SU-326-P.39-29

DESIGN DESCRIPTION OF THE NOVA 3 CARTRIDGE DISK EMULATOR ON THE STANFORD EMMY SYSTEM

by

Daniel R. Hafeman

June 1978

TECHNICAL NOTE NO. 140

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

The work described herein was supported in part by the Department of Energy under Contract No. EY-76-S-03-0326-PA 39.

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

Technical Note No. 140

June 1978

DESIGN DESCRIPTION OF THE NOVA 3 CARTRIDGE DISK EMULATOR ON THE STANFORD EMMY SYSTEM

by
Daniel R. Hafeman

ABSTRACT

This paper describes the emulation of a Nova 4047A Cartridge Disk System on the Stanford Emmy. The Cartridge Disk Emulator (CDE) serves as a peripheral to a Nova 3 processor emulator, also installed on the Emmy. With a disk system, DG disk based operating systems can be installed, allowing a wide range of programs to be studied.

The Cartridge Disk Emulator uses the Mini UNIX system to provide I/O services and manage the physical disk resources. CDE transforms Nova I/O commands into UNIX file access requests and processes all status resulting from those requests.

All other peripheral emulators currently installed on Novaem are small simple programs that may easily be understood from the program listing. CDE, however, is in itself a very complex program. It is comparable in complexity and code space, to the entire Nova Code Emulation Program. Therefore, a document dedicated to its description is required.

Appendix A of this document discusses how to add a new peripheral to NovaemU and NovaemCD.

The work described herein was supported in part by the Department of Energy under Contract No. EY-76-S-03-0326-PA 39.

1.Ø INTRODUCTION

This paper describes the emulation of a Nova 4Ø47A cartridge Disk System on the Stanford Emmy. The cartridge Disk Emulator (CDE) serves as a peripheral to a Nova 3 processor emulator, also installed on the Emmy. With a disk system, DG disk based operating systems can be installed, allowing a wide range of programs to be studied.

The cartridge Disk Emulator uses the Mini UNIX system to provide I/O services and manage the physical disk resources. CDE transforms Nova I/O commands into UNIX file access requests and processes all status resulting from those requests.

All other peripheral emulators currently installed on Novaem are small simple programs that may easily be understood from the program listing. CDE, however, is in itself a very complex program. It is comparable in complexity and code space, to the entire Nova Code Emulation Program. Therefore, a document dedicated to its description is required.

Appendix A of this document discusses how to add a new peripheral to NovaemU and NovaemCD.

2.0 FUNCTIONAL DESCRIPTION OF CDE

CDE attempts a class A emulation of the DG 4047A cartridge disk system. Obviously, complete emulation of a disk system is impossible especially when dealing with timing and error processing functions. It is hoped, however, that CDE presents an interface enough like a real 4047A so that Disk operating system software will run without modification. However, there are functional differences and these will be discussed.

2.1 Capacity

CDE can support up to four drives with 1,247,232 word of storage per drive. Each Drive corresponds to a UNIX device file as shown.

Drive	0	UNIX Device #	270
	1		271
	2		272
	3		273

The ready status of each drive is maintained by UNIX via the device ready and Device not ready commands. In UNIX, a device ready command is issued to CDE whenever the user assigns a file to the device number, and a device not ready command generated when the file is closed. When CDE is first loaded, it presets Drive 0 ready and Drives 1-3 not ready. The ready bits are available to Nova code via RegA status and thus can be (and probably is) used by the Nova disk driver to determine capacity of the system.

2.2 Functions

The DIA-DOC instructions are executed as described in the Peripherals Reference Manual and need no further discussion here. These instructions, with special function code zero, do not cause any CDE-UNIX communication. They only update the A-C registers.

The special functions, START, CLEAR, and IOPULSE originate all I/O and accordingly affect CDE-UNIX communication.

START - (Read/Write Command). Using the sector and side information in RegC, and the cylinder number from the specified drive status word, a UNIX byte address is formed. This address calculation algorithm assumes that the first word on sector zero, side zero, cylinder zero corresponds to the first word in the UNIX device file assigned to the drive.

Next, a seek command is issued on the drive specified in RegC followed by on or two read/write commands. Two commands are issued if the read/write buffer in Nova memory overflows; that is, if it wraps around through zero. The first command processes the buffer from the starting address to 77777 and the second processes the remainder of the buffer starting at Nova address 0. This is required to emulate the module 32K memory address counter in the 4047A.

A read command is issued if the command field in RegA =2, a write command if it =3, and a NOP results if it =0, or 1 (Note, the Peripherals Reference Manual does not discuss the response of the 4047A when a Nova Start Function is executed and the command field specifies a seek).

All Status is updated as defined for the 4047A.

When the command completes, the drive's Status word and Registers A-C are updated to their final values as specified. Notice that the MAR (RegB) and the sector count fields are updated in one step at completion (In a real 4047 these fields are updated continuously during the operation). A Nova interrupt is issued as specified.

IOPULSE - (Seek Command). Using the cylinder and command fields in RegA, a UNIX byte address is calculated. If the command field does not specifiy a seek or recalibrate command, then the IOPLSE is treated as a NOP. Otherwise, a seek command is issued on the drive specified in RegC. The drive ready flag is cleared, the seeking flag set, and the cylinder field in the device status word updated from the equivalent field in RegA.

At completion of the operation, the drive is returned to ready state and the seeking flag cleared. A Nova interrupt is issued as specified.

> Note: The IOPULSE function could have been handled as a NOP since a UNIX seek command is required prior to any read/write operation. It was implemented, however, to achieve timing characteristics at least similar to a real disk system. Since the UNIX disk devices map onto a real disk drive, then real time seek control might improve Nova operating system performance.

CLEAR - (reset all CDE I/O). All disk operations are aborted using the UNIX abort(device) command. When UNIX has acknowledged the abort, CDE can safely initialize the drive status words and clear any Nova interrupt.

2.3 Status

The real 4047A maintains a 7 bit status field for each drive. In addition, the current cylinder is implicitly known by the position of the head. CDE keeps equivalent information on a drive in the status vector, CDS, (see Section 3, figure 3-1).

As stated earlier, the drive's cylinder number is preset to the value specified in the last DOA instruction when a seek command (IOPULSE) is performed. The cylinder number, unlike most other variables, is not cleared during a reset command. It can only be modified by the IOPULSE special function.

The 7 bit status field is now defined for CDE. The names of the indiviual bits are consistent with DG terminology.

Drive Ready (bit 9). At load time this bit is preset to 1 on drive zero and to 0 for drives 1-3. Thereafter it is cleared when:

1) A device not ready UNIX command specifying this drive is received.

2) A Nova seek command (IOPLSE) is successfully issued on this drive.

The bit is set (drive goes ready) when:

1) A device ready UNIX command specifying this drive is received.

2) A Nova seek command completes for any reason (successful or not).

CDE will not issue any command except abort(device) to UNIX on a drive that is not ready.

Seek Error (bit 10). A seek error occurs when an out of bounds cylinder number is specified in a Nova seek command, or when UNIX returns an error on a seek finish.

End Error (bit 11). The error occurs when the sector count specifies that a read/write operation should continue beyond the last surface of the disk cartridge. This error is actually detected by CDE when the Nova read/write command is issued. CDE records the error, modifies the calculated buffer length count, and then issues the correct command to UNIX. Thus, the portion of the buffer on the valid cylinder is processed as in the real 4047. When the read/write finish command is received, CDE sets the end error in the Drive's Status word.

The end error is also set if the sector number is out of bounds when the command is issued. In this case, CDE aborts the instruction not initiating any UNIX I/O. In a real 4047, an illegal sector causes an infinite sector search that can be cleared only by an IORST. It was just too painful to emulate such a bug.

Unsafe (Bit 12). It never happens in CDE.

Check Error (Bit 13). This bit is set when an unsuccessful read/write finish command is received. Thus, any error that UNIX encounters in processing a read/write command, causes a check error.

Data Late (Bit 14). It never happens in CDE.

Error (Bit 15). Error <- 1 whenever any of bits 10-14 are set or when a read/write command is issued on a not ready drive. Note, the second condition is not specified for a real 4047A. The DG documentation doesn't define the results of such an access.

3.0 DATA STRUCTURE

CDE, like all other peripherals, uses the I/O table described in "The Design of a Nova 3 Emulator On the Stanford Emmy", as the interface to the code emulator and to the CDE u-interrupt handlers. In addition, a four word drive status vector not in the I/O table, is allocated to CDE. This vector is accessible only to CDE and maintained by the CDE code emulator and u-interrupt handlers. Figure 3-1 defines these variables and their bit assignments.

3.1 Drive Status

Each of the four drives owns an entry in the four word drive status vector (CDS) and indexes that word with its ordinal number. Complete status on a drive, including any outstanding UNIX I/O activity, resides in its status word. All status could have been compacted into the I/O table, nice from an overal NovaemCD point of view, but intolerable from and accessiblity standpoint. The current structure allows fast indexing of status. The CDE record in the I/O table cannot be increased in size to include CDS as it would prohibit the homogeneous treatment of I/O devices by Novaem when executing global Nova I/O instructions, like IORST.

Referencing figure 3-1, bits 0-6 of the drive status word correspond to the drive error bits defined in section 2. Bits 16-23 define the current cylinder number. This field specifies the position of the emulated head for the drive, and is used in UNIX seek address calculation. The field is updated during execution of a Nova seek command (IOPLSE).

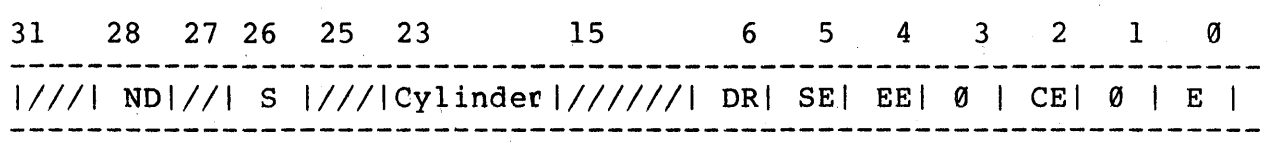
Bit 26 is the seek flag, and is set by the Nova IOPLSE command. It is cleared by the u-interrupt handler when the seek completes,i.e. when UNIX sends seek complete mail. The u-interrupt handler uses this bit to determine the cause of a seek command. If the seek was issued during execution of an IOPLSE, then the CDE seek done flag (see section 3.2) is set and a Nova Interrupt issued. Otherwise, caused by a Nova read/write command, only the drive status is updated. This Seek complete is only the first of two or three UNIX responses that must occur before the CDE Done flag can be set.

The seeking flag is also used by the seek clear routine, P24SKAB, to determine if an outstanding UNIX seek command exists on the drive. If one does then a UNIX abort command must be generated to clear the seek.

Bit 27 is the read/write flag which is currently unused.

The no done flag, Bit 28, gets set whenever two UNIX read/write commands are generated during the execution of the Nova START function. Recall that two commands are required whenever the Read/write data buffer overflows, modulo 32K, through zero. When read/write finish mail is received from UNIX, the no done flag is tested. If it =1, then the CDE done flag cannot be set since another UNIX read/write finish response is required before the operation is

CDSn: $0 \leq n \leq 3$



Bits 0-6 define drive status:

E --Error.
CE --Check error; Set when UNIX has reported a read/write error on this device.
EE --End error; Set when the read/write buffer extends beyond the last cylinder, or an illegal sector has been specified in a read/write request.
SE --Seek error.
DR --Drive ready.

Cylinder --Defines the current position of the emulated head. It is used to calculate UNIX byte addresses on a read/write operation.

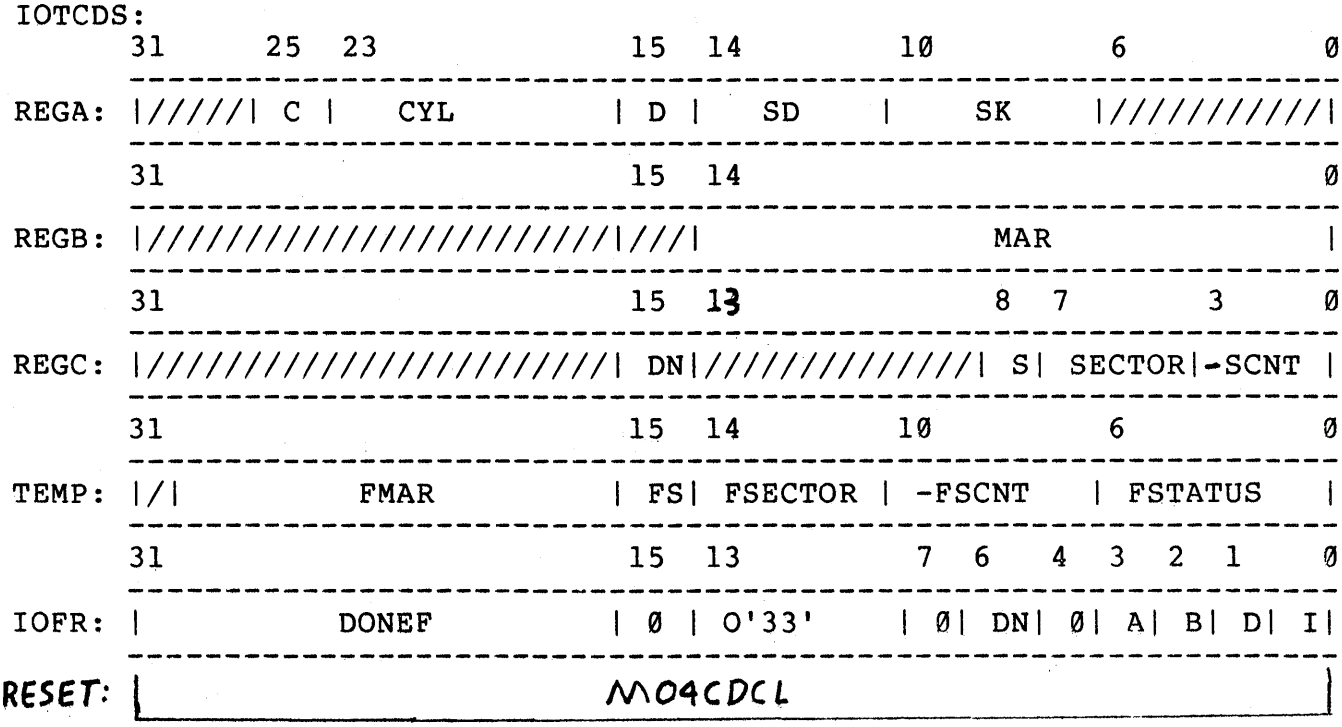
S --Seeking; A Seek is in progress on this drive. That is, a UNIX seek command, issued by the IOPLSE special function, hasn't completed yet.

ND --No Done; Two read/write UNIX commands are outstanding on this drive.

Drive Status Vector

FIGURE 3-1 CDE Data Structure (page 1 of 3)

IOTCDS:



SK --Seeking flags in DIA operand.

SD --Seek done flag in DIA operand.

D --Read/write done flag in DIA operand. It is a copy of D in IOFR.

CYL --Cylinder field in DOA operand.

C --Command field in DOA operand.

MAR --Memory address register.

-SCNT --Sector count field in DIC/DOC operand.

SECTOR --Sector # in DIC/DOC operand.

S --Side # in DIC/DOC operand.

DN --Selected drive # in DIC/DOC operand.

IOT Record
Figure 3-1 CDE Data Structure (page 2 of 3)

Register TEMP contains read/write completion values for all variables in Reg B and Reg C along with final status for the selected drive.

I --CDE Nova Interrupt; When I = 1 CDE has set IR and incremented ICNT.

D --Read/write Done.

B --Read/write Busy; When B =1 a read or write operation is in progress.

A --Abort; A=1 when a UNIX abort(device) command is in progress on one of the CDE devices.

DN --Ordinal number of drive last involved in a read or write operation.

DONEF --Nova I/O Bus done field. Bit 23 of DONEF <- 1 whenever any of the seek done flags in Reg A or the read/write done flag=1. It is cleared when all done flags = Ø.

RESET -- Termination word for IORST.

IOT Record
Figure 3-1 CDE Data Structure (Page 3 of 3)

complete. The no done flag is cleared. Otherwise, (no Done =Ø), the mail indicates that the data transfer is complete and that the CDE Done Flag should be set. In addition, the CDE busy flag is cleared and a Nova Interrupt issued.

3.2 The CDE I/O Record

The cartridge disk device registers reside in the first three words of CDE's I/O table record.

Registers B, and C are simple 16 bit bidirectional registers and their bit assignments are consistent with the specification in the Peripherals Reference Manual. Bits 16-31 of these registers are unused and may assume any value.

Register A has a more complex representation for two reasons:

1) The format for RegA on a DOA (output) instruction differs from that of the DIA (input) instruction.

2) Bits Ø-6 of the DIA operand contain the status of the selected drive; that drive defined in RegC.

For these reasons, the information stored in RegA does not match the format defined by the DIA and DOA instructions, adding some complication to the DIA and DOA execution routines.

Bits Ø-6 of Reg A are unused. On a DIA instruction, bits Ø-6 of the operand are fetched from the selected drive's status word.

Bits 7-15 contain the seeking flags, the seek done flags, and the read/write done flag. These are used in DIA operand formation. A seeking flag is set by the IOPLSE execution routine . It is cleared by the u-interrupt handlers and clear routines. A seek done flag is set by the u-interrupt handler when seek complete mail is received on a seek that was requested by the IOPLSE execution routine. It is cleared by the DOA execution routine and the clear routines. The read/write done flag is a copy of the CDE done flag in IOFR. It is maintained by the Start, u-interrupt, and clear routines.

Bits 16-23 contain the cylinder field of the DOA operand. It is used only by the IOPLSE execution unit to update the selected drive's status word and to form the byte address parameter in the UNIX seek command.

Bits 24 and 25 represent the command field of the last DOA operand, and are used by the IOPLSE and START execution units.

Reg Temp provides temporary storage required by the execution of the START command. The START execution unit, MØ4CD1ØØ, calculates final values for the read/write variables contained in Regs B and C, and generates final drive status. This information is stored in Reg Temp and remains there until the UNIX I/O completes, at which time the read/write u-interrupt handler uses it to form the final parameters in Registers B and C, and to form final drive status.

IOFR is identical to the Nova Bus status words of all other peripherals. However, there are some unique characteristics:

1) Bits 5 and 6 of this word contain the ordinal number of the drive last used in a read/write command. Thus, if a read/write operation is in progress, bits 5-13 of IOFR will define the UNIX device number involved. This infomation is used by the read/write clear procedure, P21RWC, to abort a data transfer operation.

2) Unlike all other peripherals, a Nova interrupt request (bit 23=1) may be pending, even though done =Ø. This is so because the done flag in CDE represents only read/write done status. There are, in addition, the four seek done flags found in RegA. An interrupt request is generated whenever any of the seek or read/write done flags =1, and the request cleared only when they all =Ø.

The last word in CDE's record is a pointer to the CDE power on reset procedure, MØ4CDCL, to be used by the IORST execution routine when intializing CDE.

4.Ø ROUTINES

Table 4-1 lists all routines used to implement the CDE interface. Like other devices, the cartridge disk I/O instructions are parsed in two steps.

1) The opcode field is decoded and all register transfers performed.

2) If the opcode did not specifiy a skip, then the special function field is decoded to determine what I/O, if any, is to be performed. If the opcode specifies a skip, an exit to the global skip routine, MØ4SKP, occurs.

The CDE u-interrupt handlers (I127), process all UNIX-to-Emmy mail directed to the disk drives, devices O'33Ø'--O'333'.

CDE was designed with readability and reliability as the Key design goals. Speed was hardly a consideration because the I/O instructions should be executed relatively infrequently. Since CDE was designed to incomplete documentation, some functional changes will probably be required before a disk based operating system can be reliably supported. Therefore, it is essential that CDE be well structured and easy to understand. The author sincerely hopes that CDE has met its design goals.

CDE uses 455 words of control store. This includes the CDE execution routines and u-interrupt Handlers.

ROUTINES

NAME	DESCRIPTION
M04CDS	CDE instruction decode routine.
M04CD20--70	DIA-DOC instruction execution routines.
M04CD10	CDE special function decode routine.
M04CD100	START special function execution routine; initiates UNIX read/write and seek commands.
M04CD200	CLEAR special function execution routine, issues UNIX abort(device) command.
M04CD300	IOPLSE special function execution routine; initiates UNIX seek command.
M04CDCL	CDE clear procedure used by P04IORST.
M04CDNR M04CDEE M04CDSE	Error exit routines used by M04CD100-300.
I127CDS	CDE u-interrupt handlers.

Table 4-1. CDE Routines and Procedures.

PROCEDURES

NAME	DESCRIPTION
P21RWC	Clears any read/write command currently in execution on UNIX.
P22ICL	Removes the CDE interrupt request from the Nova I/O Bus.
P23MUL	32 bit multiply routine
P24SKAB	Clears the specified drive's status word and aborts all UNIX I/O on the device. Issues UNIX abort(device) command.
P25SKD	Sets the specified seek done flag and issues a Nova interrupt.
P25RWD	Sets the read/write done flag and issues a Nova Interrupt.

Table 4-1. CDE Routines and Procedures - Continued

(PAGE 2 OF 2)

5.0 STATUS OF THE PROJECT

Unfortunately, time did not allow adequate verification of the storage system. The NovaemCD Emulator, however, contains the complete CDE device and interrupt routines, and simple Nova I/O programs have been run to verify at least functional integrity.

Because CDE cannot perform real time emulation of the 4047A, Nova disk diagnostic programs will probably not be usable. This is only conjecture, however, as no attempt has been made to run such programs.

6.0 CONCLUSIONS

CDE turned out to be a very difficult program to develop, largely due to the complexity of the 4047 hardware interface. It is clear that the 4047A evolved under severe hardware and compatability constraints resulting in a very unstructured and difficult to document interface.

90% of the 4047A function could have been emulated with a much simpler program. However, the value of such an emulator is questionable since the disk driver in the operating system will have to be rewritten to compensate for the other 10% of the function. But, if driver rewrite is required, then a single disk I/O emulation package encompassing both the driver and the disk system is a better solution. The new driver for this system would contain a single currently undefined Nova instruction which is recognized by the Emmy code emulator as a request to execute the I/O emulator. Thus, the emulation interface resides at the I/O driver call level instead of at the hardware level. But, this level of interface, in a typical minicomputer operating system, is even less well defined and documented, and may indeed be more complex, requiring the emulator writer to have intimate knowledge of the operating system primitives.

CDE attempts to perform non real time class A emulation of the hardware interface with the goal of supporting any DG 4047A driver. If sucessful, then any of Data General's commercial systems can be easily installed on NovaemCD.

APPENDIX A. INSTALLING A NOVA I/O HANDLER ON NOVAEMU OR NOVAEMCD

Installing a new I/O emulator and Novaem is a well defined task and requires only minor modification to the current code emulator. The new device will contain two programs; the device code emulator, and the u-interrupt handler.

To install the device, the following modifications to Novaem must be made:

1) The starting address of the code emulator must be entered in the device code jump tables at M01IO and at I100UNIX. Its location, in the tables, is determined by its device code.

2) An I/O record must be created for the device in the I/O table (IOT). Its location in this table is arbitrary but must follow IOTZERO and precede IOTCPU. The location determines its priority on the emulated I/O Bus.

The I/O record must be six words in length. The first four are totally specified by te I/O Emulator design. The fifth word "IOFR" must adhere to the format defined in "The Design of a Nova 3 Emulator on The Stanford Emmy" since it is used by the the global I/O instructions. The sixth word must contain the address of the new device's reset routine. If the device has a simple I/O structure, then procedure P04CL can be used.

3) The number of devices variable, NUM_OF_DEV, must be incremented to reflect the new entry.

4) Declarations defining the device code and mask should be entered in the I/O device construction table.

The designer must strictly adhere to the busy/done protocols defined for the emulated Nova I/O Bus. If the device is simple, then he may use P01CLEAR and P02DONE to assist in the management of the busy, done, and interrupt flags.

REFERENCES

1) GENACC Program Listing
Digital Systems Lab, Stanford Electronics Lab
Stanford University, March 1978

2) Charles Neuhauser
"An Emmy Based PDP11/20 Emulation"
Digital Systems Lab, Stanford Electronics Lab
Stanford University, March 1977

3) Charles Neuhauser
"Emmy System Processor -- Principles of Operation"
Digital Systems Lab, Stanford Electronics Lab
Stanford University, May 1977

4) Lee W. Hoevel
"Deltran: Principles of Operation. A directly Executed Language For Fortran II"
Digital Systems Lab, Dept. of Electrical Engineering and Computer Science
Stanford University, March 1977

5) Digital Equipment Corp
"PDP11/03 Processor Handbook"
Copyright 1975

6) Data General Corporation
"Programmer's Reference Manual--Nova Line Computers"
Doc # 015-000023-04
Copyright August 1976

7) Data General Corporation
"Programmer's Reference Manual--Peripherals"
Doc # 015-000021-00 Rev. 00
Copyright November 1974

8) Data General Corporation
"Exerciser for NOVA 3"
Doc # 096-000363-02
Copyright 1976