ORNL/TM--8106

DE83 005152

ORNL/TM-8106
Dist. Category UC-20

Contract No. W-7405-eng-26

FUSION ENERGY DIVISION

A COMMUNICATIONS INTERFACE FOR THE PDP-11 AND THE NUCLEAR DATA 6600

K. G. Young, D. E. Greenwood, and R. D. Burris

Date Published - December 1982

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37830
operated by
UNION CARBIDE CORPORATION
for the
DEPARTMENT OF ENERGY

CONTENTS

ABSTRACT

A communications interface has been designed to send data from the Nuclear Data 6600 (ND6600), a computer-based multichannel analyzer, to a PDP-11/34, operating under RSX-11M. The hardware consists of a Nuclear Data Serial Communications Processor (SCP) and a CAMAC Kinetics Systems 3340 communications interface.

On the PDP-11, Digital Data Communications Message Protocol (DDCMP) software was written to check and reply to each data buffer. The PDP-11 code also saves the data on disk and interfaces with existing communications software. On the Nuclear Data, the DDCMP is contained in firmware on the SCP board. Software was written to retrieve data from disk and to interface with the system software that accesses the SCP board.

1. INTRODUCTION

Soft and hard x-ray data from the ELMO Bumpy Torus (EBT), a research device for studying the feasibility of nuclear fusion as a source of energy, are acquired, temporarily stored, and briefly analyzed on a Nuclear Data 6600 (ND6600). Long-term storage in the EBT data base and detailed analysis are done on a DECSystem-10. The ND6600-PDP-11 communications interface was developed to transfer data from the ND6600 to the PDP-11 and thus into the Fusion Energy Division network,[1-3] where it could be properly routed to its destination on the DECSystem-10.

This document was written to expedite the use of and possible future modification of the ND6600-PDP-11 communications system. It is directed to the experienced programmer, with the exception of Sect. 3, "Instructions for the User."

2. SPECIFICATIONS

It was required that the software for this interface be able to perform certain functions related to the organization, labelling, and transmission of the data. The organization and labelling have been discussed in detail elsewhere [2,4] and will be briefly mentioned here.

Information to be transferred from the ND6600 to the PDP-11 is organized into a label or header and the data. In the current case on the ND6600, the data consist of a 256-channel spectrum. The 65-character header contains the time, date, and machine condition sequence number (a pointer to data characterizing the operation of the EBT -- i.e., power, pressure, etc. -- acquired simultaneously on another computer). The header also contains other items, including a

diagnostic identifier and sequence number, which effectively becomes the name of the file on the PDP-10. For example, the first spectrum taken on the ND6600 would be A0001.NAX. (NAX is the identifier for x-ray data.)

For the communications hardware, a Kinetic Systems communications interface CAMAC module is used on the PDP-11. This is connected to a Nuclear Data Serial Communications Processor (SCP). The SCP consists of a microprocessor coupled with RS-232 or RS-422 I/O ports. The RS-232 port is used in this application.

Requirements for this system include the capability to:

(1) Send files from the ND6600 to the PDP-11 without errors.

(2) Send files from the PDP-11 to the ND6600 without errors.

(3) Get the time, date, and machine condition sequence number from the PDP-11 before acquiring data.

(4) Get the diagnostic sequence number from the PDP-11.

(5) Put the header on disk on the ND6600 with the data.

(6) Interface with existing general communications software.

(7) Perform all user control from the ND6600. The user need not be aware of the PDP-11. Therefore, the program to receive data from the ND6600 should always be available on the PDP-11.

(8) Send all data taken that day with a minimum of typing; i.e., one should not need to type the name of each spectrum.

(9) Mark all spectra on the ND6600 that have been sent to the PDP-11. There should be a resend option if one wishes to send a spectrum already sent.

3. INSTRUCTIONS FOR THE USER

Each time data are acquired on the ND6600, a program to request the machine condition sequence number from the PDP-11 must be run. This number is stored in a disk file on the ND6600. The user then acquires data. When the user records the data on disk, the machine condition sequence number is placed in the header that is recorded with the data.

When the user wishes to send the data to the PDP-10, he runs a program that will send one or many spectra. The user may also retrieve a file from the PDP-11. Detailed instructions for performing these operations are discussed below.

The user should be aware of the fact that there is a difference in the definition of file as used on the ND6600 and on the PDP-11. On the Nuclear Data, information is stored as elements which are members of files. For example, the file DATA may contain elements DATA.A01, DATA.TMP, etc. On the PDP-11, however, elements do not exist. Thus, on the PDP-11 each element is a file. That is, DATA.A01 would be one file and DATA.TMP would be another.

3.1 RETRIEVING THE MACHINE CONDITION SEQUENCE NUMBER

The user enters JOB AUTO.BC. This job stream gets the time, date, and machine condition sequence number from the PDP-11. It writes this information in the header and on the user's terminal. For example, type

```
JOB AUTO.BC
```

Within seconds, the line

Ans A000021-NOV-19?1 09:47:18

should appear on the ND6600 terminal. In this example, the machine condition sequence number is A0000 (default machine condition sequence number when no machine state is defined), the date is the 21st of November, 1981, and the time is 09:47:18.

3.2 TRANSMITTING FILES

To send files, first list the directory by typing FDI DATA,,SCR2. This puts the directory of the file DATA on SCR2 so the program can read it. Then, enter JOB AUTO.IGO. The program will ask how many spectra to send. Enter the desired number (integer format). The program then checks each spectrum to see if it has already been sent.

If so, it asks if the user wishes to resend. The program then sends all spectra requested. After each spectrum is sent, its name appears on the terminal. For example, to send two spectra from the file DATA, type

FDI DATA,,SCR2

JOB AUTO.IGO

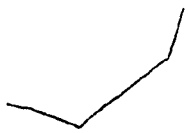
The program will ask

No. spec to send = #

(# is the prompt for input.) Enter the number of spectra that you wish to send. If DATA.A001 (the first spectrum in the file DATA) has been sent, the program asks

Do you wish to resend DATA.A001 ? y or n #

If you wish to resend, type Y for yes:



Y

When transmission of one spectrum is completed, the line

DATA.A001 transmitted

will appear on the terminal.

3.3 RETRIEVING FILES

To get a file from the PDP-11, run the job stream AUTO.IGO. Enter 0 in response to the number of spectra to send. The program will then ask for the name of the file on the PDP-11. When it gets the file it will place it in the file SOURCE.GRABIT and will indicate completion on the user's terminal. For example, to get the file DUMY.FTN from the PDP-11, type

JOB AUTO.IGO

The program will respond

No. spec to send = #

Enter

0

The program will ask

Name of file on PDP-11 #

Enter

DUMY.FTN

When the program has accomplished this, it will type

DUMY.FTN received.

4. PROGRAMMER'S DOCUMENTATION

In this section, the protocol, hardware, initialization, and software are discussed.

On the Nuclear Data the programs used are APS.TTYDEF, APS.INISCP, LD.MTALK, and LD.ITALK. (APS.TTYDEF and APS.INISCP were supplied by Nuclear Data.) The job streams (command files) used are AUTO.BC, AUTO.IGO, and AUTO.SKP. These are discussed in detail in the following sections. The table below shows the programs contained in each job stream:

JOB STREAM:	PROGRAMS:
AUTO.BC	Iniscp, Mtalk
AUTO.IGO	Iniscp, Italk
AUTO.SKP	Iniscp

On the PDP-11, a command file is run which installs the communications program. Once the program on the PDP-11 is installed, all user input is at the ND6600 terminal.

4.1 PROTOCOL

This system has a maximum buffer size of 128 bytes and 3 levels of protocol. The buffer size is dictated by the CAMAC communications module (see section on hardware for more information), which can handle a maximum of 128 bytes. A diagram of the message format showing the three levels of protocols is shown below:

/-- DATA --/						
DDCMP HEADER	CRC	NDCMP	INFORMATION	IBC	FN CODE	CRC
6 bytes	2 bytes	4 bytes	1-96 bytes	2 bytes	2 bytes	2 bytes

The first level of protocol is Digital Data Communications Message Protocol (DDCMP). The second and third levels are included in the data portion of a message.

The second level is Nuclear Data DDCMP (NDCMP), which is present but not used in the current application. The SCP automatically supplies values at the ND6600 and is supplied with innocuous values by the program on the PDP-11.

The third level was established to transmit an internal byte count (IBC) and a function code. Since it is simpler and less error prone to use a fixed size buffer, the message contains an internal byte count telling how many of the bytes sent are actually valid. The function code tells the PDP-11 what to do or expect, i.e.,

REQUEST	CODE
More buffers of this spectrum to come	0
End of one spectrum	1
End of transmission for now	3
Send diagnostic sequence number	10
Send file from PDP-11	11
Send machine condition sequence number	12
Buffer contains command string	21
Buffer contains data base header	22
Buffer contains comment field	23

In sending a spectrum to the PDP-11, the command string is sent first, followed by the data base header, the comment field, and the data itself. The command string tells the communications program on the PDP-11 where to send the spectrum. For example,

```
FED  EBTA:A0001.NAX[210,2740]=NDNAM/DB
```

where NDNAM is the name of the spectrum on the ND6600. The 65-character data base header is described in Ref. 4. A 60-character comment field input by the user is sent in the next message, followed by 11 messages containing spectral data. Following each message the SCP board outputs a seven-byte trailer and, usually, an eight-byte REP followed by a seven-byte trailer. A detailed discussion of DDCMP is located in Appendix A.

4.2 HARDWARE

The SCP communications board on the ND6600 has RS-232 or RS-422 compatible drivers and receivers, is capable of full duplex asynchronous operation, and supports various communication protocols, including DDCMP. The SCP board handles the DDCMP protocol transparently to the user via its own processor, which executes instructions directly from main memory. On the PDP-11 a Kinetic Systems CAMAC communications interface module #3340 is used. This module interfaces the CAMAC Dataway directly to terminals, modems, printers, etc., and uses a 128-character FIFO buffer for input and output.

4.3 INITIALIZATION

Before running the communications program from the Nuclear Data, one must set the terminal characteristics and initialize the SCP board.

4.3.1 Terminal Characteristics

The terminal characteristics have been selected and set for communications with the PDP-11 and should not need to be changed or redone. These characteristics are stored in the bootstrap section of the disk. They will only need to be reset if the contents of the disk are destroyed or if one installs a new operating system.

The terminal characteristics can be reset by running an editor program (APS.TTYDEF), placing the results in the bootstrap section of disk, and restarting the system. To do this, the following five steps should be followed (for a detailed explanation of the system commands, see Ref. 5):

(1) Sign on as the control operator by typing control C and define the logical units by typing:

```
DEF 5,TTY(operator input)
DEF 6,TTY(listing device)
DEF 7,TTY(prompts and messages for operator)
DEF 9,AUTO.TTABLE(input file)
DEF 10,SCR1(output file)
```

The file assigned to logical unit 10 must be at least four sectors in length. The data type of this file must be 11. To check these, type

FDI SCR1

The following information will appear:

NAME	CURR START	CURR END	BYTES	SIZE	PROT	OWN	REC SIZE	DT	DATE
SCR1	280	288	136	100		255	0	1	30Nov78

The field SIZE indicates that this file is 100 sectors in length. This should not need to be changed, since SCR1 is always kept at 100 sectors. The field DT indicates that the data type is 1. To change this, type

ALT SCR1,DT=11

(2) Run the program TTYDEF

R TTYDEF

The program will respond

Type RX for "menu".

Input command #

The menu will list all the commands TTYDEF can execute. For the current application, we will want to read in the terminal characteristics already stored on disk, select the terminal

number, and examine or modify the terminal characteristics. For example, type

RD

to read in the terminal characteristics. Next type

TN

The program will reply

Input terminal number (1-16) 1 #

The terminal we are using for the communications is 7, so respond

7

Then to list the characteristics, type

ET

A table showing the terminal characteristics and their current values will appear.

Code	Terminal characteristic	Value
TN	Terminal number	7
BD	Baud	9600
OW	Output width	0
NU	Nulls after	0
PN	Port number	9
BT	Board type	S
CT	Convert tab to spaces	N
AF	Allow formfeed	N
AL	Allow lower case	N
RE	Rubout echo mode	R
IC	Input conversion	N
OC	Output conversion	N
SE	Suppress echo	N
BI	Binary input	N
IT	Input transparent	N
OT	Output transparent	N
DO	Disable output	N
PI	Preload input buffer	N

To change any of the characteristics, enter the two letter code, wait for the prompt, and enter the desired value. For example, to change the baud rate, type

BD

The program will prompt

Baud(110-9600) 9600 #

If you wish to use 2400 baud, type

2400

The program will then type

Input command #

and wait for your next response. If you wish to exit the program, type

EX

It is not necessary to read a previous file from disk. It is simply easier to modify a file than to enter each characteristic one by one.

(3) Type

MOV SCR1,,TT

to put the table in the bootstrap section of disk.

(4) Reload the system:

Hit the keys BREAK and 0 together

Hit the keys BREAK and 1 together

Hit the key labelled AUTO.

(5) To save a copy of the terminal characteristics for future reference or modification, type

REP SCR1,AUTO.TTABLE

For more information, see Ref. 6.

4.3.2 Initialization of the SCP Board

The SCP board is initialized every time a communications program is run. This initialization starts the link, and once the data are sent the link is cancelled. The link is not maintained because it uses memory that is needed for data acquisition. The initialization is done by typing

JOB AUTO.SKP

This will cause a command file to be executed. The command file contains the following:

```
R INISCP
1
DD
9600
N
```

```
2
7
7
ENDJOB
```

The file runs the program APS.INISCP, taking the following seven lines as input data. These values are defined in the table below.

Items for initialization of SCP board	Value
Input channel number	1
Input mode	DD
Input baud rate	9600
Input parity	N
Input number of stop bits	2
Read acknowledge threshold	7
Write acknowledge threshold	7

The program then issues a START every 3 seconds until a START or STACK (start acknowledge) is received from the PDP-11. The PDP-11 program, installed and checkpointable,[7] waits for input. When it receives the START, it sends a STACK. The link is now established. To cancel the link, type

```
CAN SCP
```

4.3.3 Initialization of the PDP-11

To set up on the PDP-11, install the communications program END... which issues a START and then waits for input. This setup is automatically done when the system is booted up. To start the program manually, type @[7,37]END. (Note that the program should be started at the console device, since the program END... will be aborted if the terminal from which it is run is signed off.) END is a command file containing the following:

```
ENABLE QUIET
IFACT END...ABO END...
IFINS END...REM END...
INSCCEND/TASK =END
RUNEND
INS[7,37]CCI
CCIEND/ND6600,3;+ND6
REM...CCI
```

4.4 SOFTWARE

4.4.1 The ND6600

To use the communications on the ND6600, one first starts the link with the initialization program APS.INISCP described above. To use the communications link, the programmer sets up buffers of data and then calls the appropriate ND6600 subroutines (i.e., CCREAD and CCWRIT). The mechanics of the DDCMP protocol (CRC calculations, error checking, DDCMP header, etc.) are handled transparently by the SCP board. This has several advantages — it is fast and the code for the protocol does

not have to be written. Unfortunately, the programmer has no control over the board. He cannot modify it to make it compatible with another system nor can he control the speed of the protocol response. The major part of the coding on the ND6600 involved fulfilling the requirements outlined in Sect. 2.

On the Nuclear Data, the programs used are APS.TTYDEF, APS.INISCP, LD.ITALK, and LD.MTALK, and the job streams used are AUTO.SKP, AUTO.IGO, and AUTO.BC. TTYDEF and INISCP are initialization routines obtained from Nuclear Data. The program ITALK was written to send data to the DECSystem 10 via the PDP-11 and to receive files from the PDP-11. The program MTALK is a version of ITALK that does not require any user input; it simply gets the machine condition information from the PDP-11 and stores it on disk in the header file. The subroutines used in ITALK and MTALK are described below.

(1) GCOM -- contains control logic and initialization

(2) MAING -- sets up buffers for I/O for the communications subroutines CCWRIT and CCREAD

(3) SHED -- reads header from disk

(4) SDAT -- reads data from disk

(5) LE.Y -- defines the logical units

(6) DIRK -- reads the directory from SCR2

(7) MOHED -- writes the time, date, and machine condition sequence number in the header

The job stream AUTO.SKP initializes the SCP board as described in the section on initialization. The user does not need to run this job stream explicitly, but the programmer may wish to use it for testing purposes. The job stream AUTO.IGO initializes the link, defines the appropriate logical units, runs the program ITALK, and cancels the link. This job stream contains the following:

```
R INISCP
1
DD
9600
N
1
7
7
DEF 5 TTY
DEF 6 TTY
DEF 7 TTY
RUN LD.ITALK

CAN SCP

ENDJOB
```

The parameters in this job stream are discussed in Sect. 4.3, "Initialization." The program ITALK will ask the user for input (see Sect. 3, "Instructions for the User"). The job stream AUTO.BC is the

same as AUTO.IGO except for one line — RUN LD.ITALK becomes RUN LD.MTALK.

4.4.2 The PDP-11

On the PDP-11, in addition to fulfilling the requirements in Sect. 2, the DDCMP protocol had to be provided. A standard CRC calculation subroutine was used, but all the error checking and protocol handling were coded in a compatible manner with the ND6600.

The program on the PDP-11, CCEND, uses the general CAMAC communications modules CCPARM, CCROOT, CCPROC, CCFCN, and HEADER. The following subroutines were written to receive data from the ND6600 and to pass it to these communications modules.

(1) CNTRL -- reads the function code (see Sect. 4.1, "Protocol") and calls appropriate subroutines

(2) KRC16 -- calculates CRC

(3) INIT -- performs initialization

(4) TALK -- sets up link to ND

(5) GETB -- receives and checks data from ND

(6) SEND -- sends data to ND

(7) LGREAD -- reads 128 bytes from the CAMAC module 3340

(8) SMREAD -- reads 15 bytes from the CAMAC module 3340

(9) SMBYE -- sends 15 bytes to the ND

(10) LGBYE -- sends 128 bytes to the ND

(11) ORDER -- checks bytes for proper order

(12) RDDK -- reads files from disk on the PDP-11

(13) QIOA -- contains the CAMAC I/O code

(14) STORE -- contains arrays for data and messages

5.0 CONCLUSION

The communication interface described above is used to get the machine condition sequence number from the PDP-11, to send data from the ND6600 to the DECSystem 10 via the PDP-11, and to send files from the PDP-11 to the ND6600. All user input is at the ND6600. DDCMP software at the PDP-11 end and DDCMP firmware at the ND6600 end are used to provide error-free transmission.

ACKNOWLEDGEMENTS

Communications between two dissimilar systems was facilitated by discussions with experienced personnel from both systems. We are indebted on the one end to Bob Schlossman and Gregg Dotty of Nuclear Data for patiently explaining the inner workings of the Nuclear Data system and on the other end to Chuck Kemper of the Fusion Energy Division who contributed his knowledge of the PDP-11 system. The authors express their appreciation to Janice Hughes for the preparation of this document. This project was funded by the Computer Support Group of the Fusion Energy Division.

Appendix A: DIGITAL DATA COMMUNICATIONS MESSAGE PROTOCOL

The purpose of this appendix is to define some fundamental concepts in communications; in particular, the definition of protocol (DDCMP) and its components will be given. This appendix is based on information obtained from Refs. 6 and 8.

A protocol is a set of rules for operating a communications system. The rules are designed to solve operating problems in the following areas: framing, error control, sequence control, transparency, line control, timeout control, and startup control. This appendix will define DDCMP and point out how DDCMP handles the above problems.

Digital Data Communications Message Protocol is a very general protocol; it can be used on synchronous or asynchronous, half or full duplex, serial or parallel, and point-to-point or multipoint systems. This appendix will be concerned with DDCMP as applied to an asynchronous, full duplex, serial, point-to-point system.

In DDCMP there are three classes of messages: data, maintenance, and control. A data message consists of a six-byte header followed by a two-byte Cyclic Redundancy Check (CRC),[7] which is used for error checking, followed by the data itself with its two-byte CRC. A maintenance message is typically a bootstrap message that contains load programs in the data field. A control message consists of six bytes followed by the two-byte CRC. There are five types of control messages. The control messages are: START, STACK (start acknowledged), ACK (message received correctly), NAK (message not received correctly), and REP (tells transmitter last number of correctly received message).

The first eight bytes of a DDCMP message are shown in Table A.1. The first byte of the message is the class of message indicator. Octal values for class of message indicator are: Control - 5; Data - 201; and Maintenance - 220. The second and third bytes may be interpreted independently or together, depending on the situation. For control messages, the second byte designates the type of control message. If the message is a NAK, the first six bits of the third byte indicate the reason for the NAK. If the message is a data or maintenance message, the second byte and the first six bits of the third byte (14 bits) represent the number of characters that will follow the header and will form the data part of the message. The last two bits of the third byte are the flag and are set to 3 for asynchronous transmission.

The fourth byte contains the response field, which indicates the number of the last message correctly received. The fifth byte contains the sequence field, which is used in a data message for the sequence number of the message as designated by the transmitting station. In a REP message, it is used as part of the question: "Have you received all messages up through message number (specify) correctly?" In point-to-point operation, a station sends address 1 but ignores the address field on reception.

Message framing in DDCMP is accomplished by searching for the starting byte or byte pattern to locate the beginning of the message. If the message is a control message it will be eight bytes long — this defines the end of the message. If it is a data message, the character count indicates the end of the message.

Transparency — the ability to send control characters as part of the data — is accomplished by not recognizing a byte as a control character until the character count has been satisfied. Errors are detected by means of the CRCs. If correct, an ACK is sent - if not a NAK (negative acknowledgment and request for retransmission) is sent. Cyclic Redundancy Checks is an error detection scheme in which the check character is generated by taking the remainder after dividing all the serialized bits in a block of data by a predetermined binary number. (For CRC-16 used for eight-bit characters, the binary number is 11000000000000101). Sequence control is accomplished via the sequence field in DDCMP format. Startup is accomplished via the START and STACK messages. One system sends a START and waits for a STACK. If the STACK is received the link is established. If a START is received while waiting for a STACK, a STACK is sent in reply and the link is established.

If during the course of transmission, the transmitter does not receive any response in a given period of time, it will then send a REP with the message number of the message previously sent. The receiver may then respond with an ACK or NAK. If a NAK is the response the message will be retransmitted.

Line control is accomplished by use of the address field. In point-to-point operation this is effectively unnecessary.

TABLE A.1

DDCMP HEADER

MESSAGE TYPE	MESSAGE CLASS	COUNT	FLAG	RESPONSE	SEQUENCE	ADDRESS	CRC 1
	8 bits	14 bits	2 bits	8 bits	8 bits	8 bits	16 bits
	XXXXXXXX	XXXXXXXXXXXXXX	XX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXXXXXXXXXX
Data Messages	10000001	Character count	11	Resp.No.	Message#	00000001	
Acknowledgment	00000101	00000001000000	11	Resp.No.	00000000	00000001	
Negative Acknowledge	00000101	00000010------	11	00000000	00000000	00000001	
Reply Message	00000101	00000011000000	11	00000000	Last M.#	00000001	
Start Message	00000101	00000110000000	11	00000000	00000000	00000001	
Start Acknowledgment	00000101	00000111000000	11	00000000	00000000	00000001	
Maintenance Messages	10010000	Character count	11	00000000	00000000	00000001	

Reasons for negative acknowledge: (lower 6 bits of "COUNT")		
	BCC Header Error	000001
	BCC Data Error	000010
	REP Response	000011
	Buffer Unavailable	001000
	Receiver Overrun	001001
	Message Too Long	010000
	Header Format Error	010001

"Resp.No." refers to response number; this is the number of the last message received correctly.

"Message #" is the sequentially assigned number of the message. Numbers are assigned by the transmitting station module 256; i.e., message 000 follows 255.

"Last M.#" is the number of last message transmitted by the station.

FLAG (2 bits) is set to 11 for asynchronous transmission.

ADDRESS is always 1 for point-to-point transmission.

REFERENCES

1. C. E. Hammons, Fusion Energy Division Computer Systems Network, ORNL/TM-7296, Oak Ridge, Tennessee (December 1980).
2. R. D. Burris, D. E. Greenwood, J. S. Stanton, K. A. Geoffroy, EBT Data Acquisition and Analysis System, ORNL/TM-7464, Oak Ridge, Tennessee (October 1980).
3. R. D. Burris, Data Format Translation Routines, ORNL/CSD/TM-137, Oak Ridge, Tennessee (February 1981).
4. J. S. Stanton, ELMO Bumpy Torus Data Base, ORNL/CSD/TM-136, Oak Ridge, Tennessee (March 1981).
5. K. G. Young, The ND6600 Computer in Fusion Energy Research, ORNL/TM-8107, Oak Ridge, Tennessee (to be published).
6. ND6600/ND66 Communications Operational Instruction Documentation, Nuclear Data, Inc., Schaumberg, Illinois (July 1980).
7. Introduction to RSX-11M, Digital Equipment Corporation, Maynard, Massachusetts (1979).
8. J. E. McNamara, Technical Aspects of Data Communication, Digital Equipment Corporation, Bedford, Massachusetts (1977).