB-NMBT ←-- 128
**NO-MAPPING	←-----	NIOB-NMBT > 128
AC1 = Random record number	←-----→	NIOB-RREC

NOVA I/O TRANSFORMATIONS Page 2 of 2

STATUS RETURNS Used for DSMN-I/O and SSNM-I/O.

NOVA STATUS

NI0B-STNT =

No Error, Return-2 ←-----→ 0

ERRORS, Return-1, AC2 =

GENERAL

0---Illegal channel number	←-----→	1
3---Illegal command for device	←-----→	2
15--File not yet opened	←-----→	3
34--File direct block I/O only	-----→	12
47--Simultaeneous reads, same QTY line	-→	12
74--Address outside space	-----→	12
101-Timed out after 10 seconds	-----→	12
106-QTY/MCA IN terminated by close	-----→	12

READ ONLY

6---End of file	←-----→	6
7---Read protected file	←-----→	4
26--Access beyond NMAX	←-----→	5
30--File read error	-----→	12
33--Attempted to read into system area	-→	12

WRITE ONLY

10--File write protected	←-----→	4
27--Out of disk space	←-----→	7
103-No complimentary MCA request	-----→	12
104-MCA request incomplete--short	-----→	12

Requires network Error-Action ←----- 12

INTERDATA 8/32 GENERAL CHARACTERISTICS

1. All non-contiguous files assume system buffering.
2. Contiguous files assume user buffering and blocking.
3. Protection keys are used to determine if access privileges should be granted. There is a key for write protection and a key for read protection.
4. When volumes are mounted on devices, the device is mapped to the volume and references made to the volume are automatically mapped into the correct device.
5. Contiguous files on the Interdata have variable length records so the length must be specified at access time.
6. The Interdata requires fixed length records in both sequential (chained) files and random (indexed) files.

INTERDATA 8/32 CREATE TRANSFORMATIONS Page 1 of 2

NFMB-CODE = 1 Used for SRMN-CREATE and DRNM-CREATE.

INTERDATA 8/32 PARAMETERSNFMB PARAMETERS

<u>CONTIGUOUS File</u>	←-----→	<u>CONTIGUOUS File</u>
CMD = 80	←-----→	NFMB-CODE = 1
MOD-FT = 0	←-----→	NFMB-MDFT = 0
MOD-BF = 1	←-----→	NFMB-MDBF = 1
**NO-MAPPING	←-----→	NFMB-MDBF = 0
WK-RK	←-----→	NFMB-PKEY
FDVL	←-----→	NFMB-FDVL
FDFN	←-----→	NFMB-FDFN
FDEX	←-----→	NFMB-FDEX
SIZEFL	-----→	NFMB-SZFL ←-- SIZEFL * 256
SIZEFL ←-- NFMB-SZFL ÷ 256	←-----→	NFMB-SZFL
No LRECL given	-----→	NFMB-LRCL ←-- -1
**NO-MAPPING	←-----→	NFMB-LRCL ‡ -1

<u>CHAINED File</u>	←-----→	<u>CHAINED File</u>
CMD = 80	←-----→	NFMB-CODE = 1
MOD-FT = 1	←-----→	NFMB-MDFT = 1
MOD-BF = 0	←-----→	NFMB-MDBF = 0
**NO-MAPPING	←-----→	NFMB-MDBF = 1
WK-RK	←-----→	NFMB-PKEY
FDVL	←-----→	NFMB-FDVL
FDFN	←-----→	NFMB-FDFN
FDEX	←-----→	NFMB-FDEX
SIZEFL	-----→	NFMB-SZFL ←-- SIZEFL * 256
SIZEFL ←-- NFMB-SZFL ÷ 256	←-----→	NFMB-SZFL

INTERDATA 8/32 CREATE TRANSFORMATIONS Page 2 of 2INTERDATA 8/32 PARAMETERSNFMB PARAMETERS

LRECL	←-----→	NFMB-LRCL ≠ -1
**NO-MAPPING	←-----→	NFMB-LRCL = -1

<u>INDEXED File</u>	←-----→	<u>INDEXED File</u>
CMD = 80	←-----→	NFMB-CODE = 1
MOD-FT = 2	←-----→	NFMB-MDFT = 2
MOD-BF = 0	←-----→	NFMB-MDBF = 0
**NO-MAPPING	←-----→	NFMB-MDBF = 1
WK-RK	←-----→	NFMB-PKEY
FDVL	←-----→	NFMB-FDVL
FDFN	←-----→	NFMB-FDFN
FDEX	←-----→	NFMB-FDEX
SIZEFL	-----→	NFMB-SZFL ←-- SIZEFL * 256
SIZEFL ←-- NFMB-SZFL ÷ 256	←-----→	NFMB-SZFL
SIZENX	-----→	NFMB-SZNX ←-- SIZENX * 256
SIZENX ←-- NFMB-SZNX ÷ 256	←-----→	NFMB-SZNX
LRECL	←-----→	NFMB-LRCL

STATUS RETURNS Used for DSMN-CREATE and SSNM-CREATE.

INTERDATA 8/32 STAT =NFMB-STNT =

0---Successful	←-----→	0
1---Illegal function	←-----→	1
3---Volume not mounted	←-----→	3
4---File already exists	←-----→	5
5---Size error	←-----→	10
10--Type error	←-----→	11
11--File descriptor error	←-----→	8
13--Size error	←-----→	10
Requires network Error-Action	←-----→	12 and all others.

INTERDATA 8/32 OPEN TRANSFORMATIONS

NFMB-CODE = 2 Used for SRMN-OPEN and DRNM-OPEN.

INTERDATA 8/32 PARAMETERSNFMB PARAMETERS

CMD = 40	←-----→	NFMB-CODE = 2
MOD-AP =	←-----→	NFMB-MDAP =
0--Shared Read/Write	←-----→	0--SRW
1--Shared Read, Exclusive Write	←-----→	1--SREW
2--Shared Read Only	←-----→	2--SRO
3--Shared Write Only	←-----→	3--SWO
4--Exclusive Read/Write	←-----→	4--ERW
5--Exclusive Read, Shared Write	←-----→	5--ERSW
6--Exclusive Read Only	←-----→	6--ERO
7--Exclusive Write Only	←-----→	7--EWO
WK-RK	←-----→	NFMB-PKEY
FDVL	←-----→	NFMB-FDVL
FDFN	←-----→	NFMB-FDFN
FDEX	←-----→	NFMB-FDEX
LU	←-----→	NFMB-LUNM

STATUS RETURNS Used for DSMN-OPEN and SSNM-OPEN.

INTERDATA 8/32 STAT =NFMB-STNT =

0---Successful	←-----→	0
1---Illegal function	←-----→	1
2---Illegal logical unit	←-----→	2
3---Volume error	←-----→	3
4---File does not exist	←-----→	4
5---Size error	←-----→	10
6---Protection error	←-----→	6
7---Privilege access error	←-----→	7
8---Buffer error--No room	←-----→	10
9---Logical unit assign error	←-----→	9
10--Type error	←-----→	11
12--Attempt to assign bad device	←-----→	12
128-255--I/O errors	←-----→	12
Requires network Error-Action	←-----→	12 and all others.

INTERDATA 8/32 CLOSE TRANSFORMATIONS

NFMB-CODE = 4 Used for SRMN-CLOSE and DRNM-CLOSE.

INTERDATA 8/32 PARAMETERSNFMB PARAMETERS

CMD = 04

←-----→ NFMB-CODE = 4

LU

←-----→ NFMB-LUNM

STATUS RETURNS Used for DSMN-CLOSE and SSNM-CLOSE.

INTERDATA 8/32 STAT =NFMB-STNT =

0---Successful

←-----→ 0

2---Illegal logical unit

←-----→ 2

9---Logical unit assign error

←-----→ 9

129-192--I/O errors

-----→ 12

Requires network Error-Action

←----- 12 and all others.

INTERDATA 8/32 DELETE TRANSFORMATIONS

NFMB-CODE = 8 Used for SRMN-DELETE and DRNM-DELETE.

INTERDATA 8/32 PARAMETERSNFMB PARAMETERS

CMD = 02	←-----→	NFMB-CODE = 8
WK-RK	←-----→	NFMB-PKEY
FDVL	←-----→	NFMB-FDVL
FDFN	←-----→	NFMB-FDFN
FDEX	←-----→	NFMB-FDEX

STATUS RETURNS Used for DSMN-DELETE and SSNM-DELETE.

INTERDATA 8/32 STAT =NFMB-STNT =

0---Successful	←-----→	0
3---Volume error	←-----→	3
4---Name error	←-----→	4
6---Protect error, bad keys	←-----→	6
7---Privilege error, write protect	←-----→	6
11--File descriptor error	←-----→	8
129-192-- I/O errors	-----→	12
Requires network Error-Action	←-----	12 and all others.

INTERDATA 8/32 I/O TRANSFORMATIONS Page 1 of 2

NIOB-CODE = 0 Used for SRMN-I/O and DRNM-I/O.

INTERDATA 8/32 PARAMETERS

NIOB PARAMETERS

FC(0) = 0	←-----→	NIOB-CODE = 0
FC(1-2) = 01	←-----→	NIOB-FUNC(0) = 0
FC(1-2) = 10	←-----→	NIOB-FUNC(0) = 1
FC(3) = 0	←-----→	NIOB-FUNC(4) = 0
FC(3) = 1	←-----→	NIOB-FUNC(4) = 1
FC(4) = 0	←-----→	NIOB-FUNC(6) = 0
FC(4) = 1	←-----→	NIOB-FUNC(6) = 1
FC(5) = 0	←-----→	NIOB-FUNC(1-3)=000
FC(5) = 1	←-----→	NIOB-FUNC(1-3)=001
***NO-MAPPING	←-----→	NIOB-FUNC(1-3)=010
***NO-MAPPING	←-----→	NIOB-FUNC(1-3)=011
***NO-MAPPING	←-----→	NIOB-FUNC(1-3)=100
FC(6) = 0	←-----→	NIOB-FUNC(7) = 0
FC(6) = 1	←-----→	NIOB-FUNC(7) = 1
FC(7) = 0	←-----→	NIOB-FUNC(5) = 0
FC(7) = 1	←-----→	NIOB-FUNC(5) = 1
LU	←-----→	NIOB-LUNM
BUFSADR	←-----→	NIOB-SADR
BUFEADR	←-----→	NIOB-EADR
RNRCDNUM	←-----→	NIOB-RREC
DATLGNTN	-----→	NIOB-NMBT ←--
		BUFEADR - BUFSADR
DATLGNTN	←-----→	NIOB-NMBT

INTERDATA 8/32 I/O TRANSFORMATIONS Page 2 of 2

STATUS RETURNS Used for DSMN-I/O and SSNM-I/O.

INTERDATA 8/32 STAT =

NIOB-STNT =

0---Successful	←-----→	0
2---Illegal function	←-----→	2
4---Device unavailable	-----→	12
8---End of medium	←-----→	5
16--End of file	←-----→	6
32--Unrecoverable error	-----→	12
64--Recoverable error	-----→	12
128-Illegal LU	←-----→	1
Requires network Error-Action	←-----	12 and all others.

INTERDATA 7/16 GENERAL CHARACTERISTICS

1. All non-contiguous files assume system buffering.
2. Contiguous files assume user buffering and blocking.
3. Protection keys are used to determine if access privileges should be granted. There is a key for write protection and a key for read protection.
4. When volumes are mounted on devices, the device is mapped to the volume and references made to the volume are automatically mapped into the correct device.
5. Contiguous files on the Interdata have variable length records so the length must be specified at access time.
6. The Interdata requires fixed length records for random (indexed) files.

INTERDATA 7/16 CREATE TRANSFORMATIONS Page 1 of 2

NFMB-CODE = 1 Used for SRMN-CREATE and DRNM-CREATE.

INTERDATA 7/16 PARAMETERS

NFMB PARAMETERS

<u>CONTIGUOUS File</u>	←-----→	<u>CONTIGUOUS File</u>
CMD = 80	←-----→	NFMB-CODE = 1
MOD-FT = 0	←-----→	NFMB-MDFT = 0
MOD-BF = 1	←-----→	NFMB-MDBF = 1
**NO-MAPPING	←-----→	NFMB-MDBF = 0
WK-RK	←-----→	NFMB-PKEY
FDVL	←-----→	NFMB-FDVL
FDFN	←-----→	NFMB-FDFN
FDEX	←-----→	NFMB-FDEX
SIZEFL	-----→	NFMB-SZFL ←--
		SIZEFL * 256
SIZEFL ←-- NFMB-SZFL ÷ 256	←-----→	NFMB-SZFL
No LRECL given	-----→	NFMB-LRCL ←-- -1
**NO-MAPPING	←-----→	NFMB-LRCL ≠ -1

**NO-MAPPING	←-----→	<u>CHAINED File</u>

<u>INDEXED File</u>	←-----→	<u>INDEXED File</u>
CMD = 80	←-----→	NFMB-CODE = 1
MOD-FT = 2	←-----→	NFMB-MDFT = 2
MOD-BF = 0	←-----→	NFMB-MDBF = 0
**NO-MAPPING	←-----→	NFMB-MDBF = 1
WK-RK	←-----→	NFMB-PKEY
FDVL	←-----→	NFMB-FDVL
FDFN	←-----→	NFMB-FDFN
FDEX	←-----→	NFMB-FDEX

INTERDATA 7/16 CREATE TRANSFORMATIONS Page 2 of 2

INTERDATA 7/16 PARAMETERS

NFMB PARAMETERS

SIZEFL	----->	NFMB-SZFL <--
		SIZEFL * 256
SIZEFL <-- NFMB-SZFL ÷ 256	<-----	NFMB-SZFL
SIZENX	----->	NFMB-SZNX <--
		SIZENX * 256
SIZENX <-- NFMB-SZNX ÷ 256	<-----	NFMB-SZNX
LRCL	<----->	NFMB-LRCL

STATUS RETURNS Used for DSMN-CREATE and SSNM-CREATE.

INTERDATA 7/16 STAT =

NFMB-STNT =

0---Successful	<----->	0
1---Illegal function	<----->	1
3---Volume not mounted	<----->	3
4---File already exists	<----->	5
5---Size error	<----->	10
7---Disc write protected	<----->	7
10--Type error, wrong volume	<----->	11
11--File descriptor error	<----->	8
129-192-- I/O errors	----->	12
Requires network Error-Action	<-----	12

INTERDATA 7/16 OPEN TRANSFORMATIONS

NFMB-CODE = 2 Used for SRMN-OPEN and DRNM-OPEN.

INTERDATA 7/16 PARAMETERSNFMB PARAMETERS

CMD = 40	←-----→	NFMB-CODE = 2
MOD-AP =	←-----→	NFMB-MDAP =
0--Shared Read/Write	←-----→	0--SRW
1--Shared Read, Exclusive Write	←-----→	1--SREW
2--Shared Read Only	←-----→	2--SRO
3--Shared Write Only	←-----→	3--SWO
4--Exclusive Read/Write	←-----→	4--ERW
5--Exclusive Read, Shared Write	←-----→	5--ERSW
6--Exclusive Read Only	←-----→	6--ERO
7--Exclusive Write Only	←-----→	7--EWO
WK-RK	←-----→	NFMB-PKEY
FDVL	←-----→	NFMB-FDVL
FDFN	←-----→	NFMB-FDFN
FDEX	←-----→	NFMB-FDEX
LU	←-----→	NFMB-LUNM

STATUS RETURNS Used for DSMN-OPEN and SSNM-OPEN.

INTERDATA 7/16 STAT =NFMB-STNT =

0---Successful	←-----→	0
1---Illegal function	←-----→	1
2---Illegal logical unit	←-----→	2
3---Volume error	←-----→	3
4---File does not exist	←-----→	4
6---Protection error	←-----→	6
7---Privilege error	←-----→	7
8---Buffer error, no room	←-----→	10
10--Type error, device off-line	←-----→	11
11--File descriptor error	←-----→	8
12--Attempt to assign bad device	-----→	12
129-192--I/O errors	-----→	12
Requires network Error-Action	←-----→	12 and all others.

INTERDATA 7/16 CLOSE TRANSFORMATIONS

NFMB-CODE = 4 Used for SRMN-CLOSE and DRNM-CLOSE.

INTERDATA 7/16 PARAMETERSNFMB PARAMETERS

CMD = 04

←-----→ NFMB-CODE = 4

LU

←-----→ NFMB-LUNM

STATUS RETURNS Used for DSMN-CLOSE and SSNM-CLOSE.

INTERDATA 7/16 STAT =NFMB-STNT =

0---Successful

←-----→ 0

2---Illegal logical unit

←-----→ 2

9---Logical unit assign error

←-----→ 9

129-192--I/O errors

-----→ 12

Requires network Error-Action

←----- 12 and all others.

INTERDATA 7/16 DELETE TRANSFORMATIONS

NFMB-CODE = 8 Used for SRMN-DELETE and DRNM-DELETE.

INTERDATA 7/16 PARAMETERSNFMB PARAMETERS

CMD = 02	←-----→	NFMB-CODE = 8
WK-RK	←-----→	NFMB-PKEY
FDVL	←-----→	NFMB-FDVL
FDFN	←-----→	NFMB-FDFN
FDEX	←-----→	NFMB-FDEX

STATUS RETURNS Used for DSMN-DELETE and SSNM-DELETE.

INTERDATA 7/16 STAT =NFMB-STNT =

0---Successful	←-----→	0
1---Illegal function	←-----→	1
3---Volume error	←-----→	3
4---Name error, bad file	←-----→	4
6---Protect error, bad keys	←-----→	6
7---Privilege error, write protect	←-----→	6
9---Assign error, file assigned	←-----→	7
10--Type error, cannot delete device	←-----→	11
11--File descriptor error	←-----→	8
129-192-- I/O errors	-----→	12
Requires network Error-Action	←-----→	12 and all others.

INTERDATA 7/16 I/O TRANSFORMATIONS Page 1 of 2

NIOB-CODE = 0 Used for SRMN-I/O and DRNM-I/O.

INTERDATA 7/16 PARAMETERS

NIOB PARAMETERS

FC(0) = 0	←-----→	NIOB-CODE = 0
FC(1-2) = 01	←-----→	NIOB-FUNC(0) = 0
FC(1-2) = 10	←-----→	NIOB-FUNC(0) = 1
FC(3) = 0	←-----→	NIOB-FUNC(4) = 0
FC(3) = 1	←-----→	NIOB-FUNC(4) = 1
FC(4) = 0	←-----→	NIOB-FUNC(6) = 0
FC(4) = 1	←-----→	NIOB-FUNC(6) = 1
FC(5) = 0	←-----→	NIOB-FUNC(1-3)=000
FC(5) = 1	←-----→	NIOB-FUNC(1-3)=001
**NO-MAPPING	←-----→	NIOB-FUNC(1-3)=010
**NO-MAPPING	←-----→	NIOB-FUNC(1-3)=011
**NO-MAPPING	←-----→	NIOB-FUNC(1-3)=100
FC(6) = 0	←-----→	NIOB-FUNC(7) = 0
FC(6) = 1	←-----→	NIOB-FUNC(7) = 1
FC(7) = 0	←-----→	NIOB-FUNC(5) = 0
FC(7) = 1	←-----→	NIOB-FUNC(5) = 1
LU	←-----→	NIOB-LUNM
BUFSADR	←-----→	NIOB-SADR
BUFEADR	←-----→	NIOB-EADR
RNRCDNUM	←-----→	NIOB-RREC
DATLGNTN	-----→	NIOB-NMBT ←--
		BUFEADR - BUFSADR
DATLGNTN	←-----→	NIOB-NMBT

INTERDATA 7/16 I/O TRANSFORMATIONS Page 2 of 2

STATUS RETURNS Used for DSMN-I/O and SSNM-I/O.

INTERDATA 7/16 STAT =

NIOB-STNT =

0---Successful	←-----→	0
2---Illegal function	←-----→	2
4---Device unavailable	-----→	12
8---End of medium	←-----→	5
16---End of file	←-----→	6
32--Unrecoverable error	-----→	12
64--Recoverable error	-----→	12
128-Illegal LU	←-----→	1
Requires network Error-Action	←-----	12 and all others.

APPENDIX B

MACHINE DEPENDENT PARAMETER BLOCKS

NOVA FILE MANAGEMENT PARAMETERS

Calling Sequence and Parameters

.SYSTEM

.CMD N

Error Return-1

Normal Return-2

CMD Command code, INIT---Initialize device/directory
 CRAND--Create random file
 CREAT--Create sequential file
 CCONT--Create contiguous file
 OPEN---Shared RD/WR
 EOPEN--Shared RD, Exclusive WR
 ROPEN--Shared RD only
 CLOSE--Unassign file
 DELET--Deallocate file

N Channel number assigned to a specific file/device,
 used in opening and closing files.

Accumulators

AC0 For INIT = Pointer to three character device code.
 For CREAT = Pointer to Filename.Extension string.
 For OPEN = Pointer to Filename.Extension string.
 For CLOSE = Not used.
 For DELET = Pointer to Filename.Extension string.

AC1 For INIT =0--if you want new current directory.
 =1--if you want device cleaned up.
 For CREAT = number of disk blocks in contiguous FL.
 For OPEN = inhibit mask for the channel.
 For CLOSE = Not used.
 For DELET = Not used.

AC2 Error status code returned if Return-1 is taken.

NOVA I/O PARAMETERSCalling Sequence and Parameters

.SYSTEM

.CMD N

Error Return-1

Normal Return-2

CMD I/O command code, RDS---Read sequential
 RDR---Read random
 WRS---Write sequential
 WRR---Write random

N The channel number assigned to a file/device at
 OPEN time to provide device independent I/O.

Accumulators

AC0 The starting byte address of the buffer.

AC1 For sequential I/O = number of bytes to transfer.
 For random I/O = random record number to be used.

AC2 Error status code returned if Return-1 is taken.

INTERDATA 8/32 FILE MANAGEMENT PARAMETER BLOCK Page 1 of 2

CMD (1)	MOD-____ (1) AP BF FT	STAT (1)	LU (1)
WK-RK (2)	LRECL (2)	FD____ (15) VL FN EX	SIZE____ (4) FL NX

(#) indicates field length in bytes

CMD	Command code,	80---CREATE
		40---OPEN
		04---CLOSE
		02---DELETE

```

MOD-__      Modifications argument,
AP--Access privileges, 0--Shared RD/WR
              (3 bits)    1--Shared RD, Exclusive WR
                          2--Shared RD only
                          3--Shared WR only
                          4--Exclusive RD/WR
                          5--Exclusive RD, Shared WR
                          6--Exclusive RD only
                          7--Exclusive WR only

BF--Buffering provided, 0--System supplied
              (2 bits)    1--User supplied

FT--File type, 0--Contiguous
              (3 bits)  1--Chained
                          2--Indexed

```

```
STAT      Status result, = 0--successful
          #0--code indicating error
```

LU Logical unit assigned to specified file/device.

WK-RK File WR/RD access protection keys.

LRECL Logical record length for a buffered file.

FD__ File descriptor specifying exact file desired.
VL--4 bytes for the volume.
FN--8 bytes for the Filename in ASCII format.
EX--3 bytes for the Extension to describe content.

INTERDATA 8/32 FILE MANAGEMENT PARAMETER BLOCK Page 2 of 2

SIZE__ Size parameter in bytes,
FL--Contiguous file--Number of sectors in file.
Chained file--Number of sectors/block.
Indexed file--Number of sectors/data block.
NX--Indexed file--Number of sectors/index block.

INTERDATA 8/32 I/O PARAMETER BLOCK Page 1 of 2

FC (1)	LU (1)	DVINSTAT (1)	DVDPSTAT (1)
BUFSADR (2)	BUFEADR (2)	RNRCDNUM (4)	DATLGTH (2)

(#) indicates field length in bytes

FC Function description,
 BIT

0 = 0 to indicate a data transfer.
1-2 = 10--Read
 = 01--Write
 = 11--Test and Set
 = 00--Test I/O complete
3 = 0---ASCII formatting
 = 1---Binary formatting
4 = 0---Continue processing
 = 1---Wait for I/O to complete
5 = 0---Process next record sequentially
 = 1---Use random field to process randomly
6 = 0---Wait for I/O device
 = 1---Immediate reject if device busy
7 = 0---Use device/file for formatting
 = 1---No formatting

LU Logical unit number from 1-254 assigned to a
 particular device/file by the operator or an SVC 7
 request to provide device independent I/O.

DVINSTAT The device independent status indicating the kind
 of error that occurred when it is non-zero.

DVDPSTAT The device dependent status which contains the
 device number when an error occurs.

BUFSADR Buffer starting address; the first byte in the
 buffer which must be on boundary and in the same
 logical segment as the ending byte.

BUFEADR The last byte in the buffer.

INTERDATA 8/32 I/O PARAMETER BLOCK Page 2 of 2

RNRCDNUM The random record number to be used for random access (0 relative).

DATLGNTN Returns the actual number of bytes transferred.

INTERDATA 7/16 FILE MANAGEMENT PARAMETER BLOCK Page 1 of 2

CMD (1)	MOD-__ (1) AP BF FT	STAT (1)	LU (1)
WK-RK (2)	LRECL (2)	FD__ (15) VL FN EX	SIZE__ (4) FL NX

(#) indicates field length in bytes

CMD Command code, 80---CREATE
 40---OPEN
 04---CLOSE
 02---DELETE

MOD-__ Modifications argument,
 AP---Access privileges, 0--Shared RD/WR
 (3 bits) 1--Shared RD, Exclusive WR
 2--Shared RD only
 3--Shared WR only
 4--Exclusive RD/WR
 5--Exclusive RD, Shared WR
 6--Exclusive RD only
 7--Exclusive WR only

BF---Buffering provided, 0--System supplied
 (2 bits) 1--User supplied

FT---File type, 0--Contiguous
 (3 bits) 1--Not valid
 2--Indexed

STAT Status result, = 0--successful
 ≠ 0--code indicating error

LU Logical unit assigned to specified file/device.

WK-RK File WR/RD access protection keys.

LRECL Logical record length for a buffered file.

FD__ File descriptor specifying exact file desired.
 VL---4 bytes for the volume.
 FN---8 bytes for the Filename in ASCII format.
 EX---3 bytes for the Extension to describe content.

INTERDATA 7/16 FILE MANAGEMENT PARAMETER BLOCK Page 2 of 2

SIZE__ Size parameter in bytes,
FL--Contiguous file--Number of sectors in file.
Indexed file--Number of sectors/data block.
NX--Indexed file--Number of sectors/index block.

INTERDATA 7/16 I/O PARAMETER BLOCK Page 1 of 2

FC (1)	LU (1)	DVINSTAT (1)	DVDPSTAT (1)
BUFSADR (2)	BUFEADR (2)	RNRCDNUM (4)	DATLGNTN (2)

(#) indicates field length in bytes

FC Function description,
 BIT

0 = 0 to indicate a data transfer.
1-2 = 10--Read
 = 01--Write
 = 11--Test and Set
 = 00--Test I/O complete
3 = 0---ASCII formatting
 = 1---Binary formatting
4 = 0---Continue processing
 = 1---Wait for I/O to complete
5 = 0---Process next record sequentially
 = 1---Use random field to process randomly
6 = 0---Wait for I/O device
 = 1---Immediate reject if device busy
7 = 0---Use device/file for formatting
 = 1---No formatting

LU Logical unit number from 1-31 assigned to a
 particular device/file by the operator or an SVC 7
 request to provide device independent I/O.

DVINSTAT The device independent status indicating the kind
of error that occurred when it is non-zero.

DVDPSTAT The device dependent status which contains the
device number when an error occurs.

BUFSADR Buffer starting address; the first byte in the
buffer which must be on boundry and in the same
logical segment as the ending byte.

BUFEADR The last byte in the buffer.

INTERDATA 7/16 I/O PARAMETER BLOCK Page 2 of 2

RNRCDNUM The random record number to be used for random
 access (0 relative).

DATLGNTN Returns the actual number of bytes transferred.

APPENDIX C

MACHINE/COMMAND DEPENDENT PARAMETERS

NOVA COMMAND DEPENDENT PARAMETERS

COMMAND	CMD	CHANNEL #	AC2	AC1	ACO
INIT	X		X	X	X
CREATE	X		X	X	X
OPEN	X	X	X	X	X
CLOSE	X	X	X		
DELETE	X		X		X

ACO = Pointer to Filename.Extension.

AC1 = File size, inhibit mask, or initialize flag.

AC2 = Error status code.

=====

I/O	CMD	CHANNEL #	AC2	AC1	ACO
-----	-----	-----------	-----	-----	-----

ACO = Pointer to the first byte of the buffer. .

AC1 = The record number or the number of bytes to transfer.

AC2 = Error status code.

INTERDATA 7/16 and 8/32 COMMAND DEPENDENT PARAMETERS

<u>COMMAND</u>	<u>CMD</u>	<u>MOD</u>	<u>STAT</u>	<u>LU</u>	<u>WK-RK</u>	<u>LRECL</u>	<u>FD</u>	<u>SIZE</u>
CREATE	X	X	X		X	X	X	X
OPEN	X	X	X	X	X		X	
CLOSE	X		X	X				
DELETE	X		X		X		X	

<u>COMMAND</u>	<u>FC</u>	<u>STAT</u>	<u>LU</u>	<u>SBUF</u>	<u>EBUF</u>	<u>RRECNUM</u>	<u>DATLGTH</u>
I/O-SEQ	X	X	X	X	X		X
I/O-RAN	X	X	X	X	X	X	

APPENDIX D

MACHINE FILE STRUCTURE COMPARISONS

MACHINE FILE STRUCTURE COMPARISONS

NETWORK STANDARD INDEXED (RANDOM) FILE

INTERDATA 8/32 and 7/16

1. User specifies LRECL.
2. Physical size of index and data blocks is a number of 256 byte sectors.
3. System buffering.
4. Open ended--can grow.

NOVA

1. LRECL = 128 bytes.
2. Block size = 512 bytes.
3. System buffering.
4. Open ended.

NETWORK STANDARD CONTIGUOUS FILE

INTERDATA 8/32 and 7/16

1. User specifies LRECL.
2. Physical size = the number of 256 byte sectors reserved for the file.
3. User buffering, no blocking.
4. Fixed file size.

NOVA

1. LRECL = 128 bytes.
2. File size = number of 512 byte blocks reserved.
3. System buffering and blocking.
4. Fixed file size.

NETWORK STANDARD CHAINED (SEQUENTIAL) FILE

INTERDATA 8/32 (Not valid on 7/16)

1. LRECL fixed length.
2. Physical size of blocks = the number of 256 byte sectors specified.
3. System buffering.
4. Open ended file.

NOVA

1. LRECL variable, specified at request time.
2. Block size = 512 bytes.
3. System buffering.
4. Open ended file.

APPENDIX E

MACHINE DEPENDENT COMMAND HIPOS

NOVA SRMN-CREATEINPUTS

1. Machine dependent arguments---.CMD CHNL# , ACO =
Pointer to Filename.Extension, AC1 = number of disk
blocks if request is for a contiguous file create.

PROCESSING REQUIRED

1. NFMB-STNT <-- -1
2. NFMB-CODE <-- 1
3. NFMB-FDFN <-- Filename
4. NFMB-FDEX <-- Extension
5. NFMB-PKEY <-- 0
6. If CMD = .CCONT Then ASSRT Contiguous file create.
 NFMB-MDFT <-- 0
 NFMB-MDBF <-- 0
 NFMB-SZFL <-- C(AC1) * 512
 NFMB-LRCL <-- 128
 RETURN ASSRT NFMB file create ready.
7. If CMD = .CREAT Then ASSRT Chained file create.
 NFMB-MDFT <-- 1
 NFMB-MDBF <-- 0
 NFMB-SZFL <-- 512
 NFMB-LRCL <-- -1 (to show not used)
 RETURN ASSRT NFMB file create ready.
8. If CMD = .CRAND Then ASSRT Index file create.
 NFMB-MDFT <-- 2
 NFMB-MDBF <-- 0
 NFMB-SZFL <-- 512
 NFMB-LRCL <-- 128
 RETURN ASSRT NFMB file create ready.

OUTPUTS

1. NFMB CREATE request ready to go to the destination machine.

DISCUSSION

On the Nova sequential files are not created with a fixed length logical record size and do not retain a logical record length for the file. All accesses are required to specify a record length. Therefore, NFMB-LRCL is set to a negative one to identify it as an inactive argument.

NOVA SRMN-OPENINPUTS

1. Machine dependent arguments---.CMD CHNL# , ACO = Pointer to Filename.Extension, AC1 = Channel inhibit mask.

PROCESSING REQUIRED

1. NFMB-STNT ←-- -1
2. NFMB-CODE ←-- 2
3. NFMB-FDFN ←-- Filename
4. NFMB-FDEX ←-- Extension
5. NFMB-PKEY ←-- 0
6. NFMB-LRCL ←-- C(AC1)
7. NFMB-LUNM ←-- CHNL#
8. If CMD = .OPEN Then NFMB-MDAP ←-- 0
9. If CMD = .EOPEN Then NFMB-MDAP ←-- 1
10. If CMD = .ROPEN Then NFMB-MDAP ←-- 2

OUTPUTS

1. NFMB OPEN request ready to go to the destination machine.

NOVA SRMN-CLOSEINPUTS

1. Machine dependent arguments---.CLOSE CHNL#.

PROCESSING REQUIRED

1. NFMB-STNT <-- -1
2. NFMB-CODE <-- 4
3. NFMB-LUNM <-- CHNL#

OUTPUTS

1. NFMB CLOSE request ready to go to the destination machine.

NOVA SRMN-DELETEINPUTS

1. Machine dependent arguments---.DELET, ACO = Pointer to
Filename.Extension.

PROCESSING REQUIRED

1. NFMB-STNT <-- -1
2. NFMB-CODE <-- 8
3. NFMB-PDFN <-- Filename
4. NFMB-FDEX <-- Extension
5. NFMB-PKEY <-- 0

OUTPUTS

1. NFMB DELETE request ready to go to the destination machine.

NOVA SRMN-I/OINPUTS

1. Machine dependent arguments---.CMD CHNL#, ACO = Pointer to the first byte of the buffer, AC1 = number of bytes to be transferred or the random record number to use.

PROCESSING REQUIRED

1. NIOB-STNT \leftarrow -1
2. If CMD = .WRS Then NIOB-FUNC(0-3) \leftarrow 0000
3. If CMD = .RDS Then NIOB-FUNC(0-3) \leftarrow 1000
4. If CMD = .WRR Then NIOB-FUNC(0-3) \leftarrow 0001
5. If CMD = .RDR Then NIOB-FUNC(0-3) \leftarrow 1001
6. NIOB-FUNC(4-7) \leftarrow 1110
7. NIOB-LUNM \leftarrow CHNL#
8. NIOB-SADR \leftarrow C(ACO)
9. If NIOB-FUNC(0-3) = 1000 or 0000 Then ASSRT Sequential I/O.
 NIOB-NMBT \leftarrow C(AC1)
 NIOB-EADR \leftarrow C(AC1) - C(ACO)
 RETURN ASSRT Sequential I/O ready.
10. If NIOB-FUNC(0-3) = 1001 or 0001 Then ASSRT Random I/O.
 NIOB-NMBT \leftarrow 128
 NIOB-EADR \leftarrow 128 + C(ACO)
 NIOB-RREC \leftarrow C(AC1)
 RETURN ASSRT Random I/O ready.

OUTPUTS

1. NIOB ready to go to the destination machine.

INTERDATA 8/32 SRMN-CREATEINPUTS

1. Machine dependent arguments---CMD = 80, MOD-FT, WK-RK, FDVL, FDFN, FDEX, SIZEFL, LRECL, and SIZENX.

PROCESSING REQUIRED

1. NFMB-STNT <-- -1
2. NFMB-CODE <-- 1
3. NFMB-FDVL <-- FDVL
4. NFMB-FDFN <-- FDFN
5. NFMB-FDEX <-- FDEX
6. NFMB-PKEY <-- WK-RK
7. NFMB-SZFL <-- SIZEFL * 256
8. NFMB-MDFT <-- MOD-FT
9. If MOD-FT = 0 Then ASSRT Contiguous file create.
 NFMB-MDBF <-- 1
 NFMB-LRCL <-- -1 (to show not used)
 RETURN ASSRT NFMB file create ready.
10. If MOD-FT = 1 Then ASSRT Chained file create.
 NFMB-MDBF <-- 0
 NFMB-LRCL <-- LRECL
 RETURN ASSRT NFMB file create ready.
11. If MOD-FT = 2 Then ASSRT Indexed file create.
 NFMB-MDBF <-- 0
 NFMB-SZNX <-- SIZENX * 256
 NFMB-LRCL <-- LRECL
 RETURN ASSRT NFMB file create ready.

OUTPUTS

1. NFMB CREATE request ready to go to the destination machine.

DISCUSSION

Contiguous files have variable length records.

INTERDATA 8/32 SRMN-OPENINPUTS

1. Machine dependent arguments--- CMD = 40, MOD-AP, MOD-BF, LU, WK-RK, FDVL, FDFN, and FDEX.

PROCESSING REQUIRED

1. NFMB-STNT ←-- -1
2. NFMB-CODE ←-- 2
3. NFMB-FDVL ←-- FDVL
4. NFMB-FDEX ←-- FDEX
5. NFMB-FDFN ←-- FDFN
6. NFMB-PKEY ←-- WK-RK
7. NFMB-LUNM ←-- LU
8. NFMB-MDAP ←-- MOD-AP

OUTPUTS

1. NFMB OPEN request ready to go to the destination machine.

INTERDATA 8/32 SRMN-CLOSEINPUTS

1. Machine dependent arguments---CMD = 04, and LU.

PROCESSING REQUIRED

1. NFMB-STNT ←-- -1
2. NFMB-CODE ←-- 4
3. NFMB-LUNM ←-- LU

OUTPUTS

1. NFMB CLOSE request ready to go to the destination machine.

INTERDATA 8/32 SRMN-DELETEINPUTS

1. Machine dependent arguments---CMD = 02, WK-RK, FDVL, FDFN, and FDEX.

PROCESSING REQUIRED

1. NFMB-STNT ←-- -1
2. NFMB-CODE ←-- 8
3. NFMB-PKEY ←-- WK-RK
4. NFMB-FDVL ←-- FDVL
5. NFMB-FDFN ←-- FDFN
6. NFMB-FDEX ←-- FDEX

OUTPUTS

1. NFMB DELETE request ready to go to the destination machine.

INTERDATA 8/32 SRMN-I/OINPUTS

1. Machine dependent arguments---FC, LU, BUFSADR, BUFEADR, RNRCDNUM, and DATLGNTN.

PROCESSING REQUIRED

1. NIOB-STNT <-- -1
2. If FC(0) = 1 Then ASSRT I/O command, REWIND etc.
 NIOB-CODE <-- 1
 NIOB-FUNC <-- FC
 Continue from step 10.
3. NIOB-CODE <-- FC(0)
4. NIOB-FUNC(0) <-- FC(1-2) - 1
5. NIOB-FUNC(4) <-- FC(3)
6. NIOB-FUNC(6) <-- FC(4)
7. NIOB-FUNC(1-3) <-- FC(5)
8. NIOB-FUNC(7) <-- FC(6)
9. NIOB-FUNC(5) <-- FC(7)
10. NIOB-LUNM <-- LU
11. NIOB-SADR <-- BUFSADR
12. NIOB-EADR <-- BUFEADR
13. NIOB-RREC <-- RNRCDNUM
14. NIOB-NMBT <-- BUFEADR - BUFSADR

OUTPUTS

1. NIOB ready to go to the destination machine.

INTERDATA 7/16 SRMN-CREATEINPUTS

1. Machine dependent arguments---CMD = 80, MOD-FT, WK-RK, FDVL, FDFN, FDEX, SIZEFL, and SIZENX.

PROCESSING REQUIRED

1. NFMB-STNT <-- -1
2. NFMB-CODE <-- 1
3. NFMB-FDVL <-- FDVL
4. NFMB-FDFN <-- FDFN
5. NFMB-FDEX <-- FDEX
6. NFMB-PKEY <-- WK-RK
7. NFMB-SZFL <-- SIZEFL * 256
8. NFMB-MDFT <-- MOD-FT
9. If MOD-FT = 0 Then ASSRT Contiguous file create.
 NFMB-MDBF <-- 1
 NFMB-LRCL <-- -1
 RETURN ASSRT NFMB file create ready.
10. If MOD-FT = 2 Then ASSRT Indexed file create.
 NFMB-MDBF <-- 0
 NFMB-SZNX <-- SIZENX * 256
 NFMB-LRCL <-- LRECL
 RETURN ASSRT NFMB file create ready.

OUTPUTS

1. NFMB CREATE request ready to go to the destination machine.

DISCUSSION

Contiguous files have variable length records and user buffering is required. A logical record length is not specified until access time.

INTERDATA 7/16 SRMN-OPEN

Same as SRMN-OPEN for the Interdata 8/32.

INTERDATA 7/16 SRMN-CLOSE

Same as SRMN-CLOSE for the Interdata 8/32.

INTERDATA 7/16 SRMN-DELETE

Same as SRMN-DELETE for the Interdata 8/32.

INTERDATA 7/16 SRMN-I/O

Same as SRMN-I/O for the Interdata 8/32.

NOVA DRNM-CREATE

Page 1 of 2

INPUTS

1. NFMB CREATE request from the source machine.

PROCESSING REQUIRED

1. Perform .INIT using NFMB-FDDV and NFMB-FDDR.
2. ACO \leftarrow Pointer to NFMB-FDFN and NFMB-FDEX
3. If initialization not complete Then
 NO-MAPPING
 RETURN ASSRT Bad file descriptor.
4. If NFMB-MDBF \neq 0 Then ASSRT Buffering incorrect.
 NO-MAPPING
 RETURN ASSRT No action taken.
5. If NFMB-MDFT = 0 Then
 AC1 \leftarrow NFMB-SZFL \div 512
 CMD \leftarrow .CCONT
 RETURN ASSRT Ready to execute.
6. If NFMB-MDFT = 1 Then ASSRT Sequential file create.
 If NFMB-SZFL $>$ 512 Then ASSRT Block size too big.
 NO-MAPPING
 RETURN ASSRT No action taken.
 If NFMB-LRCL \neq -1 Then ASSRT Need fixed rcd length.
 NO-MAPPING
 RETURN ASSRT No action taken.
 CMD \leftarrow .CREAT
 RETURN ASSRT Ready to execute.
7. If NFMB-MDFT = 2 Then ASSRT Index file create.
 If NFMB-LRCL $>$ 128 Then ASSRT Rcd length too big.
 NO-MAPPING
 RETURN ASSRT No action taken.
 If NFMB-SZFL or NFMB-SZNX $>$ 512 Then ASSRT Size too big.
 NO-MAPPING
 RETURN ASSRT No action taken.
 CMD \leftarrow .CRAND
 RETURN ASSRT Ready to execute.

OUTPUTS

1. Machine dependent arguments ready to be executed---.CMD,
AC1 = number of blocks, ACO = Pointer to Filename.Extension,
or a NO-MAPPING condition.

DISCUSSION

If the Nova does not have the necessary information to initialize the device and directory, then it cannot perform the requested action. Since the Nova also requires system buffering, if the user specifies buffering it is incompatible. However, within the network there will be system software doing all I/O and therefore user specified buffering might be able to be ignored completely at request time. The other major mapping problems all concern the assumed fixed sizes of blocks and records on the Nova. For sequential or chained files the Nova assumes 512 bytes per block and anything smaller is rounded up with space wasted, while anything larger is incompatible. This type of file also allows for a variable length record within these blocks and therefore does not use or store a specified fixed record length. To provide fixed length sequential records the Nova would have to emulate other architectures by storing the information in the file and then using it when record length is not specified at access time. For the indexed or random file on the Nova, the blocks are fixed at 512 bytes also, with the addition that the random records are fixed at 128 bytes. The same rules apply towards mappability and wasted space.

NOVA DRNM-OPENINPUTS

1. NFMB OPEN request from the source machine.

PROCESSING REQUIRED

1. Same as steps 1-3 for DRNM-CREATE
2. CHNL# \leftarrow NFMB-LUNM
3. If NFMB-MTPS = NFMB-MTPD Then ASSRT Same machines.
AC1 \leftarrow NFMB-LRCL ASSRT Inhibit mask set.
4. If NFMB-MDAP = 0 Then CMD \leftarrow .OPEN
5. If NFMB-MDAP = 1 Then CMD \leftarrow .EOPEN
6. If NFMB-MDAP = 2 Then CMD \leftarrow .ROPEN
7. Else NO-MAPPING ASSRT Incompatible.

OUTPUTS

1. Machine dependent arguments ready to be executed---
.CMD CHNL# , ACO = Pointer to Filename.Extension, AC1 =
Inhibit mask if applicable, or a NO-MAPPING condition.

DISCUSSION

At this time no additional access modes are set up for mapping on the Nova; the three modes shown above are the only ones supported directly. Defaults could be set up in some cases but would require decisions on usage and environment to be answered first.

NOVA DRNM-CLOSEINPUTS

1. NFMB CLOSE request from the source machine.

PROCESSING REQUIRED

1. CMD <-- .CLOSE
2. CHNL# <-- NFMB-LUNM

OUTPUTS

1. Machine dependent arguments ready to be executed---.CLOSE
CHNL#.

NOVA DRNM-DELETEINPUTS

1. NFMB DELETE request from the source machine.

PROCESSING REQUIRED

1. Same as steps 1-3 for DRNM-CREATE.
2. CMD <-- .DELET

OUTPUTS

1. Machine dependent arguments ready to be executed---
.DELET, ACO = Pointer to Filename.Extension, or a NO-MAPPING
condition because of initialization problems discussed
in DRNM-CREATE.

INPUTS

1. NIOB request from the source machine.

PROCESSING REQUIRED

1. If NIOB-FUNC(4-7) \neq 1110 Then NO-MAPPING RETURN
2. If NIOB-FUNC(0-3) = 1000 Then CMD \leftarrow .RDS, go to step 6
3. If NIOB-FUNC(0-3) = 0000 Then CMD \leftarrow .WRS, go to step 6
4. If NIOB-FUNC(0-3) = 1001 Then CMD \leftarrow .RDR, go to step 6
5. If NIOB-FUNC(0-3) = 0001 Then CMD \leftarrow .WRR, go to step 6
Else (other commands not considered at this time)
6. CHNL# \leftarrow NIOB-LUNM
7. ACO \leftarrow NIOB-SADR
8. If CMD = .RDS or .WRS Then ASSRT Sequential I/O.
If NIOB-NMBT \neq -1 Then ASSRT Length is specified.
AC1 \leftarrow NIOB-NMBT
RETURN ASSRT Ready for I/O.
Else NO-MAPPING ASSRT Rcd length not given.
RETURN
9. If CMD = .RDR or .WRR Then ASSRT Random I/O.
If NIOB-NMBT \leq 128 Then ASSRT Rcd within limits.
AC1 \leftarrow NIOB-RREC
RETURN ASSRT Ready for I/O
Else NO-MAPPING
RETURN ASSRT Rcd is too big.

OUTPUTS

1. Machine dependent I/O arguments ready to be executed---
.CMD CHNL#, ACO = first byte in buffer, AC1 = record
number or the logical record length, or NO-MAPPING occurred.

DISCUSSION

The Nova requires sequential I/O and random I/O to be non-formatted and binary with all waiting conditions assumed. Sequential I/O also requires the record length to be specified since it is treated as variable length with no fixed information retained anywhere. Random I/O assumes fixed length records of 128 bytes, so anything smaller is rounded up and space is wasted, whereas those larger are incompatible.

INPUTS

1. NFMB CREATE request from the source machine.

PROCESSING REQUIRED

- ```

1. CMD ←-- 80
2. MOD-FT ←-- NFMB-MDFT
3. WK-RK ←-- NFMB-PKEY
4. FDVL ←-- NFMB-FDVL
5. FDFN ←-- NFMB-FDFN
6. FDEX ←-- NFMB-FDEX
7. If NFMB-FD__ is not adequate, Then NO-MAPPING, RETURN
8. SIZEFL ←-- NFMB-SZFL ÷ 256
9. If MOD-FT = 0 Then , ASSRT Contiguous file create.
 If NFMB-MDBF = 0 Then NO-MAPPING, RETURN
 If NFMB-LRCL ≠ -1 Then NO-MAPPING, RETURN
 MOD-BF ←-- NFMB-MDBF
 RETURN ASSRT Ready to execute.
10. If MOD-FT = 1 Then ASSRT Chained file create.
 If NFMB-MDBF = 1 Then NO-MAPPING, RETURN
 If NFMB-LRCL = -1 Then NO-MAPPING, RETURN
 MOD-BF ←-- NFMB-MDBF
 LRECL ←-- NFMB-LRCL
 RETURN ASSRT Ready to execute.
11. If MOD-FT = 2 Then ASSRT Indexed file create.
 If NFMB-MDBF = 1 Then NO-MAPPING, RETURN
 MOD-FT ←-- NFMB-MDBF
 LRECL ←-- NFMB-LRCL
 SIZENX ←-- NFMB-SZNX ÷ 256
 RETURN ASSRT Ready to execute.

```

OUTPUTS

1. Machine dependent arguments ready to be executed--- CMD = 80, FDVL, FDFN, FDEX, WK-RK, SIZEFL, and (LRECL, MOD-BF, and SIZENX set according to MOD-FT), or a NO-MAPPING occurred.

DISCUSSION

Interdata 8/32 contiguous files require user buffering and variable length records to be specified at request time. Chained files require fixed length records and system buffering. These assumptions within the Interdata 8/32 dictate NO-MAPPING conditions in the transformation tables. If buffering becomes a network responsibility then some of these algorithms can be changed to ignore the buffer modification field in the network standard block.

INTERDATA 8/32 DRNM-OPENINPUTS

1. NFMB OPEN request from the source machine.

PROCESSING REQUIRED

1. CMD ←-- 40
2. Same as steps 3-7 of DRNM-CREATE
3. LU ←-- NFMB-LUNM
4. MOD-AP ←-- NFMB-MDAP

OUTPUTS

1. Machine dependent arguments ready to be executed---CMD =  
40, FDVL, FDFN, FDEX, WK-RK, LU, and MOD-AP, or a  
. NO-MAPPING condition because of invalid descriptor.

INTERDATA 8/32 DRNM-CLOSEINPUTS

1. NFMB CLOSE request from the source machine.

PROCESSING REQUIRED

1. CMD ←-- 04
2. LU ←-- NFMB-LUNM

OUTPUTS

1. Machine dependent arguments ready to be executed---  
CMD = 04, and LU.

INTERDATA 8/32 DRNM-DELETEINPUTS

1. NFMB DELETE request from the source machine.

PROCESSING REQUIRED

1. CMD ←-- 02
2. Same as steps 3-7 of DRNM-CREATE.

OUTPUTS

1. Machine dependent arguments ready to be executed---CMD =  
02, WK-RK, FDVL, FDFN, and FDEX, or a NO-MAPPING condition  
because of and invalid file descriptor.

INTERDATA 8/32 DRNM-I/OINPUTS

1. NIOB request from the source machine.

PROCESSING REQUIRED

1. If NIOB-CODE  $\neq$  0 Then (I/O command, REWIND etc.) RETURN
2. FC(0)  $\leftarrow$  NIOB-CODE
3. FC(1-2)  $\leftarrow$  NIOB-FUNC(0) + 1
4. FC(3)  $\leftarrow$  NIOB-FUNC(4)
5. FC(4)  $\leftarrow$  NIOB-FUNC(6)
6. FC(5)  $\leftarrow$  NIOB-FUNC(3)
7. If NIOB-FUNC(1-3)  $>$  1 Then NO-MAPPING, RETURN
8. FC(6)  $\leftarrow$  NIOB-FUNC(7)
9. FC(7)  $\leftarrow$  NIOB-FUNC(5)
10. LU  $\leftarrow$  NIOB-LUNM
11. BUFSADR  $\leftarrow$  NIOB-SADR
12. BUFEADR  $\leftarrow$  NIOB-EADR
13. RNRCDNUM  $\leftarrow$  NIOB-RREC
14. DATLGNTN  $\leftarrow$  NIOB-NMBT

OUTPUTS

1. Machine dependent arguments ready to be executed---FC, LU, BUFSADR, BUFEADR, and (RNRCDNUM, DATLGNTN) when applicable, or a NO-MAPPING condition.

DISCUSSION

The Interdata 8/32 does not have free form and direct block I/O. A combination of its ASCII format type and other bits in the FC may eventually be used in a mapping for the line I/O type in the network standard.

INTERDATA 7/16 DRNM-CREATE

Everything is the same as on the Interdata 8/32 for the DRNM-CREATE with the exception of step 10 in the processing section. Since the Interdata 7/16 does not have chained files, if MOD-FT = 1, then NO-MAPPING occurs because it is incompatible at this time.

INTERDATA 7/16 DRNM-OPEN

Same as DRNM-OPEN on the Interdata 8/32.

INTERDATA 7/16 DRNM-CLOSE

Same as DRNM-CLOSE on the Interdata 8/32.

INTERDATA 7/16 DRNM-DELETE

Same as DRNM-DELETE on the Interdata 8/32.

INTERDATA 7/16 DRNM-I/O

Same as DRNM-I/O on the Interdata 8/32.

NOVA DSMN-CREATEINPUTS

1. Machine dependent status arguments---Instruction Counter (IC), and AC2 = error status if IC is pointing to Return-1.

PROCESSING REQUIRED

1. If IC = Return-2 Then ASSRT Successful.  
NFMB-STNT ←-- 0  
RETURN ASSRT Ready to return.
2. NFMB-STAT ←-- C(AC2)
3. If AC2 = X Then NFMB-STNT ←-- Y for the following (X,Y) pair values. (1,1) (11,5). ASSRT Status converted.  
RETURN ASSRT Ready to return.
4. NFMB-STNT ←-- 12 ASSRT Non-standard error.

OUTPUTS

1. NFMB to return to the source machine indicating the status of the CREATE request.

NOVA DSMN-OPENINPUTS

1. Machine dependent status arguments---Same as DSMN-CREATE.

PROCESSING REQUIRED

1. Same as for DSMN-CREATE except the (X,Y) pairs are (0,2) (1,1) (12,4) (21,9) (31,3) (53,8) (57,10) (60,7) (66,8).

OUTPUTS

1. NFMB for source machine with status of the OPEN request.

NOVA DSMN-CLOSEINPUTS

1. Same as for DSMN-CREATE.

PROCESSING REQUIRED

1. Everything is the same as DSMN-CREATE except for the (X,Y) pairs in step 3. For a CLOSE they are (0,2) (15,9).

OUTPUTS

1. NFMB to return to the source machine indicating the completion status of the CLOSE request.

NOVA DSMN-DELETEINPUTS

1. Same as for DSMN-CREATE.

PROCESSING REQUIRED

1. Everything is the same as DSMN-CREATE except for the (X,Y) pairs in step 3. For a DELETE they are (1,1) (12,4) (13,6) (60,7).

OUTPUTS

1. NFMB to return to the source machine indicating the completion status of the DELETE request.



NOVA DSMN-I/OINPUTS

1. Machine dependent I/O status arguments--- Instruction Counter (IC), and AC2 = error status if IC = Return-1.

PROCESSING REQUIRED

1. If IC = Return-2 Then ASSRT Successful I/O.  
NIOB-STNT  $\leftarrow$  0  
RETURN ASSRT Ready to return.
2. NIOB-STD1  $\leftarrow$  C(AC2)
3. If AC2 = X Then NIOB-STNT  $\leftarrow$  Y for the following (X,Y) pair values. (3,2) (15,3) (47,12) (101,12) (106,12).  
RETURN ASSRT Status converted.
4. If CMD = .RDS or .RDR Then ASSRT Read unique error.  
If AC2 = X Then NIOB-STNT  $\leftarrow$  Y for the following (X,Y) pair values. (6,6) (7,4) (26,5) (30,12) (33,12).  
RETURN ASSRT Status converted.
5. If CMD = .WRS or .WRR Then ASSRT Unique write error.  
If AC2 = X Then NIOB-STNT  $\leftarrow$  Y for the following (X,Y) pair values. (10,4) (27,7) (103,12) (104,12).

OUTPUTS

1. NIOB to return to the source machine indicating the completion status of the I/O request.

INTERDATA 8/32 DSMN-CREATEINPUTS

1. Machine dependent status arguments---STAT.

PROCESSING REQUIRED

1. NFMB-STAT <-- STAT
2. If STAT = X Then NFMB-STNT <-- Y for the following (X,Y) pair values. (0,0) (1,1) (3,3) (4,5) (5,10) (10,11) (11,8) (13,10).

OUTPUTS

1. NFMB to return to the source machine indicating the completion status of the CREATE request.

INTERDATA 8/32 DSMN-OPENINPUTS

1. Machine dependent status argument---STAT.

PROCESSING REQUIRED

1. NFMB-STAT <-- STAT
2. If STAT = X Then NFMB-STNT <-- Y for the following (X,Y) pair values. (0,0) (1,1) (2,2) (3,3) (4,4) (5,10) (6,6) (7,7) (8,10) (9,9) (10,11). RETURN
3. NFMB-STNT <-- 12

OUTPUTS

1. NFMB for source machine with status of OPEN request.

INTERDATA 8/32 DSMN-I/OINPUTS

1. Machine dependent I/O status arguments---DVINSTAT, and DVDRSTAT.

PROCESSING REQUIRED

1. NIOB-STDD  $\leftarrow$  DVDPSTAT
2. NIOB-STD I  $\leftarrow$  DVINSTAT
3. If DVINSTAT = X Then NIOB-STNT  $\leftarrow$  Y for the following (X,Y) pair values. (0,0) (2,2) (8,5) (16,6) (128,1).  
RETURN ASSRT Status converted.
4. NIOB-STNT  $\leftarrow$  12

OUTPUTS

1. NIOB to return to the source machine indicating the completion status of the I/O request.

INTERDATA 7/16 DSMN-CREATE

Everything is the same as DSMN-CREATE for the Interdata 8/32 except for the (X,Y) pair values in step 2 of the processing steps. For the Interdata 7/16 the pair values are (0,0) (1,1) (3,3) (4,5) (5,10) (7,7) (10,11) (11,8).

INTERDATA 7/16 DSMN-OPEN

Everything is the same as DSMN-OPEN for the Interdata 8/32 except for the (X,Y) pair values in step 2 of the processing steps. For the Interdata 7/16 the pair values are (0,0) (1,1) (2,2) (3,3) (4,4) (6,6) (7,7) (8,10) (10,11) (11,8).

INTERDATA 7/16 DSMN-CLOSE

Same as DSMN-CLOSE for the Interdata 8/32.

INTERDATA 7/16 DSMN-DELETE

Everything is the same as DSMN-DELETE for the Interdata 8/32 except for the (X,Y) pair values in the processing steps. For the Interdata 7/16 the pair values are (0,0) (1,1) (3,3) (4,4) (6,6) (7,6) (9,7) (10,11) (11,8).

INTERDATA 7/16 DSMN-I/O

Same as DSMN-I/O for the Interdata 8/32.

NOVA SSNM-CREATEINPUTS

1. NFMB from the destination machine with the request status.

PROCESSING REQUIRED

1. If NFMB-STNT = 0 Then ASSRT File create good.  
     IC  $\leftarrow$  Return-2  
     RETURN ASSRT Return to user.
2. If NFMB-MTPS = NFMB-MTPD Then ASSRT Same machine types.  
     AC2  $\leftarrow$  NFMB-STAT  
     IC  $\leftarrow$  Return-1  
     RETURN ASSRT Return error to user.
3. If NFMB-STNT = X Then AC2  $\leftarrow$  Y for the following (X,Y)  
     pair values. (1,1) (5,11).  
     IC  $\leftarrow$  Return-1  
     RETURN ASSRT Error converted.
4. IC  $\leftarrow$  Error-Action ASSRT Non-standard error.

OUTPUTS

1. The IC is set to the necessary return point for the requested action and AC2 contains the correct error code if a Nova recognizable error occurred.

NOVA SSNM-OPEN

Everything is the same as SSNM-CREATE except for the (X,Y) pair values in step 3 of the processing steps. For the OPEN command they are (2,0) (1,1) (4,12) (9,21) (3,31) (8,53) (10,57) (7,60) (8,66).

NOVA SSNM-CLOSE

Everything is the same as SSNM-CREATE except for the (X,Y) pair values in step 3 of the processing steps. For the CLOSE command they are (2,0) (9,15).

NOVA SSNM-DELETE

Everything is the same as SSNM-CREATE except for the (X,Y) pair values in step 3 of the processing steps. For the DELETE command they are (1,1) (4,12) (6,13) (7,60).

NOVA SSNM-I/OINPUTS

1. NIOB from the destination machine with the status of the completed I/O request.

PROCESSING REQUIRED

1. If NIOB-STNT = 0 Then ASSRT Successful I/O.  
     IC  $\leftarrow$  Return-2  
     RETURN ASSRT Return to user.
2. IC  $\leftarrow$  Return-1
3. If NIOB-MTPS = NIOB-MTPD Then ASSRT Same machines.  
     AC2  $\leftarrow$  NIOB-STD I  
     RETURN ASSRT Return error to user.
4. If NIOB-STNT = X Then AC2  $\leftarrow$  Y for the following (X,Y) pair values. (1,0) (2,3) (4,10) (3,15) (7,27) (6,6) (5,26) (4,7).  
     RETURN ASSRT Error converted.
5. IC  $\leftarrow$  Error-Action ASSRT Non-standard error,

OUTPUTS

1. The IC is set to the necessary return point for the requested action and AC2 contains the correct error code if a Nova recognizable error occurred.

INTERDATA 8/32 SSNM-CREATEINPUTS

1. NFMB from the destination machine with the status of the requested action.

PROCESSING REQUIRED

1. If NFMB-MTPS = NFMB-MTPD Then ASSRT Same machines.  
     STAT  $\leftarrow$  NFMB-STAT  
     RETURN ASSRT Return to user.
2. If NFMB-STNT = X Then STAT  $\leftarrow$  Y for the following (X,Y) pair values. (0,0) (1,1) (3,3) (5,4) (10,5) (11,10) (8,11) (13,10).  
     RETURN ASSRT Error converted.
3. Error-Action necessary because non-standard error code.

OUTPUTS

1. Machine dependent status in STAT or the Error-Action call.

INTERDATA 8/32 SSNM-OPEN

Everything is the same as SSNM-CREATE except for the (X,Y) pair values in step 2 of the processing steps. For the OPEN command they are (0,0) (1,1) (2,2) (3,3) (4,4) (10,5) (6,6) (7,7) (10,8) (9,9) (11,10).



INTERDATA 8/32 SSNM-CLOSE

Everything is the same as SSNM-CREATE except for the (X,Y) pair values in step 2 of the processing steps. For the CLOSE command they are (0,0) (2,2) (9,9).

INTERDATA 8/32 SSNM-DELETE

Everything is the same as SSNM-CREATE except for the (X,Y) pair values in step 2 of the processing steps. For the DELETE command they are (0,0) (3,3) (4,4) (6,6) (6,7) (8,11).

INTERDATA 8/32 SSNM-I/OINPUTS

1. NIOB from the destination machine with the status of the completed I/O request.

PROCESSING REQUIRED

1. If NIOB-MTPS = NIOB-MTPD Then ASSRT Same machines.  
     DVDPSTAT  $\leftarrow$  NIOB-STDD  
     DVINSTAT  $\leftarrow$  NIOB-STD I  
     RETURN ASSRT Return to user.
2. If NIOB-STNT = 0 Then ASSRT Successful.  
     DVDPSTAT  $\leftarrow$  0  
     DVINSTAT  $\leftarrow$  0  
     RETURN ASSRT Return to user.
3. If NIOB-STNT = X Then DVINSTAT  $\leftarrow$  Y for the following (X,Y) pair values. (2,2) (5,8) (6,16) (1,128).
4. Error-Action necessary because non-standard error code.

OUTPUTS

1. Machine dependent status in DVINSTAT or Error-Action.

INTERDATA 7/16 SSNM-CREATE

Everything is the same as SSNM-CREATE on the Interdata 8/32 except for the (X,Y) pair values in step 2 of the processing steps. For the CREATE command on the Interdata 7/16 the pair values are (0,0) (1,1) (5,4) (10,5) (7,7) (11,10) (8,11).

INTERDATA 7/16 SSNM-OPEN

Everything is the same as SSNM-CREATE on the Interdata 8/32 except for the (X,Y) pair values in step 2 of the processing steps. For the OPEN command on the Interdata 7/16 the pair values are (0,0) (1,1) (2,2) (3,3) (4,4) (6,6) (7,7) (10,8) (11,10) (8,11).

INTERDATA 7/16 SSNM-CLOSE

Same as SSNM-CLOSE on the Interdata 8/32.

INTERDATA 7/16 SSNM-DELETE

Everything is the same as SSNM-CREATE on the Interdata 8/32 except for the (X,Y) pair values in step 2 of the processing steps. For the DELETE command on the Interdata 7/16 they are (0,0) (1,1) (3,3) (4,4) (6,6) (6,7) (7,9) (8,11) (11,10).

INTERDATA 7/16 SSNM-I/O

Same as the SSNM-I/O on the Interdata 8/32.

STANDARDIZATION OF BASIC FILE MANAGEMENT SERVICES  
AND I/O DATA TRANSFERS FOR HETEROGENEOUS MINI  
COMPUTER NETWORKS

by

DOLAN M. MCKELVY

B. S., UNITED STATES AIR FORCE ACADEMY, 1971

---

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY  
Manhattan, Kansas

1976

The tremendous interest and potential in mini computer networks becomes more and more evident with the continually increasing processing capabilities and decreasing cost of the smaller more compact mini computers. The large number and variation of computer architectures make heterogeneous mini computer networks very desirable.

This report is an initial attempt to analyze the File Management and I/O commands of several mini computers with real-time operating systems. Sharing of resources between heterogeneous and homogeneous mini computers is a primary goal. It is accomplished by integrating the parameters of File Management and I/O commands available on several mini computers into network standard control blocks. Transformation tables are then generated to map network standard command parameters with machine dependent command parameters. These transformations and top down modular structured programming techniques are used to explicitly define machine independent and machine dependent modules within the network conversion process. Overall conceptual correctness of the modular design and transformation tables is shown by hand simulating the network conversion process for several commands.

Machine incompatibilities are isolated and possible emulation techniques discussed where applicable. Future demands for presently incompatible heterogeneous File Management and I/O capabilities will determine where additional research is required.