Preliminary

PDL
Operator's Guide

DEC-11-OPUGA-A-D

Preliminary

Order additional copies as directed on the Software Information page at the back of this document.

digital equipment corporation · maynard, massachusetts

First Printing, June 1975

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

The software described in this document is furnished to the purchaser under a license for use on a single computer system and can be copied (with inclusion of DIGITAL's copyright notice) only for use in such system, except as may otherwise be provided in writing by DIGITAL.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.

The HOW TO OBTAIN SOFTWARE INFORMATION page, located at the back of this document, explains the various services available to DIGITAL software users.

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

CDP	DIGITAL	INDAC	PS/8
COMPUTER LAB	DNC	KA10	QUICKPOINT
COMSYST	EDGRIN	LAB-8	RAD-8
COMTEX	EDUSYSTEM	LAB-8/e	RSTS
DDT	FLIP CHIP	LAB-K	RSX
DEC	FOCAL	OMNIBUS	RTM
DECCOMM	GLC-8	OS/8	RT-11
DECTAPE	IDAC	PDP	SABR
DIBOL	IDACS	PHA	TYPESET 8
			UNIBUS

PREFACE

The PDL system is primarily a system designed for data acquisition and analysis. It will collect, record and process data from 15 instruments such as Autoanalyzers, SMA's or Coulter model "S" while simultaneously doing calculations with the data being collected or perform any other services of BASIC.

The PDL operators guide provides the user with information on system usage, initialization of instrument parameters, cassette manipulation, software interfacing, report generating, sample programs, error conditions and error procedures. Also included is an introduction to the BASIC CAPS language. These chapters are structured in the logical fashion to orient you with the system from installation to daily operation.

The following manuals provide the PDL user with additional information.

PDL Programmers Manual DEC-11-OPPMA-A-D

BASIC Language Reference Manual DEC-11-LIBBA-A-D

CAPS 11 Users Guide DEC-11-OTUGA-A-D

CAPS/BASIC Users Manual DEC-11-LIBCA-A-D

CONTENTS

CONTENTS (Cont.)

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The PDL Operator's Guide provides an overview of the PDL system, its function, and its overall operating procedures. The chapters include information on system usages, initialization, cassette manipulation, software interfacing, report generating, and error procedures. Also included is an introduction to the BASIC-CAPS language. The information is set forth in a logical fashion to increase familiarity with PDL and to simplify system use on a daily basis.

1.2 INTENDED USE

PDL (Programmable Data Logger) has been developed specifically for use with clinical laboratory instruments. PDL's primary functions are to aquire data from such instruments as Autoanalyzers, SMAs, and coulter model "S" counters, to manipulate the data mathematically, and to produce formatted reports. (The reports are printed on the Console terminal or any other designated printing device.) PDL may also be used as a programmable calculator, tailored to allow individual modification within the laboratory.

1.2.1 The System

PDL is built around the PDP-11/10 processor. The PDL system includes such features as dual cassette drives (drives 0 and 1), a console terminal (a DECwriter), and 24K (1K = 1024 words) of core memory. Local operator console terminals are provided for each laboratory instrument interface.

Input/Output (I/O) devices connect local operator terminals, clinical instruments, and the central processor. For such clinical instruments as the coulter, the I/O devices are digital; for autoanalyzers or SMAs, analog/digital converters are provided.

Clinical instruments must be connected to the computer by cable to a screw terminal.

1.2.2 The Console

The console contains switches through which information necessary to the computer operation may be directly inserted to the programs. This is a convenient manual method of updating and controlling the status of the operations. A complete explanation of the console switches and their use will be found in the CAPS-11 USERS GUIDE. However the running of PDL does not require the use of these console switches.

1.3 THE DECWRITER

The DECwriter (LA36) with its printer and keyboard acts as the computer typewriter. It prints both the input data typed by the user and the data output from the computer, at a rate of 30 characters per second.

The DECwriter keyboard resembles the keyboard of a typewriter, with several additional keys that serve specifically as communications control keys. The carriage return key (whose use is indicated in this manual by the notation <CR>) is pressed to tell the computer that the current line of typed input is complete. This action automatically moves the print head to the left margin to await further input. Carriage return is pressed at the end of each line of operator dialogue.

The line feed key (indicated as LF in this manual) returns the print head to the left margin but does not tell the computer that the current line of input is complete. This allows for continuation of a long input line on the next typing line.

The ALT-MODE key (indicated by ALT) may appear on the keyboard as the ESCape key. It has many functions which are not described here because PDL does not use this key.

The RUBOUT key erases characters, one by one, in reverse order. Press the RUBOUT key once for each character to be erased, then type the correct characters. To erase a complete line, hold down the CTRL (control) key and type the U key. Then resume typing the correct input.

1.3.1 Special Switches on the LA36 Terminal

The baud switch controls the speed at which the terminal can receive and send characters. When PDL is being used, the Baud switch must be set at 300.

Information on all of these functions and a complete description of the DECwriter will be found in the CAPS-11 Users Guide (DEC-11-OTUGA-A-D).

1.4 LOCAL OPERATOR'S CONSOLE

The local operator's console looks like a small black box. Such a box is attached to every laboratory instrument. The local operator's console is the means through which the laboratory instruments

communicate with the computer. The box contains three white buttons, above which are three small lights, (see Figure 1-1) To run PDL effectively it is necessary to understand the functions of both buttons and lights.

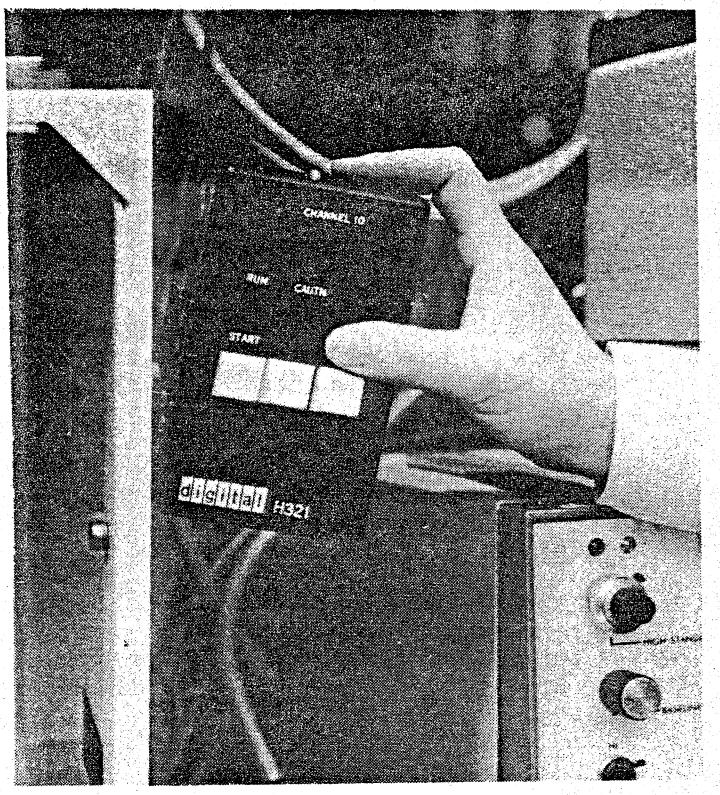

Figure 1-1
Local Operator's Console

The button on the left is the START button. Pressing this button causes the computer to begin taking data from laboratory instruments.

The light on the left (above the start button) is the READY/RUN light. The flashing of this light indicates that PDL is ready to be started. When the light remains on, PDL is taking data from the instrument.

The middle button is not assigned an activity. This particular button may be used to control a function needed in the user's program, possibly a halt.

The light in the middle is the BUFFER STATUS LIGHT. This light flashes as a warning whenever buffer storage areas are almost full. The flashing reminds the user that report generation is necessary to remove data from the instrument's buffer. If the light is ignored it will eventually remain on, indicating that the buffer in question has overflowed, and that results are being lost.

The button on the right is the MARK BUTTON. It is used to set selective indicators on data taken from the laboratory instrument, so that this particular data may be treated by the Report Generating program in a specific manner. The pressing of this button once

initiates this action. A second pressing of the button cancels the action. An example of the use of this feature is to indicate the separation of several samples of water so that they are not included in a calculation.

The right hand light indicates the status of the MARK BUTTON. If the light is on, the MARK function is in use.

The local operator's console enables the user to remotely control the computer's interaction with a particular laboratory instrument, the transfer of data to the computer, and the monitoring of the whole procedure.

1.5 CASSETTE SYSTEM

PDL is tailored to the cassette system. The choice of cassettes for use with this system contributes to the ease, directness, and simplicity of approach that is inherent in the system. Loading, unloading, and general manipulation are greatly simplified because of this feature.

The small, sturdy cassette is capable of carrying large quantities of data and is easy to handle. Storage problems are nonexistent; many cassettes may be stored in a relatively small area.

Cassettes are inexpensive to buy. Therefore, their use as backup storage, for important data, is both efficient and economical. Advantage should be taken of the back-up storage feature in both the BASIC CAPS system and the PDL system. Further information on cassettes will be found in Chapter 2.

NOTE

Do not use audio cassettes on this system. Use only computer cassettes.

1.6 BASIC LANGUAGE OVERVIEW

BASIC is the language chosen for use with the PDL system. BASIC is a high level, English-like language that can be learned quickly and easily. It is, perhaps, the simplest of all programming languages to use. The small number of powerful statements and commands that are necessary to produce good results, together with the ease of problem solving, make BASIC ideal for use with PDL

Although BASIC is similar to other high-level programming languages, its conversational nature makes it a convenient language for use with PDL. A conversational language allows two-way communication between the programmer and the language processor. The typing of input data onto the terminal invokes a reciprocal response from the computer, which results in output data being printed on the terminal. Use of BASIC in the cassette environment is referred to as BASIC/CAPS.

BASIC-CAPS facilitates the easy writing of report generating programs. Data acquisition and processing actions continue while the user is writing or testing programs, or otherwise using BASIC as a calculator. See Report Generators, Chapter 7.

CHAPTER 2

THE COMPUTER CASSETTE

The computer cassette is a convenient magnetic tape device used to simplify the manipulation of data within a computer environment. In size, shape, and construction, the computer cassette is similar to the popular cassette commonly used for home tape recording. However, the computer cassette is structured specifically for use with the computer.

On one edge of the cassette there are two flexible, plastic orange tabs. They are called 'WRITE PROTECT TABS'. They are situated one at each end of the cassette where they can govern the write capability on the tape.

Unlike home tape cassettes, the computer cassette is used to record information in one direction only. Therefore only one orange tab (the one marked with an arrow on the cassette label) need be used. (See Figure 2-1.)

When the orange tab is pulled over toward the middle of the cassette (i.e., the hole is uncovered), the tape is considered to be in WRITE-LOCK position. WRITE-LOCK means that data cannot be written onto the tape. This simple protection device has proven a very useful tool. If an attempt is made to write onto a cassette which is in WRITE-LOCK, an error message (WRT LOCK) will result.

Pushing the orange tab to the outside of the cassette (i.e., covering the hole) makes the tape WRITE-ENABLED. WRITE-ENABLED means that data may be written onto the tape.

The orange tab, while controlling the cassette's ability to receive data, does not affect the tape's ability to be read. Thus, the cassette tape may be read, regardless of the position of the tab.

As a safeguard against abrasion, both ends of the tape consist of clear plastic. The clear plastic is either leader (or trailer tape). Magnetic tape is quite susceptible to dust, grit, and fingerprints. Therefore the clear plastic leader protects the tape during handling and storage.

To ensure proper use of this protect feature the tape should be rewound to its beginning (plastic leader) at all times other than when the cassette is actually in use.

2.1 RECORD FORMATTING

The computer cassette tape contains data in the form of files and records. A file is defined as the largest self-contained unit of data on the tape. A file consists of one or more records. A record is defined as the smallest self-contained unit of data on the tape. It is the only type of unit within a file.

The size of a data-file is determined by the amount of data it encompasses. There are no set number of files on a cassette tape. There may be as few as one file or there may be many files, each containing one or more small data records. A file may not extend over more than one cassette tape.

Each file on the cassette tape is separated from its neighbor by an inter-file gap. The inter-file gap is one quarter inch long. Its purpose is both to separate the individual files and to serve as a point of reference for any program that is seeking a given file. Therefore, it is important that such a gap also appear before the first file on the tape. The seeking program will ignore any data it finds ahead of the first inter-file gap.

The data-record is structured in the same manner as the file. Because there are often many records within a file, each record is separated from the next by an inter-record gap. Within a file, each record must be in both logical and physical sequence. The record's number, its position in the file, and general information controlling its function, are present, together with the file name, file type, and file length, in the first record of that particular file. This record is called the header record.

2.2 USING THE CASSETTE

Before mounting the cassette on the cassette drive, the orange tab should be set to the desired position (WRITE-LOCK or WRITE ENABLE) The locking-bar (a protect feature on the cassette drive) is removed by pushing (sliding) it to the right, away from the drive (see Figure 2-1).

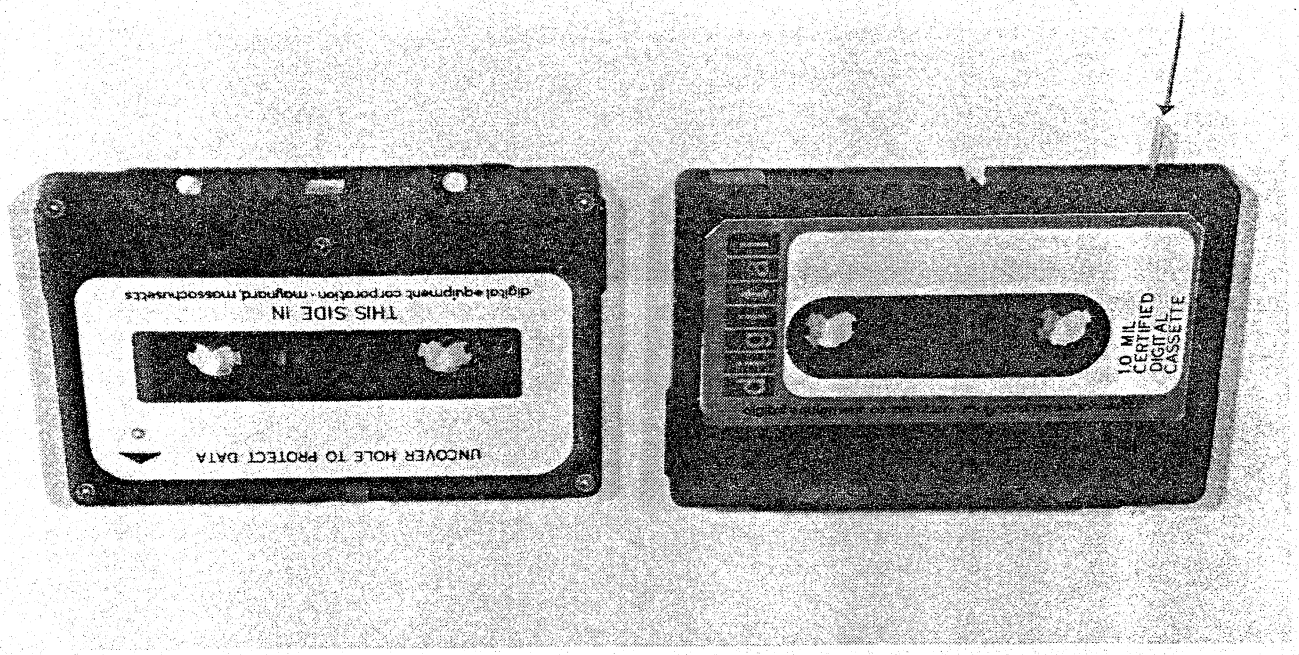

Figure 2-1
Cassettes

The cassette is then held, with the open (tape) edge facing to the left. In this position the cassette's flat surface, which is facing the operator, has the rewound tape at the top. The cassette is held so that the left (tape) edge is tilted inwards towards the computer and the right (back) edge is outwards from the computer at approximately a 45-degree angle. The cassette is then inserted by pushing slightly upward as the cassette is pressed firmly home onto the drive sprockets. The cassette tape is now in contact with the read/write heads. The cassette locking-bar is then replaced (see Figure 2-2).

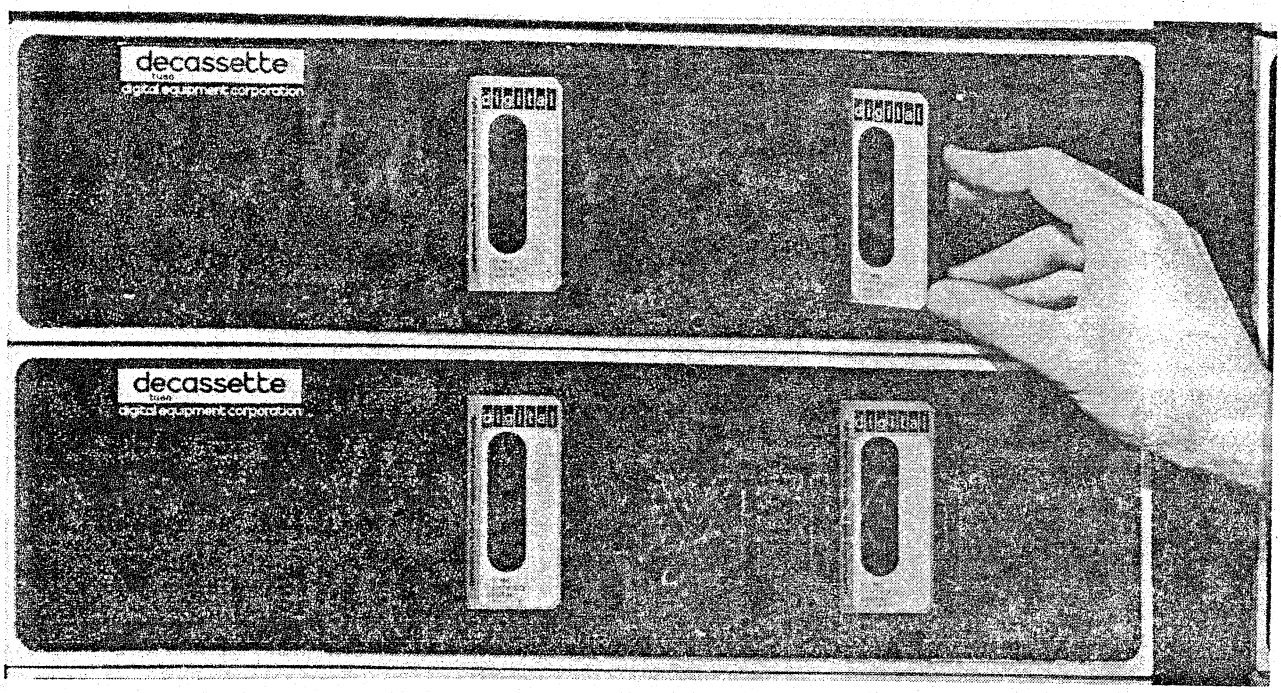

Figure 2-2
Cassette Drive

To remove a cassette from a cassette drive, the locking-bar is pushed away from the drive (i.e to the right) and the release of pressure causes the cassette to pop out.

Each cassette drive has a small black rewind button. This button is located on the right of the particular drive. One press of this button causes the tape to rewind to its beginning. The tape should be positioned at its beginning before work commences.

Newly shipped cassettes should be prepared for use in the following manner. The cassette is first mounted on a drive so that the cassette's DIGITAL label faces inwards to the drive unit. This is the opposite way to normal procedure, and is only performed this way when a new tape is being used. The tape is then rewound by pressing the rewind button. Following this, the cassette is removed from the drive, turned over so that the label faces outwards, then remounted on the drive. The tape is rewound again. This procedure stacks the tape neatly within the cassette and provides the adequate tension for its regular use.

When not in use, cassettes may be conveniently stored in the small plastic boxes provided for this purpose by the manufacturer. The computer's facing panel has provision for four of these plastic boxes (including the cassettes) to be held for easy access during operations.

NOTE

As mentioned earlier it is a good practice to keep tapes in their rewound state and in their plastic cases at all times other than when they are actually in use.

2.3 THE SYSTEMS CASSETTE

Each programming system is contained on a single cassette. This cassette is called the system cassette. It contains the programs that govern the loading and the running of all of the systems programs.

NOTE

As the systems cassette is not normally written upon it is advisable to make sure that the orange protect tab is always in WRITE-LOCK.

Because of its obvious importance to the system, the information contained on the systems cassette should be copied onto a spare cassette at the earliest opportunity. In the event that the system cassette information is destroyed, such a back-up saves time, money, and frustration. To make copies of system cassette, refer to the section on copying cassettes.

2.4 LOADING PROGRAMS FROM A CASSETTE

To operate the systems programs, it is necessary to have them in the computer's memory. The loading of these programs into memory is effected by the BOOTSTRAP program.

The BOOTSTRAP program is a self-loading program: a series of instructions (residing on the beginning of the system tape) enable the system programs to be transferred into memory. Once loaded, the BOOTSTRAP program is able to accept commands (from the console terminal keyboard) directing it to load the programs which follow on the systems tape.

There are three systems cassettes provided with the PDL system. They are:

CAPS-11 System (CAPS-11)

BASIC CAPS-11 System (BASIC)

PROGRAMMABLE DATA LOGGER (PDL)

The loading procedure for any of the above systems consists of the following steps:

1. Be sure both computer and console terminal are on line. Turn power key ON.

2. Place the specific systems cassette onto cassette drive 0. Drive 0 must always be used for the system cassette. This drive is found on the left-hand side.

NOTE

There are two cassette drives on the console: drive 0 and drive 1. It is important to remember that drive 0 is on the left.

3. There are a number of switches on the front of the computer (see Figure 2-3). Make sure that the power key is on and not in LOCK position. Press the HALT switch downward and then upward into the ENABLE position. (This cancels previous programming action and prepares the console for further work).

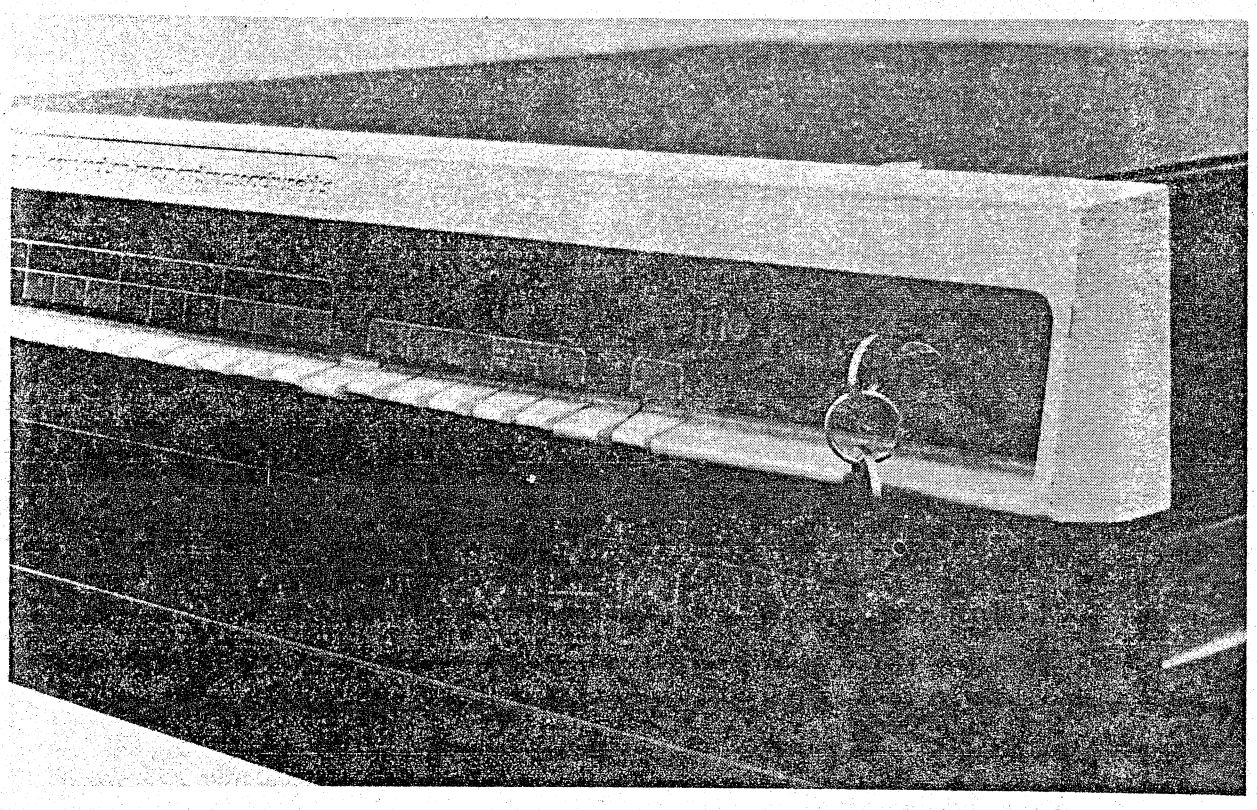

Figure 2-3
Console

There is a panel on the front of the computer. This panel contains a number designation: H324.
It also contains switches numbered 1-4 and a LOCK/UNLOCK switch. Place the switch marked LOCK/UNLOCK in the downward position. (This action prepares for loading BOOTSTRAP.) Hold it in that position until the next switch has been pressed.

Press the button indicated by a number 1.

Release both buttons.

NOTE

At this point, the run lamp is on and the systems cassette tape begins to move. The system is being loaded.

The system cassette tape comes to a halt when the system has been loaded. If CAPS-11 system has been used, proof of loading is seen when the identification line shown below is automatically printed on the terminal.

```
CAPS-11  V01-02
```

A dot is also printed at the terminal's left margin. This dot is the indication that the system is in the ready state (i.e., awaiting a command from the terminal).

NOTE

If a loading error occurs, the systems cassette tape stops moving and the computer halts. This situation requires a restart of the whole procedure. The system must be REBOOTED (i.e., the BOOTSTRAP must be reloaded). If the error reoccurs, refer to the error procedures in Appendix C in the CAPS-11 User's Guide DEC-11-OTUGA-A-D.

When in the ready state, as indicated by the dot printed at the left margin of the terminal printer, the system accepts any of the following eight commands:

```
DATE
ZERO
SENTINAL
DIRECTORY
RUN
LOAD
START
VERSION
```

A complete description of the commands, together with their uses, can be found in the CAPS-11 User's Guide (DEC-11-OTUGA-A-D) Section 3.3.

2.5 ZEROING A CASSETTE

One of the most useful commands available in the CAPS system is the ZERO command. The command deletes (erases) any data or files present on a given cassette. This is done by writing a Sentinel file at beginning of tape. Refer to CAPS-11 Users Manual on definition of Sentinel file. The purpose of this procedure is to enable a previously used cassette to be reused without the possibility of previous data and present data inadvertently being mixed.

The procedure for zeroing a cassette tape is shown below:

1. Place cassette to be zeroed on drive 1.

2. In response to the dot on the terminal type the letters ZER, followed by the drive number, followed by a colon.

   Example: to zero the tape on drive 1 type:

   ZER1:<CR>

   The tape is now automatically zeroed.

NOTE

All new cassettes should be zeroed to ensure proper operation. Unzeroed cassettes can not be written on. When zeroing previously used cassettes, care should be used so that important data are never accidently removed.

2.6 COPYING A CASSETTE

The copying of a cassette tape is a function that will often be performed.

Copying a cassette is aided by the Peripheral Interchange Program (PIP). PIP is a useful utility program which allows copying, deleting, zeroing, and multiple copying of cassette tapes. Information regarding each of these features is found in the CAPS-11 Users Manual.

PIP resides on the CAPS-11 systems cassette. The CAPS system tape is booted. This results in a dot appearing on the left margin of the terminal printer. In response to the dot the PIP program is called by typing the following statement on the terminal.

R PIP <CR>

PIP is automatically loaded into memory. An asterisk printed on the left margin of the terminal printer, is PIP's response. The system is ready for the next command which in this case is a string command (i.e., it contains several actions) in the format shown below.

CT#: = CT#:/OPTION

In the above symbolic statement CT#: refers to the cassette drive number, and OPTION refers to the type of action that PIP is to perform.

For copying, only one input and only one output device are required. Therefore the command to be typed would be:

1:=0: <CR>

where 1 is the output drive, 0 is the input drive, and C means copy.

The output cassette (on drive 1) is automatically zeroed. The contents of the cassette on drive 0 are then copied onto the cassette on drive 1.

NOTE

The protect tab should be in WRITE-LOCK position on the input tape (the tape being copied); the tab should be in ENABLE position on the output tape (the blank or zeroed tape).

To copy the BASIC-CAPS-11 systems tape or the PDL systems tape, place the systems tape on drive 0 and a receiving tape on drive 1. In response to the *, type:

1:=0: <CR>

Further information on PIP may be found in CAPS-11 User's Guide (DEC-11-OTUGA-A-D) Section 8-2.

CHAPTER 3

PDL INITIALIZATION

The initialization routine allows the user to tailor the PDL software to the unique characteristics of specific labaratory instruments. This routine is a communication between the user and the computer, which, through a series of questions and answers, allows individualized configuration of the system.

Initialization of the system with a particular instrument configuration is normally required only once. This usually is when the system is installed. From then on, the specified instruments may be used with the PDL system on a daily basis. Once the desired parameters have been incorporated into the PDL system, reinitialization is necessary only when a configuration change is required.

To proceed with the initialization routine it is necessary to know the following:

1. The laboratory instrument timing
2. The number of tests to be reported on a given channel for one sample
3. The buffer size of each channel
4. The physical channels to which the instruments are connected and to which the local operator's box is attached.

NOTE

At the end of each user communication with the computer, the carriage return key <CR> must be pressed. This signifies the end of that particular dialogue and positions the printer for the next communication.

3.1 CONSOLE PROCEDURES

To initialize the PDL system follow the procedure below.

1. Mount the PDL systems cassette on unit 0.

2. Press down the halt switch, then raise it to the ENABLE (up) position.

3. Press the UNLOCK LOCK switch on the BOOT panel downward and hold it there.

4. Press switch number 1

5. Release both switches

A line of text now appears on the console terminal:

BASIC V01-01

indicating that PDL utilizes BASIC as its base language.

3.1.1 Option Routine

NOTE

As a documentation convention in this manual, the system or computer or peripheral device "prints" output whereas the user "types" input. Both output and input are physically printed on the same device.

The second line of dialogue printed is:

OPT FNS

OPT FNS is an abbreviation for "optional functions". All available optional functions have been incorporated into the PDL system as a default. However, at the appearance of the line OPT FNS the user has the option to accept, reject, or selectively incorporate this feature. (Refer to Appendix A for a list of these functions.)

To accept the system with all optional functions	Press the carriage return key.
To selectively accept optional functions	Type I
To accept an optional function	Type A or just a carriage return
To reject an optional function	Type N

If the user types I, the system prints the individual optional functions for the user to accept or reject. After each choice is indicated by the user's typing an A or an N, pressing the carriage return key causes the system to print the next optional function.

NOTE

Elimination of optional functions increases program space. However, make

sure that any function deleted is not required in future programs.

3.2 USER FUNCTIONS

The next line to be printed is:

```
USER FNS LOADED
```

USER FNS is an abbreviation for "user functions". This message indicates that PDL routines have been loaded.

3.2.1 TERM

When the system prints:

```
TERM:
```

the user types in the number that designates the type of terminal used. PDL uses a terminal which runs at 300 baud and is designated by a 1.

```
1<CR>
```

3.2.2 DATE

Next, the system prints:

```
DATE:
```

The correct response is in the form dd-mmm-yy, where dd=day, mmm=month, and yy=year. A typical response would be:

```
09-JUN-75
```

If the date is not typed in this format, the system prints BAD DATE, and requests the user to try again. After the correct response, the system prints:

```
READY
```

which indicates that the system is successfully loaded. PDL is now awaiting a response in the form of a command, a program line, or a name.

NOTE

If the computer is turned off while BASIC is operating, the ?PWF (power fail) error message is printed when the power is turned on again.

The user types:

RUN SETUP

The initialization program is called SETUP.

SETUP now prints out the name, the date, and the BASIC version number. The initialization program has begun.

The computer at this time prints out questions referring to the first-time-initialization. The user is asked to provide the configuration of the system.

The system then prints:

FIRST TIME INITIALIZATION PROGRAM. PLEASE SPECIFY IN RESPONSE TO THE QUESTIONS TYPED HERE THE CONFIGURATION OF YOUR SYSTEM. YOU WILL HAVE TO KNOW THE PHYSICAL CHANNELS TO WHICH YOUR INSTRUMENTS AND H321 ARE ATTACHED, INSTRUMENT TIMING, THE NUMBER OF TESTS REPORTED ON A CHANNEL FOR ONE SAMPLE AND THE BUFFER SIZE FOR EACH CHANNEL.

ARE YOU READY (YES OR NO)?

The response to this is Y or YES for yes. If NO is typed, the computer prints:

THEN START ME WHEN YOU ARE.

When you are ready to restart this sequence, type RUN SETUP.

If Y is typed, the system requests further information about the first instrument to be used.

The computer prints:

PLEASE GIVE THE INFORMATION FOR INSTRUMENT NUMBER 1 TO WHICH PHYSICAL CHANNEL HAS THE INSTRUMENT BEEN CONNECTED.

The system is asking for the channel to which the instrument has been connected. The physical channel number is given in response:

0 <CR>

If the response to the question is a number between 0-15 the system prints:

THIS IS AN ANALOG CHANNEL.

If the response is 16-31 the system prints:

THIS IS A DIGITAL CHANNEL.

NOTE

Channels are either digital or analog, depending upon the instrument. If analog, they are assigned a number between 0-15. Digital channels are numbered 16 through 31. The incorrect assignment of a channel will result in an error message.

The next question from the system asks which channel is connected to the local operator console. The answer to this is the number of the physical channel on H322 distribution panel. Whenever possible it should be the same as the channel specified for the instrument.

0 <CR>

The system asks that the user check the connection of the H321 LOC to the H322 distribution panel. The message printed is:

CHECK THAT YOU ARE CONNECTED TO LOC DR-11 (EITHER 0,1, OR 2)

AND THAT YOU ARE USING TERMINALS # - #

Next the computer asks for the type of instrument in use. The answer to this is one of the following:

AA indicating Autoanalzer

SMA indicating SMA 6/60 or 12/60

CS indicating Coulter Counter Model S

OTHER indicating instrument type is not one of the above.

The next question is:

HOW MANY TESTS ARE MADE ON EACH SAMPLE.

The appropriate response to this is the number of results per sample. For example, on the SMA 6/60 there are six results per sample. On the coulter model S, the proper response is 10, because the data is cycled twice.

Then:

HOW MANY SAMPLES ARE PROCESSED PER HOUR.

If the user is running at 60 per hour, the response would be 60.

If the instrument is a coulter the response is:

WHAT DELAY DO YOU WANT IN SECONDS EXPRESSED TO THE NEAREST 0.01 SECOND.

The delay specified should be between 40 and 120 miliseconds. We recommend use of 100 miliseconds for the Coulter "S", so you would type 0.10.

Next from the system:

HOW MANY WORDS DO YOU WANT FOR THIS INSTRUMENT'S BUFFER.

(One word is required for each test result.) When running at 60 per hour and an hour's worth of data must be stored prior to printing, the response would be 60. Buffer space is allocated in 1-word increments.

NOTE

When indicating buffer space it is important to allocate only that space which is available. To change this parameter once established, see the PDL Programmers Manual.

The system then informs the user how many words of buffer space are left after each instrument's requirements have been allocated.

After this the system asks:

DO YOU HAVE ANOTHER INSTRUMENT.

If the response is YES, the procedure is repeated for the second instrument. This process continues for all the instruments in use. The response of NO causes an exit from this routine.

The computer at this time prints a summary (see example below) of all the parameters specified, and asks the user for verification by printing:

CHECK THE ABOVE SUMMARY. IS IT OK?.

The user response is:

Y or YES

N or NO

This is the last chance for corrections. Once Y is typed, all the parameters become part of the PDL system in memory and a save file containing these parameters is stored on the cassette on Unit 0.

Computer prints:

MOUNT CASSETTE FOR SAVE ON UNIT ZERO AND PRESS RETURN KEY

Computer prints:

SYSTEM INITIALIZATION IS COMPLETE. AN ASCII FILE NAMED PDLSAV.DAT HAS BEEN CREATED WHICH CONTAINS ALL OF YOUR SYSTEM PARAMETERS. ITS CONTENTS ARE:

A list of the contents is then printed onto the DECwriter.

NOTE

Channels are either digital or analog, depending upon the instrument. If analog, they are assigned a number between 0-15. Digital channels are numbered 16 through 31. The incorrect assignment of a channel will result in an error message.

The next question from the system asks which channel is connected to the local operator console. The answer to this is the number of the physical channel on H322 distribution panel. Whenever possible it should be the same as the channel specified for the instrument.

0 <CR>

The system asks that the user check the connection of the H321 LOC to the H322 distribution panel. The message printed is:

CHECK THAT YOU ARE CONNECTED TO LOC DR-11 (EITHER 0,1, OR 2)

AND THAT YOU ARE USING TERMINALS # - #

Next the computer asks for the type of instrument in use. The answer to this is one of the following:

AA indicating Autoanalzer

SMA indicating SMA 6/60 or 12/60

CS indicating Coulter Counter Model S

OTHER indicating instrument type is not one of the above.

The next question is:

HOW MANY TESTS ARE MADE ON EACH SAMPLE.

The appropriate response to this is the number of results per sample. For example, on the SMA 6/60 there are six results per sample. On the coulter model S, the proper response is 10, because the data is cycled twice.

Then:

HOW MANY SAMPLES ARE PROCESSED PER HOUR.

If the user is running at 60 per hour, the response would be 60.

If the instrument is a coulter the response is:

WHAT DELAY DO YOU WANT IN SECONDS EXPRESSED TO THE NEAREST 0.01 SECOND.

The delay specified should be between 40 and 120 miliseconds. We recommend use of 100 miliseconds for the Coulter "S", so you would type 0.10.

Next from the system:

HOW MANY WORDS DO YOU WANT FOR THIS INSTRUMENT'S BUFFER.

(One word is required for each test result.) When running at 60 per hour and an hour's worth of data must be stored prior to printing, the response would be 60. Buffer space is allocated in 1-word increments.

NOTE

When indicating buffer space it is important to allocate only that space which is available. To change this parameter once established, see the PDL Programmers Manual.

The system then informs the user how many words of buffer space are left after each instrument's requirements have been allocated.

After this the system asks:

DO YOU HAVE ANOTHER INSTRUMENT.

If the response is YES, the procedure is repeated for the second instrument. This process continues for all the instruments in use. The response of NO causes an exit from this routine.

The computer at this time prints a summary (see example below) of all the parameters specified, and asks the user for verification by printing:

CHECK THE ABOVE SUMMARY. IS IT OK?.

The user response is:

Y or YES

N or NO

This is the last chance for corrections. Once Y is typed, all the parameters become part of the PDL system in memory and a save file containing these parameters is stored on the cassette on Unit 0.

Computer prints:

MOUNT CASSETTE FOR SAVE ON UNIT ZERO AND PRESS RETURN KEY

Computer prints:

SYSTEM INITIALIZATION IS COMPLETE. AN ASCII FILE NAMED PDLSAV.DAT HAS BEEN CREATED WHICH CONTAINS ALL OF YOUR SYSTEM PARAMETERS. ITS CONTENTS ARE:

A list of the contents is then printed onto the DECwriter.

The computer then prints:

```
STOP AT LINE 50

READY
```

NOTE

If initialization parameters are to be included on the PDL System cassette, that cassette should be mounted on unit 0. This cassette will be used on a daily basis.

It is advisable to copy this system onto another cassette to save as backup. The following is an example dialog of the Setup Routine:

```
FIRST TIME SYSTEM INITIALIZATION PROGRAM.

PLEASE SPECIFY IN RESPONSE TO THE QUESTIONS TYPED
HERE THE CONFIGURATION OF YOUR SYSTEM.
YOU WILL HAVE TO KNOW THE PHYSICAL CHANNELS TO WHICH
YOUR INSTRUMENTS AND H321S ARE ATTACHED, INSTRUMENT TIMING,
THE NUMBER OF TESTS REPORTED ON A CHANNEL FOR ONE SAMPLE,
AND THE BUFFER SIZE FOR EACH CHANNEL.


ARE YOU READY (YES OR NO)? YES

PLEASE GIVE THE INFORMATION FOR INSTRUMENT NUMBER 1

TO WHICH PHYSICAL CHANNEL HAS THE INSTRUMENT BEEN
CONNECTED? 0
THIS IS AN ANALOG CHANNEL
TO WHICH CHANNEL IS THE LOC CONNECTED? 0
CHECK THAT YOU ARE CONNECTED TO LOC DR-11 # 0
AND THAT YOU ARE USING TERMINALS 0 - 2
WHAT INSTRUMENT ARE YOU USING (AA,SMA,CS,OTHER)? SMA
HOW MANY TESTS ARE MADE ON EACH SAMPLE? 6
HOW MANY SAMPLES ARE PROCESSED PER HOUR? 60
HOW MANY WORDS DO YOU WANT FOR THIS INSTRUMENT'S BUFFER
(ONE WORD IS REQUIRED FOR EACH TEST RESULT)? 200

YOU HAVE USED  200 WORDS SO FAR.

DO YOU HAVE ANOTHER INSTRUMENT? Y

PLEASE GIVE THE INFORMATION FOR INSTRUMENT NUMBER 2

TO WHICH PHYSICAL CHANNEL HAS THE INSTRUMENT BEEN
CONNECTED? 1
THIS IS AN ANALOG CHANNEL
TO WHICH CHANNEL IS THE LOC CONNECTED? 1
CHECK THAT YOU ARE CONNECTED TO LOC DR-11 # 0
AND THAT YOU ARE USING TERMINALS 3 - 5
WHAT INSTRUMENT ARE YOU USING (AA,SMA,CS,OTHER)? AA
HOW MANY TESTS ARE MADE ON EACH SAMPLE? 1
HOW MANY SAMPLES ARE PROCESSED PER HOUR? 60
HOW MANY WORDS DO YOU WANT FOR THIS INSTRUMENT'S BUFFER
(ONE WORD IS REQUIRED FOR EACH TEST RESULT)? 60

YOU HAVE USED  260 WORDS SO FAR.
```

```
DO YOU HAVE ANOTHER INSTRUMENT? Y

PLEASE GIVE THE INFORMATION FOR INSTRUMENT NUMBER 3

TO WHICH PHYSICAL CHANNEL HAS THE INSTRUMENT BEEN
CONNECTED? 16
THIS IS A DIGITAL CHANNEL
TO WHICH CHANNEL IS THE LOC CONNECTED? 3
CHECK THAT YOU ARE CONNECTED TO LOC DR-11 # 0
AND THAT YOU ARE USING TERMINALS 9 - 11
WHAT INSTRUMENT ARE YOU USING (AA,SMA,CS,OTHER)? CS
HOW MANY TESTS ARE MADE ON EACH SAMPLE? 10
WHAT DELAY DO YOU WANT IN SECONDS EXPRESSED TO THE NEAREST
0.01 SECOND? 0.10
HOW MANY WORDS DO YOU WANT FOR THIS INSTRUMENT'S BUFFER
(ONE WORD IS REQUIRED FOR EACH TEST RESULT)? 200

YOU HAVE USED  460 WORDS SO FAR.

DO YOU HAVE ANOTHER INSTRUMENT? Y

PLEASE GIVE THE INFORMATION FOR INSTRUMENT NUMBER 4

TO WHICH PHYSICAL CHANNEL HAS THE INSTRUMENT BEEN
CONNECTED? 2
THIS IS AN ANALOG CHANNEL
TO WHICH CHANNEL IS THE LOC CONNECTED? 2
CHECK THAT YOU ARE CONNECTED TO LOC DR-11 # 0
AND THAT YOU ARE USING TERMINALS 6 - 8
WHAT INSTRUMENT ARE YOU USING (AA,SMA,CS,OTHER)? OTHER
GIVE INSTRUMENT TYPE.
0=SINGLE NUMBER, 1=PEAK, 2=PLATEAU, 3=ON DEMAND? 3
GIVE THE DATA TYPE. 0=ANALOG, 1=BINARY, 2=BCD? 0
HOW MANY TESTS ARE MADE ON EACH SAMPLE? 1
WHAT DELAY DO YOU WANT IN SECONDS EXPRESSED TO THE NEAREST
0.01 SECOND? 0
HOW MANY WORDS DO YOU WANT FOR THIS INSTRUMENT'S BUFFER
(ONE WORD IS REQUIRED FOR EACH TEST RESULT)? 100

YOU HAVE USED  560 WORDS SO FAR.

DO YOU HAVE ANOTHER INSTRUMENT? N


INSTRUMENT NUMBER 1 :
IS ON CHANNEL 0 WITH LOC # 0
INSTRUMENT TYPE IS 2 , DATA TYPE IS 0
THERE ARE 6 TESTS PER SAMPLE AT A RATE OF 10 SECONDS PER TEST.
BUFFER WORDS ALLOCATED= 200


INSTRUMENT NUMBER 2 :
IS ON CHANNEL 1 WITH LOC # 1
INSTRUMENT TYPE IS 1 , DATA TYPE IS 0
NOTE: NUMBER OF TESTS IN SAVE FILE IS 2
THERE ARE 1 TESTS PER SAMPLE AT A RATE OF 60 SECONDS PER TEST.
BUFFER WORDS ALLOCATED= 60


INSTRUMENT NUMBER 3 :
IS ON CHANNEL 16 WITH LOC # 3
INSTRUMENT TYPE IS 0 , DATA TYPE IS 2
```

PDL INITIALIZATION

```
THERE ARE 10 TESTS PER SAMPLE WITH A DELAY OF .1 SECONDS PER TEST.
BUFFER WORDS ALLOCATED= 200


INSTRUMENT NUMBER 4 :
IS ON CHANNEL 2 WITH LOC # 2
INSTRUMENT TYPE IS 3 , DATA TYPE IS 0
THERE ARE 1 TESTS PER SAMPLE.
BUFFER WORDS ALLOCATED= 100


CHECK THE ABOVE SUMMARY.
IS IT ALL RIGHT? Y
```

CHAPTER 4

USING PDL

Once the installation and initialization of the system are completed, the system may be run on a daily basis. Each daily procedure requires the same fixed dialogue.

NOTE

Use the cassette that contains the PDL system and the save files.

To load PDL, and the configuration save files, and also to start the system, follow the instructions below.

1. Place PDL system cassette on unit 0. This cassette should contain the initialization file.

2. Hold down the LOCK/UNLOCK switch in the unlock position and press the switch marked 1.

   The tape begins to move and copies PDL into memory. When it completes this task, the computer responds with the system identification and version number:

   BASIC/CAPS V01

   The computer then prints:

   OPT FNS

   To which the user replies with a carriage return, unless other options are to be specified. Refer to optional functions in the CAPS/BASIC Users Guide.

   <CR>

NOTE

If N is typed, no options are included and the available program space is increased by approximately 600 words.

The computer prints:

USR FNS LOADED

This allows insertion of user functions. The response to this is followed by a carriage return.

The next question is which terminal:

TERM:

The answer to this is:

1 <CR>

This indicates that the terminal is running at 300 baud (see Section 1.3.1).

The computer's next request is:

DATE:

The reply is as before, dd-mmm-yy, where dd=the day, mmm=the month and yy=the year, Example:

12-OCT-75<CR>

The system is now successfully loaded. It prints:

READY

The user now types:

PDL <CR>

This response loads the PDL system save file (the instrument parameters), turns on all applicable LOC box lights, and initializes all instrument channels for the daily run. The computer is ready to accept data from the user's analyzer. A flashing light on a local console box indicates that the associated channel is ready to receive information. It is good practice to remove the PDL save cassette so that it cannot be written on.

NOTE

If the instrument in use is a peak or plateau device (i.e., an autoanalyzer or an SMA), the recorder must be adjusted to zero baseline after appropriate warmup and calibration procedures. It is important that all baselines are at zero before the start button on LOC is pressed.

When the start button is pressed, data acquisition begins at the first significant pen rise. The data is transferred to a temporary storage area. When the buffer area becomes 75% full, the middle light on the local operator's console (H321) begins to flash. Until the light starts flashing the user can:

1. Do calculation in the immediate mode.

2. Develop new programs in BASIC.

3. Execute previously written programs.

A previously written user program (a report generator) may be used to take stored data and write it out in the form of a report. This may be done at any time and is at the user's discretion. However, when the buffer becomes 75% full the middle warning light on the Local Operator's Console begins to flash. It is time to empty the buffer. Failure to do this at this time results in a buffer overflow. A buffer overflow causes the middle light to become steady. At this time no further data will be moved into the buffer. Time elapses between the beginning of the flashing (warning) light and the buffer overflow and is approximately a quarter of the time required to fill up the buffer. Regular use of the Report Generation program prevents buffer overflow.

NOTE

If buffer overflow has occurred, it is necessary to stop the instrument, run the report generator program and restart the instruments at baseline.

The procedure to run report generator program or any other previously written BASIC program is as follows: when the terminal is sitting idle and has typed READY, place the cassette containing the desired program on unit 1.

Type the RUN command followed by the name of the program which will report patient data. Example: for a program named RGEN, type the following:

```
RUN 1:RGEN<CR>
```

The information in the buffer (i.e., patient data) will then be printed on the console terminal. When this has been completed the terminal responds with READY and another program can be run.

NOTES

1. If the results produced during instrument checkout or calibration are not to be stored in memory, they should be perfomed prior to pressing the start button. Caution: be sure the instrument is back to baseline before starting acquisition.

2, During the time that PDL is acquiring data from the user's instruments, the computer can be used for calculating, for quality control, or to run any other program written in BASIC.

CHAPTER 5

PDL ERROR MESSAGES

An error message is produced whenever PDL encounters a condition which conflicts with the program's normal functional procedure. The purpose of an error message is to advise the programmer of a problem and suggest suitable corrective action. Under such circumstances, the execution of the erroneous command or statement is either terminated or suspended, depending on the severity of the error. The appropriate error message is then printed on the console terminal.

Errors fall into two categories:

Fatal errors
Nonfatal errors

A fatal error cancels all work in progress. The corresponding fatal error message is printed and PDL then returns to the ready state, awaiting further instructions. Although a fatal error does not stop data acquisition it does cause loss of data. Therefore, after the error has been corrected, the program must be restarted from the beginning.

A nonfatal error has less drastic results. An error message is printed out while the particular statement or command, which caused the fault, is merely suspended pending corrective action by the programmer. Data is retained during this procedure, PDL continues with the program as if it had not been interrupted.

Error messages appear in one of two formats:

Format one: ?xxx (Immediate Mode)

where xxx consists of an abbreviation (for example, ?FNO: file not open) denoting the error condition.

Format two: message at line nnnn

where nnnn is the line number of the statement causing the error.

The following error conditions apply specifically to PDL. Any error condition not found in this section may be considered to be a BASIC-CAPS-11 error condition.

A list of BASIC CAPS-11 error conditions, together with their corresponding remedial actions, is contained in Appendix B.

PDL ERROR CONDITIONS:

PDL ERROR MESSAGES

FATAL ERRORS

Message	Explanation
RSC	Reinitialization Size Change. Tried to change channel buffer area after specified by initialization. The channel buffer size can be changed only by means of the initialization program (SETUP). Remedy Create new save file by rerunning system, initialization program or use the original buffer specified.
NMB	No More Buffer. Insufficient buffer. space for all channel combined. Remedy Re-allocate less buffer space per channel.
DTS	Dimension Too Small. Data or flag array was dimensioned with an array to small to hold data. Remedy Change dimension statement in BASIC program.
TMD	Too Much Data. More data than specified with (CKBF). Remedy Reinitialize, or change the argument in CKBF.
CHN	CHaNnel specifications error. Wrong caNnel or no caNnel specfied with RDBS, CKBS, BFST, CLBS, STAT, HLTC, STRC, LOC. Remedy Correct the channel or reinitialize.
TIM	TIMing error. Clock routine is overworked. Remedy Remove instrument that has excessively high data rate, and refer the problem to a programmer for program analysis and correction.
CTO	Clock Table Overflow. Too many instruments with delays requested.

Remedy

Remove a delay instrument, and refer the problem to a programmer for program analysis and correction.

NON-FATAL ERRORS

NDC-# — Interface not connected to computer.

Remedy

One of the interfaces required for operation of this channel (AR-11, DR-11, etc.) is missing or inoperative. Fix or insert proper interface.

REINIT ON CHN# — This channel has been initialized previously.

Remedy

Warning only.

	INSTRUMENT ERRORS	REMEDY
Flashing light on H321 LOC out of Sync.	Buffer overflow. Too much data acquired for specified channel buffer.	Re-evaluate buffer space allocated
Observe error by noting erroneous results from instrument.	Instrument and computer out of sync.	Restart instrument.

	INSTRUMENT READING ERROR	REMEDY
SMA 0-7	Noise level. The number (which will be between 0 and 7) shows the deviation from the means of the SMA. 1 Deviation = 1.6%	Re-evaluate
AA 0-7	Noise level. Indicates the level of peak and the number representing Peak are out of Sync.	Re-evalute.

See PDL Programming manual for instrument errors.

CHAPTER 6

BASIC LANGUAGE INTRODUCTION

BASIC is a conversational programming language. It is one of the easiest computer languages to learn. The English-like statements found in BASIC, together with the familiar mathematical notations, make BASIC a direct yet simple language that allows intricate problems to be expressed and performed with ease and efficiency.

BASIC CAPS is an individually tailored BASIC language used in a cassette-based operating system. BASIC CAPS has been chosen for use with PDL because of its simplicity and because the data file structure helps in the quick understanding, development, and versatile utilization of PDL.

6.1 BASIC CAPABILITIES

BASIC has the capability of operating in either the programmed mode or the immediate mode. BASIC is in the programmed mode when a computer program is actually being manipulated within the computer, and the core memory is being used for storage of data. When the computer is used as a desk calculator, BASIC is in immediate mode.

In the immediate mode, temporary data such as calculations are handled quickly. Such data is erased by subsequent calculations.

BASIC consists of statements, commands, and functions, each of which is designed to make BASIC an effective computer tool. A number at the beginning of a BASIC statement is the criterion by which BASIC determines the mode treatment of that particular statement. The presence of such a number indicates the programmed mode. The absence of such a number dictates immediate mode. Thus, the computer easily distinguishes between the two modes. If a line is preceded by a line number, the line is stored internally but not executed. If no line number precedes the statement, the line is executed immediately.

Examples:

```
10 PRINT "TEST PERIOD IN HOURS"  (programmed mode)

PRINT "TEST PERIOD IN HOURS"  (immediate mode)
```

The above two statements are shown as representative examples of the two individual modes. These modes are not interchangeable within a given program or calculation.

The first statement is treated internally as an integral part of a given program. The second statement is treated as a transitional statement and is executed immediately. The only way to save immediate data is to have the results written onto either a cassette or a printer.

6.2 IMMEDIATE MODE

The immediate mode feature greatly enhances the user's ability to obtain quick results to immediate problems, while avoiding the time and necessity for writing a program. This feature is especially useful for debugging purposes and is of particular interest where simple or infrequent calculations are to be performed.

Example: immediate calculation of the CREATININE CLEARANCE TEST.

```
PRINT  100*1000/(1.1*1440)
```

The calculated results of this statement are written onto the printer immediately. The calculation itself is not retained.

6.3 PROGRAMMED MODE

The number placed at the beginning of each statement in the programmed mode is an ideal identifier for each statement within a program. The statement number may be as long as five digits. All statements are executed in their numerical sequence regardless of the order in which they were originally typed. To allow easy insertion of additional statements at a later time, statement numbering should be written in increments of 10.

Example:
```
10    statement one
20    statement two
30    statement three
          etc...
```

6.3.1 Statements

Each statement (with or without a number) begins with a directive. This is an English word that specifies the type of operation to be performed by the part of the statement which follows. One such directive is PRINT.

Example: Immediate mode statement

```
PRINT  "CREATININE CLEARANCE CALCULATION"
```

The results on the specified terminal would be:

```
CREATININE CLEARANCE CALCULATION
```

BASIC statements are usually written as 1-line statements. This is a recommended procedure for ease of reading, program maintenance, and debugging. When space is limited or when the incremental nature of a program does not permit additional numeric insertions, multistatement lines may be used. A multistatement line consists of two or more statements written on the same line. Each statement is preceded by a backslash and in the case of programmed mode, all are referenced collectively by the number placed at the beginning of the line.

Example: multistatement line programmed mode

```
20 PRINT X=11\LET X=10\PRINT X,Y Z
```

Example: statement line statement Immediate mode

```
LET A=7\PRINT A
```

The immediate results on the specified terminal would be:

```
7
```

NOTE

For a list of BASIC statements, commands and functions refer to Appendix A of this manual.

6.4 MATHEMATICAL OPERATIONS

BASIC performs all the standard mathematical operations:

addition	A+B
subtraction	A-B
multiplication	A*B
division	A/B
exponentiaion	A^B

BASIC also performs relational operations, evaluates expressions, processes character strings, and uses variables. All these features are explained in the following paragraphs.

6.4.1 Relational Operation

Relational operations are alphabetic expressions used to compare arithmetic values. The usual relational operations are as follows:

equals	A=B
less than	A<B
greater than	A>B
not equal	A<>B

6.4.2 Evaluation of Expressions

An expression is a group of numbers, variables, or functions that are separated by arithmetic or relational operators. Such expressions may be used individually or in combination.

Example:

```
A7*(B+2-1)
```

6.4.3 Character Strings

A character string is a sequence of characters treated as a single unit. A string may consist of letters, numbers, special characters, or any combination thereof.

A character string may be the program's message to the operator. Example:

```
PRINT "CREATININE CLEARANCE TEST"
```

All characters between the quotes are considered part of the character string.

A character string enclosed in quotes is known as a string constant.

Example:

```
"NAME"
```

A string constant may also be used to assign a value to a string variable.

6.4.4 Variable

A variable is defined as a single alphabetic character or an alphabetic character followed by a single numeric character. An example of an alphanumeric variable is the algebraic symbol A2. A variable is used as the name of the storage location for data. This allows the user to store, update, and retrieve data by using the location name (variable).

Values are usually assigned to variables by indicating the value in a "LET" statement.

Example:

```
LET X=1
PRINT
1
```

Here the value 1 is assigned to X and the value is printed.

Variables may be:

singular	A2
subscripted	A2(2)
string variable	A$

A singular variable has the following formats:

	Example
single character	A or T
single character plus single digit	B3

It is called a singular variable because it has only one value assigned to it at a time. If a new value is assigned to a variable it replacs the old value.

Unacceptable formats are:

	Example
numbers	1 or 22
number plus letter	2C
two letters	RS

Subscripted variables have the following format:

	Example
Single letter followed by a number or numbers in parentheses.	Z(3)

The number in parentheses indicates the element within the variable group that is being referenced. The above example shows that the third element of group Z is being referenced.

The statement LET Z(3)=5 assigns the value 5 to the example above. A list can be considered a matrix of one dimension; and 2-dimensional matrices (i.e., doubly subscripted variables) are often used with data having two parameters. Example: T(2,5) which references the element of matrix T that is in row 5, column 2.

String variables allow the setting up of a group of characters as a unit i.e., a single variable. A string can be composed of letters, numbers, spaces, or combinations of any of these. Refer to the BASIC Language Reference Manual.

The following demonstration program introduces the user to BASIC. One of the uses of a string variable is to store, compare text. The statement A$=B$ implies that the text strings A$ and B$ are alphabetically identical.

EXAMPLE PROGRAM

This program performs the calculation of Creatinine Clearance. Each statement performs the action described to the right of the statement. For example, the first statement, PRINT "CREATININE CLEARANCE TEST", directs the computer to print on the termintal the message between the quotes.

STATEMENT	ACTION OF STATEMENT
10 PRINT "CREATINE CLEARANCE TEST"	prints on the terminal
20 PRINT "TYPE VOLUME OF URINE COLLECTED IN ML";	
30 INPUT V	Accepts urine volume typed on terminal and
40 PRINT "TEST PERIOD IN HOURS";	Gives V the value
50 PRINT T	
60 LET T=T*60	Assigns the value t*60 to the variable t
70 PRINT "URINE CREATININE CONCENTRATION";	
80 INPUT U	
90 PRINT "SERUM CREATININE CONCENTRATION";	
110 INPUT P	
120 PRINT "DO YOU WISH BODY AREA CORRECTION";	
130 INPUT Y$	
140 LET A=1 73	
150 IF Y$="Y" THEN 160 iF Y$<>"YES" THEN 190	Check answer and direct program control to line
160 PRINT "BODY AREA IN SQUARE METERS";	indicated after THEN
170 INPUT A	
180 IF A=0 THEN 120	
190 PRINT "ML OF PLASMA CLEARED PER MINUTE =";	
200 PRINT U*V\(P*T)*(1 73/A)	Performs the calculation
210 PRINT "ANOTHER CALCULATION";	
220 INPUT Y$	
230 IF Y$="Y" THEN 20	
240 STOP	Terminates the program

When the program is running, the following dialog will be printed on the terminal. The underlined sections are the operator's response to the program request for input.

```
RUN

CCTC        BASIC V01-05

CREATININE CLEARANCE TEST


TYPE VOLUME OF URINE COLLECTED IN ML?1000
TEST PERIOD IN HOURS?24
URINE CREATININE CONCENTRATION?100
SERUM CREATININE CONCENTRATION?100
DO YOU WISH BODY AREA CORRECTION?N

ML OF PLASMA CLEARED PER MINUTE = 69.4444


ANOTHER CALCULATION?N


STOP AT LINE 250

READY
```

NOTE

All numbers within BASIC are considered to be decimal numbers. If the decimal point is not shown, it is assumed to follow the last digit.

CHAPTER 7

REPORT GENERATORS

This chapter presents examples of report generation programs, explains how these programs interact with the PDL system, provides an overview of the associated subroutines, and gives directions for incorporating report generation into the system.

Chapters 3 and 5 discussed initialization (SETUP) and the operation of PDL on a daily basis. It has been stated that PDL takes data from the laboratory instruments and stores it temporarily in specific core areas (i.e., buffers). The user needs to access this data to print a report on the terminal. This is achieved by a BASIC program called the REPORT GENERATOR. The REPORT GENERATOR locates the data, associates it with a specific laboratory instrument, formats the results, scales the clinical units, and then prints the data on a DECwriter or sends it to another computer.

Report generation programs are user written. To simplify the writing procedure, a set of subroutines has been provided. These subroutines perform such functions as reading the data from temporary locations, initializing channels, and performing the various housekeeping and checking tasks. These subroutines are described in the PDL Programming Manual under BASIC subroutines. A sample program containing the subroutines is written in Section 7.2 as an example to provide the user with a general understanding of the relationship between a program and its subroutines and also to show their normal functions.

A program is merely a list of instructions that direct the computer to perform a job. The program performs this job in much the same manner as Laboratory personnel perform their jobs. In other words, laboratory personnel must go through a set procedure to achieve a given goal. If the procedure is altered, the results are different. Similarly, the instructions within a program are performed in sequence to achieve the desired result. Changes to the program instruction will alter the sequence and results of that program.

Just as laboratory personnel find it necessary to break their jobs into individual tasks (i.e., test on patients, samples, preparation of reagents, reporting of results, etc.) the program has its job broken into smaller tasks called subroutines. A subroutine defines a small number of instructions which have been selected and placed together to be used for a specific task within the programs. A subroutine may be an integral part of the program or it may be stored elsewhere in memory, awaiting a call from the main program. Whichever the case, all sections of a program (whether they are used or not) are organized so that when required, they will do their part in producing the desired results.

7.1 SUBROUTINES USED

The subroutines used in the example program are discussed briefly below. More information on these subroutines will be found in the PDL Programming Manual under BASIC SUBROUTINES.

7.1.1 INTL Subroutine

The first subroutine discussed is INTL. The Programmable Data Logger assigns a 20-word initialization file that describes each channel (instrument) that is monitored. (See Section 4.1 of the PDL Programmers Manual for complete description.) The portion of this initialization file that is specified by the user contains seven items. Each item is referenced by a capital letter. The seven items are:

CAPITAL LETTER	ITEM	DESCRIPTION
C	CHANNEL NUMBER Channel Number must be a number between 0-31. Analog Channels such as AA's and SMA's must be a number between 0-15. Digital channels such as the Coulter Model S, must be a number between 16-31. This allows the computer to know what type of instrument it is looking at. The channel number is never changed. The descriptive file might be moved to another channel, but the channel numbers are constant.	This designation tells the computer which instrument to reference by its physical channel number. The computer associates each instrument with a physical channel number. By physical we mean where on the computer the cable is connected.
L	LOC Number. This is number which designates where the LOC is physically connected. The number must be between 0-15.	A number is associated with the local operator's console attached to your instrument. It is generally thought best to make LOC number and channel number the same if possible.
I	Instrument type Numbers are 0-3 0 = single number 1 = peak instrument 2 = plateau instrument 3 = on demand	This designation tells the computer the type of instrument and the particular program to use to analyze the data. A peak instrument would be an AA, a plateau instrument would be an SMA, a single number would be a Coulter, on demand could be any type instrument. The data is read when the user presses the start button on the LOC box.

D	Data Type 0 Analog 2 BCD	Data type tells the computer the format in which the result is being presented and the appropriate action to take. All AA's and SMA's are analog. Coulter Model S is BCD.
N	Number of Data in each sample	This tells the computer how many results are to be taken on a sample. Baseline results on the SMAs are not stored so that the number of data is equal to the number of tests per sample, i.e., an SMA 6/60 would have 6 data results per sample. The one exception is on the 12/60. It has 13 data points to optimize timing. An autoanalyzer has two because both the peak and the valley are read so that baseline corrections can be made, if desired. The Coulter has 10 data per sample. The data is cycled twice but printed only once.
T	Time per test	This item tells the computer the time between tests on peaks or plateaus, etc. in intervals of 0.01 sec, for example, SMA 6/60 has 6 tests per sample which is 1 test every 10 sec, or 1 test every 1000 intervals of .01 sec. To calculate this easily, multiply the time per test (not per sample) by 100. Note: For instrument type 12/60 T = 462 intervals.

Descriptive items (arguments) allow communication with subroutines. Some of the arguments hold information used by the subroutine such as instrument and temporary location of data or they can contain information returned from the subroutine.

7.1.2 RDBF Subroutine

In RDBF (read buffer) we have five descriptive arguments (items).

They are:

C Channel number

S Sample or Cup number

A Data Buffer area. If there is more than one result per sample, then it must be in the form of a dimensioned array. See BASIC-11 Language Reference manual. (DEC-11 -LIBBA-A-D), Section 2.4.

F Storage area for flag information. This data storage should also be an array if more than one result is anticipated. A flag is a means of displaying error condition information. For a more complete description of flags, refer to PDL Programmers Manual.

N Optional argument. Refer to the PDL Programmers Manual.

Example:

```
        C
RDBF (0, S, A, F)
```

This statement tells the program to read the buffer associated with channel 0 and put sample (cup) number in location S and data in location A, and error condition (flag) information in location F. This information can then be printed by means of a BASIC PRINT command.

Example: If S is a single value, the command to print out would be: PRINT S. This prints the sequential sample number. To equate cup number with sequential sample number, subtract the number of first cup and add result to S in a BASIC program.

Example: If you begin at cup 100, subtract 1 and add 99 to the variable S. S then represents 100.

7.1.3 BFST Subroutine

The subroutine called BFST (buffer status) has the descriptive arguments:

C Channel number
X number of unused storage words in the buffer for channel C.

The purpose of using this subroutine is to return to the user the number of unused buffer words. Optional arguments can be used with this subroutine, refer to the PDL Programmers Manual for description.

Example:

```
BFST (0, X) PRINT "X =";X.
```

This example will print the number of unused words on channel 0 as the value of X.

X = 0 means not ready
X = 1 means ready
X > 0 Data partially removed

7.1.4 STAT Subroutine

STAT is a subroutine that allows the user to access a buffer area for the data (result) associated with a sample (cup) number. This allows the user the availability of printing out STAT results, without waiting for all the results to print. Like the RDBF subroutine, STAT has several arguments, except the user must input the sample number instead of receiving one.

C Channel number
S Sample number
A Data Buffer Area
F Flag storage location
N Optional, refer to the PDL Programmers Manual

7.2 SAMPLE PROGRAM

The previously described subroutines and others are used in report generation programs to aid the user in accessing the data that is being collected by PDL. The following short program uses some of the subroutines mentioned and explains their use. It is assumed that SETUP was used for channel descriptions, therefore the INTL subroutine is not used. The program will generate a report based on data from an SMA 6/60. First, assume that the appropriate channel (call it 0) has been set up and PDL has been called so that the computer is ready to collect data from the instrument. A RDBF call must be used to bring the data to BASIC so that the routine can be as follows (review the REM, DIM, IF, FOR, NEXT, GOTO, and PRINT statements in the BASIC-11 Language Reference manual):

```
10    REM SMA 6/60 PROGRAM
20    DIM A(5)
30    RDBF (0, S, A)
40    IF A(0) = -1 GOTO 30
50    PRINT "SMA 6/60 S=";S
60    FOR I = 0 TO 5
70    PRINT A(I);
80    NEXT I
90    PRINT
100   GOTO 30
110   END
```

The first statement, 10, is a REMark statement and is not printed when the program is run. It is included to remind the programmer why the program or part of the program was written. In this case, it serves to remind the programmer that this is an SMA 6/60 report generation.

The second statement, 20, allocates room for six values to be stored in an array called A. The first of these values is referenced by A(0), the second by A(1), etc.

The third statement is a call to the PDL subroutine RDBF to get the results obtained for a given sample from the data buffer area. The variable S contains (is set equal to) the current sequential sample number while the array, A, holds the six results for channel 0. Line 40 is an IF statement which returns to line 30 if A(0) is equal to -1. This condition will be true whenever a full sample is not ready in the data buffer area. Therefore, lines 30 and 40 wait for a full sample to be processed. Line 50 simply prints a header with the sequential sample number. The output might look like the:

```
SMA 6/60 S=5    (S = 5 is the sample number)
```

Lines 60 through 80 are a FOR loop. Their effect is the same as the 1 line statement:

```
70 PRINT A(0);  A(1);  A(2);  A(3);  A(4);  A(5);
```

Line 90 forces a new line to be printed. Line 100 returns to waiting for another sample. This program lacks several obvious and important features. It does not provide any way to exit, so it just keeps running forever. There are no headings for the data and the values printed are not in usual laboratory units.

Take the last problem first. The actual computer viewpoint numbers corresponding to the laboratory viewpoint results must be known in order to convert the first to the second.

Normally a simple multiplication by a number known as scale factor for each test in a sample must be determined. A routine listed in the PDL Programmers Manual will do the calculation of the scale factors automatically. All that is needed in the example program is to read these numbers and use them. The following program reflects these changes (review the INPUT and LET statements the BASIC-11 Language Reference manual):

```
10    REM SMA 6/60 PROGRAM
20    DIM A(5), Q(5)
30    FOR I = 0 TO 5
40    INPUT Q (I)
50    NEXT I
60    RDBF (0, S, A)
70    IF A(0) = -1 GOTO 60
80    PRINT "SMA 6/60 S=";S
90    FOR I = 0 TO 5
100   PRINT A(I) * Q(I);
110   NEXT I
120   PRINT
130   GOTO 60
140   END
```

Note that an additional item in the dimension list has been added for the scale factors and that the INPUT statement (in a FOR loop) is used to read these factors as they are typed. The only other change is the use of these scale factors on line 100 to do the conversion.

Next, the program must be altered so that it is possible to stop it without losing data and then to start it up again. A special PDL subroutine, CNDX, is needed for this step. CNDX(X) will return with X=1 if a CTRL/V (hold CTRL key down while typing V) has been typed out since the last call to CNDX. It will return X=0 otherwise. Since it is a good idea to test for this feature continuously, not just when a sample has been printed, the following program provides this call while waiting for a new sample to be complete. (Review the STOP statement in the BASIC-11 Language Reference manual):

```
10    REM SMA 6/60 PROGRAM
20    DIM A(5), Q(5)
30    PRINT "TYPE THE SIX SCALE FACTORS"
40    FOR I = 0 TO 5
50    INPUT Q(I)
60    NEXT I
```

```
70    RDBF (0, S, A)\IF A(0) = -1 GOTO 130
80    PRINT "SMA 6/60 S=";S
90    FOR I = 0 TO 5
100   PRINT A(1) *Q(I);
110   NEXT I
120   PRINT
130   CNDX (X)\ IF X = 0 GOTO 70
140   PRINT "EXIT TAKEN"
150   STOP
160   END
```

Since the appearance of a question mark on the terminal with no explanation can be disconcerting, a message to the user has been added. Also, where logical groupings make the combination clear, multistatement lines have been added at lines 70 and 130.

The backslash (\) separates the statements. Note that waiting now consists of cycling between lines 70 and 130. The message, EXIT TAKEN, will appear if a CTRL/V is typed followed by the BASIC messages:

```
STOP AT LINE 150

READY
```

We have included the following sample report generator programs, Auto Analyzer Report Generator, Coulter Report Generator, SMA Report Generator. For an example of a program that combines all three, see the PDL Programmers Manual.

```
10 REM SIMPLE COULTER COUNTER REPORT GENERATOR
20 PRINT "COULTER REPORT GENERATOR"\PRINT\PRINT
30 DIM A(9),F(9),H$(9),U$(9),S(9)
40 PRINT "TO WHICH CHANNEL IS THE COULTER ATTACHED";\INPUT C
50 C=INT(C)\T$="TIT"&STR$(C)\IF C>15 GOTO 70
60 PRINT "NOT A LEGAL CHANNEL FOR DIGITAL INPUT"\GOTO 40
70 CALL "BEST"(C,X,S,N,T,Z)
80 PRINT "CHANNEL";C;"INITIALIZED FOR";N;"DATA PER SAMPLE"
90 PRINT "WITH";Z;"WORD BUFFER. ";X;"WORDS LEFT."
100 PRINT "BUFFER WILL FILL AFTER";INT(X/N);"MORE SAMPLES"
110 PRINT "CURRENT SAMPLE NUMBER IS";S
120 H$ = ""\F$=""
130 FOR I=0 TO N-2
140 H$=H$&" >###### "\F$=F$&"    -    "
150 NEXT I
160 U$=H$
170 PRINT "IS FORMATTING INORMATION ON CASSETTE";
180 INPUT Y$\IF Y$='Y' GOTO 190\IF Y$<>'YES' GOTO 290
190 OPEN T$ FOR INPUT AS FILE #1
200 FOR I=0 TO N-2
210 INPUT #1:H$(I),U$(I),S(I)
220 NEXT I
230 CLOSE #1
240 A$=""
250 FOR I=0 TO N-2
260 GOSUB 1000
270 NEXT I
280 GOTO 660
290 PRINT\PRINT "GIVE HEADINGS FOR EACH TEST IN ORDER"
300 PRINT "MAXIMUM OF 6 CHARACTERS EACH"
310 FOR I=0 TO N-2
320 PRINT I+1;":";\INPUT H$(I)
```

```
330 NEXT I
340 PRINT "GIVE THE UNITS FOR EACH TEST IN ORDER"
350 FOR I=0 TO N-2
360 PRINT I+1;":";\INPUT US(I)
370 NEXT I
380 PRINT "GIVE THE NUMBER OF DIGITS AFTER THE DECIMAL"
390 PRINT "POINT(0-3) FOR EACH TEST IN ORDER."
400 PRINT "I.E., 3.45 WOULD BE 2, 123 WOULD BE 0"
410 AS=""
420 FOR I=0 TO N-2
430 PRINT I+1;":";\INPUT S(I)
440 S(I)=INT(S(I))
450 GOSUB 1000
460 S(I)=10^-S(I)
470 NEXT I
480 PRINT "DO YOU WANT THESE FACTORS SAVED ON CASSETTE";\INPUT YS
490 IF YS='Y' GOTO 500\IF YS<>'YES' GOTO 660
500 OPEN TS FOR OUTPUT AS FILE #1
510 FOR I=0 TO N-2
520 PRINT #1:HS(I)
530 PRINT #1:US(I)
630 PRINT #1:S(I)
640 NEXT I
650 CLOSE #1
660 CALL "RDBF"(C,S,A,F)\IF A(0)<0 THEN 660
670 PRINT USING HS,HS(0),HS(1),HS(2),HS(3),HS(4),HS(5),HS(6),HS(7),HS(8)
680 PRINT USING US,US(0),US(1),US(2),US(3),US(4),US(5),US(6),US(7),US(8)
690 FOR I=0 TO N-2
700 A(I)=A(I)*S(I)
710 NEXT I
720 PRINT USING FS,F(1),F(2),F(3),F(4),F(5),F(6),F(7),F(8)
730 PRINT USING AS,A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8)
740 PRINT
750 CALL "CNDX"(X)\IF X=0 GOTO 660
760 PRINT "REQUESTED HALT"
770 STOP
1000 AS=AS&" -"&SEG$("###.###",S(I)+1,S(I)+4)&"  "
1010 RETURN
9999 END


10 REM SIMPLE AUTOANALYZER REPORT GENERATOR
20 PRINT "AUTOANALYZER REPORT GENERATOR"\PRINT\PRINT
30 DIM S(20),F(2),S1(20)
40 PRINT "TO WHICH CHANNEL IS THE AUTOANALYZER ATTACHED";\INPUT C
50 C=INT(C)\CALL "BFST"(C,X,S1,N1)\IF N1=2 THEN 70
60 PRINT "THIS CHANNEL INITIALIZED FOR"N1"DATA PER SAMPLE"\STOP
70 PRINT "FOR QUICK REPORT, MOUNT THE SAVE CASSETTE AND TYPE Y"
80 GOSUB 2000\IF YS<>'Y' THEN 90\Z=1\GOTO 640
90 Z=0
100 PRINT "THERE ARE";N;" WORDS LEFT IN THE BUFFER FOR CHANNEL";C
110 PRINT "SAMPLE NUMBER IS"S1
120 PRINT "ARE STANDARDS TO BE RUN";\GOSUB 2000
130 IF YS<>'Y' THEN 730
140 PRINT "HOW MANY STANDARDS ARE THERE";\INPUT N\N=INT(N)
150 PRINT "WHAT IS THE SEQUENTIAL SAMPLE NUMBER OF THE FIRST STANDARD";
160 INPUT S
170 PRINT "PROGRAM WILL WAIT UNTIL LAST STANDARD IS IN BUFFER"
180 CALL "STAT"(C,S+N-1,S)
190 IF S(0)<>-2 THEN 200\PRINT "DATA HAS ALREADY BEEN REMOVED"\GOTO 20
200 IF S(0)=-1 THEN 180
210 PRINT "LAST DATUM IS IN BUFFER"
220 FOR I=1 TO N
```

```
230 CALL "STAT"(C,S+I-1,S)\IF S(0)<0 THEN 190
240 S(I+1)=S(1)\REM 2ND DATUM IS HEIGHT
250 NEXT I
260 PRINT "TYPE THE STANDARD VALUES"
270 FOR I=1 TO N\INPUT S1(I)\NEXT I
280 PRINT "LINEAR(1) OR CURVED(2) FIT";\INPUT K\K=INT(K)
290 IF K>2 GOTO 280\IF K<1 GOTO 280
300 Y=0\Y1=0\Y2=0\X1=0\X2=0\X3=0\X4=0
310 FOR I=1 TO N
320 Y=Y+S1(I)
330 J=I+1\Y1=Y1+S1(I)*S(J)
340 S2=S(J)*S(J)
350 Y2=Y2+S1(I)*S2
360 X1=X1+S(J)
370 X2=X2+S2
380 X3=S3+S(J)*S2
390 X4=X4+S2*S2
400 NEXT I
410 IF K<2 GOTO 480
420 D=X4*(X2*N-X1*X1)-X3*(X3*N-X2*X1)+X2*(X3*X1-X2*X2)
430 IF (ABS(C))^(1/6)<.0001*X1/N THEN 480
440 A2 = (Y2*(N*X2-X1*X1)-Y1*(N*X3-X2*X1)+Y*(X3*X1-X2*X2))/D
450 A1 = (Y2*(N*X3-X2*X1)-Y1*(X4*N-X2*X2)+Y*(X4*X1-X3*X2))/D
460 A0 = (Y2*(X3*X1-X2*X2)-Y1*(X4*X1-X3*X2)+Y*(X4*X2-X3*X3))/D
470 GOTO 560
480 A2=0
490 D=N*X2-X1*X1
500 IF SQR(ABS(D))<.0001*X1/N THEN 540
510 A1 =(N*Y1-X1*Y)/D
520 A2 =(X2*Y-X1*X1)/D
530 GOTO 560
540 A1 = 0
550 A0 = Y/N
560 PRINT "THE EQUATION IS:"
570 IF A2=0 THEN 590
580 PRINT A2;"X^2";
590 IF A1=0 THEN 610
600 PRINT "+";A1;"X";
610 IF A0=0 THEN 630
620 PRINT "+";A0;
630 PRINT\PRINT "COEFFICIENTS:";A2,A1,A0
640 F$="STD"&STR$(C)\IF Z=1 THEN 780
650 PRINT\PRINT "DO YOU WANT THESE COEFFICIENTS SAVED"
660 GOSUB 2000\IF Y$<>'Y' THEN 800
670 PRINT "PLACE APPROPRIATE CASSETTE ON UNIT 0"
680 PRINT "TYPE ANY CHARACTER WHEN READY"\GOSUB 2000
690 OPEN F$ FOR OUTPUT AS FILE #1
700 PRINT #1:A2,A1,A0\CLOSE #1
710 PRINT "A FILE NAMED ";F$".DAT HAS BEEN CREATED"
720 GOTO 800
730 PRINT "ARE THE COEFFICIENTS ON CASSETTE"\GOSUB 2000
740 IF Y$='Y' THEN 780
750 PRINT "THEN TYPE THEM IN ORDER"
760 INPUT A2\INPUT A1\INPUT A0
770 GOTO 800
780 OPEN F$ FOR INPUT AS FILE #1
790 INPUT #1:A2,A1,A0\CLOSE #1
800 T$="TIT"&STR$(C)\IF Z=1 THEN 880
810 PRINT "ARE THE HEADINGS ON CASSETTE";\GOSUB 2000
820 IF Y$='Y' THEN 880
830 PRINT "WHAT IS THE HEADING"\INPUT H$
840 PRINT "UNITS"\INPUT U$
```

```
850 PRINT "SAVE ON CASSETTE";\GOSUB 2000\IF Y$<>'Y' THEN 880
860 OPEN T$ FOR OUTPUT AS FILE #1
870 PRINT #1:H$,U$\CLOSE #1\GOTO 900
880 OPEN T$ FOR INPUT AS FILE #1
890 INPUT #1:H$,U$\CLOSE #1
900 PRINT "CURRENT SEQUENTIAL SAMPLE NUMBER IS";S\IF S=1 THEN 1000
910 PRINT "DO YOU WANT TO SKIP OVER THE STANDARDS"?
920 GOSUB 2000\IF Y$<>'Y' THEN 970
930 FOR I=S1 TO S+N-1
940 CALL "RDBF"(C,X,S)
950 NEXT I
960 PRINT "NEW SEQUENTIAL SAMPLE NUMBER IS";S+N
970 PRINT "WHAT CUP NUMBER DO YOU WANT FOR THIS SAMPLE";\INPUT C1
980 CALL "BFST"(C,X,N1,S1)
990 C1 = C1-S1+1\REM S1 IS LAST SMP NUM
1000 PRINT\PRINT
1010 PRINT H$
1020 PRINT "CHANNEL";C
1030 CALL "RDBF"(C,S,S,F)\IF S(0)<0 GOTO 1180
1040 PRINT "SAMPLE";S+C1
1045 S1=S(1)\S(1)=A0+S1*(A1+A2*S1)
1050 S(2)=INT(S(1))\S(3)=INT(100*(S(1)-S(2))+.5)
1060 PRINT STR$(S(2));".";STR$(S(3));" ";U$;
1070 IF F(0)=0 THEN 1170
1080 F(0)=F(1)\F(1)=INT(F(0)/8)\F(2)=INT((F(0)-F(1)/4)
1090 F(0)=F(0)-F(1)*8-F(2)*4
1100 IF F(1)=0 THEN 1120
1110 PRINT "*";
1120 PRINT
1130 IF F(2)=0 THEN 1150
1140 PRINT "TIMING"
1150 IF F(0)=0 THEN 1170
1160 PRINT "NOISE LEVEL";INT(1.6*F(0));"%"
1170 PRINT\PRINT
1180 CALL "CNDX"(X)
1190 IF X=0 GOTO 1030
1200 STOP
2000 INPUT Y$\IF Y$<>'YES' GOTO 2020
2010 Y$='Y'
2020 RETURN
9999 END

10 REM SMA REPORT GENERATOR--ASSUME PROPER INTL AND PDL CALLS
20 DIM H$(12),U$(12),S(12),L$(12),A(12),F(12),N$(12)
30 PRINT "SMA REPORT GENERATOR"
40 PRINT "WHICH CHANNEL";\INPUT C
50 IF C>15 GOTO 40\IF C<0 GOTO 40
60 CALL "BFST"(C,X,S,N,T)\IF X>-1 THEN 80
70 PRINT "CHANNEL";C;"IS NOT INITIALIZED."\STOP
80 PRINT "THIS CHANNEL HAS BEEN INITIALIZED FOR ";N;"TESTS PER SAMPLE"
90 PRINT "THERE ARE ";Z;"WORDS REMAINING IN THE BUFFER."
100 PRINT "AT A RATE OF";T/100;"SECONDS PER POINT,"
110 PRINT "THE BUFFER WILL FILL IN";X*T/6000;"MINUTES."
120 PRINT "GIVE THE HEADINGS FOR EACH TEST IN ORDER"
130 PRINT " (MAXIMUM 5 CHARACTERS)"
140 FOR I=0 TO N-1
150 PRINT "HEADING FOR COLUMN #";I+1
160 INPUT H$(I)
170 NEXT I
180 PRINT "GIVE THE UNITS FOR EACH TEST (MAXIMUM 5 CHARACTERS)"
190 FOR I=0 TO N-1
200 PRINT "UNITS FOR";H$(I);"(COLUMN";I+1;")";\INPUT U$(I)
```

```
210 NEXT I
220 PRINT "GIVE THE FULL SCALE VALUE OF EACH TEST."
230 FOR I=0 TO N-1
240 PRINT H$(I);"(COLUMN";I+1;") IN UNITS OF";U$(I)
250 PRINT "MAX IS--";\INPUT S(I)
260 S(I)=S(I)/980\NEXT I
270 FOR I=0 TO N-1
280 L=6-LEN(H$(I))\GOSUB 2000\L$(I)=L$
290 L=6-LEN(U$(I))\GOSUB 2000\M$(I)=L$
300 NEXT I
310 PRINT\PRINT
320 FOR I=0 TO N-1
330 PRINT L$(I);H$(I);
340 NEXT I
350 PRINT
360 FOR I=0 TO N-1
370 PRINT M$(I);U$(I)
380 NEXT I
390 PRINT\A$=''
400 CALL "RDBF"(C,S,A,F)\IF A(0)<0 THEN 400
410 PRINT\PRINT "SAMPLE NUMBER";S
420 FOR I=0 TO N-1
430 F=INT(F(I)/16)\F1=INT((F(I)-16*F)/8)\F2=F(I)-16*F1-8*F2
440 F$=''\IF F=0 THEN 450\F$='M'\GOTO 470
450 IF F1=0 THEN 460\F$='T'\GOTO 470
460 IF F2=0 THEN 470\F$=STR$(F2)
470 A(I)=A(I)*S(I)
480 IF A(I)<100000 GOTO 500
490 A$='***** '\GOTO 510
500 A$=SEG$(STR$(A(I)),1,5)&F$
510 PRINT A$;
520 NEXT I
530 CALL 'CNDX'(X)
540 IF X=0 GOTO 390
550 PRINT 'REQUESTED HALT'
560 STOP
2000 L$=''
2010 FOR I=1 TO L
2020 L$=L$&' '
2030 NEXT I
9999 END
```

These programs can be used on any system provided the user accepts the output format of the results. If the results produced do not meet the users needs, the programs will have to be modified.

To use the programs as they are: Mount a new cassette on unit one. The cassette should have been previously zeroed. If PDL is not running, load PDL on cassette unit 0 and boot it. Follow through with the normal dialog until the computer issues the message READY. If PDL is already running the CTRL key must be held down while V is typed. The message printed will be:

```
READY
```

SCR<CR> is then typed. This creates a clean memory so that the program will not contain any errors.

Next, RENAME (Name of Program) is typed.

Example:

```
RENAME SMA <CR>
```

Each line of the program must be typed sequentially (as you see it written).

Example:

```
10    REM SIMPLE AUTO ANALYZER REPORT GENERATOR <CR>
20    ETC.
```

NOTE

Remember to press the <CR> key after each line. When typing of the program has been completed, type the following:

SAVE 1: <CR>

This puts the above program you have typed on to cassette unit # 1. For future use this program may now be called from cassette instead of having to be retyped. If the user wants to use the other programs, simply repeat the procedure and save them on cassette 1.

When PDL is already running, simply load the cassette containing the report generators, on either unit 0 or 1, then call the program which is to be run, by name.

Example: PDL is doing data acquisition on an SMA 6/60 and it is necessary to have a report printed when the terminal says READY. The cassette containing the SMA 6/60 report generator program is loaded onto cassette unit 1.

Type RUN SMA6 <CR> (SMA6 is the name of the program). The program will be loaded into memory and the SMA 6 data will be printed out.

APPENDIX A

BASIC STATEMENTS, COMMANDS, FUNCTIONS

A.1 BASIC/CAPS

The following summary of BASIC statements defines the general format for each statement and gives a brief explanation of its use.

CALL (function name, argument list)	Used to reference assembly language user functions from a BASIC program.
DATA value list	Used in conjunction with READ to input data into an executing program.
DEF function (argument)=expression	Defines a user function to be used in the program.
DIM variable(n),variable(n,m),variable$(n),variable$(n,m)	Reserves space for lists and tables according to subscripts specified after variable name.
END	Placed at the physical end of the program to terminate program execution.
FOR variable = expression 1 TO expression 2 STEP expression 3	Sets up a loop to be executed the specified number of times.
GOSUB line number	Transfers control to the first line of a subroutine.
GOTO line number	Unconditionally transfers control to the program line specified by the line number.
IF expression {THEN / GOTO} line number	Conditionally transfers control to the specified program line.
INPUT list	Used to input data from the terminal keyboard or paper tape reader.

INPUT #expression: list	Inputs from a particular device.
[LET] variable = expression	Assigns a value to the specified variable(s).
NEXT variable	Placed at the end of a FOR loop to return control to the FOR statement.
PRINT list	Prints output data on the terminal printer.
PRINT expression	Prints result of expression.
PRINT "string"	Prints a character string.
PRINT #expression: expression list	Outputs to a particular output device.
PRINT TAB(x)	Spaces the printing head to the specified column.
RANDOMIZE	Causes the random number generator to calculate different random numbers every time the program is run.
READ variable list	Assigns the values listed in a DATA statement to the specified variables.
REMARK comment	Inserts explanatory comments into a BASIC program.
RESTORE	Resets data block pointer so the same data can be used again.
RETURNS	Returns program control to the statement following the last executed GOSUB statement.
STOP	Terminates execution of the program.

A.2 COMMANDS

The following key commands halt program execution, erase characters, or delete lines.

Key	Explanation
ALTMODE	Deletes the entire current line. Prints DELETED message (same as CTRL/U). On some terminals the ESC key must be used. This key is not used in PDL.
CTRL/C	Terminates program execution. BASIC prints READY.

CTRL/O	Stops output to terminal and returns BASIC to READY message when program or command execution is completed.
CTRL/U	Deletes the entire current line. Echoes DELETED message (same as ALTMODE).
RUBOUT	Deletes the last character typed and echoes a backarrow (same as _).

The following commands list, punch, erase, execute, and save the program currently in core.

Command	Explanation
CLEAR	Sets the array and string buffers to nulls and zeroes.
LIST	Prints the user program currently in core on the terminal.
LIST line number	Prints the program from the line specified to the end of the program.
OLD #n	Does a SCRatch and inputs the program from the terminal (n=0), or high-speed paper tape reader (n=1).
RUN	Executes the program in the buffer area.
SAVE #n	Outputs the program in core to the terminal (n=0), paper tape punch (n=1), or line printer (n=2).
SCRatch	Erases the entire storage area.

A.3 FUNCTIONS

The following functions perform standard mathematical operations in BASIC.

Name	Explanation
ABS(x)	Returns the absolute value of x.
ATN(x)	Returns the arctangent of x as an angle in radians in the range + or - pi/2.
COS(x)	Returns the cosine of x radians.
EXP(x)	Returns the value of e^x where e=2.71828.
INT(x)	Returns the greatest integer less than or equal to x.

LOG(x)	Returns the natural logarithm of x.
RND(x)	Returns a random number between 0 and 1.
SGN(x)	Returns a value indicating the sign of x.
SIN(x)	Returns the sine of x radians.
SQR(x)	Returns the square root of x.
TAB(x)	Causes the terminal type head to tab to column number x.

The string functions are:

ASC(x$)	Returns as a decimal number the 7-bit internal code for the 1-character string (x$).
CHR$(x)	Generates a 1-character string having the ASCII value of x.
LEN(x$)	Returns the number of characters in the string (x$).
POS(x$,y$,z)	Searches for and returns the position of the first occurrence of Y$ in x$ after skipping z characters.
SEG(x$,y,z)	Returns the string of characters in positions Y through z in x$.
STR$(x)	Returns the string which represents the numeric value of x.
VAL(x$)	Returns the number represented by the string (x$).

APPENDIX B

BASIC ERROR MESSAGES

When BASIC encounters an error in the execution of a statement or command, it halts. An error message is then printed out on the DECwriter.

BASIC error messages are considered either fatal or nonfatal errors.

A fatal error cancels all work in progress and returns BASIC to the READY state. At this point, if the erroneous condition is removed, the program maybe restarted from the beginning.

A nonfatal error allows the removal of the error condition, without the need for a return to the beginning of the procedure. When the error has been removed, the suspended action of BASIC continues as if uninterrupted.

All error messages are printed in one of the following formats

```
message
```

or

```
message AT LINE xxxxx
```

where xxxxx is the line number of the statement containing the error. Error messages produced in immediate mode statements or commands will not include AT LINE xxxxx. Nonfatal error messages do not include AT LINE xxxxx and do not return to READY.

Table B-1 lists all BASIC error messages. The message produced will be the abbreviation unless BASIC-11 has been assembled from the sources with longer error messages specified. All error messages are fatal unless the explanation specifies nonfatal.

Table B-1
BASIC Error Messages

Abbreviation	Longer Message	Explanation
?ARG	ARGUMENT ERROR	Arguments in a function call do not match, in number or in type, the arguments defined for the function.
?ATL	ARRAYS TOO LARGE	Not enough memory is available for the arrays specified in the DIM statements. If the arrays can not

		be redimensioned to a smaller dimension, then reduce the size of the program by eliminating remarks or by using subroutines, user-defined functions, chaining, or overlaying.
B?		Nonfatal, BASIC/CAPS system overlay file not found on cassette unit 0. Mount a new cassette and type any character to continue. Occurs only in 8K version.
BAD DATE		Nonfatal, an illegal date was typed during initial diaogue. Retype to continue.
?BDR	BAD DATA READ	Item input from data statement list by READ statement is bad.
?BOV	BAD OVERLAY FILE	Nonfatal, BASIC/CAPS system overlay file was found but contained an error. Also prints B? message. Retry or mount new cassette with overlay files. BASIC/CAPS may also be rebooted. occurs only in 8K version.
?BRT	BAD DATA-RETYPE FROM ERROR	Nonfatal, item entered to input statement was bad. Retype and program will continue.
?BSO	BUFFER STORAGE OVERFLOW	Not enough room available for file buffers.
?DCE	DEVICE CHANNEL ERROR	The device channel number, also called the logical unit number, specified in OPEN statement has been previously opened or is out of range (1-7).
?DNR	DEVICE NOT READY	A hardware I/O device referred to by an OLD, SAVE, or PRINT command is not on-line or the file does not contain any legal BASIC program lines.
?DV0	DIVISION BY 0	Program attempted to divide some quantity by 0.
?ETC	EXPRESSION TOO COMPLEX	The expression being evaluated caused the stack to overflow usually because the parentheses are nested too deeply. The degree of complexity that produces this error varies

		according to the amount of space available in the stack at the time. Breaking the statement up into several simpler ones eliminates the error.
?FIO	FILE I/O ERROR	A hardware I/O error occurred. All files are automatically closed.
?FNF	FILE NOT FOUND	The file requested was not found on the specified device. May be caused by an OPEN FOR INPUT, OVERLAY, or CHAIN statement or an OLD, APPEND, RUN[NH] file descriptor , or REPLACE command.
?FNO	FILE NOT OPEN	The logical unit number specified is not associated with an open file. May be caused by PRINT #, INPUT #, or CLOSE statements.
?FRM	FORMAT ERROR	Format string error occurred in PRINT USING statement or an attempt was made to print string in numeric field or vice versa.
?FSE	FILE SPECIFICATION ERROR	The file descriptor contained an illegal device or character. Legal characters are A through Z and 0 through 9.
?FWN	FOR WITHOUT NEXT	The program contains a FOR statement without a corresponding NEXT statement to terminate the loop.
?GND	GOSUBS NESTED TOO DEEPLY	Program GOSUB nested to more than 20 levels.
?IDF	ILLEGAL DEF	The DEFine function statement contains an error.
?IDM	ILLEGAL DIM	In either a DIMENSION or a COMMON statement, subscript is not an integer number or array has been dimensioned previously.
?ILN	ILLEGAL NOW	Execution of INPUT statement was attempted in immediate mode.
?ILR	ILLEGAL READ	A write-only device was opened for input or an attempt was made to input from a file opened for output.

?LTL	LINE TOO LONG	The line being typed is longer than 120 characters; the line buffer overflows.
?NBF	NEXT BEFORE FOR	The NEXT statement corresponding to a FOR statement precedes the FOR statement.
?NER	NOT ENOUGH ROOM	Cassette is full and there is not enough room to open file. May be caused by an OPEN FOR OUTPUT statement or a SAVE or REPLACE command.
?NPR	NO PROGRAM	The RUN command has been specified, but no program has been typed in.
?NSM	NUMBERS AND STRINGS MIXED	String and numeric variables may not appear in the same expression, nor may they be set equal to each other; for example, A$=2.
OFFLINE{0 ? 1}		Nonfatal, cassette unit indicated was off-line. Cassette should be mounted and operation will continue.
?OFFLINE{0 1}		Fatal cassette error, occurs when cassette becomes offline after I/O has started.
?OFO	OUTPUT FILE OVERFLOW	Reached end of cassette while outputting to file.
?OOD	OUT OF DATA	The data list was exhausted and a READ requested additional data.
?OVF	OVERFLOW	The result of a computation is too large for the computer to handle.
?PTB	PROGRAM TOO BIG	The line just entered caused the program to exceed the user code area. Reduce program size by eliminating remarks and by using subroutines, user-defined functions, overlaying, and chaining.
?PWF	POWER FAIL	A power fail interrupt occurred while the specified program line was executing. All files are closed.
?RBG	RETURN BEFORE GOSUB	A RETURN was encountered before execution of a GOSUB statement.

?RPL	USER REPLACE	File saved already existed on device. Caused by SAVE command.
?SOB	SUBSCRIPT OUT OF BOUNDS	The subscript computed is greater than 32,767 or is outside the bounds defined in the DIM statement.
?SSO	STRING STORAGE OVERFLOW	Not enough memory is available to store all the strings used in the program.
?STL	STRING TOO LONG	The maximum length of a string in a BASIC statement is 255 characters.
?SYN	SYNTAX ERROR	The program has encountered an unrecognizable statement. Common examples of syntax errors are misspelled commands, unmatched parentheses, and other typographical errors.
?TIMING{0 / 1}		Cassette hardware timing error.
?TLT	LINE TOO LONG TO TRANSLATE	Lines are translated as entered and the line just entered exceeds the area available for translation.
?UFN	UNDEFINED FUNCTION	The function called was not defined by the program or was not loaded with BASIC or there was a syntax error in the first keyword on a line and BASIC translated the line as an implied CALL statement.
?ULN	UNDEFINED LINE NUMBER	The line number specified in an IF, GO TO or GOSUB statement does not exist anywhere in the program.
?WLO	WRITE LOCKOUT	Tried to write on a file opened for input or tried to open for output a read-only device.
WRT LOCK{0 / 1 / ?}		Nonfatal, cassette on unit indicated was write-locked. Write enable cassette and continue (removing cassette to write enable it will cause the nonfatal OFFLINE message).
?WRT LOCK	0 1	Cassette on unit indicated was write-locked while I/O was in progress.

?^ER/A^ ERROR/aThe program tried to compute the value A^B, where A is less than 0 and B is not an integer. This produces a complex number which is not represented in BASIC.

When the message ?DNR AT LINE xxxxx is printed because the device referred to is not on-line, turn the device on and issue a GO TO xxxxx statement. Execution of the program resumes at the line (xxxxx) specified. This message may also indicate that a program file does not contain any legal BASIC program lines.

Where the message ?OOD AT LINE xxxxx is printed because the file referred to by an INPUT# statement is not ready, prepare the file and issue a GO TO statement to resume execution.

Function Errors

The following errors can occur when a function is called improperly.

?ARG — The argument used is the wrong type. For example, the argument was numeric and the function expected a string expression.

?SYN — The wrong number of arguments was used in a function, or the wrong character was used to separate them. For example, PRINT SIN(X,Y) produces a syntax error.

In addition, the functions give the errors listed below.

FNa(...) ?UFN — The function a has not been defined (function cannot be defined by an immediate mode statement).

?SYN — A syntax error has been found in the expression in the DEF statement which defined the function. This error message is produced by statements evaluating user-defined functions, not by the statement defining the function.

RND or RND(expr)		No errors
SIN(expr)		No errors
COS(expr)		No errors
PI	?SYN	An argument was included
SQR(expr)	?ARG	Expression is negative
ATN(expr)		No errors
EXP(expr)	?^ER	Expression is greater than 87
LOG(expr)	?ARG	Expression is negative or 0
LOG10(expr)	?ARG	Expression is negative or 0

ABS(expr)		No errors
INT(expr)		No errors
SGN(expr)		No errors
TAB(expr)	?ARG	Expression is not in the range 0<x<256
LEN(string expr)		No errors
ASC(string expr)	?ARG	String expr is not a string of length 1
CHR$(expr)	?ARG	Expression is not in the range 0<x<256
POS(string expr1,string expr2,expr)		No errors
SEG$(string expr,expr1,expr2)		No errors
VAL(string expr)	?ARG	String expr is not a valid number
STR$(expr)		No errors
TRM$(string expr)		No errors
BIN(string expr)	?ARG	Character other than blank, 0 or 1 in string
OCT(string expr)	?ARG	Character other than blank or 0 through 7 in string

HOW TO OBTAIN SOFTWARE INFORMATION

SOFTWARE NEWSLETTERS, MAILING LIST

The Software Communications Group, located at corporate headquarters in Maynard, publishes software newsletters for the various DIGITAL products. Newsletters are published monthly, and keep the user informed about customer software problems and solutions, new software products, documentation corrections, as well as programming notes and techniques.

There are two similar levels of service:

- The Software Dispatch
- The Digital Software News

The Software Dispatch is part of the Software Maintenance Service. This service applies to the following software products:

PDP-9/15
RSX-11D
DOS/BATCH
RSTS-E
DECsystem-10

A Digital Software News for the PDP-11 and a Digital Software News for the PDP-8/12 are available to any customer who has purchased PDP-11 or PDP-8/12 software.

A collection of existing problems and solutions for a given software system is published periodically. A customer receives this publication with his initial software kit with the delivery of his system. This collection would be either a Software Dispatch Review or Software Performance Summary depending on the system ordered.

A mailing list of users who receive software newsletters is also maintained by Software Communications. Users must sign-up for the newsletter they desire. This can be done by either completing the form supplied with the Review or Summary or by writing to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

SOFTWARE PROBLEMS

Questions or problems relating to DIGITAL's software should be reported as follows:

North and South American Submitters:

Upon completion of Software Performance Report (SPR) form remove last copy and send remainder to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

The acknowledgement copy will be returned along with a blank SPR form upon receipt. The acknowledgement will contain a DIGITAL assigned SPR number. The SPR number or the preprinted number should be referenced in any future correspondence. Additional SPR forms may be obtained from the above address.

All International Submitters:

Upon completion of the SPR form, reserve the last copy and send the remainder to the SPR Center in the nearest DIGITAL office. SPR forms are also available from our SPR Centers.

PROGRAMS AND MANUALS

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center.

Digital Equipment Corporation
Software Distribution Center
146 Main Street
Maynard, Massachusetts 01754

Digital Equipment Corporation
Software Distribution Center
1400 Terra Bella
Mountain View, California 94043

Outside of the United States, orders should be directed to the nearest Digital Field Sales Office or representative.

USERS SOCIETY

DECUS, Digital Equipment Computers Users Society, maintains a user exchange center for user-written programs and technical application information. The Library contains approximately 1,900 programs for all DIGITAL computer lines. Executive routines, editors, debuggers, special functions, games, maintenance and various other classes of programs are available.

DECUS Program Library Catalogs are routinely updated and contain lists and abstracts of all programs according to computer line:

- PDP-8, FOCAL-8, BASIC-8, PDP-12
- PDP-7/9, 9, 15
- PDP-11, RSTS-11
- PDP-6/10, 10

Forms and information on acquiring and submitting programs to the DECUS Library may be obtained from the DECUS office.

In addition to the catalogs, DECUS also publishes the following:

DECUSCOPE	-The Society's technical newsletter, published bi-monthly, aimed at facilitating the interchange of technical information among users of DIGITAL computers and at disseminating news items concerning the Society. Circulation reached 19,000 in May, 1974.
PROCEEDINGS OF THE DIGITAL EQUIPMENT USERS SOCIETY	-Contains technical papers presented at DECUS Symposia held twice a year in the United States, once a year in Europe, Australia, and Canada.
MINUTES OF THE DECsystem-10 SESSIONS	-A report of the DECsystem-10 sessions held at the two United States DECUS Symposia.
COPY-N-Mail	-A monthly mailed communique among DECsystem-10 users.
LUG/SIG	-Mailing of Local User Group (LUG) and Special Interest Group (SIG) communique, aimed at providing closer communication among users of a specific product or application.

Further information on the DECUS Library, publications, and other DECUS activities is available from the DECUS offices listed below:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

DECUS EUROPE
Digital Equipment Corp. International
(Europe)
P.O. Box 340
1211 Geneva 26
Switzerland

Please cut along this line.

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form (see the HOW TO OBTAIN SOFTWARE INFORMATION page).

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you do not require a written reply, please check here. ☐