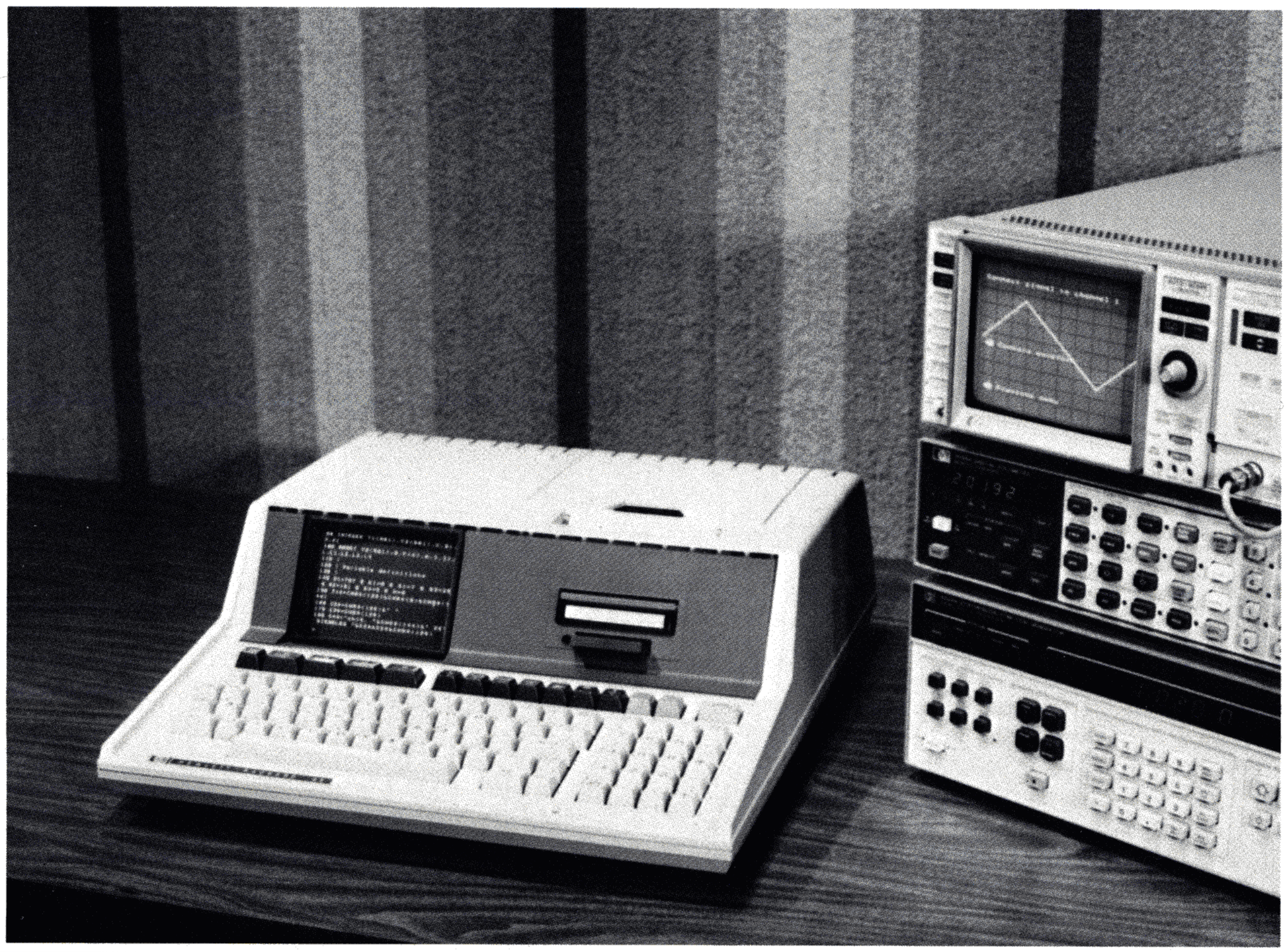
HP-IB

Systems Training With The HP-85 As A Controller



HEWLETT PACKARD

HEWLETT PACKARD

TRAINING

ON

HP-IB

PROGRAMMING TECHNIQUES

FOR THE

HP-85

HEWLETT hp PACKARD

AGENDA

Day 1	(8:30 - 5:00)	HP-85 Basic Usage
Day 2	(8:30 - 5:00)	Basic Usage, Structured Programming
Day 3	(8:30 - 5:00)	Introduction to I/O, Formatting
Day 4	(8:30 - 5:00)	HP-IB (IEEE-488) Usage
Day 5	(8:30 - 2:30)	Advanced I/O, Interrupts, Buffering & Miscellaneous Topics*

*If time permits (note: Graphics is considered a misc. topic)

NOTES:

Please fill out and return to class instructors by the end of the day.

NAME:

COMPANY:

PHONE:

PRODUCTS USING:

WHY YOU NEED THIS TRAINING? WHAT WILL YOU DO WITH IT?

HP-IB CLASS EVALUATION

DATE:

I) COURSE MATERIAL

☐ Too Basic ☐ Too General

☐ Too Advanced ☐ About Right

Recommended Changes:

II) I learned the most on day

☐ #1 ☐ #3 ☐ #5

☐ #2 ☐ #4

III) Equipment Useage

☐ Class had adequate equipment

☐ Needed more controllers

☐ Needed more instumentation

Which types? _____

_____

_____

☐ Needed more hands-on time

☐ Needed less hands-on time

Comments:

IV) Presentation

☐ Instructor(s) was knowledgeable & helpful

☐ Instructor(s) was a hinderance to learning process by

- ☐ Style
- ☐ Knowledge
- ☐ Other _____

_____

V) Change Class

☐ OK except as noted above

☐ Class too long;

- ☐ Delete material _____

_____

- ☐ Divide into 2 classes - 1 on HP-85 useage, 1 on HP-IB I/O

☐ Class too short

Material covered too quickly

Cover additional topics of _____

_____

☐ Offer specialized classes on specific equipment

_____

_____

VI) Other Comments:

HEWLETT hp PACKARD

HELPFUL HINTS

TEAM UP

SAVE YOUR PROGRAMS

USE STRUCTURED PROBLEM SOLVING
AND PROGRAMMING TECHNIQUES

NOTES:

HELPFUL HINTS

Consider teaming up with someone at a different knowledge level. If you are a novice, team up with someone who has programmed before. If you have insight to offer, please do so.

Successive lab exercises build on earlier lab programs. BE SURE to store the programs you develop in each lab for later referral or use.

Please use the following procedure in order to best utilize equipment:

A) Develop your program on paper first.

B) Enter into your computer.

C) Debug as much of your program as possible without the use of instrumentation.

D) Then request to hookup to the instrumentation if someone else is on it. The group currently using the equipment should relinquish it as soon as possible.

E) Hookup and test your program. If there is a problem and someone else wants to use the equipment, let them use it while you debug your program.

NOTES:

HEWLETT hp PACKARD

SOURCES OF REFERENCE

HEWLETT PACKARD HP-85 OWNER'S MANUAL AND PROGRAMMING GUIDE

Part number 00085-90002

HEWLETT PACKARD HP-85 I/O PROGRAMMING GUIDE

Part number 00085-90142

TUTORIAL DESCRIPTION of the HEWLETT-PACKARD INTERFACE BUS (PN 5952-0156)

NOTES:

Often the material presented here will refer to these manuals.

PREREQUISITES

Source: HP-85 Owner's Manual & Programming Guide

A) Section 1 (pgs 17 - 31)

B) Section 2 (pgs 33 - 40)

C) Appendix B (pgs 269 - 289)

NOTES:

This information is assumed to be known!

PREREQUISITE NOTES

TOPICS INTRODUCED:

* Power on, loading paper, loading ROMS
* Manual calculator mode
* Keyboard overview
* Display Editing
* Simple Variables

* Entering a program
* Running a program (RUN)
* Halting a program (PAUSE)
* Erasing a program from memory (SCRATCH)

* Data cartridge care and information
* Loading a program from the data cartridge (LOAD "name")
* Recording a program on data cartridge (STORE "name")
* Erasing a data cartridge (ERASETAPE)
* Erasing a program or data file (PURGE "name")

NOTES:

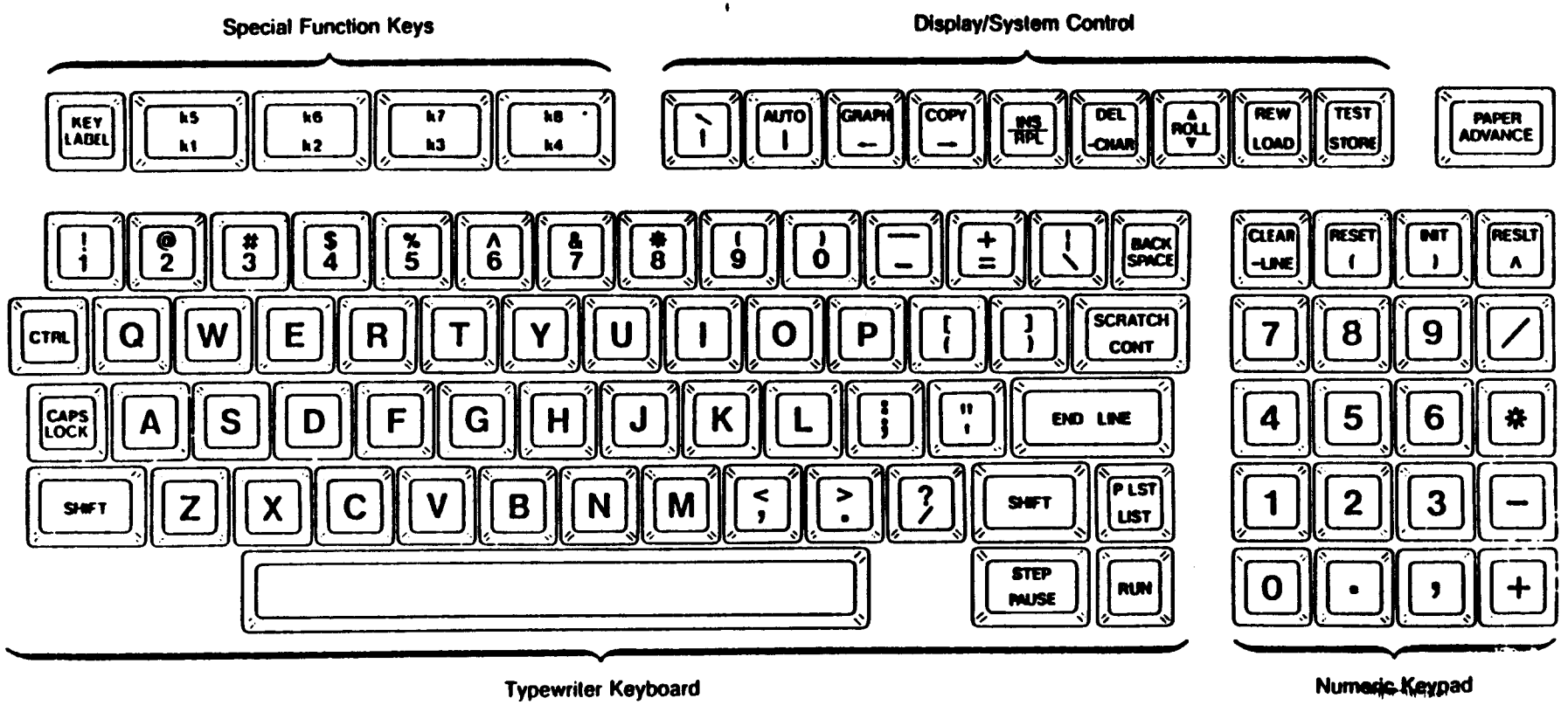
KEYS USED:

END-LINE
BACK SPACE
CURSOR KEYS
CLEAR DISPLAY
CLEAR TO END OF LINE
DELETE CHARACTERS/LINE
INSERT CHARACTERS/LINE
KEY LABEL
ROLL
COPY
LOAD/STORE
RESET

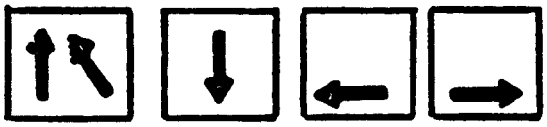
COMMANDS USED:

CLEAR
PRINT ALL
NORMAL
DISP"--------"
PRINT"-------"
COPY
ERASETAPE
AUTO
CAT
LOAD"--", STORE"---"
PURGE"--"
RESET

HP-85 Keyboard



DISPLAY CONTROL KEYS (pgs 12,19,38,39)



Move cursor to desired location without erasing characters.

clear line	* UNSHIFT = Clears line from cursor to end * SHIFT = Clears the display
back space	* UNSHIFT = Backup cursor while erasing * SHIFT = Backspaces rapidly
ins rps	TOGGLES BETWEEN INSERT AND REPLACE MODES. * Insert mode inserts characters pushing right. * Replace mode replaces character at cursor position.
del -char	* UNSHIFT = Delete character above the cursor. * SHIFT = "Delete" command typing aid.
roll	* Roll display up or down with respect to cursor. * UNSHIFT = Roll down * SHIFT = Roll up
graph	* Put into graphic mode, showing any current graphic display. Push any alphanumeric key to exit mode.

NOTES:

EDITING KEYS

LIST

* Enters program development mode and lists 1 page of current program to specified CRT.

Possible listing devices

CRT IS	1	END-OF-LINE	-	HP-85 CRT
CRT IS	2	END-OF-LINE	-	HP-85 Printer
CRT IS	701	END-OF-LINE	-	External Printer

PLIST

* Same as above but lists entire program in memory to specified printer

PRINTER IS 701 END-OF-LINE

AUTO

* Enter program development mode and automatically number lines.

NOTES:

?? WHAT DOES "LIST 120" END-OF-LINE DO??

?? WHAT DOES "LIST 9999" END-OF-LINE DO??

?? WHAT DOES "AUTO 100,20" END-OF-LINE DO??

?? WHAT DOES "PLIST 20,200" END-OF-LINE DO??

HEWLETT hp PACKARD

EXECUTION KEYS

END-OF-LINE

* Executes pending instruction line
* Enter information into running program
* Store program line

RUN

* Runs program currently in memory after initializing it.

STEP PAUSE

SHIFT = Step thru program executing one line at a time (INIT or RUN must have been previously executed)

UNSHIFT = Pauses program currently executing

SCRATCH CONT

SHIFT = "SCRATCH" command typing aid.

UNSHIFT = Restarts program at pause point

RESET

* Resets hung program
* Resets I/O buses

NOTES:

?? WHAT DOES INITIALIZING A PROGRAM MEAN??

?? CAN YOU INITIALIZE A PROGRAM WITHOUT RUNNING IT??

ESSENTIAL KNOWLEDGE - QUIZ

Given this program;

```
10 REM PROGRAM COUNTS
12 C=1
30 BEEP 100, 300
40 DISP C
42 PAUSE
50 WAIT 1000
60 C=C+1
70 GOTO 30
80 END
```

DO YOU KNOW....

??What all these statements mean??

??How would you delete lines 42 and 50??

??How would you get this program renumbered??

??How would you store this program on data cartridge??

??How would you erase this program from memory??

??How to find out how much memory this program takes??

NOTES:

With the exception of the asterisked (*) commands you should know how these commands and statements work.

Ask questions NOW or refer to indicated pages of HP-85 Onwer's Manual at 1st chance. This information must be thoroughly understood.

NON PROGRAMMABLE COMMANDS

AUTO [beginning line number [,increment value]] - Autonumber program lines	Page 80
CONT [statement number] - Continue running program from here	Page 98
DELETE first statement [,last statement number] - delete program lines	Page 95
INIT - Allocates memory to variables , initialize them	Page 99
LOAD program name - Load program from data cartridge	Page 179
REN [first statement number [,increment value]] - Renumber program lines	Page 96
RUN [statement number] - Run the program from specified line number	Page 99
SCRATCH - Deletes current program & all variables	Page 78
STORE program name - Store program to data cartridge	Page 176

PROGRAMMABLE COMMANDS

*CAT - Lists the program & data files on cartridge	Page 175
*COPY - Copy contents of CRT to HP-85 printer.	Page 35
*CTAPE - Conditions the data cartridge for optimim life.	Page 282
*ERASETAPE - Erases old data cartridge directory.	Page 175
FLIP - Flip between normal & inverted typewriter mode.	Page 34
LIST [beginning statement number [,ending statement number]] - to CRT	Page 97
PLIST [beginning statement number [,ending statement number]] - to printer	Page 97
PRINT ALL - print all messages and entries	Page 35
*REWIND - the data cartridge	Page 280
NORMAL - cancels PRINT ALL	

STATEMENTS

BEEP [tone,duration] - Make an audible beep	Page 89
CRT IS output code number - specify CRT select code	Page 169
END - Indicates end of program	Page 77
GOTO statement number - Branch to specified line number	Page 91
PAUSE - Pause a running program	Page 99
REM [any combination of characters] - Documentation Remarks	Page 83
WAIT number of milliseconds - Pause for specified time	Page 100

HP-85 TIDBITS

* INTERACTIVE BASIC OPERATING SYSTEM AND BUFFERED CRT:
  + Orientated for easy program development & modification
  - Not as fast as a "compiled" system

* 8 BIT MICROPROCESSOR BASE = Dictates computational speed

* I/O PROCESSOR ON INTERFACE CARDS:
  + Relieves HP-85 of some I/O management and allows for overlapped I/O and computation

* MULTIPLE STATEMENTS PER LINE:
  + Efficient use of program memory
  + Slightly faster execution speed
  - Affects response time to interrupts
  - Can contribute to "unreadable" programs

* TIME CAPABILITY
  + Internal clock to keep time of day or pace program or data acquisition

* "DEFAULT ON" + Math errors will not halt program execution
  + Unitialized variables initialized to zero
  - Warning message displayed on CRT

NOTES:

HEWLETT hp PACKARD

HP-85 TIDBITS

* INTERRUPT DRIVEN KEYBOARD
  - ? Allows certain keys to interrupt and HALT execution of a running program
  - - CONT key must be used to restart program
  - + Examine and change variables
  - + Valuable debugging tool
  - + Selective keyboard disabling

* SUBROUTINE CAPABILITY
  - = Referenced by line number

* VARIABLES APPLY THROUGHOUT PROGRAM

* WAKES UP IN RADIAN MODE

NOTES:

HP-85 MODES OF OPERATION

1. Halted or Idle
   * Performs commands
   * Available for program modification / entry
   * In th is mode upon power - on, or after PAUSE, END, etc.

2. Program Execution
   * RUN causes program execution

   Note: Keyboard being touched will halt execution.

3. Keyboard Input
   * Input data expected.

NOTES:

PROGRAM TRANSPORTABILITY

IN GENERAL A PROGRAM TO BE RUN ON AN HP-85 MUST HAVE BEEN WRITTEN ON ANOTHER HP-85 WITH THE SAME ROM CONFIGURATION.

PROGRAMS WRITTEN USING PARTICULAR ROM STATEMENTS CANNOT BE LOADED INTO ANOTHER HP-85 WITHOUT THE ROM's USED BY THE PROGRAM BEING PRESENT. THE LOADING PROCESS IS ABORTED AND ONE CANNOT EVEN VIEW THE STATEMENT IN QUESTION.

A SOLUTION MAY BE TO USE A BINARY PROGRAM WHICH ALLOWS THE PROGRAM TO BE "SAVED" (AS A DATA FILE) RATHER THAN STORED. THEN A CORRESPONDING "GET" COMMAND WILL RELOAD THE PROGRAM AND ANY STATEMENTS NOT UNDERSTOOD BECAUSE OF A MISSING ROM WILL BE COMMENTED OUT (!). THIS WILL ALLOW THE OPERATOR TO BE AWARE OF THE PROBLEM SO THAT THE MISSING ROM's CAN BE IDENTIFIED AND OBTAINED, OR THE PROGRAM MODIFIED TO RUN WITHOUT THE ROM.

NOTES:

COMMANDS COMMON TO BOTH A ROM AND THE MAINFRAME (eg PRINTER IS) MUST HAVE THE ROM PRESENT FOR RELOADING IF THE COMMAND WAS STORED WITH THE ROM PRESENT (eg I/O ROM).

SPECIAL CHARACTERS

@ Enables multiple statements per line;

```
550  TRIGGER 722 @ ENTER 722;v(I)  @ PRINT "READING NUMBER";A
```

! Remarks follow;

```
120  DIM A$ [100]   ! STRING VARIABLE HOLDING INSTRUCTIONS
```

? INPUT prompt;

Input items are expected

NOTES:

LAB 1

ENTER THIS PROGRAM

```
10 ! EXAMPLE - NAME - DATE
20 ! HP-85 PROGRAM
30 !
40 PRINTER IS 2 ! PAPER
50 CRT IS 1 @ CLEAR ! CRT
60 RANDOMIZE ! RESET RANDOM
70 ! NUMBER GENERATOR
80 N1=RND @ DISP "N1"
90 N2=RND @ DISP N2
100 N3=RND @ DISP N3
110 N4=RND @ DISP N4
120 A=(N1+N2+N3+N4)/N5 ! COMPUTE
130 ! AVERAGE
140 PRINT "AVERAGE IS ";A
150 DISP "''CONT'' TO DO AGAIN"
160 PAUSE
170 GOTO 60
180 END
```

See B-12B

NOTES:

Practice line entering and line editing by correcting errors.

How much memory did your program take?

Modify this program to have it BEEP before it pauses.

Modify program name, programmer's name and date to suit you.

STORE program for later use.

```
 10 ! EXAMPLE 1 - BYRNE 1/81
 20 ! HP-85 PROGRAM
 30 !
 40 PRINTER IS 2 ! PAPER
 50 CRT IS 1 @ CLEAR ! CRT
 60 RANDOMIZE ! RESET RND # GEN
 70 !
 80 N1=RND @ DISP N1
 90 N2=RND @ DISP N2
100 N3=RND @ DISP N3
110 N4=RND @ DISP N4
120 !
130 ! COMPUTE AVERAGE
140 A=(N1+N2+N3+N4)/4
150 PRINT "AVERAGE IS ";A
160 DISP "''CONT'' TO DO AGAIN"
170 BEEP @ PAUSE
180 GOTO 60
190 END
```

NOTES:

```
 .191399968067
 .838977509773
 .875440380566
 .880046839384
AVERAGE IS  .696466174448
''CONT'' TO DO AGAIN
```

ABOUT 400 BYTES MEMORY USED TO HOLD PROGRAM

PRINT & DISP STATEMENTS

Purpose: Output to printer or display

Examples:

```
PRINT "VOLTAGE = ",V(I)

DISP "SUM IS ";S,"AVERAGE IS - ";S/N

PRINT A,B,C;D
```

Destination: Information is output to the specified printer or CRT.

```
11Ø PRINTER IS 2                  ! print or PLIST on HP-85 printer
... PRINTER IS 1                  ! print or PLIST on HP-85 CRT
... PRINTER IS 701,80**           ! print on HP-IB line printer
                                  ! set line width to 80 columns
... CRT IS 1                      ! display or LIST on CRT
... CRT IS 2                      ! display or LIST on HP-85 printer
```

NOTES:

Two delimiters - a , or a ; delimit items in the data list.

Unless otherwise specified standard number format is used.

PRINTER IS 701 - ALLOWED WITH I/O ROM

PRINTER IS 701,72 - ALLOWED WITH PRINTER/PLOTTER ROM

Freefield Format with PRINT OR DISP

Purpose: It is a flexible default format designed to output numbers and/or text to the printer or CRT in readable form.

When used: It is used whenever no IMAGE statement is specified.

Characteristics:

* ASCII characters are transferred.
* End-of-line sequence (Default is CR/LF) sent after data list is complete.
* Standard number format is used with numbers. Digits of the number (with leading space or minus sign) are output left justified in a field of 11,21, or 32 characters. Trailing spaces are output as necessary to fill the unused portion of the field.
  Varying field widths are used to avoid having a number broken up because it is at the end of a line.
  Characters of a string are output with no leading spaces and no more than 20 trailing spaces, and are left justified.

NOTES:

Number	Standard Format
15.000	15
00.23500	.235
-.0547^9	-4.38415537301E-12
000987.5	987.5
10000^6	1.E24
.01E4	100
120E-4	.012

PRINT AND DISP DELIMITERS

There are two:

, which specifies free-field format

; which specifies compact free field format

Definition:

	Numeric Data	String Data
Compact Field ; ,	Digits of the number are output, preceded by a space (if plus) or a minus sign (if minus), and followed by one space.	Characters of the string are output with no leading or trailing spaces.
Free Field ,	Digits of the number (with leading space or minus sign) are output left-justified in a field of 11,21, or 32 characters. Trailing spaces are output as necessary to fill the unused portion of the field.	Characters of the string are output with no leading spaces and no more than 20 trailing spaces.

NOTES:

Free-field Format Practice

GIVEN A$= "NUMBER IS"

A= 9999999

B= -9999999

C= -.123456789

```
PRINT A$,A$
PRINT A$;A$
PRINT A$,A
PRINT A$;A
PRINT A$,A,B,C
PRINT A,B,C
PRINT A;B;C
```

NOTES:

These exercises can be done without writing a program.

GIVEN:

```
A$="NUMBER IS"
A=9999999
B=-9999999
C=-.123456789
```

```
                        123456789Ø123456789Ø123456789012
PRINT A$,A$             NUMBER IS              NUMBER IS

PRINT A$;A$             NUMBER ISNUMBER IS

PRINT A$,A              NUMBER IS               9999999

PRINT A$;A              NUMBER IS 9999999

PRINT A$,A,B,C          NUMBER IS               9999999
                        -9999999
                        -.123456789

PRINT A$;A,A$;B,A$,C    NUMBER IS 9999999      NUMBER IS
                        -9999999               NUMBER IS
                        -.123456789

PRINT A$;A;A$;B;A$;C    NUMBER IS 9999999  NUMBER IS
                        -9999999 NUMBER IS-.123456789

PRINT A;B;C;A;B;C        9999999 -9999999 -.123456789
                         9999999 -9999999 -.123456789
```

NOTES:

TAB FUNCTION

SYNTAX:

```
TAB (CHARACTER POSITION)
```

EXAMPLE:

```
30 PRINT "1"; TAB(5); "5"; TAB(10); "10"
40 DISP "1"; TAB(5); "5"; TAB (10); "10"
```

RESULTS
IN : 1 5 10

NOTES:

RULES:

* NEXT ITEM IN PRINT/DISP LIST IS OUTPUT BEGINNING IN SPECIFIED CHARACTER POSITION.
* SPECIFIED CHARACTER POSITION ALREADY FULL: CR/LF OUTPUT AND TABBING PERFORMED.
* CHARACTER POSITION: A NUMERIC EXPRESSION ROUNDED TO THE NEAREST POSITIVE INTEGER.
* USE SEMICOLONS BETWEEN PRINT/DISP LINES AND TABS TO SUPPRESS JUMPING TO THE NEXT CHARACTER FIELD.

HEWLETT PACKARD

MODIFY YOUR STORED PROGRAM TO PRESENT ITS DATA AS FOLLOWS:

```
CRT   OR  PRINTER   COLUMNS

12345678901234567890123456789012


NUMBER 1 IS .191399968067
NUMBER 2 IS .838977509773
NUMBER 3 IS .875440380566
NUMBER 4 IS .880046839384
           --------------

SUM IS     2.785864697790

       AVERAGE = .696466174448
```

NOTES:

HEWLETT hp PACKARD

BITS, BYTES, WORDS - DATA PRECISION

1 BIT	ON-OFF	1 or Ø PRIMARY UNIT
1 BYTE	8 BITS	BASIC UNIT OF MEMORY
1 WORD	16 BITS	2 BYTES

NUMBER		TYPE	DATA PRECISION	BYTES OF MEMORY
FULL PRECISION	-	REAL	±9.99999999999E±499	10*
SHORT PRECISION	-	SHORT	±9.9999E±99	6*
INTEGER PRECISION	-	INTEGER	-99999 thru +99999	5*

*SIMPLE VARIABLES

NOTES:

FULL PRECISION (REAL) IS THE DEFAULT PRECISION, AND THE ONE NORMALLY USED UNLESS MEMORY NEEDS TO BE CONSERVED.

HEWLETT hp PACKARD

STANDARD NUMBER FORMATS:

NUMBER	STANDARD FORMAT
9876543210.1234	9876543210.12
15.000	15
00.23500	.235
-.0547^9	-4.38415537301E-12
000987.5	987.5
10000^6	1.E24
.01E4	100
120E-4	.012

*MAXIMUM OF 12 DIGITS ARE PRINTED OR DISPLAYED.

*UNNECESSARY TRAILING ZEROS TO THE RIGHT OF THE DECIMAL POINT ARE SUPPRESSED.

*LEADING ZEROS ARE SUPPRESSED.

*NUMBERS WHOSE ABSOLUTE VALUE ≥ 1 AND $< 10^{12}$ ARE OUTPUT WITHOUT EXPONENT.

*NUMBERS BETWEEN -1 AND +1 ARE SHOWN WITHOUT EXPONENT IF POSSIBLE.

*ALL OTHERS ARE EXPRESSED IN SCIENTIFIC NOTATION.

NOTES:

RANGE OF NUMBERS WHICH CAN BE ENTERED AND STORED IS

$-9.99999999999 \times 10^{-499}$ THRU $9.99999999999 \times 10^{-499}$

HEWLETT hp PACKARD

SIMPLE NUMERIC VARIABLES (PGS. 49 - 51)

NAMES = A, AØ, A1, , , A9
B, BØ, , , B9
.
.
.
Z, ZØ, , , , Z9
} 286 total

PURPOSE = USED TO STORE SINGLE NUMBERS IN HP-85's MEMORY

ASSIGNMENT = A= -13.1234

Z9= 9.87 E + 56

B= -M8

```
11Ø DISP "ENTER # OF SCANS"
12Ø INPUT  S
```

NOTES:

REQUIRES 10 BYTES OF MEMORY (FULL PRECISION)

a=A AS FAR AS VARIABLES ARE CONCERNED

HEWLETT PACKARD

ARITHMETIC OPERATORS (pg. 295)

ORDER OF EXECUTION:

( )	Parentheses	Performed First
^	Exponentiation	
* /	Multiplication, Division	
MOD	Modulo: A MOD B=A-B*INT(A/B)	
\ or DIV	Integer divide: A DIV B=IP(A/B)	
+ -	Addition, Subtraction	Performed Last

NOTES:

*Operations within ( ) are first.

*Nested (( )), innermost ( ) are first.

Examples:

```
3 + 2 - 4 = 1

3 * 4 ^ 2 = 48

(3 * 4) ^ 2 = 144

-4 + 72/6 ^ 2 + 1 = -1

(-4 + (72/6))  2 + 1 = 65

3 * 8/4 * 3 = 18

3 * 8/(4 * 3) = 2
```

HEWLETT PACKARD

ARRAY VARIABLES (PG. 119-124)

PURPOSE: USED TO STORE GROUPS OF NUMBERS.

NAMES: A0,A1,,,,,,,,,,,,,,,,Z9

TYPES: *ONE DIMENSIONAL (SCALARS)

EG. A9

1	2	3	4		99	100
1.52	-2.34	18.23	99.45		20.12	-23.5	-777

```
10 OPTION BASE 1 ! 1st array element is 1
20 DIM A9 (100)  ! room for 100 readings
30 A9(3) =18.23  ! assign an array element
```

*TWO DIMENSIONAL

```
EG   10 OPTION BASE 1
     20 DIM Z0 (4,5)
     30 Z0(1,4)=12.5
```

	COL 1	2	3	4	5
ROW 1				12.5	
2					
3					
4					

NOTES:

Simple and array variables can share the same name.

*THREE DIMENSIONAL (NOT ALLOWED)

*STRING ARRAYS (NOT ALLOWED)

HEWLETT hp PACKARD

MORE ON ARRAY VARIABLES

RESERVING MEMORY — BEFORE AN ARRAY VARIABLE CAN BE USED, DATA PRECISION MUST BE SPECIFIED, AND MEMORY MUST BE SET ASIDE.

STORAGE REQUIREMENTS -

REAL	(12 digits)	8 bytes per element + 8 bytes overhead
SHORT	(5 digits)	4 bytes per element + 8 bytes overhead
INTEGER	(5 digits)	3 bytes per element + 8 bytes overhead

?? How many full precision numbers could you take in a standard 16K HP-85 assuming a 2K program ??

?? How many short precision ??

?? How could you represent a voltage reading as an integer number eg - 1.2345 ??

NOTES:

85A HAS 14160 BYTES MEMORY AVAILABLE WHEN ONLY I/O ROM INSTALLED.

85F (EXTRA MEMORY CARD INSTALLED) HAS 30288 BYTES MEMORY AVAILABLE WHEN ONLY I/O ROM INSTALLED.

MATRIX ROM TAKES 70 BYTES.

PRINTER/PLOTTER ROM TAKES 250 BYTES.

HEWLETT PACKARD

ARRAY VARIABLE MEMORY ALLOCATION

```
XXX  OPTION BASE 1              !  SPECIFIES WHETHER LOWEST
                                !  ARRAY ELEMENT IS Ø OR 1
                                !  MUST COME FIRST
                                !  DEFAULT IS Ø
                                !
XXX  DIM V(3,1ØØ)               !  ALLOCATES 2408 BYTES MEMORY
                                !  FOR THE 300 ELEMENTS OF ARRAY
                                !
XXX  REAL T(100)                !  808  BYTES FOR ARRAY T
                                !
XXX  SHORT X(100)               !  408  BYTES FOR ARRAY X

XXX  INTEGER A(3,4)             !   44  BYTES FOR ARRAY A

                                !  PUT LENGTH OF PROGRAM IN
                                !  BYTES HERE FOR YOUR
                                !  REFERENCE
```

DIM is the equivalent of REAL

NOTES:

OPTION BASE statement must precede the DIM, REAL, SHORT, or INTEGER statement.

?? IF OPTION BASE Ø WERE SPECIFIED, HOW MANY ELEMENTS WOULD ARRAY V CONTAIN ??

PROVE THE STORAGE REQUIREMENTS GIVEN EARLIER FOR ARRAY VARIABLES.

HEWLETT hp PACKARD

VARIABLE ASSIGNMENT

```
 70   I=1  ! INITIALIZE COUNT OF INPUT DATA
 80   OPTION BASE 1  ! 1ST ARRAY ELEMENT IS 1
 90   DIM V (1ØØØ)  !  ROOM FOR 1000 READINGS
100   N=100
110   DATA 100,56,78,34.56,DISTORTION TEST
120   READ A,B,C,D,F$
```

ASSIGNMENT FROM PROGRAM

```
200   DISP  "ENTER NUMBER OF SCANS"
210   INPUT C1   ! WITH PROMPT
```

ASSIGNMENT FROM KEYBOARD

```
500   FOR S=1 TO C1      !MEASUREMENT LOOP
510   TRIGGER 722        !TAKE READING
520   ENTER 722; V(S)    !ENTER DATA FROM INSTRUMENT
                         ! HP-IB DEVICE #22 ON BUS 7
```

ASSIGNMENT FROM EXTERNAL DEVICE

NOTES:

VARIABLE ASSIGNMENT - READ/DATA STATEMENTS

PURPOSE: ENABLES DATA ENTRY FROM WITHIN A PROGRAM

SYNTAX: READ variable 1, variable 2, ----,

DATA value 1, value 2, , , ,

EXAMPLE:

```
1Ø DATA 45, 55, 65, 72, 66
2Ø DATA 88, 92, 95, 105, 11Ø
3Ø FOR C = 1 to 1Ø
4Ø READ V @ DISP V
5Ø NEXT C
6Ø END
```

NOTES:

RULES:

* CAN HAVE MULTIPLE READ, DATA STATEMENTS.
* LOCATION OF DATA STATEMENTS WITHIN PROGRAM CAN BE ARBITRARY: DATA POINTER LOCATES DATA.
* DATA STATEMENTS PROVIDE VALUES TO READS, STARTING AT THE DATA STATEMENT WITH THE LOWEST LINE NUMBER.
* DATA POINTER KEEPS TRACK OF LOCATION OF NEXT CONSTANT TO BE READ.

?? What happens when the loop is increased to 20 values ?? Try it!

HEWLETT hp PACKARD

REPOSITIONING DATA POINTER

PURPOSE:

* REPOSITIONS POINTER AT BEGINNING OF DATA STATEMENT SELECTED. LETS YOU RE-USE PORTIONS OF DATA.

SYNTAX:

* RESTORE LINE NUMBER eg RESTORE 1Ø

RULES:

* NO LINE NUMBER ; POINTER REPOSITIONED AT BEGINNING OF LOWEST-NUMBERED DATA STATEMENT.
* YOU CANNOT ONLY JUMP BACK TO A SELECTED DATA STATEMENT; YOU CAN ALSO JUMP AHEAD.

NOTES:

Modify earlier example so that only the first 5 data values are used.

HEWLETT hp PACKARD

DATA ENTRY - REVIEW

USE READ, DATA:

WHEN YOU HAVE KNOWN, NON-VARYING DATA, REMEMBER:
YOU CANNOT CHANGE THE DATA IN YOUR PROGRAM WITHOUT CHANGING THE DATA STATEMENTS THEMSELVES.

USE INPUT:

WHEN YOU NEED INTERACTIVE, VARYING DATA, REMEMBER:
YOU MUST HAVE AN OPERATOR PRESENT TO TYPE IN THE RESPONSES.

NOTES:

HEWLETT hp PACKARD

RELATIONAL OPERATORS (pg. 53)

```
=  EQUAL TO

>  GREATER THAN

<  LESS THAN

>= GREATER THAN OR EQUAL TO

<= LESS THAN OR EQUAL TO

<>, # NOT EQUAL TO
```

Logical expressions return the value one for true, zero for false. Non-zero values are considered true. Zero values are false.

```
110 DISP"ENTER VOLTAGE RANGE (1,10,100)"
120 INPUT R
130 V=1*(R=1) + 2*(R=10) + 3* (R=100) !convert 1,10,100 to
                                      !1,2,3 respectively.

... IF A$#"YES" THEN .....
...
... IF A$<B$ THEN GOTO 1000
...
... L=L + (LEN (A$)>9)  ! add 1 to L when A$ contains more than
                        ! nine characters.
```

NOTES:

You must use reverse order or enclose relational operators in parentheses to distinguish from variable assignment.

Non-numeric values can also be compared with relational operators. Strings are compared, character by character from left to right until a difference is found. If one string is shorter than another, it is considered the lesser.

HEWLETT hp PACKARD

LOGICAL OPERATORS (pg. 54)

AND.....if all are true

OR.....if any is true

EXOR.....if one or more but not all are true

NOT

NOTES:

```
IF A > B AND C=D THEN.....
IF A$ [1,1] = "Y" OR A$ [1,1] = "y" THEN.....
IF S > 100 OR S<Ø THEN GOTO 9000
```

HEWLETT hp PACKARD

BRANCHING INTRODUCTION (pg 103-108)

TYPES:

GOTO STATEMENT

IF numeric expression is true THEN execute this portion of statement

EXAMPLES:

```
880 GOTO 100 ! BRANCH DIRECTLY TO LINE 100
...
...
...
980 IF B$="YES" THEN 920 ! If true goto 920
...
...
...
... IF N=100 THEN 2040 ELSE 1000 ! If true goto 2040
                                 ! Else goto 1000
```

NOTES:

Computed GOTO not discussed.

Branch to labels NOT allowed. eg ~~GOTO "fix"~~

IF....THEN....!WITH MULTIPLE STATEMENTS PER LINE

```
130    !
140    !
150    !
160    DISP  "ENTER VOLTAGE TO BE"
170    DISP  "OUTPUT (-10 TO +10V)"
180    INPUT V
190    IF V>10 OR V<-10 THEN BEEP @
     DISP  "VOLTAGE OUT OF LIMIT" @ W
AIT 3000 @ CLEAR @ GOTO 160
200    !
210    !



990    IF V(I,S)>=L2 THEN OUTPUT 709;
       "DO1,Ø" ELSE OUTPUT 709; "DC1,Ø"
```

NOTES:

Several RELATED commands can be lumped together, allowing PROGRAM logic to be easier to implement.

LAB 2

Modify Program written for Lab 1 as follows;

A) Dimension two numeric arrays to hold up to 100 numbers

   One array will store the random numbers generated as full precision numbers. The second array will store them as short precision numbers.

B) Have the program have the operator input the number of random numbers to be generated and averaged.

C) Use the logical "IF THEN" statement to generate the appropriate number of random numbers.

D) Compute the average as before.

E) Display or Print out the full precision and short precision numbers generated so you can compare them.

EXTRA CREDIT:

A) Check for proper numeric entry by operator.

B) Label the full and short precision numbers being displayed or printed.

NOTES:

```
10 ! LAB2 - BYRNE 2/81
20 ! HP-85 PROGRAM
30 !
40 PRINTER IS 2 ! PAPER
50 CRT IS 1 @ CLEAR ! CRT
60 !
70 RANDOMIZE
80 OPTION BASE 1 ! 1ST ARRAY ELEMENT IS 1
90 DIM R(100) ! 100 FULL PRECISION  READINGS
100 SHORT S(100) ! 100 SHORT PRECISION READINGS
110 !
120 C=1 ! INITIALIZE COUNT OF NUMBERS GENERATED
130 S=0 ! INITAILIZE SUM OF NUMBERS GENERATED
140 !
150 DISP "ENTER # OF READINGS "
160 DISP "TO BE AVERAGED (<101)"
170 INPUT N
180 IF N<0 OR N>100 THEN BEEP @ GOTO 150
190 !
200 DISP "FULL PRECISION -  SHORT PREC." @ DISP @ DISP
210 !
220 ! BEGINNING OF LOOP
230 R(C)=RND
240 S(C)=R(C)
250 DISP R(C);"    ";S(C)
260 S=S+R(C) ! COMPUTE TOTAL SUM
270 !
280 IF C<N THEN C=C+1 @ GOTO 230
290 !
300 ! COMPUTE AVERAGE
310 PRINT "AVERAGE IS ;";S/N
320 BEEP @ DISP "DONE"
330 END
```

NOTES: Output shown in "PRINT ALL" mode.

```
ENTER # OF READINGS
TO BE AVERAGED (<101)
?
105
ENTER # OF READINGS
TO BE AVERAGED (<101)
?
10
FULL PRECISION -  SHORT PREC.

 .208244947351      .20824
 .747536109474      .74754
 .751334566801      .75133
 .719220376063       71922
 .654540216962      .65454
 .161648891563      .16165
 .872216682948      .87222
 .857017795574      .85702
 .469902305241      .4699
 .113434638381      .11343
AVERAGE IS ;  .555509653035
DONE
```

FOR / NEXT LOOPS

```
....
....
1480  DISP "ENTER NUMBER OF SCANS"
1490  INPUT N
1500  DISP "ENTER NUMBER OF CHANNELS"
1510  INPUT N2
1520  !
1530  !MEASUREMENT LOOP
1540  FOR S=1 to N ! FOR N SCANS
1550  FOR C=1 to N2 ! FOR N2 CHANNELS
1560  TRIGGER 722 ! TRIGGER VOLTMETER
1570  ENTER 722; V(S,C) ! ENTER READING
1580  PRINT V(S,C); "VOLTS"
1590  WAIT 3000 ! WAIT 3 SEC
1600  NEXT C ! NEXT CHANNEL
1610  NEXT S ! NEXT SCAN
....
....
....
```

NOTES:

?? What would be the value of S after the FOR/NEXT loop shown above is complete ??

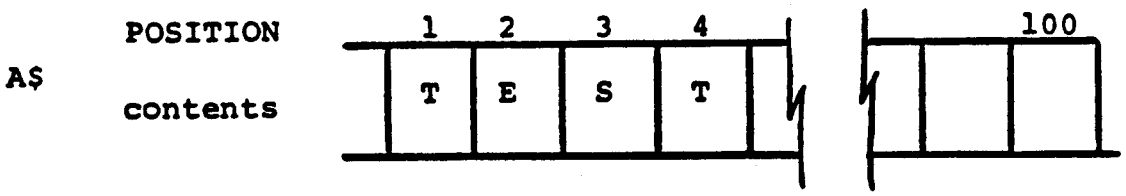
?? What would happen if the NEXT S and NEXT C statements were reversed ??

HEWLETT PACKARD

STRING VARIABLES (pgs 51,52,124-133)

Variables that can contain nothing or sequences of characters, and allow the HP-85 to modify, analyze, and combine them.

```
130  DIM A$ [100]  : Reserve memory for up to
                     100 characters
 .
 .
 .
 .
190  A$="TESTING 1 2 3"
```



NOTES:

RULES:

* Always end in a "↕"
* Implicitly dimensioned to 18 characters long.
* Must be dimensioned if more than 18 characters long.
* Dim statement specifies maximum length of string. Limited only by available memory.
* Strings and numerics cannot operate on each other:

  C = A$/B -- ERROR

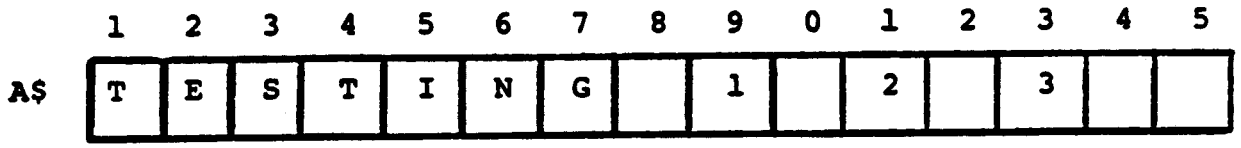
* String arrays are not allowed.
* The null string contains no characters or blanks. It provides a method of blanking out an entire string without knowing its length.

A$=""

HEWLETT hp PACKARD

SUBSTRINGS

A$ = "TESTING 1 2 3"



A$ [1,4] =

A$ [6,6] =

A$ [10,14] =

NOTES:

Purpose:

* To extract portions of long strings.
* To insert, change, or add to strings or parts of strings.

?? How would you change the string to "TESTING A B C" ??

PRINTING & DISPLAYING STRINGS

```
190 PRINTER IS 2          ! PRINTER
200 PRINT A$              ! ISSUES A CR/LF AFTER EACH
 :                        ! 32 CHARACTERS
 :
 :
 :

440 CRT IS 1              ! CRT
450 DISP A$               ! CR/LF EVERY 32 CHARACTERS
 :
 :

800 PRINTERS IS 701       ! HP-IB PRINTER (80 COLUMN)
810 PRINT A$              ! CR/LF EVERY 32 CHARACTERS
```

NOTES:

CHARACTERS ARE LEFT JUSTIFIED

WITH PRINTER/PLOTTER ROM;

```
PRINTER IS 703,78         ! ISSUE A CR/LF AFTER EVERY 78 CHARACTERS
PRINT A$
```

HEWLETT PACKARD

FUNCTIONS

They are used to perform often needed operations on numbers or strings. Most of you are probably familiar with the geometric functions SINE, COSINE, TANGENT, etc. These and many other functions are built into the BASIC operating system of the HP-85.

The programmer has the ability to define and use functions of his own design. This will be covered in the structured programming section.

NOTES:

Appendix D and reference section have a complete list of HP-85 mainframe functions.

HEWLETT hp PACKARD

TIME FUNCTIONS pg 56-57

PURPOSE: The HP-85 has an internal clock that provides time of day upon command.

On power on the HP-85 starts counting time in milliseconds. After 24 hours (86,400 sec) the date is incremented by 1, and time counts counts from zero again.

```
SETTIME seconds since midnight, day of year        } SETS
                                  or               } CLOCK
                                date in mdd        }

A = TIME                                           }
B = DATE                                           } READ
DISP "DATE IS ";B;"TIME IS ";A                     } CLOCK
```

NOTES:

Useful in measuring execution time of program segments.

If a SETTIME 36000,20 was executed, what was the time and date the clock was set to??

If you use the mdd format for the date (eg 331 represents March 31 1st) remember that at midnight the date number wil be 332 NOT 401 !

HEWLETT PACKARD

STRING FUNCTIONS

SYNTAX	DESCRIPTION
LEN (S$)	LENGTH OF STRING S$.
POS (S1$, S2$)	POSITION OF S2$ IN S1$.
UPC$ (S$)	CONVERTS LOWERCASE CHARACTERS IN S$ TO UPPERCASE.
VAL (S$)	NUMERIC EQUIVALENT OF STRING S$.
VAL$(X)	STRING EQUIVALENT OF X.
CHR$ (X)	CHARACTER WHOSE DECIMAL CODE IS X: 0<=X<=255.
NUM (S$)	DECIMAL CODE OF FIRST CHARACTER IN S$.

Refer to ASCII chart which follows.

NOTES:

RULES:

* RESULTS OF FUNCTIONS ENDING IN $ ARE STRINGS.
* RESULTS OF FUNCTIONS THAT DO NOT END IN $ ARE NUMBERS

EXAMPLES:

```
DIM A$ [10]
A$ "abc27fgh"

LEN  (A$)=8                    VAL$  (85) = "85"
POS  (A$, "fg") = 6            CHR$  (65) = "A"
UPC$  (A$) = "ABC27FGH"        NUM   (A$) = 97
VAL  ("27") = 27
```

HEWLETT PACKARD

CODES AND NUMBER BASES

DECIMAL (BASE 10)

OCTAL (BASE 8)

BINARY (BASE 16)

ASCII Codes: Assigns a number to represent each of 128 characters. ie:

ASCII CHARACTER	DECIMAL	BINARY	OCTAL	HEXADECIMAL
A	65	01 000 001	101	41
B	66	01 000 010	102	42
C	67	01 000 011	103	43
D	68	01 000 100	104	44
$	36	00 100 100	044	24

NOTES:

There are other widely used codes (EBCIDIC) but the HP-85 thinks in ASCII. Try to get used to the concept that each character has a numeric equivalent.

$16_{10} = 1 \times 10 + 6 \times 1 = 16_{10} = 10000_2$

$16_8 = 1 \times 8 + 6 \times 1 = 14_{10} = 01110_2$ (Note A)

$16_{16} = 1 \times 16 + 6 \times 1 = 22_{10} = 10110_2$ (Note B)

A. NOTE THAT BREAKING THE BINARY REPRESENTATION INTO GROUPS OF THREE GIVES YOU THE OCTAL NUMBER

eg $01\ 110_2 = 16_8$

B. BREAKING INTO GROUPS OF FOUR GIVES THE HEXADECIMAL NUMBER

$1\ 0110_2 = 16_{16}$

ASCII CHART

HP-IB		ASCII	Decimal	Binary	Octal	Hexa-decimal
Addressed Command Group -ACG	GTL	NUL	0	00 000 000	000	00
		SOH	1	00 000 001	001	01
		STX	2	00 000 010	002	02
		ETX	3	00 000 011	003	03
	SDC	EOT	4	00 000 100	004	04
	PPC	ENQ	5	00 000 101	005	05
		ACK	6	00 000 110	006	06
		BEL	7	00 000 111	007	07
	GET	BS	8	00 001 000	010	08
	TCT	HT	9	00 001 001	011	09
		LF	10	00 001 010	012	0A
		VT	11	00 001 011	013	0B
		FF	12	00 001 100	014	0C
		CR	13	00 001 101	015	0D
		SO	14	00 001 110	016	0E
		SI	15	00 001 111	017	0F
Universal Command Group -UCG	LLO	DLE	16	00 010 000	020	10
		DC1	17	00 010 001	021	11
		DC2	18	00 010 010	022	12
		DC3	19	00 010 011	023	13
	DCL	DC4	20	00 010 100	024	14
	PPU	NAK	21	00 010 101	025	15
		SYN	22	00 010 110	026	16
		ETB	23	00 010 111	027	17
	SPE	CAN	24	00 011 000	030	18
	SPD	EM	25	00 011 001	031	19
		SUB	26	00 011 010	032	1A
		ESC	27	00 011 011	033	1B
		FS	28	00 011 100	034	1C
		GS	29	00 011 101	035	1D
		RS	30	00 011 110	036	1E
		US	31	00 011 111	037	1F
Listen Address Group -LAG Note 1	L0	SP	32	00 100 000	040	20
	L1	!	33	00 100 001	041	21
	L2	"	34	00 100 010	042	22
	L3	#	35	00 100 011	043	23
	L4	$	36	00 100 100	044	24
	L5	%	37	00 100 101	045	25
	L6	&	38	00 100 110	046	26
	L7	'	39	00 100 111	047	27
	L8	(	40	00 101 000	050	28
	L9	)	41	00 101 001	051	29
	L10	*	42	00 101 010	052	2A
	L11	+	43	00 101 011	053	2B
	L12	,	44	00 101 100	054	2C
	L13	-	45	00 101 101	055	2D
	L14	.	46	00 101 110	056	2E
	L15	/	47	00 101 111	057	2F
	L16	0	48	00 110 000	060	30
	L17	1	49	00 110 001	061	31
	L18	2	50	00 110 010	062	32
	L19	3	51	00 110 011	063	33
	L20	4	52	00 110 100	064	34
	L21	5	53	00 110 101	065	35
	L22	6	54	00 110 110	066	36
	L23	7	55	00 110 111	067	37
	L24	8	56	00 111 000	070	38
	L25	9	57	00 111 001	071	39
	L26	:	58	00 111 010	072	3A
	L27	;	59	00 111 011	073	3B
	L28	<	60	00 111 100	074	3C
	L29	=	61	00 111 101	075	3D
	L30	>	62	00 111 110	076	3E
	UNL	?	63	00 111 111	077	3F

HP-IB		ASCII	Decimal	Binary	Octal	Hexa-decimal
Talk Address Group -TAG Note 2	T0	@	64	01 000 000	100	40
	T1	A	65	01 000 001	101	41
	T2	B	66	01 000 010	102	42
	T3	C	67	01 000 011	103	43
	T4	D	68	01 000 100	104	44
	T5	E	69	01 000 101	105	45
	T6	F	70	01 000 110	106	46
	T7	G	71	01 000 111	107	47
	T8	H	72	01 001 000	110	48
	T9	I	73	01 001 001	111	49
	T10	J	74	01 001 010	112	4A
	T11	K	75	01 001 011	113	4B
	T12	L	76	01 001 100	114	4C
	T13	M	77	01 001 101	115	4D
	T14	N	78	01 001 110	116	4E
	T15	O	79	01 001 111	117	4F
	T16	P	80	01 010 000	120	50
	T17	Q	81	01 010 001	121	51
	T18	R	82	01 010 010	122	52
	T19	S	83	01 010 011	123	53
	T20	T	84	01 010 100	124	54
	T21	U	85	01 010 101	125	55
	T22	V	86	01 010 110	126	56
	T23	W	87	01 010 111	127	57
	T24	X	88	01 011 000	130	58
	T25	Y	89	01 011 001	131	59
	T26	Z	90	01 011 010	132	5A
	T27	[	91	01 011 011	133	5B
	T28	\	92	01 011 100	134	5C
	T29	]	93	01 011 101	135	5D
	T30	^	94	01 011 110	136	5E
	UNT	_	95	01 011 111	137	5F
Secondary Command Group -SCG Note 3	S0	`	96	01 100 000	140	60
	S1	a	97	01 100 001	141	61
	S2	b	98	01 100 010	142	62
	S3	c	99	01 100 011	143	63
	S4	d	100	01 100 100	144	64
	S5	e	101	01 100 101	145	65
	S6	f	102	01 100 110	146	66
	S7	g	103	01 100 111	147	67
	S8	h	104	01 101 000	150	68
	S9	i	105	01 101 001	151	69
	S10	j	106	01 101 010	152	6A
	S11	k	107	01 101 011	153	6B
	S12	l	108	01 101 100	154	6C
	S13	m	109	01 101 101	155	6D
	S14	n	110	01 101 110	156	6E
	S15	o	111	01 101 111	157	6F
	S16	p	112	01 110 000	160	70
	S17	q	113	01 110 001	161	71
	S18	r	114	01 110 010	162	72
	S19	s	115	01 110 011	163	73
	S20	t	116	01 110 100	164	74
	S21	u	117	01 110 101	165	75
	S22	v	118	01 110 110	166	76
	S23	w	119	01 110 111	167	77
	S24	x	120	01 111 000	170	78
	S25	y	121	01 111 001	171	79
	S26	z	122	01 111 010	172	7A
	S27	{	123	01 111 011	173	7B
	S28	|	124	01 111 100	174	7C
	S29	}	125	01 111 101	175	7D
	S30	~	126	01 111 110	176	7E
	S31	DEL	127	01 111 111	177	7F

MATHEMATICAL FUNCTIONS

SYNTAX	DESCRIPTION
ABS (X)	ABSOLUTE VALUE OF X
SGN (X)	SGN OF X: +, Ø or -
CEIL (X)	SMALLEST INTEGER > = X
FLOOR (X)	LARGEST INTEGER < = X
INT (X)	LARGEST INTEGER < = X
FP (X)	FRACTIONAL PART OF X
IP (X)	INTEGER PART OF X
MAX (X, Y)	RETURNS LARGER VALUE OF X AND Y
MIN (X, Y)	RETURNS SMALLER VALUE OF X AND Y
EXP (X)	e^x
LOG (X)	LOG (BASE e) OF X, X > 0
LGT (X)	LOG (BASE 10) OF X, X > 0
SQR (X)	SQUARE ROOT OF X
INF	LARGEST MACHINE NUMBER:9.99999999999E499
EPS	SMALLEST MACHINE NUMBER: 1E-499
PI	3.14159265359
RND	RETURNS A NUMBER WHICH IS NEXT IN A SEQUENCE OF PSEUDO-RANDOM NUMBERS 0 < = RND < 1

MATH FUNCTIONS (cont)

SYNTAX	DESCRIPTION
SIN (X)	SINE OF X
COS (X)	COSINE OF X
TAN (X)	TANGENT OF X
CSC (X)	COSECANT OF X
SEC (X)	SECANT OF X
COT (X)	COTANGENT OF X
ASN (X)	ARC SINCE OF X - 1ST OR 4TH QUARTER
ACN (X)	ARC COSINE OF X - 1ST OR 2ND QUADRANT
ATN (X)	ARC TANGENT OF X - 1ST OR 4th QUADRANT
ATN2 (Y,X)	ARC TANGENT OF Y/X - PROPER QUADRANT
DTR (X)	DEGREES TO RADIANS CONVERSION
RTD (X)	RADIANS TO DEGREES CONVERSION

NOTE: THE FOLLOWING THREE STATEMENTS ARE ASSOCIATED WITH THE TRIGONOMETRIC FUNCTIONS:

DEG	SETS DEGREES MODE
RAD	SETS RADIANS MODE
GRAD	SETS QUAD MODE

HEWLETT hp PACKARD

MATH HIERARCHY (pgs 43-46, 55, 296)

( )	PERFORMED FIRST
Functions	↓
^	↓
NOT	
*, /, MOD, or DIV	
+, -	
Relational operators (=,>,<,>=,<=,<>, or #)	
AND	↓
OR, EXOR	↓ PERFORMED LAST

Expressions are evaluated from left to right for operators at the same level. Operations within parentheses are performed first. Nested parentheses are evaluated inward out.

NOTES:

* Scan is from left to right. Operation occurs when operator to the right has lower memory.

* Use parentheses when in doubt of priority.

$$2 + \frac{3x6}{(7-4)^2} \Rightarrow 2 + ((3*6) /((7-4) \wedge 2))$$

↑ ↑
Innermost parrantheses evaluated first.

* No implied multiply

LAB 3

MODIFY PROGRAMS WRITTEN FOR LAB 1 OR 2 AS FOLLOWS:

A) REPLACE THE IF...THEN... STATEMENTS USED IN THE LOOP WITH A FOR / NEXT STATEMENT.

B) HAVE THE PROGRAM REQUEST THE OPERATOR TO INPUT A CAPITAL LETTER "Y" IN ORDER TO PROCEED WITH THE PROGRAM. IF ANY OTHER CHARACTER IS INPUT, HAVE THE PROGRAM:

1) BEEP
2) DISPLAY A MESSAGE OF YOUR CHOICE FOR 5 SECONDS
3) CLEAR THE CRT
4) REQUEST ANOTHER INPUT

EXTRA CREDIT:

A) ALLOCATE MEMORY FOR A STRING VARIABLE LARGE ENOUGH TO HOLD 100 SHORT PRECISION NUMBERS. STORE THE SHORT PRECISION NUMBERS IN THIS STRING (AS SUBSTRINGS).

B) LATER RECONVERT THIS "UNUSEABLE" STRING DATA BACK INTO "USEABLE" NUMERIC DATA.

C) USE TIME FUNCTION TO MEASURE ELAPSED TIME OF NUMBER GENERATION.

NOTES:

?? WHAT IS THE DIFFERENCE BETWEEN USEABLE AND NONUSEABLE DATA AS MENTIONED ABOVE ??

?? IS IT PRACTICAL TO STORE THIS TYPE OF NUMERIC DATA IN A STRING ??

```
 10 ! LAB3 - BYRNE 2/81
 20 ! HP-85 PROGRAM
 30 !
 40 ! INITIALIZATION
 50 !
 60 PRINTER IS 2 ! PAPER
 70 CRT IS 1 @ CLEAR ! CRT
 80 RANDOMIZE @ OPTION BASE 1
 90 S=0 @ D=1 ! S=SUM CNT D=STRI
    NG POINTER
100 DIM R(100) ! 100 F.P. NUMBER
    S
110 SHORT S(100) ! 100 S.P. #'s
120 DIM S$[500] ! STRING STORAGE
130 !
140 DISP "ENTER # OF READINGS"
150 DISP "TO BE AVERAGED (<101)"
170 INPUT N@ IF N<0 OR N>100 THE
    N BEEP @ GOTO 140
180 !
190 DISP "READY TO START (Y/N)"
200 INPUT S$
210 IF UPC$(S$[1,1])#"Y" THEN BE
    EP @ DISP "GET READY" @ WAIT
     5000 @ CLEAR @ GOTO 190
220 !
225 !
230 ! BEGINNING OF LOOP
240 !
250 FOR C=1 TO N
260 R(C)=RND @ S(C)=R(C) @ S$[D,
    D+5]=VAL$(S(C))
280 DISP R(C);" ";S(C);"  ";S$[D
    ,D+5]
290 D=D+6 @ S=S+R(C)
300 NEXT C @ DISP
310 !
320 ! RECONVERTING STRING INTO
330 ! USEABLE DATA
340 !
350 D=1
360 FOR C=1 TO N
370 V=VAL(S$[D,D+5]) @ DISP V
380 D=D+6
390 NEXT C
394 !
395 DISP LEN(S$)/6;" READINGS ST
    ORED IN STRING S$"
400 BEEP @ DISP "DONE" @ END
```

LAB 3

MODIFY PROGRAMS WRITTEN FOR LAB 1 OR 2 AS FOLLOWS:

A) REPLACE THE IF...THEN... STATEMENTS USED IN THE LOOP WITH A FOR / NEXT STATEMENT.

B) HAVE THE PROGRAM REQUEST THE OPERATOR TO INPUT A CAPITAL LETTER "Y" IN ORDER TO PROCEED WITH THE PROGRAM. IF ANY OTHER CHARACTER IS INPUT, HAVE THE PROGRAM:

1) BEEP
2) DISPLAY A MESSAGE OF YOUR CHOICE FOR 5 SECONDS
3) CLEAR THE CRT
4) REQUEST ANOTHER INPUT

EXTRA CREDIT:

A) ALLOCATE MEMORY FOR A STRING VARIABLE LARGE ENOUGH TO HOLD 100 SHORT PRECISION NUMBERS. STORE THE SHORT PRECISION NUMBERS IN THIS STRING (AS SUBSTRINGS).

B) LATER RECONVERT THIS "UNUSEABLE" STRING DATA BACK INTO "USEABLE" NUMERIC DATA.

C) USE TIME FUNCTION TO MEASURE ELAPSED TIME OF NUMBER GENERATION.

NOTES:

?? WHAT IS THE DIFFERENCE BETWEEN USEABLE AND NONUSEABLE DATA AS MENTIONED ABOVE ??

?? IS IT PRACTICAL TO STORE THIS TYPE OF NUMERIC DATA IN A STRING ??

```
 10 ! LAB3 - BYRNE 2/81
 20 ! HP-85 PROGRAM
 30 !
 40 ! INITIALIZATION
 50 !
 60 PRINTER IS 2 ! PAPER
 70 CRT IS 1 @ CLEAR ! CRT
 80 RANDOMIZE @ OPTION BASE 1
 90 S=0 @ D=1 ! S=SUM CNT D=STRI
    NG POINTER
100 DIM R(100) ! 100 F.P. NUMBER
    S
110 SHORT S(100) ! 100 S.P. #'s
120 DIM S$[500] ! STRING STORAGE
130 !
140 DISP."ENTER # OF READINGS"
150 DISP "TO BE AVERAGED (<101)"
170 INPUT N@ IF N<0 OR N>100 THE
    N BEEP @ GOTO 140
180 !
190 DISP "READY TO START (Y/N)"
200 INPUT S$
210 IF UPC$(S$[1,1])#"Y" THEN BE
    EP @ DISP "GET READY" @ WAIT
     5000 @ CLEAR @ GOTO 190
220 !
225 !
230 ! BEGINNING OF LOOP
240 !
250 FOR C=1 TO N
260 R(C)=RND @ S(C)=R(C) @ S$[D,
    D+5]=VAL$(S(C))
280 DISP R(C);" ";S(C);"  ";S$[D
    ,D+5]
290 D=D+6 @ S=S+R(C)
300 NEXT C @ DISP
310 !
320 ! RECONVERTING STRING INTO
330 ! USEABLE DATA
340 !
350 D=1
360 FOR C=1 TO N
370 V=VAL(S$[D,D+5]) @ DISP V
380 D=D+6
390 NEXT C
394 !
395 DISP LEN(S$)/6;" READINGS ST
    ORED IN STRING S$"
400 BEEP @ DISP "DONE" @ END
```

```
 10 ! LAB3 - BYRNE 2/81
 20 ! HP-85 PROGRAM
 30 !
 40 ! INITIALIZATION
 50 !
 60 PRINTER IS 2 ! PAPER
 70 CRT IS 1 @ CLEAR ! CRT
 80 RANDOMIZE @ OPTION BASE 1
 90 S=0 @ D=1 ! S=SUM CNT D=STRI
    NG POINTER
100 DIM R(100) ! 100 F.P. NUMBER
    S
110 SHORT S(100) ! 100 S.P. #'s
120 DIM S$[500] ! STRING STORAGE
130 !
140 DISP."ENTER # OF READINGS"
150 DISP "TO BE AVERAGED (<101)"
170 INPUT N@ IF N<0 OR N>100 THE
    N BEEP @ GOTO 140
180 !
190 DISP "READY TO START (Y/N)"
200 INPUT S$
210 IF UPC$(S$[1,1])#"Y" THEN BE
    EP @ DISP "GET READY" @ WAIT
     5000 @ CLEAR @ GOTO 190
220 !
225 !
230 ! BEGINNING OF LOOP
240 !
250 FOR C=1 TO N
260 R(C)=RND @ S(C)=R(C) @ S$[D,
    D+5]=VAL$(S(C))
280 DISP R(C);" ";S(C);"  ";S$[D
    ,D+5]
290 D=D+6 @ S=S+R(C)
300 NEXT C @ DISP
310 !
320 ! RECONVERTING STRING INTO
330 ! USEABLE DATA
340 !
350 D=1
360 FOR C=1 TO N
370 V=VAL(S$[D,D+5]) @ DISP V
380 D=D+6
390 NEXT C
394 !
395 DISP LEN(S$)/6;" READINGS ST
    ORED IN STRING S$"
400 BEEP @ DISP "DONE" @ END
```

HEWLETT hp PACKARD

TAPE CAPABILITIES

1. Store and load programs

2. Store and load data (numeric & string)

3. Load new programs and run them on command from another program

4. Upon power on (or power-fail restart) load and execute any program stored with the name

   Autost

NOTES:

HEWLETT hp PACKARD

HP - 85 DATA CARTRIDGE (APPENDIX B)

General Information for Use

Rewind time	29 seconds
Initialization time	15 seconds
Search speed	60 inches per second
Read/write speed	10 inches per second
Tape length	43 meters (140 feet)
Number of tracks	2 independent tracks
Typical tape capacity	780 program records (195K bytes) 850 data records (210K bytes)
Tape directory capacity	42 files (directory entries)
Typical access rate (search speed)	7,800 bytes/second
Typical transfer rate	650 bytes/second
Typical tape life (continuous use)	50 to 100 hours
Typical error rate*	<1 in 10^8 bits (that's less than one in every 100 million!)

Protect tape from magnetic fields!

NOTES:

MASS STORAGE OPERATIONS

REVIEW:

```
ERASETAPE

CAT

STORE   "PROGRAM NAME"

LOAD    "PROGRAM NAME"

PURGE   "PROGRAM NAME"
```

NOTES: Use the program name Autost if you want automatic startup upon power on.

Program name must be less than 7 characters.

HEWLETT PACKARD

MASS STORAGE OPERATION

OF STORING DATA

IN ORDER TO STORE DATA ON A MASS STORAGE MEDIUM:

I. MEDIUM MUST BE INITIALIZED.

II. FILE MUST BE CREATED IF NEW.

III. FILE MUST BE OPENED.

IV. PRINT DATA ONTO FILE.

V. READ DATA FROM FILE.

VI. CLOSE FILE.

NOTES:

HEWLETT hp PACKARD

DATA STORE REQUIREMENTS

TYPE OF VARIABLES	NUMBERS	STRINGS
SINGLE VARIABLE	8 BYTES PER NUMBER	1 BYTF PER CHAR. + 3 BYTES PER STRING + 3 BYTES PER RECORD CROSSED
ARRAY VARIABLE	8 BYTES X NUMBER OF ARRAY ELEMENTS	N/A

NOTES:

ANY NUMBER OF ANY TYPE CONSUMES 8 BYTES

CREATING DATA FILES

PURPOSE:

* ESTABLISHES DATA FILES WITHIN A CERTAIN NAME AND A CERTAIN SIZE.

SYNTAX:

```
CREATE "FILE NAME", NUMBER OF RECORDS
[ ,RECORD LENGTH]
```

EXAMPLE:

```
50 CREATE "SCORES", 5, 100
```

NOTES:

* RECORD - SMALLEST ADDRESSABLE UNIT ON TAPE
* PHYSICAL RECORD - 256 BYTES, DEFAULT
* LOGICAL RECORD - 4 BYTES TO 32,767 BYTES
  DEFINED BY CREATE
* CREATION OF 256 BYTE RECORD LENGTH FILES MAKES BEST USE OF THE STORAGE MEDIA IN TERMS OF STORAGE CAPACITY AND TRANSFER SPEED.
* THE TOTAL TRANSFER TIME OF DATA TO TAPE WILL BE A FUNCTION OF;
  A) WHETHER THE FILE MUST BE CREATED.
  B) THE NUMBER OF RECORDS TO BE STORED.
  C) WHETHER SERIAL OR RANDOM ACCESS IS USED.
  D) THE INDIVIDUAL RECORD LENGTH.
* THE CREATE STATEMENT MUST NOT BE EXECUTED TWICE FOR THE SAME FILE NAME

OPENING A FILE

PURPOSE:

ENABLES ACCESS TO A CREATED FILE BY ASSIGNING A BUFFER TO IT.

SYNTAX:

ASSIGN # BUFFER NUMBER TO "FILE NAME"

EXAMPLE:

ASSIGN #1 TO "SCORES"

NOTES:

- Buffer no: 1 thru 10
- File names: previously created file
- Data being transferred to the file is first buffered in 256 Byte buffer
- Up to 10 files may be active at one time (ASSIGNED).
- A second ASSIGN using same buffer number deletes the first ASSIGN.

CLOSING A FILE

PURPOSE:

CLEARS BUFFER BY STORING REMAINING BUFFER CONTENTS ON STORAGE MEDIUM.

SYNTAX:

```
ASSIGN # BUFFER NUMBER TO *
```

EXAMPLE:

```
ASSIGN # 1 TO *
```

NOTES:

Closing:

Important to close all files after use - to assure buffer is cleared, and all intended information has reached mass storage medium.

SERIAL VS RANDOM ACCESS

SERIAL ACCESS

It is used when data is stored and retrieved as an entire block. An example of this might be where measurement data is taken and stored away as a group, and later loaded in as a group for reduction or presentation.

It is faster in transfer.

An array identifier can be used.

It takes less space.

RANDOM ACCESS

It is used when data needs to be pulled off the data file selectively, ie a record or selected records need to be retrieved or modified.

It is slower in transfer.

A record identifier must be specified in addition to the data value.

3 bytes per record overhead is needed.

NOTES:

STORING AND RETRIEVING DATA

SERIAL ACCESS

PURPOSE:

STORES AND RETRIEVES DATA SEQUENTIALLY. USEFUL WHEN THE ENTIRE DATA LIST IS TO BE STORED AND RETRIEVED AS A UNIT.

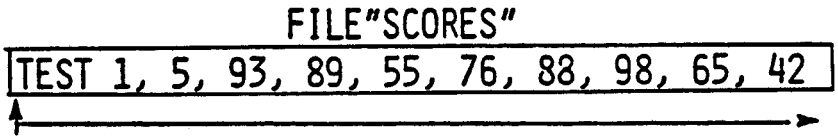
SYNTAX:

PRINT # BUFFER NUMBER; [PRINT # LIST]

READ # BUFFER NUMBER; VARIABLE LIST

```
PRINT #1; V( , )
READ  #4; A$
```

NOTES:



ITEMS ARE ACCESSED SEQUENTIALLY WITHOUT REGARD TO RECORD DIVISIONS. ACCESS THE COMPLETE FILE AS A UNIT.

SERIAL ACCESS

SERIAL PRINT

```
10  REM **PRINT  ARRAY**
20  OPTION BASE 1
30  DIM A(20,20)
40  FOR I=1 TO 20
50  FOR J=1 TO 20
60  A(I,J)=I*J
70  NEXT J
80  NEXT I
90  CREATE "MATRIX", 20, 160
100 ASSIGN# 1 TO "MATRIX"
110 PRINT # 1 ; A(,)
120 ASSIGN# 1 TO *
130 END
```

SERIAL READ

```
10  REM **READ ARRAY**
20  OPTION BASE 1
30  DIM X(20, 20)
40  ASSIGN# 5 TO "MATRIX"
50  READ# 5; X(,)
60  END
```

NOTES:

* When reading data in serial mode, the file pointer must be positioned to the beginning of the assigned file. This may be done two ways:

  . An ASSIGN statement that opens a file, positions the file pointer to the beginning of the file.

  . You may also use a random READ statement without a read list; e.g.,: READ # 1,1

  If the file pointer is not repositioned, reading begins after last item printed or read.

* Data must be read to variables that correspond in type exactly. I,e.,strings to string variables, numbers to numeric variables. Variable names can be different. Arrays can be read as simple variables and the reverse and can read back into different variables.

* You cannot access a file after it has been closed.

STORING AND RETRIEVING DATA

RANDOM ACCESS

PURPOSE:

ENABLES ACCESS TO PORTIONS OF THE DATA BY PRINTING TO, READING FROM, OR UPDATING INDIVIDUAL RECORDS OF A DATA FILE.

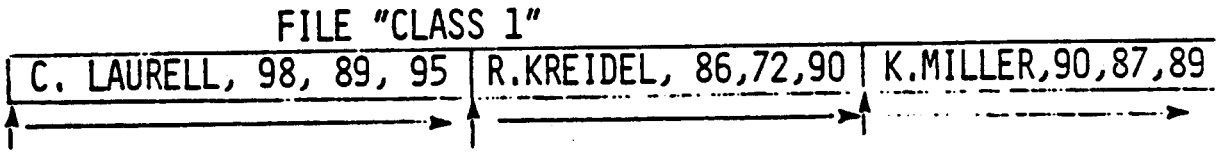
SYNTAX:

PRINT # BUFFER NUMBER,RECORD NUMBER [;[PRINT # LIST]]

READ # BUFFER NUMBER,RECORD NUMBER [;VARIABLE LIST]

```
PRINT #1,I;V(I)
```

NOTES:



ITEMS ARE ACCESSED BY RECORD NUMBER. ACCESS PORTIONS OF THE FILE BY INDIVIDUAL RECORD.

RANDOM PRINT AND READ

EXAMPLE:

```
 10  REM **RANDOM PRINT**
 20  DIM A$ [60]
 30  CREATE "NAME", 10, 63
 40  ASSIGN# 2 TO "NAME"
 50  FOR I=1 TO 10
 60  INPUT A$
 70  PRINT # 2, I ; A$
 80  NEXT I
 90  ASSIGN # 2 TO *
100  END
```

```
 10  REM **RANDOM READ**
 20  DIM X$ [60]
 30  ASSIGN# 6 TO "NAME"
 40  DISP "RECORD NUMBER";
 50  INPUT N
 60  READ# 6,N ; X$
 70  PRINT X$
 80  GOTO 40
 90  END
```

MASS STORAGE OPERATIONS

RENAME

PURPOSE:

TO CHANGE FILE NAME

SNYTAX:

RENAME "OLD FILE NAME" TO "NEW FILE NAME"

EXAMPLE:

RENAME "SCORES" TO "GRADES"

NOTES:

CAT COMMAND & FILE STORAGE NOTES

CAT - OUTPUTS A CATALOG OF TAPE CONTENTS

NAME	TYPE	BYTES	RECS	FILE
Data	DATA	256	94	1
Autost	PROG	256	37	2
97PTSc	PROG	256	24	3
3456TM	PROG	256	36	4
MERGE	BPGM	256	7	5
	NULL	256	1	6
THERMO	DATA	256	32	7
REFJCT	DATA	256	3	8
RFTEMP	PROG	256	28	9
	NULL	256	5	10
	NULL	256	1	11
	NULL	256	1	12

NOTES:

NOTES ON FILE CREATION

A) Whenever a program file is LOADED and then modified such that over 200 bytes of program has been added, when that program is later STORED it will be STORED in a different area of the data cartridge.

It's previous location will become a NULL file, and is available for future use.

The program will be stored at the first available NULL file that has sufficient record length to hold it. If none exists, the program will be stored at the end of the last file.

B) Filling the NULL file - A CREATE or STORE statement will cause the HP-85 to search its directory for an available file. If it finds ANY NULL file of sufficient size it will use it, even though it may be a tremendous waste of file space. For example if the first NULL file is 100 records long, and a CREATE "DATA",5 is performed this file will be put here, even though 95 records will be wasted.

NOTES:

```
NAME     TYPE   BYTES    RECS FILE
69735    PROG     256       2    1 -----original storage - 2 records
69751a   PROG     256      36    2
69751b   PROG     256      38    3
97INT    PROG     256       4    4
790_MT   PROG     256      71    5
751FHS   PROG     256      72    6
69720a   PROG     256      72    7
WAVEFM   PROG     256       4    8
```

```
NAME     TYPE   BYTES    RECS FILE
         NULL     256       2    1 -----original file NULLED
69751a   PROG     256      36    2
69751b   PROG     256      38    3
97INT    PROG     256       4    4
790_MT   PROG     256      71    5
751FHS   PROG     256      72    6
69720a   PROG     256      72    7
WAVEFM   PROG     256       4    8
69735    PROG     256       4    9 -----new storage - 4 records
```

HEWLETT PACKARD

MASS STORAGE OPERATION

BINARY PROGRAMS

PURPOSE:

TO STORE AND RETRIEVE BINARY PROGRAMS

SYNTAX:

STOREBIN "PROGRAM NAME"

LOADBIN "PROGRAM NAME"

NOTES:

HEWLETT hp PACKARD

BINARY UTILITY "MERGE"

IT ADDS THREE COMMANDS TO THE HP-85;

* SAVE - eg SAVE"PROG A"

  SAVE"TEST B",50,150

  It puts a program on the data cartridge as a data file rather than as a program file.

  + Allows other program segments that have also been "SAVED" to be merged together.

  + Allows transportability from HP-85 to HP-85 with different ROM configurations.

* GET - Fetches a "SAVED" program.

* MERGE - Fetches a "SAVED" program and adds it to the program currently in memory.

NOTES:

The line numbers of the progam being merged must be different from the line numbers of the program already in memory.

If the merged program is to be added to the end, the line numbers of the program should all be higher than the progam lines of the program currently in memory. This requires a little forsight in program development.

MERGING PROGRAMS - EXAMPLE

1) Develop a program or LOAD a program from data cartridge.
2) LOADBIN "MERGE"
3) Renumber the program lines as appropriate.
4) SAVE this program on data cartridge under another name.
5) Repeat steps 1 through 4 for other program segments.
6) GET first program.
7) MERGE second program.
8) MERGE repeated as needed to merge all programs .
9) Study entire merged program and correct any errors that result from the merging (GOTO's , PAUSES, GOSUB's Etc.)
10) Renumber the program
11) STORE or SAVE on data cartridge as desired.

NOTES: Loading a program erases the binary program.

USING THE 82901 M/S FLEXIBLE DISC DRIVE

* Roughly 3 times faster than the data cartridge.
* Requires the mass storage ROM.
* Capacity up to 286.72 K Bytes (Controller dependent).
* Factory select code is 00.
* Uses same commands as data cartridge after,
  MASS STORAGE IS ":D700" is executed.

NOTES:

* Use random access to save space.
* Use serial access for speed.
* Use multiple buffers with numerous records. (10 are allowed)

TRANSFER TIMES: DATA CARTRIDGE

82901 DISC DRIVE

TIME TO TRANSFER 1000 NUMERIC VALUES (8000 bytes)

STORAGE DEVICE	# of RECORDS	BYTES/RECORD	# BUFFERS	TIME (SEC)
tape	32	256	1	93
disc	32	256	1	25
tape	1000	8	1	96
disc	1000	8	1	31
			10	26
tape	1	8008	1	91
disc	1	8008	1	25
			10	26

NOTES:

USING THE TAPE DRIVE (SUMMARY)

1. Select new tape cartridge.

2. Set 'record' tab in left-most position.

3. Place cartridge in tape drive.

4. Initialize tape:

   Erasetape [ENDLINE]

   * Erases tape
   * Sets up directory
   * Use only with
     1. New tape
     2. Erasing all contents of old tape

5. To place a copy of a program on tape:

   Store "NAME" [END LINE]

6. To bring a copy of a program on tape into computer memory:

   Load "NAME" [END LINE]

7. To view the catalog of programs on tape:

   CAT [ENDLINE]

   Example: Do the following with a blank tape cartridge:

   ERASE TAPE

   CAT

   STORE "AMORT"

   CAT

8. To erase one program from the tape:

   Purge "FILE NAME" [ENDLINE]

LAB 4

A) Modify your program to store the numeric data you've generated. Store also the numeric data that is in string variable form.

B) Write a program that reads this data from the data cartridge and prints it out.

C) Merge the two programs together.

EXTRA CREDIT:

A) Experiment with the record length, serial vs random file access to find what type of access and record size will best meet your storage needs for your company's application.

B) See what happens when the data is stored in segments during the generation process rather than after it has all been generated.

NOTES:

?? Can a file that has been stored via serial access be read via a random access??

?? Vice versa??

HEWLETT PACKARD

THINKING ABOUT AND SOLVING PROBLEMS - STRUCTURED PROGRAMMING

1) Define the problem

   A) Write down the problem, objectives, and what needs to be done to attain those objectives.

   B) Identify constraints and how they affect solving the problem (memory, speed, overlapped processing, etc.)

   C) Break the problem down into smaller sections.

   D) Don't touch the computer!

2) Model a solution of the various problem sections using flowcharting or pseudocode.

3) Convert flowchart/pseudocode to HP-85 program code.

   A) One section at a time

   B) Document

4) Troubleshoot by section.

NOTES:

A structured approach to program development involves:

1) Top down development (focuses on the partially complete program running during development.)

2) Structured programming (effort to produce readable code)

3) Structured walk through (review by peers)

PROGRAM DOCUMENTATION

WHY

Documentation makes the difference between having a program which is readable and one which is not. Programs without documentation are programmer dependent - you'll always have to answer other programmer's questions about your program. Documentation will save hours of searching for a particular piece of info about a long program.

WHAT

* Provide program overview
* Description of subroutines
* Meaning/use of variables
* INPUT and OUTPUT lists, FORMATS
* Special conditions

HOW

* Use REM statements or ! to identify program beginning and offset major sections of code.

* Use comments (!) to explain complex lines

NOTES:

HEWLETT PACKARD

FLOWCHARTING OR PSEUDOCODE?

Both accomplish the same thing:

> Forces one to generate the logical sequence of events which must occur to obtain the desired objective.
>
> Has one consider the consequence of all possible alternative courses of action which could happen, (i.e. If it doesn't do this, it could do this, of that, or something else.)

Use whichever method you feel most comfortable with.

NOTES:

HEWLETT hp PACKARD

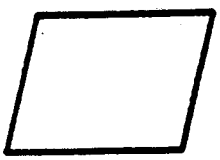
FLOWCHARTING

A GRAPHIC REPRESENTATION OF PROGRAM CONTENT

* TOOL TO DESIGN PROGRAM STRUCTURE
* TOOL TO DESCRIBE/DOCUMENT PROGRAM STRUCTURE
* TOOL TO EVALUATE PROGRAM STRUCTURE

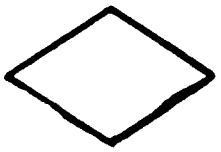
NOTES:

STARTER/TERMINATOR

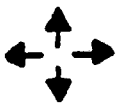


INPUT/OUTPUT

PROCESS

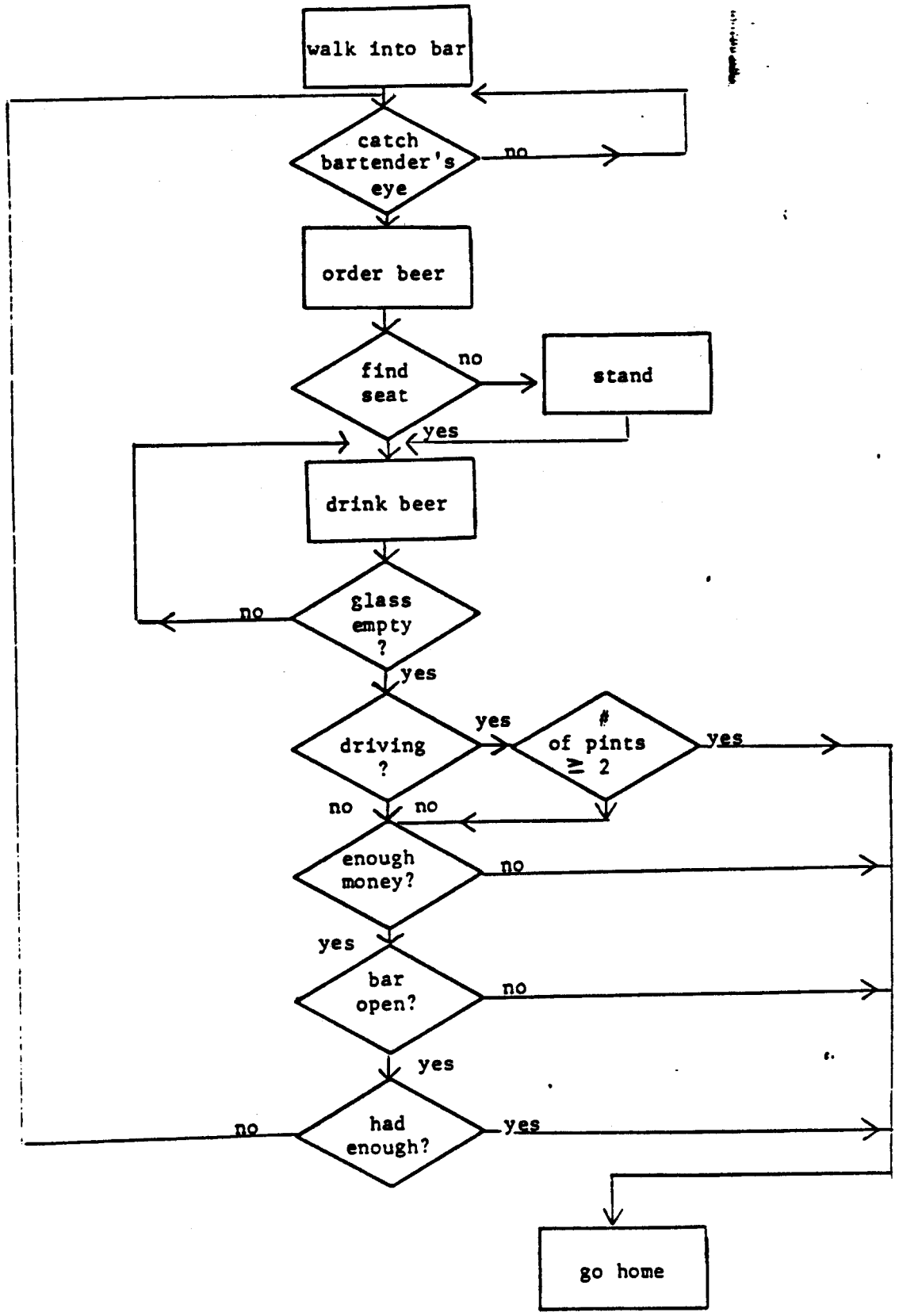


DECISION



FLOW DIRECTION

Flowchart for getting a beer.



HEWLETT PACKARD

PSEUDOCODE

* Represents an alternative to detailed flow charting

* Bridges the gap between conversation and program language

* Can be used to express a more complex operation; ie: "Find a matching record"

* Not bound by formal rules

SUGGESTIONS:

1. Write each statement on a separate line

2. Do not abbreviate too much

3. Leave space for changes

NOTES:

EXAMPLE: PSEUDOCODE FOR LAB 3

```
INITIALIZE CRT, PRINTER, LOOP COUNTER VARIABLE N, SUM
VARIABLE S, & SUBSTRING POINTER VARIABLE D

ALLOCATE MEMORY FOR STORAGE OF DATA - F(100), S(100),
S$(600)

ENTER FROM KEYBOARD NUMBER OF READINGS TO BE GENERATED.
CHECK FOR PROPER ENTRY.

LOOP FOR N TIMES
    GENERATE READING VIA RANDOM NUMBER GENERATOR & STORE
    CREATE SHORT PRECISION VERSION & STORE
    STORE SHORT PRECISION VERSION AS SUBSTRING OF
      6 CHARACTERS
    COMPUTE RUNNING SUM
    DISPLAY FULL AND SHORT PRECISION NUMBERS.
    CHECK FOR END OF LOOP.

PRINT AVERAGE OF NUMBERS AND THE ENTIRE STRING.
REINITIALIZE LOOP COUNTER & STRING POINTER.

LOOP FOR N TIMES
    COMPUTE NUMBER FROM SUBSTRING & PRINT IT.
    INCREMENT LOOP COUNT & STRING POINTER.
    CHECK FOR END OF LOOP.

END
```

NOTES:

HEWLETT hp PACKARD

BLOCK STRUCTURED PROGRAMMING TOOLS ON THE HP-85

A) SUBROUTINES:

Called to perform various tasks. Upon completion automatically returns to where it came plus 1 line.

B) FUNCTIONS:

Useful in performing same computation but with different data each time.

C) CHAINED PROGRAMS WITH COMMON MEMORY:

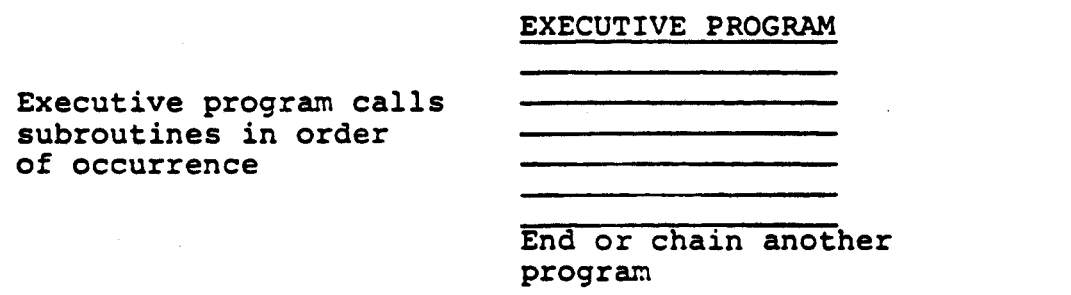
Useful when data in memory leaves limited space for programs. Allows more than 1 program to use common data.

NOTES:

HEWLETT hp PACKARD

STRUCTURED PROGRAMMING

A concept of program organization that breaks a task into logical segments. This type of organization pays benefits during 1. development 2. testing (debugging) 3. documentation



NOTES:

HEWLETT hp PACKARD

IMPLEMENTING STRUCTURED PROGRAMMING

GOSUB STATEMENT

```
                ...   ! DATA ACQUIRE       AUTHOR DATE

                ...   GOSUB 1000   !  GIVE INSTRUCTIONS
                ...   GOSUB 2000   !  PREPARE FOR MEASUREMENTS
EXECUTIVE       ...   GOSUB 3000   !  TAKE MEASUREMENTS
PROGRAM         ...   GOSUB 4000   !  REDUCE AND ANALYZE
                ...   GOSUB 5000   !  CHECK FOR DONE
                900   END

SUBROUTINES     1000  ! INSTRUCTIONS
                1010  PRINT "     "
                ...   PRINT "     "
                ...
                ...   RETURN

                2000  DIM A(1000)  ! STORE 1000 READINGS
                2010  OUTPUT 722; "F1R7T2T3" ! DCV, AUTORANGE MEASUREMENT
                ...
                ...   RETURN
                 .
                 .
                 .    etc
```

NOTES:

GOSUB LINE NUMBER ONLY - NO LABELS

PASS PARAMETERS NOT ALLOWED eg ~~GOSUB 1000 (A,D)~~

NO LOCAL VARIABLES

SUBROUTINE PRACTICE

Write a program which uses subroutines to;

A) BEEP a variable number of times as determined either by the operator or loop counter.

B) Have another subroutine indicate the number of beeps which are to be given to the CRT.

NOTES:

```
 10 ! SUBROUTINE EXERCISE
 20 !
 30 FOR L=1 TO 10
 40 !
 50 GOSUB 240 ! INDICATE COUNT
 60 GOSUB 150 ! MAKE NOISE
 70 !
 80 NEXT L
 90 !
100 STOP ! DON'T GO PAST HERE
110 !
120 !
130 ! BEEP SUBROUTINE
140 !
150 FOR J=1 TO L
160 BEEP L*50,50
170 WAIT 30
180 NEXT J
190 RETURN
200 !
210 !
220 ! DISPLAY COUNT
230 !
240 DISP "LOOP - ";L
250 RETURN
260 END
```

NOTES:

BRANCHING USING SPECIAL FUNCTION KEYS (154 - 156)

```
ON KEY #1, "READ V"  GOSUB 9ØØ
ON KEY #2, "READ F"  GOSUB 12ØØ
ON KEY #8, "DONE" GOTO 9ØØØ
```

GO WHERE SPECIFIED WHEN KEY PRESSED.

KEY LABELS OF UP TO 8 characters may be assigned.

These are displayed on the bottom two lines whenever the "Key Label" key or statement is executed.

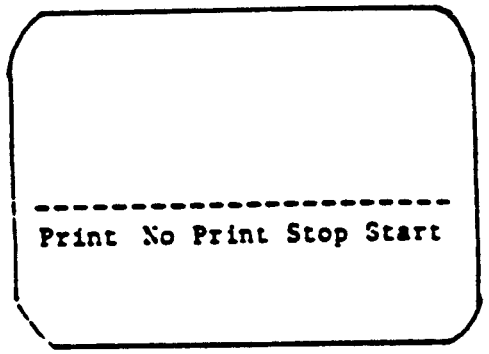
CANCELLED BY OFF KEY # key number

NOTES:

When key is pressed program goes to specified line numer AFTER present program line is finished. This is called an interrupt, and will be explained later in the interrupt section.

SPECIAL FUNCTION KEYS

HP-85 CRT



KEY LABLE	K5 K1	K6 K2	K7 K3	K8 K4

```
... ON KEY #1,"PRINT" GOSUB 2000
... ON KEY #2,"NOPRINT" GOSUB 2040
... ON KEY #5,"START" GOTO 1040
... ON KEY #4, "STOP" GOTO 4000

1010 DISP"WAITING FOR START KEY"
1020 CLEAR
1030 GOTO 1010
1040 !
1050 !
...
...
...  IF P9=1 THEN PRINT "DATA IS"
...
...
...
...
2000 ! SET PRINT MODE
2010 P9=1
2020 RETURN
2030 !
2040 ! PRINT MODE OFF
2050 P9=0
2060 RETURN
...
...
4000 END
```

NOTES:

USE TO SET OR CLEAR CERTAIN CONDITIONS OR MODES OF OPERATION.

USE TO CONTROL PROGRAM EXECUTION ONLY USING SPECIAL FUNCTION KEYS.

USER DEFINED FUNCTIONS (pg. 145 - 151)

You have used some of available HP-85 functions (RND, VAL$(X), SIN) etc. You can also define functions of your own.

Functions you define can be single line or multi-line. They may have no return variable or 1 return variable.

```
1Ø ! SINGLE LINE FUNCTION EXAMPLE

2Ø ! CONVERT deg C TO deg F
25 !
3Ø DEF FNT(T) = 9/5*T + 32
35 !
4Ø DISP "TEMP deg C ?" @ INPUT C
5Ø F=FNT(C)
6Ø DISP "TEMP deg F IS ;";F
7Ø GOTO 4Ø
8Ø END
```

NOTES:

A) Line 5Ø could be omitted and line 6Ø changed to

```
6Ø DISP "TEMP deg F IS ; FNT(C)
```

B) A function does not need an argument, eg;

```
1Ø DEF FNS$ = "10 Seconds"
20 DISP FNS$
3Ø END

RUN

10 Seconds
```

FUNCTION WITH NO ARGUMENT

MULTIPLE LINE FUNCTIONS

Used if function contains lengthly computations or branching operations. Like single line functions, they can have at most 1 argument and one return value.

```
10 ! MULTIPLE LINE FUNCTION EXAMPLE

20 ! CONVERT degC TO degF
30 ! AND DO OTHER THINGS
40 !
50 DISP "TEMP degC " @ INPUT C
60 !
70 F=FNF(C) ! PASS C TO VARIABLE ARGUMENT T
80 DISP
90 DISP "TEMP degF IS ";F
100 !
110 END
120 !
130 !
140 DEF FNF(T)  ! BEGINNING OF FUNCTION
150 !
155 DISP "T =";T
156 !
160 K=T+273
170 DISP "TEMP de K IS ";K
180 X=9/5*T+32
190 FNF=X  !X IS THE RETURN VARIABLE AND IS PASSED INTO F
200 FN END
```

NOTES:

PRINT ALL MODE PRINTOUT

```
TEMP degC
?
20
T = 20
TEMP degK IS ' 293

TEMP degF IS 68
```

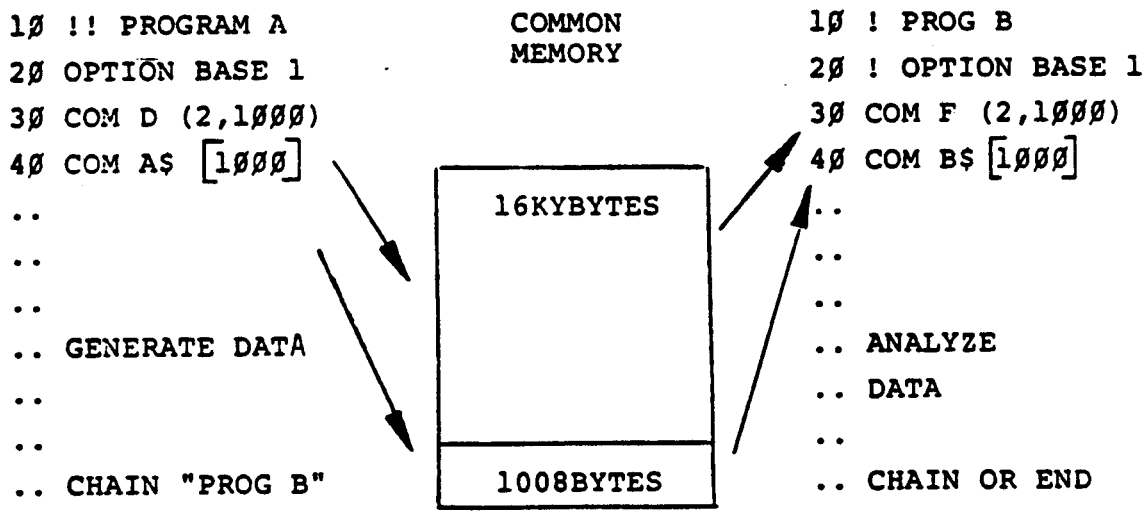
A VARIABLE CHANGED IN THE FUNCTION IS CHANGED IN THE REST OF THE PROGRAM AS WELL.

REFER Pg. 149-151 FOR INFO ON USE OF MULTIPLE LINE FUNCTIONS FOR STRING VARIABLES.

HEWLETT PACKARD

COMMON MEMORY (Pg. 123,179)

Variables can be put into common memory which allows 2 or more programs to access them



NOTES:

CHAIN command will not destroy data stored in common memory.

Variable name (s) must be of like type, but do not have to have same names.

HEWLETT hp PACKARD

DEBUGGING TECHNIQUES

A) Use PAUSE, BEEP , PRINT----------, DISP----------
to verify program areas.

B) Use ! or REM to block execution of program lines.

C) Use keyboard interrupt to stop program execution and examine variables.

D) Remove program line numbers and press END-LINE to execute single lines of code.

E) PRINT ALL mode

F) TRACE mode / STEP mode

NOTES:

LAB 5

A) Multiply your generated data by .0001 to turn your generated data into pretend microvolt data.

B) Add the subroutine given below to convert the microvolt data into degrees centigrade. Print the raw and reduced data.

C) Change from a subroutine to a function.

EXTRA CREDIT:

Write a program to sort the numeric data you have generated in ascending order. After sorting the data print it out.

A) Flowchart and Pseudocode for one sorting technique follow on next 2 pages.

B) Write your sorting program using block structured programming techniques.

C) Have the above program chain this one.

NOTES:

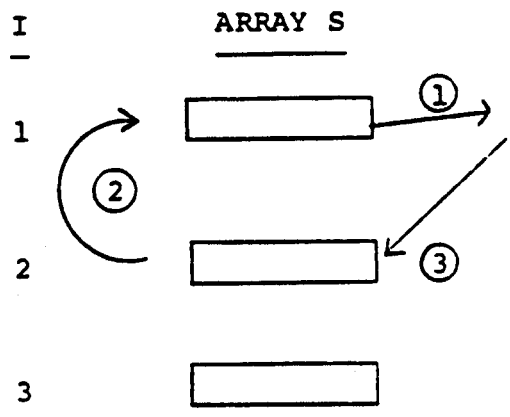
```
XXXX  ! CONVERT SUBROUTINE TYPE J
XXXX  ! uV TO DEGREES F
XXXX  V1=-2.128839E-8 + .0000503824 * T1 + 2.97284E-8
      *T1*T1 + -6.91111E-11 *T1*T1*T1
XXXX  V2=V+V1
XXXX  V3=V2-.0116
XXXX  V=17978.7 * V3 + 10837.9 *V3 * V3
      + -177978 *V3*V3*V3 +215.034
XXXX  RETURN
XXXX  ! T1= Reference junction temperature
XXXX  ! set = to 25
XXXX  ! V=microvolt data going in
XXXX  !   & degrees centigrade going out
```

If the instructor has this routine on a data cartridge MERGE it into your existing program.

PSEUDOCODE FOR SORT ROUTINE

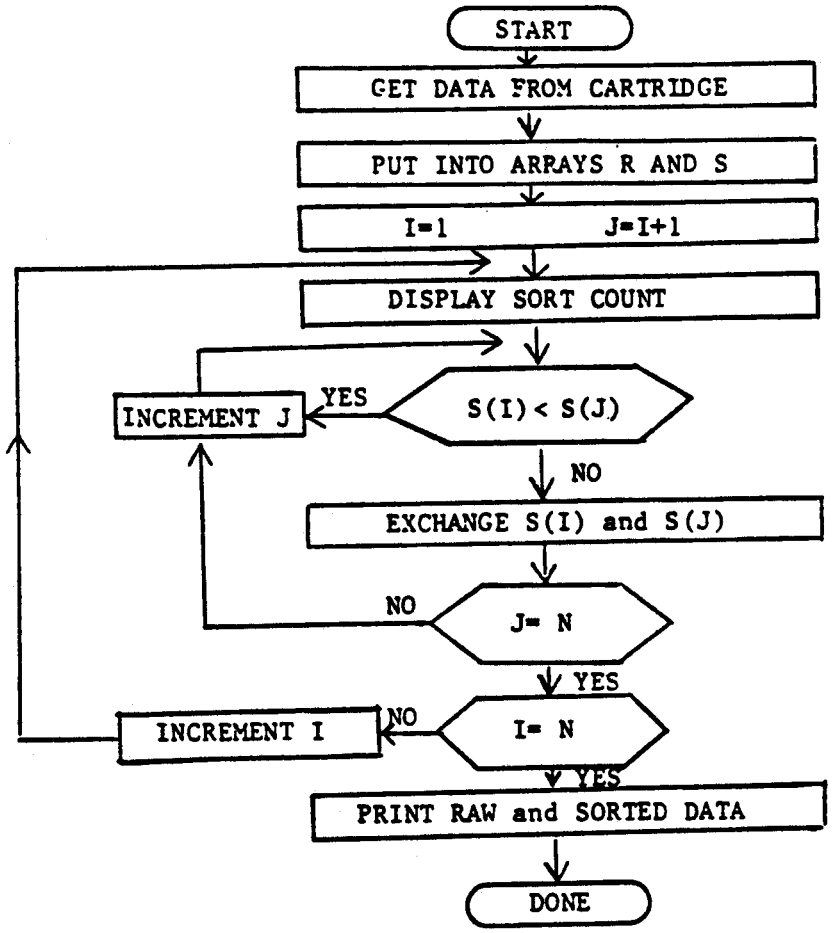
```
Create arrays to hold raw data and sorted data

Read in data from data cartridge

Transfer raw data to sorted data array

Loop A:  For I=1 to N (N=# of raw data points)

         Display the sort count

         Loop B:  For J= I + 1 to N

                  Compare the Ith and Jth elements
                  IF Ith element smaller continue
                  Loop B

                  IF Ith element larger switch Ith
                  and Jth elements

         End Loop B

End Loop A

Print out raw and sorted data
```

NOTES:



To exchange the values of positions 1 & 2

1. Save Value [1] in T
2. Store Value [2] into S Position 1
3. Store Value of T into S Postion 2



NOTES:

Array R holds raw data

Array S - at beginning also holds raw data
at end holds sorted data

N= Number of data points to be sorted

HEWLETT hp PACKARD

FORMATTING UP TO NOW

```
PRINTER IS 2
PRINTER IS 701,8Ø
CRT IS 2
```

Know how to select printer or display device.

```
PRINT A,B,C,D,E,
PRINT A;B;C;D;E
DISP  "A=";A,  "B=";B
```

Know how to use , and ; to separate items to be displayed or printed.

```
DISP  "A="; TAB(20);A
```

Know how to use TAB

KNOW WHAT FREE-FIELD FORMAT DOES ON THE CRT AND PRINTER.

Do not know how to format numbers to meet specific requirements

Do not know how to enter information.

NOTES:

Free-field Format to external device

* Format used when no IMAGE statement used.

* There are two forms of free-field format. The one used depends on whether the delimiter is a:

		Numeric Data	String Data
semicolon ;	Compact Field	Digits of the number are output, preceded by a space (if plus) or a minus sign (if minus), and followed by one space.	Characters of the string are output with no leading or trailing spaces.
comma ,	Free Field	Digits of the number (with leading space or minus sign) are output left-justified in a field of 11,21, or 32 characters. Trailing spaces are output as necessary to fill the unused portion of the field.	Characters of the string are output with no leading spaces and no more than 20 trailing spaces.

NOTES:

* PRINTER IS 701 .- Assigns device 01 on HP-IB bus 7 to be printer.

  PRINTER IS 6 - Assigns device on select code 6 to be printer.

* PRINTER IS 704,80 - Allowed with Printer/Plotter ROM

  - issue EOL sequence after 80 characters have been printed.

FREEFIELD OUTPUT EXAMPLES (WITH PRINTER/PLOTTER ROM)

GIVEN: PRINTER IS 701,70
A= 2.34E15 B= +9999999
C= -9999999 D= .0123456789 A$="TESTING 12345"

```
1234567890123456789012345678901234567890123456789012345678901234567890
```

```
PRINT A,B,C,D,A,B,C D,A

 2.34E15                9999999          -9999999
 .0123456789            2.34E15           9999999
-9999999                .0123456789       2.34E15
```

```
PRINT A;B;C;D;A;B;C D;A

 2.34E15  9999999 -9999999  .0123456789  2.34E15  9999999 -9999999
 .0123456789  2.34E15
```

```
 PRINT A$,A$,A$,A$,A,B,C

TESTING 12345          TESTING 12345      TESTING 12345
TESTING 12345           2.34E15            9999999
-9999999
```

```
PRINT A$;A$;A$;A$;A,B,C

TESTING 12345TESTING 12345TESTING 12345TESTING 12345 2.34E15  9999999
-9999999
```

NOTES:

HEWLETT hp PACKARD

FORMATTING WITH IMAGE

WHY IMAGE?

* Get rid of leading, trailing blanks.
* Columnize with respect to decimal point.
* Right - Justify columns.
* Insert commas, periods for more humanized output of long numbers.
* Easy control over number of significant digits printed or displayed.
* Neater output

NOTES:

EXAMPLE:

From this;

```
10 FOR I=1 to 10
20 PRINT RND
30 NEXT I
40 END
```

```
 .879546874885
 .510432700278
 .850038458669
 .516683883424
 .62216477247
 .900864041057
 .581942277323
 .950860003951
 .222269723105
2.90271847999E-2
```

To this

```
10 FOR I=1 to 10
20 PRINT USING "SD.DDDD"; RND
30 NEXT I
40 END
```

```
+.9914
+.0844
+.5900
+.1846
+.8404
+.6418
+.8203
+.3800
+.0432
+.0856
```

HEWLETT PACKARD

IMAGE STATEMENT

They define exactly how information will be transferred.

An IMAGE statement works in combination with a PRINT, OUTPUT, DISP, or ENTER statement. The two statements reference each other.

```
11Ø  IMAGE  specifications   ----------- HOW
                                        - WHERE
12Ø  OUTPUT  722 USING 110;  data list  - WHAT

...

81Ø  ENTER  709 USING 110;  data list

...

92Ø  PRINT USING 110;  data list
```

NOTES:

Items in data list are separated by , or ;

OUTPUT 709 USING 120; A, A$, V(C4)

IMAGE specifications can be included in OUTPUT, ENTER, DISP or PRINT statement

ENTER 722 USING "K,XX,DDD"; V(I)

For proper transfer items in the data list must correspond to the corresponding IMAGE specifications.

An IMAGE specification will be reused repetitively until exhausted by the data list.

Printer and Display Formatting via IMAGE statement

Purpose: Customize Display or Printed Cutput. The IMAGE statement defines exactly how information will be displayed or printed.

Syntax:

```
XXX   PRINT USING Line number  ;expression list
XXX   DISP USING Line number  ;expression list
XXX   IMAGE  format string
```

Examples:

```
10 PRINT USING 20; "PI=", PI
20 IMAGE 11X, 3A, D.DDDD
30 A$ = "COST = $"
40 PRINT USING 50 ; A$, 2*PI, A$, 3*PI
50 IMAGE 8A, D.DD, 7X, 8A, D.DD
60 END
```

Results in:

```
123456789Ø123456789Ø123456789Ø1
            PI=3.1416
COST = $6.28        COST= $9.42
```

NOTES:

* Line Number: Specifies desired IMAGE statement.
* Expression List: Variable names, numeric expressions, string expressions separated by commas or semicolons.
* Format String: A list of IMAGE symbols that specifies the exact output format of the expression list.
* IMAGE can appear anywhere in program.
* Many PRINT USING or DISP USING statements can reference the same IMAGE statement.
* For proper output items in the expression list must correspond to the corresponding IMAGE specifications.

HEWLETT hp PACKARD

More on the IMAGE statement

Alternate method of using IMAGE

```
11Ø A$=" VOLTS"
12Ø PRINT USING "8X,D.DDD,6A";6.324,A$
```

is the same as

```
11Ø IMAGE 8X,D.DDD,6A
12Ø PRINT USING 110;6.324,A$
```

Reuse of IMAGE specification

```
18Ø PRINT USING "10D.2D";A,B,C,D,
```

An IMAGE specification will be reused repetitively until exhausted by the expression list.

NOTES:

Separate items in IMAGE statement with comma's.

Remember: An end-of-line sequence (default CR/LF) is sent after all items in the expression list have been transferred.

HEWLETT hp PACKARD

NUMERIC OUTPUT IMAGE SPECIFIERS

MAINFRAME

```
   nD - digit field with leading blanks
   nZ - digit field with leading zeros
   n* - digit field with leading *'s (asterisks)
    E - exponent field ( 5 characters (eg E+499))
    S - sign field (+ or -)
    M - sign field (blank or -)
    . - decimal point (American radix) eg 01.234
    R - comma (European radix)  eg 01,234
    C - comma (American separator) eg 1,200,000
    P - period (European separator) eg 1.200.000
   nK - compact free-field specifier using standard number format
 n( ) - replicating field
   nX - insert blank
```

I/O ROM

```
   e - exponent field (4 characters (eg. E+99))
   B - send one eight bit byte
   W - send two   "     "  bytes (1 word)
```

means symbol can be replicated times

NOTES:

* End-of-line (EOL) Control

  I/O ROM
  / issue EOL sequence (default is CR LF)
  # suppress end-of-line sequence (default is CP LF)
  % ignored

* All numeric specifiers (except B and W) output ASCII characters

* S and M may be used only at the beginning of a number.

* Overflowing an IMAGE will cause completely unpredictable results.

* With B specifier if the number if >255 or< Ø then a MOD 256 is performed on the number. (the lower eight bits are sent out.)

* Using W on an eight bit interface (HP-IB outputs the most significant byte (bits 8-15 first, then follows the least significant byte (bits 0-7).

  W must be a number < 32767 and > -32768

FORMATTED NUMERIC OUTPUT PRACTICE

GIVEN T= 1234567 , T1= -1234567

T2= .1234

Use formatted PRINTs or DISPs to generate these outputs

```
123456789Ø123456789Ø123456789Ø12
-1234567.000
+1234567.0
 1234567
1.235E+006
 -1,234,567.ØØ
Ø.1234    Ø.1234    Ø.1234
 1234567.00 }  both in 1
 0000000.12 }  IMAGE STATEMENT
 1.235E+06
```

NOTES:

HEWLETT PACKARD

ASCII Code - A 7 Bit representation of numbers, letters and symbols

"A" = 65_{10}

"B" = 66_{10}

"C" = 67_{10}

.
.
.

The calculator automatically outputs the ASCII codes corresponding to characters unless a binary output is specified.

NOTES:

THE BINARY FORMAT SPECIFICATION

The binary specification ("B") allows the programmer to output specific digital patterns to the output lines of an I/O card. The 82937 HP-IB card has 8 data lines. A write referencing a binary format give control over all 8 lines.

BIT #	7	6	5	4	3	2	1	0
VALUE	1	Ø	1	Ø	Ø	1	Ø	Ø

To get this pattern output, compute the decimal value of the binary number ($4 + 32 + 128 = 164_{10}$) and

```
OUTPUT 1 USING"B";164
```

NOTES:

TRY THIS ON THE HP-85

```
1Ø  FOR I = 1  to 128
2Ø  IMAGE DDD,XX,B
3Ø  PRINT USING 2Ø ; I,I
4Ø  NEXT I
5Ø  END
```

ASCII CHART

HP-IB		ASCII	Decimal	Binary	Octal	Hexa-decimal
Addressed Command Group ·ACG	GTL	NUL	0	00 000 000	000	00
		SOH	1	00 000 001	001	01
		STX	2	00 000 010	002	02
		ETX	3	00 000 011	003	03
	SDC	EOT	4	00 000 100	004	04
	PPC	ENQ	5	00 000 101	005	05
		ACK	6	00 000 110	006	06
		BEL	7	00 000 111	007	07
	GET	BS	8	00 001 000	010	08
	TCT	HT	9	00 001 001	011	09
		LF	10	00 001 010	012	0A
		VT	11	00 001 011	013	0B
		FF	12	00 001 100	014	0C
		CR	13	00 001 101	015	0D
		SO	14	00 001 110	016	0E
		SI	15	00 001 111	017	0F
Universal Command Group ·UCG	LLO	DLE	16	00 010 000	020	10
		DC1	17	00 010 001	021	11
		DC2	18	00 010 010	022	12
		DC3	19	00 010 011	023	13
	DCL	DC4	20	00 010 100	024	14
	PPU	NAK	21	00 010 101	025	15
		SYN	22	00 010 110	026	16
		ETB	23	00 010 111	027	17
	SPE	CAN	24	00 011 000	030	18
	SPD	EM	25	00 011 001	031	19
		SUB	26	00 011 010	032	1A
		ESC	27	00 011 011	033	1B
		FS	28	00 011 100	034	1C
		GS	29	00 011 101	035	1D
		RS	30	00 011 110	036	1E
		US	31	00 011 111	037	1F
Listen Address Group ·LAG Note 1	L0	SP	32	00 100 000	040	20
	L1	!	33	00 100 001	041	21
	L2	"	34	00 100 010	042	22
	L3	#	35	00 100 011	043	23
	L4	$	36	00 100 100	044	24
	L5	%	37	00 100 101	045	25
	L6	&	38	00 100 110	046	26
	L7	'	39	00 100 111	047	27
	L8	(	40	00 101 000	050	28
	L9	)	41	00 101 001	051	29
	L10	*	42	00 101 010	052	2A
	L11	+	43	00 101 011	053	2B
	L12	,	44	00 101 100	054	2C
	L13	-	45	00 101 101	055	2D
	L14	.	46	00 101 110	056	2E
	L15	/	47	00 101 111	057	2F
	L16	0	48	00 110 000	060	30
	L17	1	49	00 110 001	061	31
	L18	2	50	00 110 010	062	32
	L19	3	51	00 110 011	063	33
	L20	4	52	00 110 100	064	34
	L21	5	53	00 110 101	065	35
	L22	6	54	00 110 110	066	36
	L23	7	55	00 110 111	067	37
	L24	8	56	00 111 000	070	38
	L25	9	57	00 111 001	071	39
	L26	:	58	00 111 010	072	3A
	L27	;	59	00 111 011	073	3B
	L28	<	60	00 111 100	074	3C
	L29	=	61	00 111 101	075	3D
	L30	>	62	00 111 110	076	3E
	UNL	?	63	00 111 111	077	3F

HP-IB		ASCII	Decimal	Binary	Octal	Hexa-decimal
Talk Address Group TAG Note 2	T0	@	64	01 000 000	100	40
	T1	A	65	01 000 001	101	41
	T2	B	66	01 000 010	102	42
	T3	C	67	01 000 011	103	43
	T4	D	68	01 000 100	104	44
	T5	E	69	01 000 101	105	45
	T6	F	70	01 000 110	106	46
	T7	G	71	01 000 111	107	47
	T8	H	72	01 001 000	110	48
	T9	I	73	01 001 001	111	49
	T10	J	74	01 001 010	112	4A
	T11	K	75	01 001 011	113	4B
	T12	L	76	01 001 100	114	4C
	T13	M	77	01 001 101	115	4D
	T14	N	78	01 001 110	116	4E
	T15	O	79	01 001 111	117	4F
	T16	P	80	01 010 000	120	50
	T17	Q	81	01 010 001	121	51
	T18	R	82	01 010 010	122	52
	T19	S	83	01 010 011	123	53
	T20	T	84	01 010 100	124	54
	T21	U	85	01 010 101	125	55
	T22	V	86	01 010 110	126	56
	T23	W	87	01 010 111	127	57
	T24	X	88	01 011 000	130	58
	T25	Y	89	01 011 001	131	59
	T26	Z	90	01 011 010	132	5A
	T27	[	91	01 011 011	133	5B
	T28	\	92	01 011 100	134	5C
	T29	]	93	01 011 101	135	5D
	T30	^	94	01 011 110	136	5E
	UNT	_	95	01 011 111	137	5F
Secondary Command Group SCG Note 3	S0	`	96	01 100 000	140	60
	S1	a	97	01 100 001	141	61
	S2	b	98	01 100 010	142	62
	S3	c	99	01 100 011	143	63
	S4	d	100	01 100 100	144	64
	S5	e	101	01 100 101	145	65
	S6	f	102	01 100 110	146	66
	S7	g	103	01 100 111	147	67
	S8	h	104	01 101 000	150	68
	S9	i	105	01 101 001	151	69
	S10	j	106	01 101 010	152	6A
	S11	k	107	01 101 011	153	6B
	S12	l	108	01 101 100	154	6C
	S13	m	109	01 101 101	155	6D
	S14	n	110	01 101 110	156	6E
	S15	o	111	01 101 111	157	6F
	S16	p	112	01 110 000	160	70
	S17	q	113	01 110 001	161	71
	S18	r	114	01 110 010	162	72
	S19	s	115	01 110 011	163	73
	S20	t	116	01 110 100	164	74
	S21	u	117	01 110 101	165	75
	S22	v	118	01 110 110	166	76
	S23	w	119	01 110 111	167	77
	S24	x	120	01 111 000	170	78
	S25	y	121	01 111 001	171	79
	S26	z	122	01 111 010	172	7A
	S27	{	123	01 111 011	173	7B
	S28	|	124	01 111 100	174	7C
	S29	}	125	01 111 101	175	7D
	S30	~	126	01 111 110	176	7E
	S31	DEL	127	01 111 111	177	7F

ASCII CHARACTERS AVAILABLE ON THE HP-85

HEWLETT PACKARD

STRING IMAGE SPECIFIERS

MAINFRAME

nA - single characters

"test----" - literal text

nK - compacted free-field - output string of unknown length

nX - blank

n/ - end of line sequence

n( ) - entire specifier or group of specifiers repeated n times

n means symbol can be repeated n times

NOTES:

* EOL CONTROL

  I/O ROM

  # suppress end-of-line sequence (default is CR/LF)
  % ignored

* All strings output ASCII characters.

* Overflowing an IMAGE will cause unpredictable results.

FORMAT EXAMPLE

WRITE A PROGRAM TO OUTPUT THIS DATA EXACTLY AS SHOWN

FORMATTING EXAMPLE - HP-85

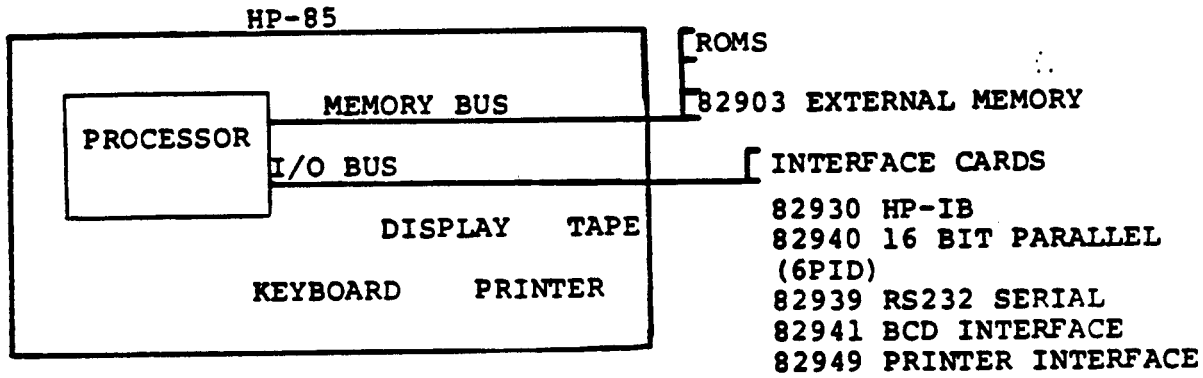
DATA PRINTOUT AT TIME: 981.132

N	N↑2	1/N
1	1	1.000
2	4	0.500
3	9	0.333
4	16	0.250
5	25	0.200
6	36	0.167
7	49	0.143
8	64	0.125
9	81	0.111
10	100	0.100

NOTES:

HEWLETT PACKARD

Select Codes & Interface Cards



SELECT CODES

CRT 1

PRINTER 2

EXTERNAL DEVICES 3-1Ø

NOTES:

more later

PROPER COMMUNICATIONS to/from EXTERNAL DEVICES

Suppose a voltmeter was to be sent the following information:

ASCII output of data, 10v range,
variable # of readings, internal trigger

and it must receive the data in ASCII characters with no spaces in between the characters, and a CR/LF as a terminator.

eg "FIR31ØØST1CRLF" for 1ØØ readings

WHICH ONE OF THESE WOULD SEND THE CORRECT INFORMATION

```
 1Ø N=1ØØ
 2Ø PRINT  "F1R3",N,"ST1"
 3Ø PRINT
 4Ø PRINT "F1R3";N;"ST1"
 5Ø PRINT
 6Ø OUTPUT 2 ; "F1R3";N;"ST1"
 7Ø PRINT
 8Ø OUTPUT 2 USING "K,DDD,K" ; "
    F1R3",N,"ST1"
 9Ø PRINT
1ØØ END
```

NOTES:

Try these on your HP-85

OUTPUT STATEMENT

```
1Ø1Ø PRINTER IS 722 @ PRINT A$   ! PRINT statement does not
....                             ! specifiy destination
....
2Ø1Ø OUTPUT 722;A$               ! OUTPUT statement specifies
....                             ! destination

1Ø1Ø OUTPUT 711;"FIR3N10ST2" ! SET UP DEVICE

....

....

2Ø2Ø IMAGE "F1R3N", DDDD, "STZ" ! PROGRAM DEVICE

2Ø3Ø OUTPUT 722 USING 2020;N

....

5Ø5Ø OUTPUT 701 USING "B,B";13,10 ! ISSUE CR/LF TO PRINTER
```

NOTES: It is a more specific form of the PRINT statement.

It can be used most anywhere that a PRINT would be proper.

It SHOULD be used when talking to external devices (especially when the printer/plotter ROM is not present).

NOTE: A PRINT USING ..... statement may yield an unwanted EOL sequence after every 32 characters even if an IMAGE statement specifies otherwise.

An OUTPUT USING..... statement will correct this.

It always sends ASCII data unless otherwise specified.

A CR/LF is issued after the data list completes unless suppressed.

If free-field format is used (ie no IMAGE) the same field widths as in the PRINT statement are used.

HEWLETT hp PACKARD

ENTER STATEMENT

It is the statement which allows data to be entered via the interfaces.

It is more involved than the OUTPUT statement because it must not only receive incoming data - it must also be able to put it into various destination variables or turn it into numbers.

```
110  ENTER 709;A$,A,V(22)  ENTER using freefield format

...

250  ENTER 709 USING "5A,K,5D.2D";A$,A,V(22) Formatted ENTER

...

300  IMAGE 5A,K,5D,2D

310  ENTER 709 USING 300;A$,A,V(22)
```

NOTES:

HEWLETT PACKARD

ENTER IMAGE Specifiers for STRINGS

SYMBOL			PURPOSE
n	A	-	enter single character
n	K	-	enter string of unknown length

ENTER statement not complete until line-feed (LF) is seen, even if more characters are received than the string is dimensioned for.

n - means symbol can be replicated

NOTES: EXAMPLES:

```
DIM A$ [100]
ENTER 709 USING "K"; A$

      * will not terminate until a Lf is seen.
      * if more than 100 characters sent 1st 100 are retained.

DIMX$[100], Y$ [100]
ENTER 709 USING "K,K";X$,Y$

      *LF needed to terminate both X$,Y$
      * if X$ overflowed, Y$ starts to fill.

ENTER 709 USING "5A";A$

      * take the 1st 5 characters - throw away any additional
        characters
      * terminate on LF
```

HEWLETT hp PACKARD

ENTER IMAGE Specifiers for Numbers

The number builder is not concerned as to which specifiers are used. It is ONLY concerned with the TOTAL NUMBER OF CHARACTERS IN THE NUMERIC FIELD.

SYMBOL			NUMBER OF CHARACTERS CONTRIBUTED TO NUMERIC FIELD
n	D	digit field (blanks)	1
n	Z	digit field (zero's)	1
n	*	digit field (asterisks)	-1
n	.	decimal point	1
	S	sign field (t or -)	1
	M	sign field (blank or -)	1
	( )	replicating fields	As Applicable
	E	exponent field ($E^{\pm}499$)	5
	e	exponent field ($E^{\pm}99$)	4
	C	causes ENTER to ignore Commas	1
	K	free-field	
R,P,		not allowed	

NOTES:

Number builder terminates on seeing non numeric character.

ie any other ASCII character than;

+ - . 1 2 3 4 5 6 7 8 9 Ø E e

```
ENTER 709 USING "3DE";X
                "4De";X
                "S2D.3DC";X
                "8*";
```

All would interpret 123.4567 the same!

HEWLETT PACKARD

ENTER IMAGE FOR NUMBERS - QUESTIONS

?? How would 123.45, 123.456789, 234,567 be interpreted by this statement

```
ENTER 709 USING "8D";X  ??
```

?? Given the string "123,456,789" and

```
ENTER 709 USING "8D,4D";X,Y
ENTER 709 USING "7DC,4D";X,Y
```

What will X and Y equal in both cases ??

?? Given the string 1234ABCDE and

```
ENTER 709 USING "K,K";X,Y$
```

what is X, Y$??

NOTES:

HEWLETT PACKARD

ENTER IMAGE SPECIFIERS - BINARY

Unless these format specifiers are used input data is interpreted as ASCII characters.

SYMBOL	MEANING
B	Interpret 1 byte as binary data
W	Interpret 2 bytes as binary data (if an 8 bit interface is used (eg 82937A HP-IB); the HP-85 assumes the most significant byte is sent first

NOTES:

Enter 6 using "#,W";A

(Example of entering data from 6940B multiprogrammer on 82940A GPIO interface)

HEWLETT PACKARD

ENTER IMAGE SPECIFIERS - TERMINATORS

SYMBOL	MEANING
#	Don't hunt for anything
	Hunt for a line-feed (LF)
%	Hunt for EOI or Line feed (LF)
#% or %#	Hunt for EOI (end or identify line on IEEE-488 interface.

NOTES:

```
ENTER 709 Using "5A";A$          *Throw away anything over 5 characters
                                 *Terminate on line-feed

ENTER 709 using "#,5A";A$        *Terminate after 5 characters

ENTER 709 using "%,5A";A$        *Throw away anything over 5 characters
                                 *Terminate on LF or EOI

ENTER 709 using "#%,5A";A$       *Throw away anything over 5 characters
                                 *Terminate on EOI
```

HEWLETT hp PACKARD

ENTER IMAGE SPECIFIERS - SKIP CONTROL

SYMBOL	MEANING
/	Look for a line-feed (LF) before moving on ie, throw away characters until a line-feed is found, then proceed to next field. eg ENTER 709 USING"D,1,D";X,Y
n X	Skip a character eg ENTER 709 USING "2D,X,2D,X,2D";X,Y,Z

NOTES:

?? How would the latter IMAGE interpret
12345678LF

X=
Y=
Z= ??

HEWLETT hp PACKARD

Write the IMAGE for the following;

x = space

---

xx46.230xxmsecCRLF

OP,7,100.000,T CRLF

variable

MI,23.456GZ CRLF

variable

WF,8.0,2500,T CRLF

variables

send the binary equivalent of the octal number 170240

send the binary equivalent of decimal 12 (page eject)

incoming data; +138.5000E-6,+0220.900E-6,+0276.000E-6

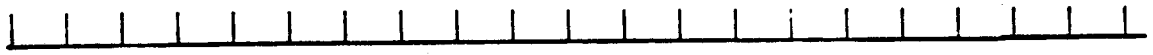
NOTES:

HEWLETT PACKARD

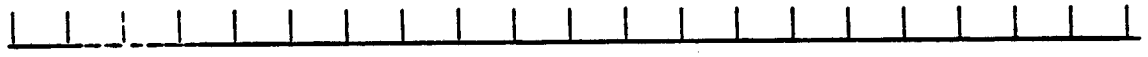
DIAGRAM THE OUTPUT FOR THE FOLLOWING WRITES

```
OUTPUT 2 USING "4D";14
```

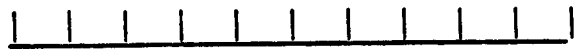
```
OUTPUT 701 USING "2D.D";-6
```



```
500 IMAGE "column 1",2D.2D,"mg"
 10 OUTPUT 706 USING 500;12.1
```



```
OUTPUT 2 USING"B";65
```



NOTES:

Summary of OUTPUT Image Specifiers

Image	Meaning
A	Output one string character
B	Output number as one 8-bit byte
C	Output a comma separator in a number
D	Output one digit character; blank for leading zero
E	Output exponent information; five characters
e	Output exponent information; four characters
K	Output a variable in free-field format
M	Output number's sign if negative, blank if positive
P	Output a period separator in a number
R	Output a European radix point (comma)
S	Output number's sign, plus or minus
W	Output number as two 8-bit bytes (16-bit word)
X	Output one blank
Z	Output one digit character, including leading zeros
"..."	Output a literal
#	Suppress end-of-line sequence at end of statement
*	Output one digit character; asterick for leading zero
.	Output an American radix point (decimal point)
/	Output an end-of-line sequence

Summary of ENTER Image Specifiers

Image	Meaning
A	Demands one string character
B	Enter number as one 8-bit byte
C	Demand one character for a numeric field; allows commas to be skipped over
D	Demand one character for a numeric field
E	Demand five characters for a numeric field
e	Demand four characters for a numeric field
K	Enter a variable in free-field format
M	Demand one character for a numeric field
S	Demand one character for a numeric field
W	Enter number as two 8-bit bytes (16-bit word)
X	Skip one character
Z	Demand one character for a numeric field
#	Suppress requirement for a line-feed to terminate statement or field
%	Allow EOI to terminate statement or field
*	Demand one character for a numeric field
.	Demand one character for a numeric field
/	Demand a line-feed

HEWLETT PACKARD

HP-IB TOPICS

I Overview of what it is

II Overview of how it works

III How to use it with the HP-85

- A) Manual Operation
- B) Device Addresses
- C) Bus Messages - Instrument Capability
- D) Program Codes of Instrument
- E) Device setup and measurement data return
- F) Control statements

IV HP-IB Advanced topics - Reference Book

NOTES:

HEWLETT PACKARD

FORMATTING LAB

6

Write a program that reads in 60 voltage readings that are stored on the instructor's data cartridge. Your program must ask the name of the data file that it is to load.

After loading print out the data exactly as follows;

TEST 1 RESULTS

DATE: 145 TIME: 2403.78

SCAN	CHANNEL 1	CHANNEL 2	CHANNEL 3	CHANNEL 4	CHANNEL 5	CHANNEL 6
1	value	value	value	value	value	value
2	"	"	"	"	"	"
3	"	"	"	"	"	"
4	"	"	"	"	"	"
5	"	"	"	"	"	"
10	"	"	"	"	"	"

NOTES: EXTRA CREDIT:

A) REDUCE AND DISPLAY THE DATA AGAIN REDUCED IN SOME WAY. (temperature in deg F)

LAB 6 SOLUTION

```
10 OPTION BASE 1
20 DIM A$[100],V(60),V1(10,6)
30 DISP "ENTER FILE NAME" @ INPUT A$
40 ASSIGN@ 1 TO A$[1,6]
50 READ@ 1 ; V1(,)
60 A$[1,15]="TEST 1 RESULTS"
70 C=DATE @ D=TIME
80 IMAGE 23X,K,2/,18X,"DATE",4D,2X,"TIME: ",5D.3D,2/,%
90 PRINTER IS 706
100 OUTPUT 706 USING 80 ; A$[1,15],C,D
110 OUTPUT 706 USING "@,6X" ;
120 IMAGE @,"CHANNEL ",D,X
130 FOR I=1 TO 6
140 OUTPUT 706 USING 120 ; I
150 NEXT I
160 OUTPUT 706
170 OUTPUT 706 USING "4A,/,%" ; "SCAN"
180 FOR R=1 TO 10
190 OUTPUT 706 USING "@4D" ; R
200 FOR C=1 TO 6
210 OUTPUT 706 USING "@,3D.6D" ; V1(R,C)
220 NEXT C
230 OUTPUT 706
240 NEXT R
```

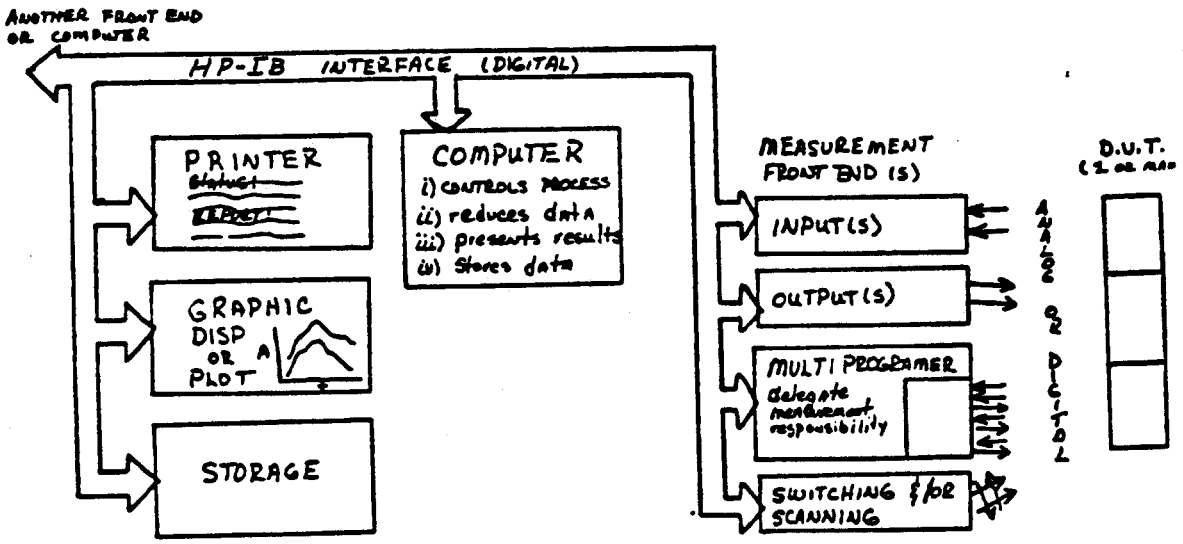
TEST 1 RESULTS

DATE 0 TIME: 4306.642

SCAN	CHANNEL 1	CHANNEL 2	CHANNEL 3	CHANNEL 4	CHANNEL 5	CHANNEL 6
1	.000513	.000513	.000514	.000515	.000515	.000515
2	.000517	.000520	.000522	.000525	.000525	.000522
3	.000519	.000517	.000515	.000504	.000486	.000470
4	.000458	.000441	.000427	.000417	.000407	.000395
5	.000382	-.000238	-.000675	-.000799	-.000873	-.000896
6	-.000965	-.000986	-.001000	-.001020	-.001036	-.001045
7	-.001049	-.001050	-.001051	-.001053	-.001035	-.001013
8	-.000990	-.000963	-.000936	-.000910	-.000886	-.000863
9	-.000844	-.000826	-.000810	-.000791	-.000771	-.000754
10	-.000738	-.000724	-.000712	-.000672	-.000374	-.000165

HEWLETT hp PACKARD

HP-IB OVERVIEW



NOTES:

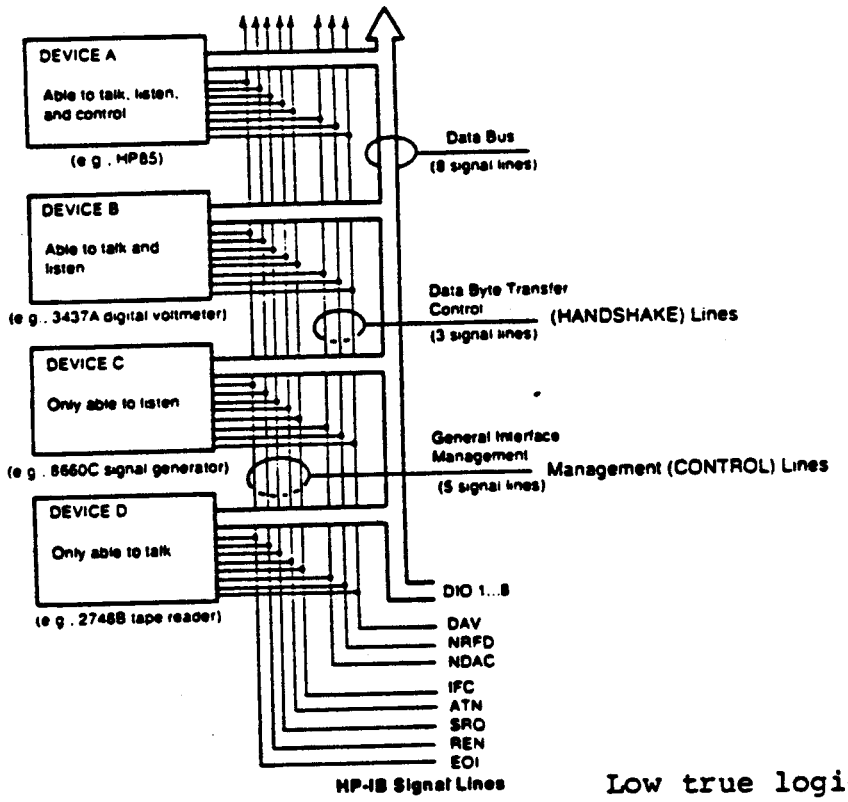
*It is the primary means that Hewlett Packard uses to connect instruments to each other and to a system controller (usually a computer).

*HP-IB refers to HP's implementation of the IEEE-488/ANSI MC1.1 standards. HP-IB (Hewlett Packard Interface Bus) is totally consistent with these standards.

*HP-IB also refers to enhancements in our computers and instrumentation that eases the utilization of this interface bus in automatic test systems. It deals with conventions used by HP equipment to ease computer-to-instrument communications.

HEWLETT hp PACKARD

HP-IB BUS CONCEPT



Low true logic applies

NOTES:

Every HP-IB device must be capable of performing one or more of the following interface functions (roles):

a. LISTENER - A device capable of receiving data from other devices over the interface when addressed. Examples of this type of devices are: printers, display devices, programmable power supplies, programmable signal sources and the like. There can be up to 14 active listeners simultaneously on the interface.

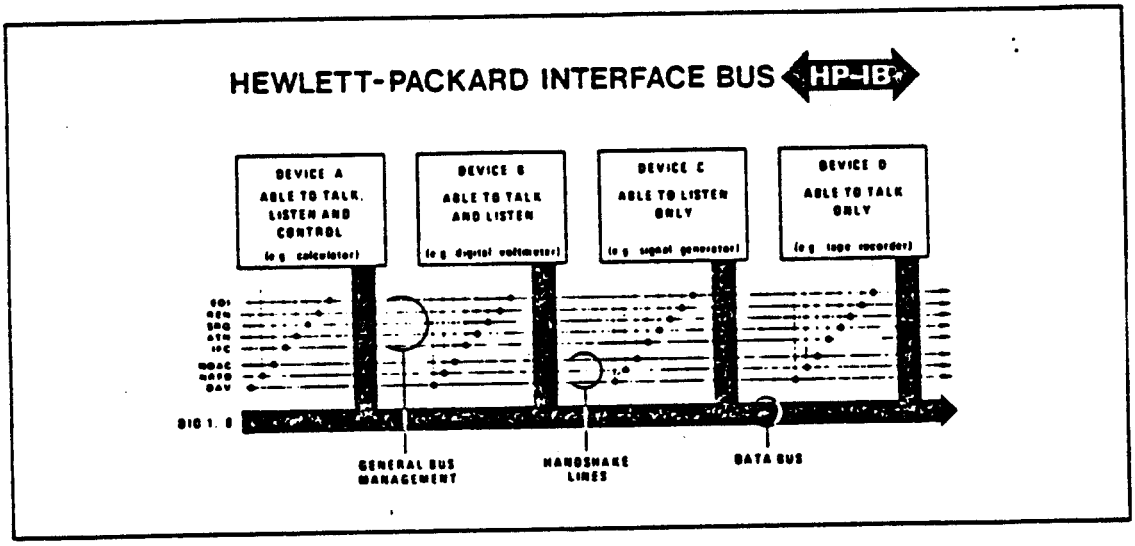
b. TALKER - A device capable of transmitting data (but not commands) to other devices over the interface when addressed. Examples of this type of devices are: voltmeters that are outputting data, counters that are outputting data, and so on. There can be only one active talker on the interface at a time.

c. CONTROLLER - A device capable of this includes specifying the talker and listeners for an information transfer (including itself). A computer with an appropriage I/O card is an example of this type of device. There can be only one active controller on the interface at a time. In multiple controller systems only one can be a SYSTEM CONTROLLER (MASTER).

d. SYSTEM CONTROLLER - This is an instrument on the bus which has all the features of a standard controller with the added ability to control the IFC and REN lines. The system controller will take control of the bus when power is turned on or when it determines that something has gone wrong with normal bus operations. The system controller can pass control to other controllers but always retains the system controller status.

HEWLETT hp PACKARD

OVERVIEW OF THE CONTROL LINES

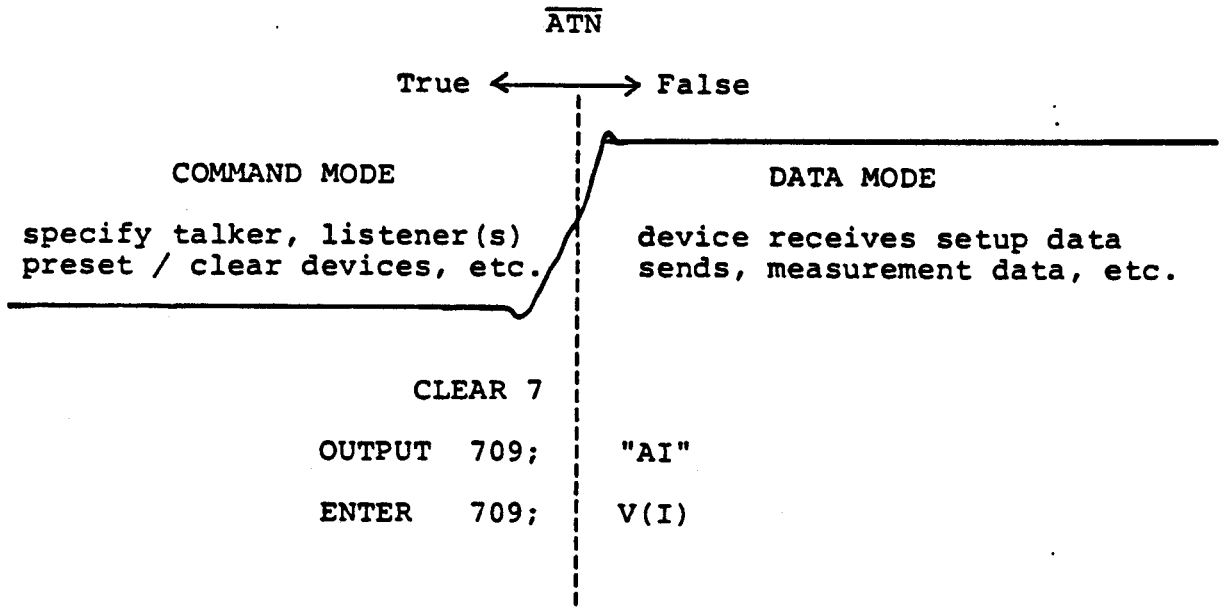


Structure of the HP-IB

NOTES:

ATN (Attention)	When true (low interface is in the command mode, and devices receive data which tells them whether or not they are to do something. When false interface is in the data mode and device dependent data (eg setup or measurement data) is passed between the talker and listeners.
IFC (Interface Clear)	Terminates all bus activity.
SRQ (Service Request)	Used to tell the controller that a device needs attention.
REN (Remote Enable)	Enables devices to respond to remote program (computer) control when addressed. If REN false devices return to local ("front panel") operation.
EOI (End or Identify)	Indicates last byte of data in multibyte sequence.

COMMAND MODE VS. DATA MODE



NOTES:

$\overline{\text{REN}}$ MUST ALSO BE TRUE

HEWLETT hp PACKARD

HANDSHAKE LINES OVERVIEW

* These lines coordinate the transfer of data over the data bus from the talker or controller to one or more receiving devices.
* A 3-wire handshake guarantees data transfer integrity among devices operating at different transfer rates. The data transfer runs at the rate of the slowest active device.
* Every byte transferred undergoes the handshake.
* HP-IB signal lines use a low-true logic convention to implement the wired or convention of the NRFD and NDAC lines, provide active true-state assertion, and reduce noise susceptibility in the true state.
* Patented by Hewlett Packard.

NOTES:

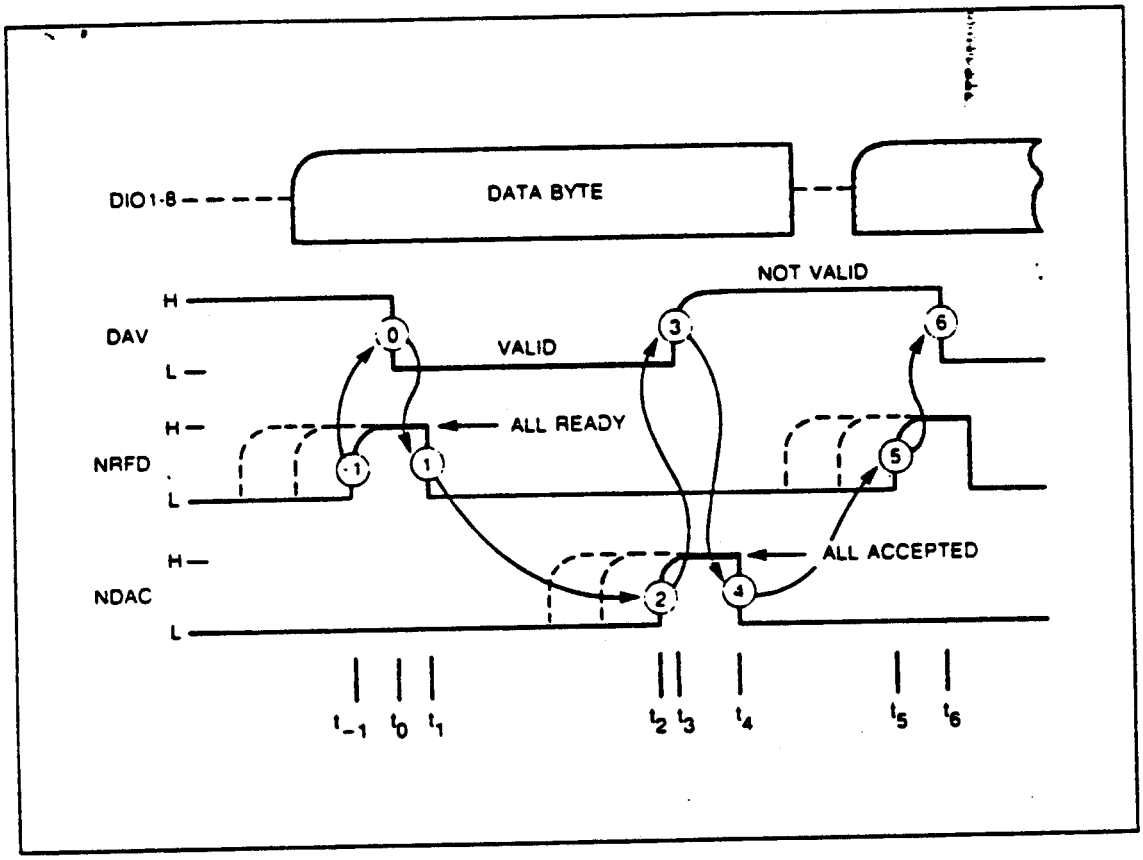
The three handshake lines are:

DAV - Data Valid. This line is controlled by the source (active talker of controller). When true (low) it indicates that data is stable on the DIO lines and auailable to be accepted by the receiver.

NRFD - Not Ready For Data. This line is controlled by the acceptors (active listeners) or all devices receiving interface commands. When false (high) H indicates to the source that the device is ready to receive data.

NDAC - Not Data Accepted. This line is controlled by the acceptors (active listeners) or all devices receiving interface commands. When set false (high) it indicates to the source that data has been accepted. It does NOT mean that the data was acted upon by the acceptor - which is determined by the acceptor's internal logic.

THE HANDSHAKE TIMING SEQUENCE



Data byte transfer

NOTES:

Preliminary: Source checks for listeners and places data byte on data lines.

t_{-1}: All acceptors become ready for byte. NRFD goes high with slowest one.

t_0: Source validates data (DAV low)

t_1: First acceptor sets NRFD low to indicate it is no longer ready for a new byte.

t_2: NDAC goes high with slowest acceptor to indicate all have accepted the data.

t_3: DAV goes high to indicate this data byte is no longer valid.

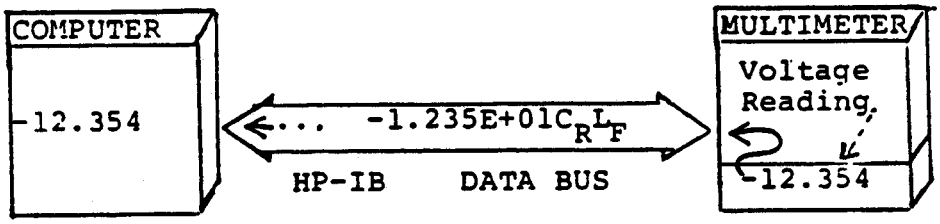
t_4: First acceptor sets NDAC low in preparation for next cycle.

t_5: Back to t_{-1} again.

DATA BUS OVERVIEW

8 data lines - transfer both command and data information.

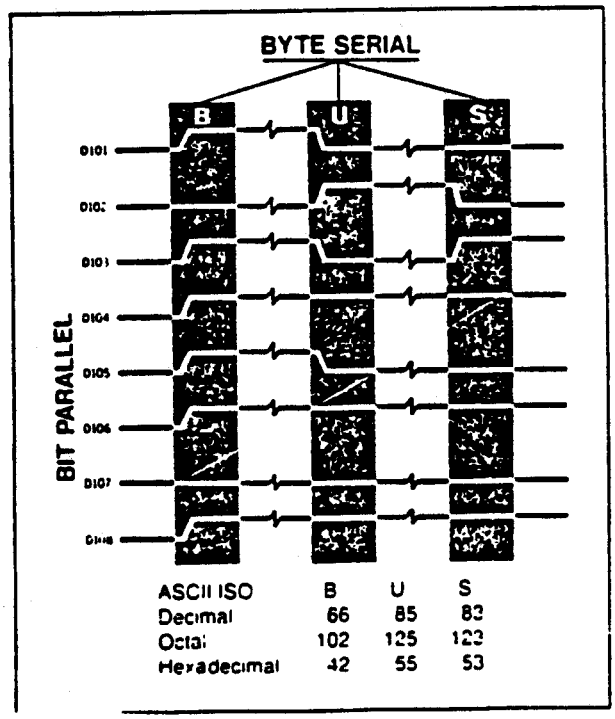
HP devices typically send and receive ASCII characters.



?? Is this data transfer in the command mode or data mode ??

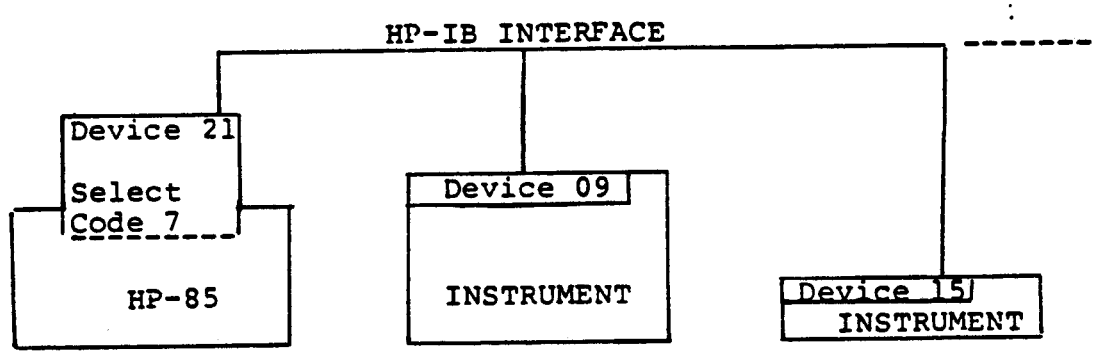
NOTES:

The transfer of the 3 byte sequence "BUS" would occur as shown here over the Data Lines. Hence the BIT PARALLEL . . . BYTE SERIAL description.



HEWLETT PACKARD

SELECT CODES AND DEVICE ADDRESSES



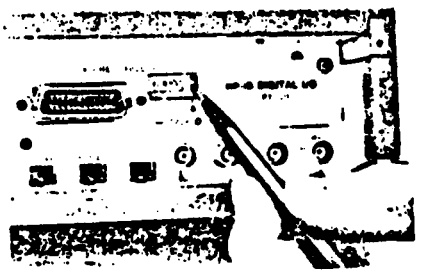
NOTES:

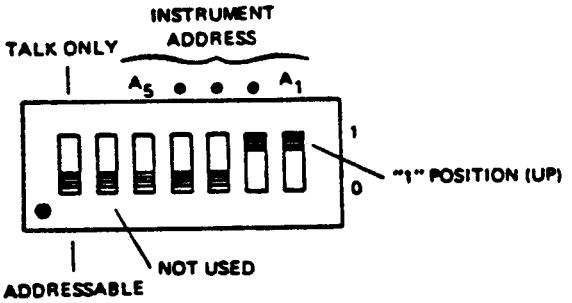
The system device number (SYSTEM ADDRESS) is the combination of the 82937A select code (factory setting of 7) and the particular device address. Thus a device having a device address of 9 has a system address of 709.

HEWLETT PACKARD

SETTING DEVICE ADDRESSES

* Valid Address Ranges are ØØ thru 31
* The HP-85's 82937A interface card is factory set to select code 7 and device address 21. Do Not Use!
* The address switches are usually on the rear of the instrument. Set according to the table below.
* Put a device # table on the instrument front panel.





NOTE: THE TALK ONLY SWITCH SHOULD BE CHANGED ONLY WHEN THE INSTRUMENT IS OFF.

NOTES:

ASCII CODE CHARACTER		ADDRESS SWITCHES					DECIMAL EQUIVALENT OF BINARY SWITCH SETTING
LISTEN	TALK	A_1	A_2	A_3	A_4	A_5	
SP	@	0	0	0	0	0	00
!	A	0	0	0	0	1	01
"	B	0	0	0	1	0	02
#	C	0	0	0	1	1	03
$	D	0	0	1	0	0	04
%	E	0	0	1	0	1	05
&	F	0	0	1	1	0	06
'	G	0	0	1	1	1	07
(	H	0	1	0	0	0	08
)	I	0	1	0	0	1	09
*	J	0	1	0	1	0	10
+	K	0	1	0	1	1	11
,	L	0	1	1	0	0	12
-	M	0	1	1	0	1	13
.	N	0	1	1	1	0	14
/	O	0	1	1	1	1	15
0	P	1	0	0	0	0	16
1	Q	1	0	0	0	1	17
2	R	1	0	0	1	0	18
3	S	1	0	0	1	1	19
4	T	1	0	1	0	0	20
5	U	1	0	1	0	1	21 ---------- factory setting of HP-85
6	V	1	0	1	1	0	22
7	W	1	0	1	1	1	23
8	X	1	1	0	0	0	24
9	Y	1	1	0	0	1	25
:	Z	1	1	0	1	0	26
;	[	1	1	0	1	1	27
<	\	1	1	1	0	0	28
=	]	1	1	1	0	1	29
>	^	1	1	1	1	0	30

HEWLETT hp PACKARD

DISTINGUISHING BETWEEN TALK AND LISTEN ADDRESSES
THEIR UTILIZATION IN THE COMMAND MODE

The 5 address switch settings are pertain to the 5 least significant bits of the data bus when in the command mode. They work in conjunction with the 3 most significant bits of the data bus (which are controlled automatically by the I/O ROM) to determine who is to receive or send information.

The talk and listen addresses are distinguished by the setting of bits 5 and 6 as indicated above.

ie Device Address 09 breaks down into

TALK SYMBOL I = X1001001

LISTEN SYMBOL) = X0101001

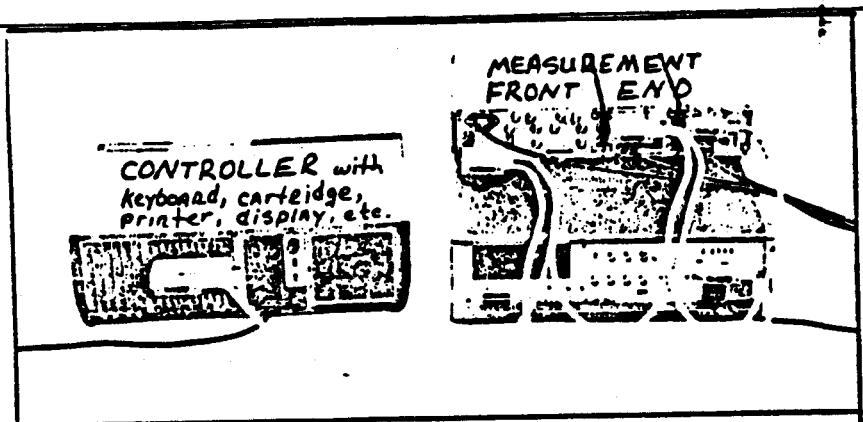
└ Set automatically via high level commands such as OUTPUT or ENTER.

NOTES:

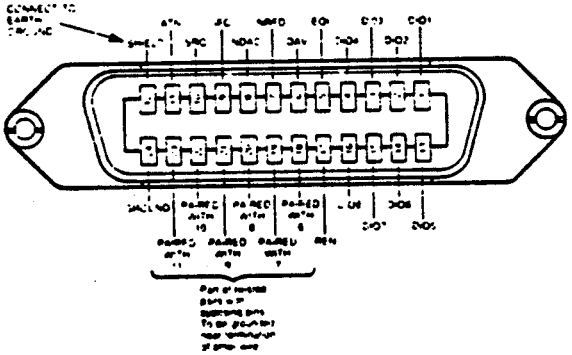
COMMAND MODE PARAMETERS (PARTIAL)	DATA BUS 7654321Ø
BUS COMMAND	X00CCCCC
TALK ADDRESS	X10TTTTT
EXTENDED TA*	X10SSSSS
LISTEN ADDRESS	X01LLLLL
2ND LISTEN ADR*	X01LLLLL
EXAMPLE	
ALL UNLISTEN	X0011111
DEVICE 23 TALKS	X1010111
2NDARY ADDRESS 1Ø	X0110101
DEVICE 01 LISTEN	X0100001

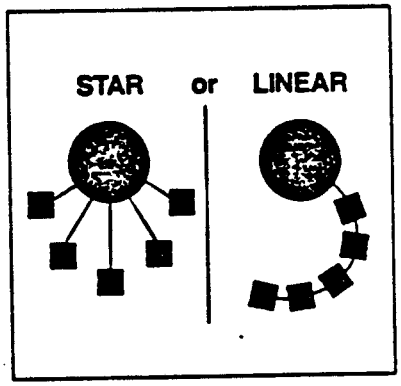
HEWLETT hp PACKARD

MECHANICAL ASPECTS



Rear view of 5-device HP-IB bench system. Note single and stacked connections.





Cabling arrangements

NOTES:

* 15 devices per bus
* An overall cabling restriction of 20 meters TOTAL or 2 meters per device, the lesser of the two applies. The length between adjacent devices is not critical as long as the overall restriction is met.
* The STAR cabling configuration will minimize worst-case transmission path lengths but can lump large capacitance values at a single plane on the line. The LINEAR cabling configuration may produce longer electrical lengths but provides more control to distribute capacitive line loads for maximum error-free transmission.

?? What is the maximum cable length for a system consisting of 1 controller and 2 instruments ??

HP-IB PROGRAMMING STEPS

A) Understand the measurement to be made.

B) Understand how to make the measurement manually.

C) Set or check HP-IB select code and device addresses.

D) Learn function codes necessary to exercise each system device (eg instrument set-ups)

E) Learn proper HP-85 commands to transmit and receive information (i.e. OUTPUT, ENTER, CLEAR, etc.)

F) Hookup computer to instrument and test communications via simple bus commands.

   i) remote check

   ii) instrument set-ups

   iii) entering data into computer

G) Create your program on paper

   i) flowchart or pseudocode (logical English)

   ii) subdivide into segments or blocks (for easier reading and testing)

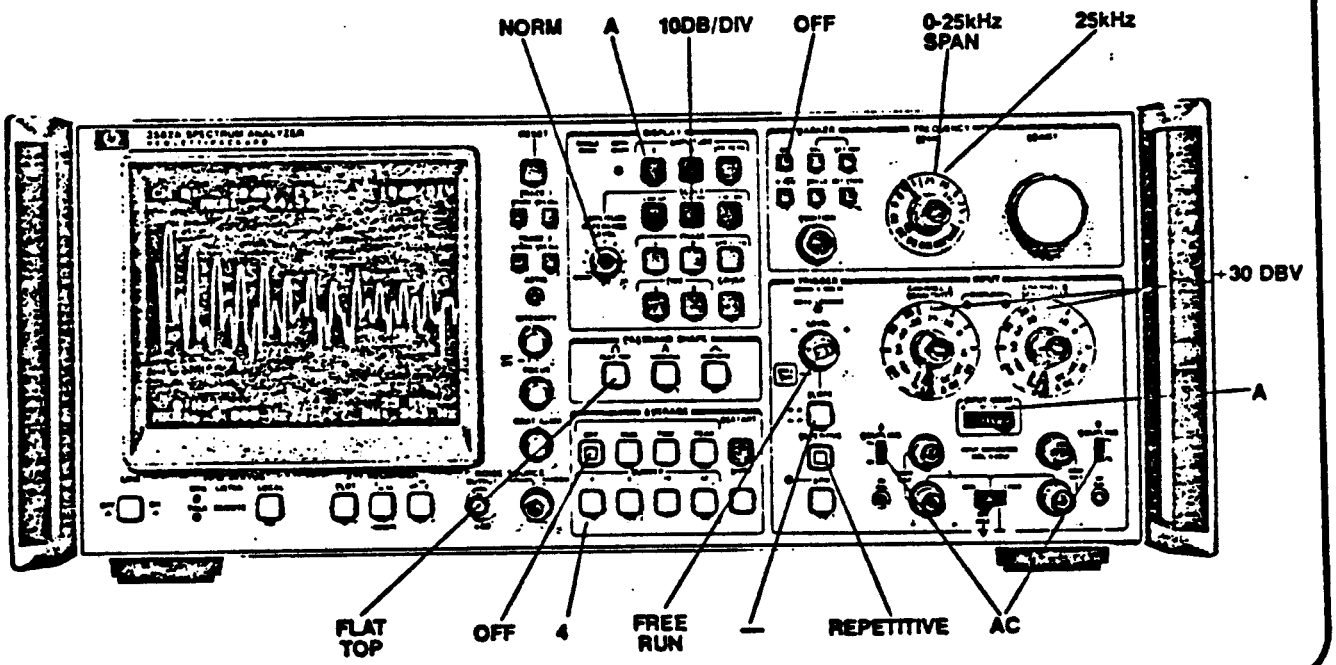
H) Enter program into computer

   De-bug program segments

   i) live keyboard to check variables

   ii) stop or pause statements or program stepping as de-bus tools

HP-IB PROGRAMMING STEP: KNOW MANUAL OPERATION



NOTES:

HP - IB PROGRAMMING STEP

CHECKING INTERFACE CAPABILITIES OF A DEVICE

Interface functions are pre-defined capabilities which COULD BE designed into an HP-IB device. The total available set is shown below. A designer is free to choose which are implemented in a given device.

Check the device you are using to see what capabilities it has.

Available Interface Functions

Interface Functions that may be included in an HP-IB device	Mnemonic	Comments
Talker or Extended Talker	T,TE	Capability required for a device to be a "talker".
Listener or Extended Listener	L,LE	Capability required for a device to be a "listener".
Source Handshake	SH	This provides a device with the capability to properly transfer a multiline message.
Acceptor Handshake	AH	This provides a device with the capability to guarantee proper reception of remote multiline messages.
Remote/Local	RL	Provides capability to select between two sources of input information. Local corresponds to front panel controls and remote to the input information from the bus.
Service Request	SR	This capability permits a device to asynchronously request service from the controller.
Parallel Poll	PP	Provides capability for a device to uniquely identify itself if it requires service and the controller is requesting a response. This capability differs from service request in that it requires a commitment of the controller to periodically conduct a parallel poll.
Device Clear	DC	This function allows a device to be initialized to a pre-defined state. A device with this capability will have the effect of this command described in its operating manual.
Device Trigger	DT	This function permits a device to have its basic operation initiated by the talker on the Bus.
Controller	C	This function permits a device to send addresses, universal commands and addressed commands to other devices on the HP-IB. It may also include the ability to conduct polling to determine devices requiring service.
Drivers	E	This code describes the type of electrical drivers used in a device.

HP - IB PROGRAMMING STEP

CHECKING BUS COMMANDS

The interface functions are performed using bus commands.

An HP-85 program statement (discussed later) causes the $\overline{REN}$ and $\overline{ATN}$ lines to go true and puts data on the data bus. This table show what action will be performed for specific data values being present on the the data bus.

SUMMARY OF BUS COMMANDS THAT MOST INSTRUMENTS WILL RECOGNIZE

	COMMAND	GPIB CODE	OCTAL CODE	PURPOSE	ASCII CHAR	DECIMAL
UNADDRESS COMMANDS	UNLISTEN	UNL	077	Clears Bus of all listeners.	?	063
	UNTALK	UNT	137	Unaddresses the current talker so that no talker remains on the Bus.*	-	137
UNIVERSAL COMMANDS	Local Lockout	LLO	021	Disables front panel local-reset button on responding devices.	DC1	017
	Device Clear	DCL	024	Returns all devices capable of responding to pre-determined states, regardless of whether they are addressed or not.	DC4	020
	Parallel Poll Unconfigure	PPU	025	Sets all devices on the HP-IB with Parallel Poll capability to a predefined condition.	NAK	021
	Serial Poll Enable	SPE	030	Enables Serial Poll Mode on the Bus.	CAN	024
	Serial Poll Disable	SPD	031	Disables Serial Poll Mode on the Bus.	EM	031
ADDRESSED COMMANDS	Selective Device Clear	SDC	004	Returns addressed devices, capable of responding to pre-determined states.	EOT	004
	Go to Local	GTL	001	Returns responding devices to local control.	SOH	001
	Group Execute Trigger	GET	010	Initiates a simultaneous pre-programmed action by responding devices.	BS	008
	Parallel Poll Configure	PPC	005	This command permits the DIO lines to be assigned to instruments on the Bus for the purpose of responding to a parallel poll.	ENQ	005
	Take Control	TCT	011	This command is given when the active controller on the Bus transfers control to another instrument.	HT	009

NOTE. Talkers can also be unaddressed by transmitting an unused talk address on the Bus. This is not true with listeners.

HEWLETT hp PACKARD

HP-IB PROGRAMMING STEP: LEARN DEVICE SYNTAX CONDITIONS

The answers to these questions should be understood;

* How does it receive numbers?
* How does it receive ASCII characters?
  * Are small letters interpreted the same as capital letters? ie does upper case = lower case?
* How are commands separated?

  ASCII blanks, commas, semicolons, CR, LF, etc.
* Will ASCII blanks have an effect on command interpretation?
* What are the allowable command terminators?
* How does the device send its data?
* What is it's output terminator(s)?

NOTES:

HP-IB PROGRAMMING STEP: KNOW PROGRAM CODES

These are the codes (usually ASCII characters) that cause the device to perform it's various functions.

NOTES:

HEWLETT hp PACKARD

EXAMINING THE INTERFACE FUNCTIONS , BUS COMMANDS , AND PROGRAMMING CODES OF VARIOUS HEWLETT PACKARD INSTRUMENTS

NOTES:

HEWLETT hp PACKARD

5335 COUNTER INTERFACE CAPABILITIES
(Reference: 5335 Manual)

The capability of a device connected to the bus is specified by its interface functions. The following table lists the 5335A Interface using the terminology of the IEEE 488-1978 standard. These features are also listed below the rear panel HP-IB connector, as follows:

SH1, AH1, T1, TEØ, L2, LEØ, SR1, RL1, PPØ, DC1, DT1, CØ

INTERFACE FUNCTION SUBSET IDENTIFIER	INTERFACE FUNCTION DESCRIPTION
SH1	Complete source handshake capability.
AH1	Complete acceptor handshake capability.
T1	Talker (basic talker, serial poll, talk only mode)
TE0	No extended talker capability.
L2	Listener (basic listener, no listen only mode, does not unaddress to listen if addressed to talk).
LE0	No extended listener capability.
SR1	Service request capability.
RL1	Complete remote/local capability.
PP0	No parallel poll capability.
DC1	Device clear capability.
DT1	Device trigger capability.
C0	No controller capability.

NOTES:

LISTEN: When addressed as a Listener, the instrument can accept any number of commands from a controller on the bus. These commands will usually be used to program the instrument operation.

SERVICE REQUEST: SRQ can be sent out to the bus at the end of measurements and on error or failure messages. Normally SRQ is inhibited, but certain commands will enable this feature. See "WA" and "SR".

REMOTE/LOCAL: Normally the 5335A is under local control. In order to program the instrument it must be in Remote. Once in Remote, all programmable controls are in remote and cannot be affected by manual command. The RESET key may be used to manually return to local control only if Local Lockout is OFF. If Local Lockout is ON, the RESET key is ignored.

PARALLEL POLL: No parallel poll capability in the 5335A.

DEVICE CLEAR: When a universal or selected device clear is received, the instrument clears out all input buffers and resets the hardware for a new measurement. The display will flash momentarily. SRQ is also cleared. Device clear can be used to clear an ERROR message.

DEVICE TRIGGER: When a device trigger is received, a new measurement is started.

CONTROLLER: No controller capability in the 5335A.

BUS MESSAGE IMPLEMENTATION OF THE 5316A COUNTER

(REFERENCE 5316A MANUAL)

Table 3-3. Bus Messages

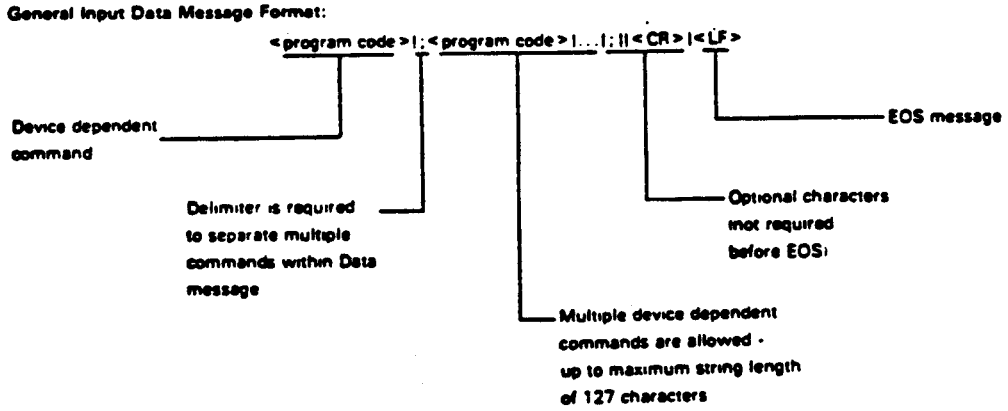
HP-IB MESSAGE	DESCRIPTION/RESPONSE	SAMPLE 9825A (address = 20)	SAMPLE 9835A/45A (address = 20)
DATA	MEANS TO SEND COMMANDS TO 5316A AND RECEIVE MEASUREMENT DATA.	wrt 720, "FN1" red 720, A	OUTPUT 7,20; "FN1" ENTER 7,20; A
TRIGGER	STARTS NEW MEASUREMENT. IF 5316A IS IN LOCAL, IT WILL REMAIN IN LOCAL AND NO TRIGGER OCCURS.	trg 7	TRIGGER 7
	STARTS NEW MEASUREMENT. IF THE 5316A IS IN LOCAL THE 5316A WILL GO INTO REMOTE.	trg 720	TRIGGER 7,20
CLEAR	STARTS NEW MEASUREMENT (ACTS AS RESET).	clr 7 clr 720	CLEAR 7 CLEAR 7,20
REMOTE	FRONT PANEL FUNCTION AND SLOPE SWITCHES ARE DISABLED; COUNTER DEFAULTS TO FREQUENCY A, ALL SLOPES TO POSITIVE UNLESS PREVIOUSLY PROGRAMMED.	rem 7 rem 720	REMOTE 7 REMOTE 7,20
LOCAL	RETURNS TO LOCAL (FRONT PANEL) OPERATION.	lcl 7 lcl 720	LOCAL 7 LOCAL 7,20
LOCAL LOCKOUT	DISABLES FRONT PANEL RESET; ONLY CONTROLLER CAN RETURN 5316A TO LOCAL. NOTE: IF IN REMOTE, FRONT PANEL FUNCTION AND SLOPE SWITCHES ARE ALSO DISABLED.	llo 7	LOCAL LOCKOUT 7
GOTO LOCAL AND CLEAR LOCAL LOCKOUT	5316A RETURNS TO LOCAL (FRONT PANEL) CONTROL; LOCAL LOCKOUT CLEARED.	lcl 7	LOCAL 7
SERVICE REQUEST	5316A WILL REQUEST SERVICE AT END OF MEASUREMENT IF SRQ AND WAIT STATE ENABLED.	rds (720) DEVICE STATUS	STATUS 7,20
STATUS BYTE	PRESENTS STATUS INFORMATION. BIT 7 IS SET IF SERVICE IS REQUESTED.	rds (7) BUS STATUS	STATUS 7
STATUS BIT	NOT APPLICABLE.		
PASS CONTROL	NOT APPLICABLE.		
ABORT	TERMINATES THE BUS COMMUNICATIONS; TELLS ALL DEVICES TO UNLISTEN; 5316A ADRSD LIGHT WILL GO OFF.	cli 7	ABORTIO 7

LEARNING SYNTAX CONDITIONS FOR THE 1980 OSCILLOSCOPE

(REFERENCE: 1980 QUICK REFERENCE GUIDE)

DATA SENT TO 1980

General Input Data Message Format:



Format rules:

code

The 1980A/B sends and receives all Data messages except the Learn String in standard ASCII code. The Learn String consists of 80, eight bit bytes of binary data.

terminators

A linefeed (<LF>) is used as the End-of-String (EOS) message for all Data message transfers except when the 1980 outputs the Learn String. The 1980A/B sets the EOI bus control line true to signal EOS during the 80th byte of the Learn String.

Except during Learn String transfers, the carriage return character (<CR>) is not required before <LF>. Preceding <LF>, <CR> is treated as "no operation" and may be repeated as many times as permitted by the maximum string length limitation.

delimiters

When several program codes are sent in a Data message, a semicolon (;) must be used to delimit each program code except the last one in the string.

Program codes consist of a two-character identifier (prefix) and a parameter field containing zero, one or several parameters.

etc.

Multiple parameters within a program code are delimited by a comma (,).

Unsigned parameters are interpreted as positive values.

In integer parameters, leading zeroes may be omitted.

The character "E" or "e" is used to delimit the mantissa of exponential parameters.

Exponential parameters may be entered in scientific or engineering notation.

In Data messages, spaces (<SP>) are permitted only following program code identifiers and parameter delimiters.

The maximum Data message length is 127 characters including: <CR>, <LF>, <SP>, comma and semicolon.

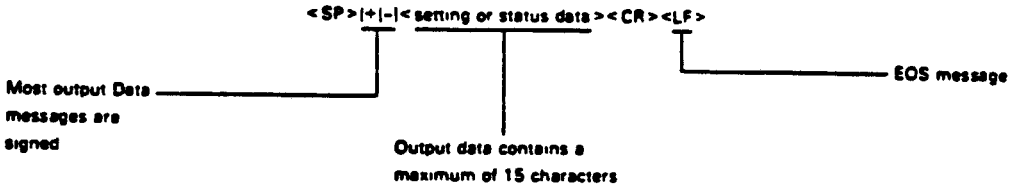
The instrument cannot be unaddressed during input or output Data message transfers. If the instrument is unaddressed and then readdressed, the data transfer is aborted and a syntax error is reported.

Errors in Data message syntax are trapped and can be reported via the HP-IB. Table 2 lists the syntax errors detected and the corresponding ASCII codes. Refer to the function "INSTRUMENT STATUS" in table 3 for the program code used to read syntax error codes.

Program Order Considerations: Measurement System functions may be programmed in practically any order from the HP-IB. However, it is recommended that program code sequences should be in the same order used for front-panel operation. Generally, this requires that functions be set up starting with the most basic parameter to be changed. For example, before entering channel 2 deflection factor, turn on channel 2.

DATA SENT TO HP-85 FROM 1980

General Output Data Message Format:



The instrument can send Data messages in local or remote mode, when it is addressed to talk or in the talk-only mode.

Note

Before the instrument is addressed to talk, the desired output data must be specified with the appropriate input Data message. Otherwise, the Measurement System outputs the ASCII character "E" by default to complete the bus transaction.

Output Data Message Format. Output Data messages include the settings of individual functions, instrument status information or binary Learn String data. Excluding the Learn String, there are three output data types: integer, decimal, and exponential. All output Data messages contain a leading space (<SP>) followed by the function value or status data. <CR> and <LF> are sent as the EOS message for all output data except the Learn String. The Learn String uses the EOI bus control line to signal end-of-string.

Note

Exponential values are sent by the 1980A/B with the ASCII character "E" (uppercase) as the delimiter between the mantissa and the exponent.

Table 3. Program Codes and Format Summary

Function	Program Code (ASCII)
ADVISORY MESSAGES	AV<state>
off	state ::= 0
on	1
AUTOSCOPE	
Execute autoscope	AS
Execute selective autoscope	SA
BANDWIDTH LIMIT	BW<state>
off	state ::= 0
on	1
CALIBRATOR LEVEL	CL<level>
0.02 V p-p	level ::= 1
0.1 V p-p	2
0.2 V p-p	3
1 V p-p	4
10 V p-p	5
CHARACTER GENERATOR	CG<state>
readout off	state ::= 0
readout on	1
CONTROL KNOB	RC<entry>[,<mode>]
Assign control knob	
hold	entry ::= 0
channel 1 deflection	1
channel 2 deflection	2
main sweep speed	3
delayed sweep speed	4
channel 1 position	5
channel 2 position	6
channel 1 position	7
channel 2 position	8
dual separation	9
horizontal position	10
main trigger level	11
delayed trigger level	12
delay	13
delay	14
delay	15
trace intensity	16
character intensity	17
panel intensity	18
Select step resolution:	
coarse steps	mode ::= 0
fine steps	1
DELAY	
Enter delay time (seconds)	DY<value> value ::= exponential [+]d.d[d][.]{E|e}[+|-][d]d [up to 9 digits] +0.00e-09 to +9.999999999e+00 sec min step = 100 psec
Enter digital delay (delayed trigger events)	DD<value> value ::= integer[1] [d]...[d] [up to 7 digits] 0 to 9999999 events
DELTA TIME	DZ<state>
ΔT mode off	state ::= 0
ΔT mode on (and zeroed)	1
DELTA VOLTS	DV<channel>,<mode>
vertical channel 1	channel ::= 1
vertical channel 2	2
ΔV off	mode ::= 0
ΔV on (and zeroed)	1
HORIZONTAL MODE	HM<mode>
main	mode ::= 1
intensified	2
delayed	3
dual	4

Function	Program Code (ASCII)
HORIZONTAL POSITION	HP<value>
(divisions)	value ::= decimal [+|-][d]d.d[d] -6.00 to +6.00 div minimum step = .02 div
HP-IB STATUS ADVISORY	BA<state>
off	state ::= 0
on	1
INITIALIZE	
Execute initialize:	IN
INSTRUMENT STATUS	
Specify data to be read	OQ<code>
HP-IB syntax error code	code ::= 1
last key code	2
trigger flag state	3
advisory or error code	4
internal option code	5
plug-in option code	6
	output format: <SP>ddd<CR><LF>
INTENSITY, READOUT	FP<char>[,<lamp>]
CRT Readout intensity	char ::= integer[1] dd 0 to 99%
(% full character brightness)	
Lamp (and LED) intensity:	lamp ::= integer[1] dd 0 to 99%
(% full lamp brightness)	
INTENSITY, TRACE	CI<value 1>[,<value 2>]
Main intensity level	value 1 ::= integer[1] dd 0 to 99%
(% max intensity)	
Delayed intensity level	value 2 ::= integer[1] dd 0 to 99%
(% max intensity)	
KEY	KY<code>[,<code>]
Valid keycodes are listed in table 3-27	code ::= integer[1] dd
LEARN MODE	TE
Specify Learn String output.	output format 80 eight bit bytes EOS = EOI bus control line true
Configure the 1980A/B using the Learn String.	<80 byte string><CR><LF>
	Note The Learn String must be transferred without sending UNL or UNT

READING VALUES			
Select function to be read	OF<code>		Format #
channel 1 deflection factor	code ::=	1	F1
channel 2 deflection factor		2	F1
main sweep speed		3	F2
delayed sweep speed		4	F2
channel 1 ΔV		5	F1
channel 2 ΔV		6	F1
channel 1 position		7	F3
channel 2 position		8	F3
dual separation		9	F3
horizontal position		10	F3
main trigger level		11	F4
delayed trigger level		12	F4
delay time		13	F5
ΔT		14	F6
digital delay		15	F7

NOTE: In numeric parameters, leading zeroes may be omitted.

APPENDIX III. HP-IB PROGRAM CODES

Here is the syntax used when programming the 3456A via HP-IB:

FUNCTION	(Unshifted)	(Shifted)
F1	DCV	DC/DC
F2	ACV	AC/DC
F3	AC+DC	(AC+DC)/DC
F4	2WR	O.C. 2WR
F5	4WR	O.C. 4WR
S1	Function SHIFT ON	
S0	Function SHIFT Off	

RANGE	(Volts)	(Ohms)
R1	AUTO Range	AUTO Range
R2	.1 (DCV only)	100
R3	1	1 K
R4	10	10 K
R5	100	100 K
R6	1 K	1 M
R7	- - -	10 M
R8	- - -	100 M
R9	- - -	1 G

AUTO ZERO

Z1	On
Z0	Off

FILTER

FL1	On
FL0	Off

MATH

M0	OFF
M1	PASS/FAIL
M2	STAT
M3	NULL
M4	dBm (R)
M5	THMS °F
M6	THMS °C
M7	(X-Z)/Y
M8	100 (X-Y)/Y
M9	20 LOG X/Y

TRIGGER

T1	INTernal
T2	EXTernal
T3	SINGLE
T4	HOLD

STORE and RECALL

STa or REa where (a) is any of the abbreviations listed below except where noted:

G	N DIG DISP
N	N RD/TRIG
D	DELAY SEC
I	N CYC INT
M	MEAN (Recall Only)
V	VARIANCE (Recall Only)
L	LOWER
U	UPPER
R	R
C	COUNT (Recall Only)
Z	Z
Y	Y

STore takes the contents of the display and places it in the specified register.
REcall takes the contents of the specified register and places it in the display and on the bus.

NUMERIC ENTRY

W ± DXXXXXX E ± X

Where

D is a 1 or 0
X is a digit 0 - 9.
W is an optional ASCII character.

Entry is free field in that signs assume default values and an implied decimal point is to the right of the last entered digit.

INSTRUMENT CONTROL

TE1	TEST On
TE0	TEST Off
RS1	RDGS STORE On
RS0	RDGS STORE Off
P1	Packed Mode On
P0	Packed Mode Off
D1	Display On
D0	Display Off
L1	Load Internal Memory On
Q	Load Internal Memory Off
X1	Execute Stored Program
SO1	System Output Mode On
SO0	System Output Mode Off
CL1	CLEAR CONTinue Asserted
H	Software RESET
SW1	Front/Rear Switch Sense
O1	Set EOI On
O0	Set EOI Off

SERIAL POLL STATUS BYTE

Bit 0	Front Panel SRQ
1	Execution of Program in Memory Complete
2	Data Ready
3	Trigger Too Fast (External Trigger Only)
4	Illegal Instrument State/Internal Error/Interpreter Syntax Error
5	Program Memory Error
6	rsv
7	Limits Failure (Pass/Fail)

Note: All bits masked at power-on.

SMDDD Set the Service Request Mask where DDD is the octal representation of the Serial Poll Mask Byte.

HP-IB COMMANDS

Follows are the HP-85 commands which send the various bus messages.

NOTES:

REMOTE

11Ø REMOTE 722 - Put device #22 on bus 7 into remote

- must be bus controllers

- Except for "local" or "return-to-local" key front panel not useable

23Ø REMOTE 722, 723

NOTES:

REMOTE 7 asserts REN control line. Devices do not go into remote state until they are addressed to listen

REMOTE 722 bus implementation:

CONTROL LINES		DATA LINES			
REN	ATN	OCTAL	DEC	ASCII	HP-IB
T	T	077	63	?	UNL (unlisten)
T	T	125	85	U	21 TALK
T	T	066	54	6	22 LISTEN

REMOTE 722 @ RESUME 7: USE RESUME TO SET ATN FALSE

LOCAL LOCKOUT 7

* LOCK OUT FRONT PANEL CONTROL OF ALL DEVICES ON BUS 7 (CAPABLE OF RESPONDING TO THIS COMMAND)

* "RETURN-TO-LOCAL" KEY DISABLED

* MUST BE SENT BY ACTIVE CONTROLLER

NOTES:

LOCAL LOCKOUT 7 @ RESUME 7: sets ATN false

LOCAL LOCKOUT 7 Bus Implementation:

CONTROL LINES		DATA LINES		
REN	ATN	OCTAL	ASCII	HP-IB
T	T	021	DC1	LLO

LOCAL 7

* Sets REN false
* Cancels a LOCAL LOCKOUT 7 command
* Must be issued by the system controller

LOCAL 709, 722,,,

* Must be all on SAME select code
* Returns to local front panel operation the devices at the specified device addresses.

NOTES:

LOCAL 709 Bus Implementation:

CONTROL LINES		DATA LINES			
REN	ATN	OCTAL	DEC	ASCII	HP-IB
T	T	077	63	?	UNL (unlisten)
T	T	125	85	V	21 talks
T	T	051	41	)	9 listens
T	T	001	01	SOH	GTL (go to local)

LOCAL 709 @ RESUME - sets ATN false

CLEAR 7

* Re-initiate each device on bus 7 capable of responding to this command
* must be system controller to send

CLEAR 722, 723

* CLEAR ONLY DEVICE #22 on bus 7 and device #23 on bus 7

Reinitialization is a function of the particular device. Typically it is the power-on state.

NOTES:

CLEAR 7 Bus implementation:

	CONTROL LINES		DATA LINES			
	REN	ATN	OCTAL	DEC	ASCII	HP-IB
	T	T	024	20	DC4	DCL (device clear)
CLEAR 722, 723	T	T	077	63	?	UNL (unlisten)
	T	T	125	85	V	21 talk
	T	T	066	54	6	22 listen
	T	T	067	55	7	23 listen
	T	T	004	04	EOT	SDC (selected device clear)

RESET 7

* Does a complete power on sequence

  includes

  1) Terminates current operation

  2) If system controller, pulse IFC, then put REN false then true.

  3) Perform self test (error 11Ø for failure)

  4) Clear interrupt mask

  5) Set default EOL sequence
     Default CR/LF

NOTES:

ABORTIO 7

* If system controller
  1) Pulse IFC
  2) Sets REN

* If active (but not system) controller
  1) Asserts ATN true
  2) Puts out its talk address

* If neither of the above
  1) Terminates I/O operation
  2) Leave bus in present state

HALT 7

1) Terminates I/O operation.
2) Become ready for next operation.
3) Leave bus in present state.

NOTES:

TRIGGER 7

* Must be active controller.
* All current listeners (capable of being triggered) are tirggered.

TRIGGER 709, 708

* Must be active controller.
* Trigger only specified devices.

TRIGGER 709, 709 @ RESUME

* Sets ATN false

NOTES: Remember that the trigger function of each device may be different.

BUS IMPLEMENTATION

	CONTROL LINES		DATA LINES			
	REN	ATN	OCTAL	DEC	ASCII	HP-IB
TRIGGER 709,708	T	T	077	63	?	UNL unlisten
	T	T	125	85	U	#21 talker
	T	T	050	40	(	#08 listen
	T	T	051	41	)	#09 listen
	T	T	010	08	BS	GET (group execute trigger)
TRIGGER 7	T	T	010	08	BS	GET

OUTPUT STATEMENT

Used to transmit information from the HP-85 to one or more devices on the bus.

EXAMPLES:

OUTPUT 722 USING "K"; "F2" (TELL DMM TO GO TO FUNCTION 2)

OUTPUT 722, 706 USING "K"; F2

OUTPUT 7 USING "K"; F2

* Must be the talker

NOTES:

BUS IMPLEMENTATION:

	CONTROL LINES		DATA LINES			
	REN	ATN	OCTAL	DEC	ASCII	HP-IB
OUTPUT 722, 706	T	T	077	63	?	UNL unlisten
	T	T	125	85	U	#21 talks
	T	T	066	54	6	#22 listens
	T	T	046	38	&	#06 listens
	T	F	106	70	F	Data
	T	F	062	50	2	Data
	T	F	015	13	CR	Terminator
	T	F	012	1Ø	LF	Terminator
OUTPUT 7	NO SEQUENCE GENERATED					

NOTE ON THE FREEFIELD OUTPUT STATEMENT

eg OUTPUT 722; "F2"

The default freefeild Output Statement sends 21 characters whether they are specified or not. In this Example 19 ASCII blanks are sent to device #22. It is a waste of time, and device #22 may react negatively to receiving them. So....

Use IMAGE Specifiers

eg OUTPUT 722 USING "K"; F2

NOTES:

BUS IMPLEMENTATION:

	DATA LINES			
ATN	OCTAL	DEC	ASCII	HP-IB
T	077	63	?	UNL unlisten
T	125	85	U	#21 talk
T	060	54	6	#22 listen
F	106	70	F	Data
F	062	71	2	Data
F	040	32		Data

19 ASCII BLANKS

F	015	13	CR	Terminator
F	012	10	LF	Terminator

ENTER STATEMENT

Used to get information from a device on the bus. It unlistens all devices, designates itself as the listener, assigns the appropriate device as the talker, and then read the data on the data bus per the IMAGE or free-field specifiers.

```
ENTER 709; V(S)

ENTER 709 Using "#,K" A$
```

NOTES: Statement implementation on the bus:

CONTROL LINES		DATA LINES			
REN	ATN	OCT	DEC	ASCII	HP-IB
T	T	077	63	?	UNL unlisten
T	T	065	53	5	#21 listens
T	T	126	86	V	#22 talks
T	F	055	45	-	Data
T	F	060	48	Ø	
T	F	067	55	7	
T	F	056	46	.	
T	F	066	54	6	
T	F	070	56	8	
T	F	062	50	2	
T	F	071	57	9	
T	F	066	54	6	
T	F	105	69	E	
T	F	053	43	+	
T	F	060	48	0	Data
T	F	015	13	CR	Terminator
T	F	012	10	LF	Terminator

HEWLETT hp PACKARD

SENDING CUSTOM BUS COMMANDS

Sometimes one must configure the bus without using normal ENTER/OUTPUT addressing. (which is performed automatically).

Examples of this are;

A) Receiving or sending information to/from devices which use secondary addressing. eg 6942 Multiprogrammer

B) Sending measurement data from a device to both the HP-85 and another device.

C) Sending data from one device to another device with the HP-85 not being involved.

D) Communications with devices when the OUTPUT/ENTER addressing doesn't work. (sometimes encountered with non HP devices.)

NOTES:

HEWLETT hp PACKARD

SEND COMMAND

* Must be active controller to send commands.
* Commands are sent with ATN true.
* Data is sent with ATN false.

Syntaxes

SEND 7; CMD list DATA list
SEND7; UNL UNT MTA LISTEN number(s)
SEND7: UNL MTA LISTEN number DATA list

NOTES:

Valid Parameters:

CMD	command list
DATA	dat list then EOL sequence
TALK	address(es)
LISTEN	address(es)
SCG	secondary address number
UNL	unlisten
UNT	untalk
MLA	my listen address #
MTA	my talk address #

HEWLETT hp PACKARD

SECONDARY ADDRESSING EXAMPLE

Reading in data from 6942 (device 23) to HP-85 (device 21).

```
110  REM READ CLOCK
120  OUTPUT 723 USING "K"; "RC"
130  !
140  !   MAKE DEVICE 23 SECONDARY ADDRESS
150  !   14 THE TALKER
160  SEND 7; UNL TALK 23 SCG 14 MLA
170  ENTER 7; D,H,M,S
```

NOTES:

Line 160 could also have been written as

```
16Ø  SEND 7; CMD"?Wn5"
```

HEWLETT hp PACKARD

MORE CUSTOM BUS COMMAND EXAMPLES

A) Sending data from multimeter (device 22) to line printer (device 01)

```
510 SEND 7; UNL TALK 22 LISTEN 01
520 RESUME 7
```

B) Adding the HP-85 (device 21) as a listener to above

```
610 SEND 7; UNL TALK 22 LISTEN 01, 21
620 ENTER 7; V
```

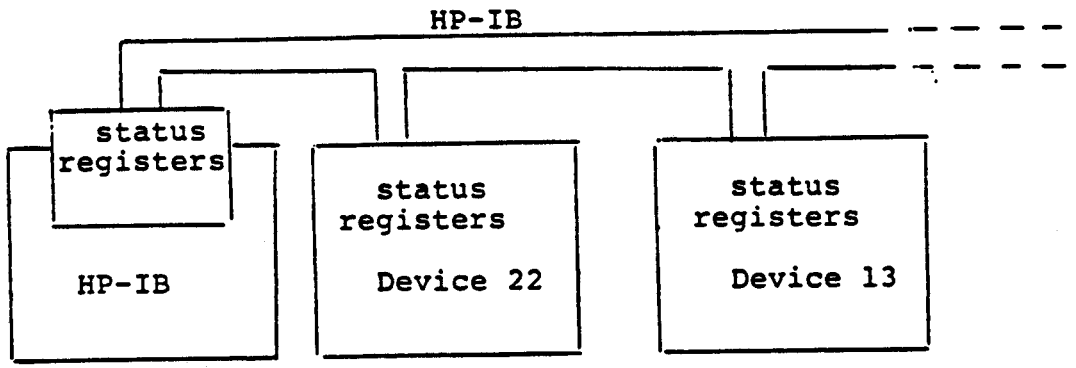
C) Custom device communications.

```
SEND 7: MTA UNL LISTEN 23 ! 9835/45 PROTOCOL
SEND S; CMD "U?%" DATA "TESTING 12"
```

NOTES:

HEWLETT PACKARD

STATUS OF THE BUS AND DEVICES ON IT



There are registers in the 82937A interface and in most HP-IB devices which provide information on the state of the interface or a device on it.

Reading and interpreting these registers can tell the HP-85 what has happened so that appropriate action can be taken.

NOTES:

Examples on use:

A) To check that a line printer has paper before we write to it.

B) To check whether we have misprogrammed a device.

C) To check whether a measurement has been completed before we ask for the data.

STATUS OF THE HP-IB VIA 82937

The 82937A has 7 status registers.

STATUS select code , reg # ; return variable

STATUS 7,1;S - Returns the status of the interrupt cause register (register 1)

?? if S = 8 what has happened ??

?? What does STATUS 7,2;A
with A = 17 mean ??

NOTES:

HP-IB Status Registers

Status Register Number	Bit Number 7	6	5	4	3	2	1	0	Default Value	Register Function
SR0	0	0	0	0	0	0	0	1	1	Interface Identification
SR1	IFC	LA	CA	TA	SRQ	DCL or SDC	GET	SCG	0	Interrupt Cause
SR2	0	REN	SRQ	ATN	EOI	DAV	NDAC	NRFD	64	HP-IB Control Lines
SR3	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	Not Applicable	HP-IB Data Lines
SR4	0	0	SC	A4	A3	A2	A1	A0	53	HP-IB Address/ System Controller
SR5	SC	LA	CA	TA	SPE	Parity Error	REN	LLO	160	State Register
SR6	0	0	0	SC5	SC4	SC3	SC2	SC1	0	Secondary Commands

HEWLETT hp PACKARD

STATUS OF HP-IB DEVICES

The STATUS or SRQ register(s) of a device is read with a;

Q = SPOLL (device selector)

Q = SPOLL (723) { reads the status register of device 23 on bus 7

Q = SPOLL (7) - read status byte of current talker

NOTES:

* This is called serial polling a device (explained later).
* Q is just one of many variables which could contain the value of the status register.
* Q contains a decimal number which represents the binary equivalent value of the 8 bits in the status register.
* The meaning of Q is a function of the individual device (see next page).
* Most HP-IB devices do not support parallel polling.

INTERROGATING THE STATUS BYTE(S)

```
120 S     = SPOLL (723)
130 IF S  = 65 THEN PRINT "3455 DATA READY"
140 IF S  = 66 THEN PRINT "SYNTAX ERROR"
150 IF S  = 72 THEN PRINT "TRIGGERED TOO FAST"
```

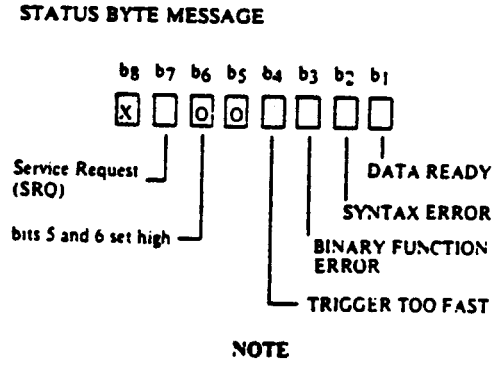
ANOTHER MORE FLEXIBLE METHOD USES THE BIT FUNCTION

```
120   X  = SPOLL (723)
130   IF BIT (X,6) = Ø THEN GOTO 200 ! NOT 3455
140   IF BIT (X,Ø) THEN PRINT "3455 DATA READY"
150   IF BIT (X,1) = 1 THEN PRINT "SYNTAX ERROR"
160   IF BIT (X,3) THEN PRINT "TRIGGERED TO FAST"
...
...
```

NOTES:

* Also applies to STATUS 7, 1;S

?? Why is using the BIT FUNCTION more powerful ??

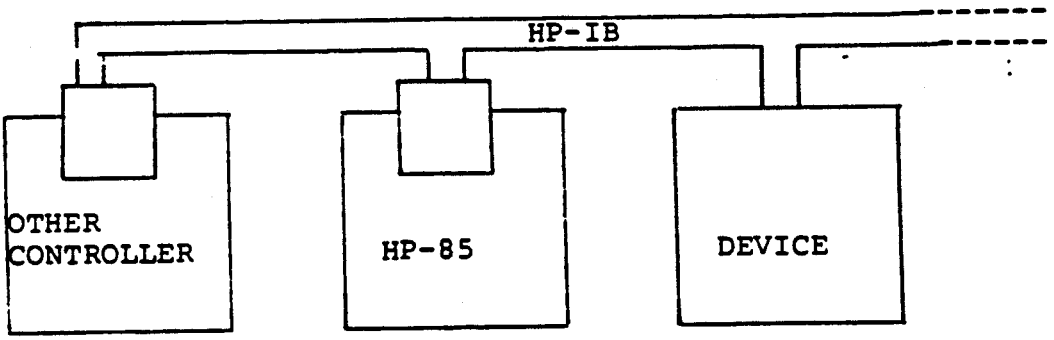


NOTE

All "bits" are low true; bit 8 is not used.

Status Byte Code		
ASCII CHAR	Decimal Code	
A	65	Data Ready - Indicates to the controller that measurement data is available. Applies to DATA READY Request feature.
B	66	Syntax Error - Indicates improper program code. Example - Program Code "F7" would cause a syntax error since the FUNCTION program set is only defined for codes F1 through F6.
D	68	BINARY FUNCTION Error - Indicates improper BINARY PROGRAM code or incomplete binary message. Similar to syntax error.
H	72	Trigger too Fast - Indicates the 3455A has been triggered while measurement data is being output to the bus. Warns of possible incorrect measurement information.

MORE THAN 1 BUS CONTROLLER



There can be more than 1 controller on the HP-IB. One must be designated the system controller (by switch settings in the interface card) so that when the IFC line is asserted or power up occurs, the system controller is in command.

Active control can be passed back and forth between the controllers.

NOTES:

HEWLETT PACKARD

MULTIPLE CONTROLLER COMMANDS

PASS CONTROL 715

* Must be active controller to send
* Passes control to device #15

PASS CONTROL 7

* Must be active controller to send
* Passes control to current talker.

REQUEST 7;X

* Cannot be sent by active controller.
* Used by 85 to request service.
* Used by 85 to ask to become controller.
* If bit 6 or X is true SRQ is set true.
* 85 sends X in response to a serial poll and sends SRQ false if was true.

NOTES:

BUS IMPLEMENTATION

	CONTROL LINES		DATA LINES			
	REN	ATN	OCTAL	DEC	ADCII	HP-IB
PASS CONTROL 715		T	077	63	?	UNL unlisten
(from device #21)		T	065	43	5	#21 listen
		T	117	79	O	#15 talk
		T	077	63	?	UNL unlisten
		T	011	09	HT	TCT take control
		F				
PASS CONTROL 7		T	077	63	?	UNL unlisten
		T	011	09	HT	TCT take control
		F				

HP-IB PROGRAM STATEMENTS NOT COVERED

ASSERT 7;X	Write to interface register CRZ
CONTROL 7,n;X	Write to interface register CRn
PPOLL (7)	Parallel Poll

These are not covered as they are not usually needed.

NOTES:

HEWLETT PACKARD

HP-IB OPERATION IN GREATER DETAIL

Refer to sections of

Tutorial Description of the
Hewlett-Packard Interface Bus
(PN 5952-0156)

NOTES:

Modify previous program to:

A. Replace random number generator portion with section that takes data from an available instrument.

B. Exercise the HP-IB statements you have learned about to see how they affect your instrument.

C. Send the instrument variable data (eg various number of readings, delay, etc.) to make sure it understands it.

D. If possible program the device to make a measurement that is slow. Trigger the instrument and monitor its status byte to know when the data is ready. Until it is ready display a "waiting" message.

E. If not D program your instrument incorrectly and read back the status byte to see if it indicates improper programming.

EXTRA CREDIT

F. Monitor the 82937 register to know when your instrument asserts the SRQ line.

G. Have your instrument send its data to a line printer rather than the 85.

H. Use your imagination!

HEWLETT PACKARD

INTRODUCTION TO INTERRUPTS

A running program needs the ability to respond to real time events which may occur at any time.

EXAMPLES:

A. Perform a task every 30 seconds
B. Perform a task at 12:00 P.M.
C. REspond to button pushed
D. Get a reading when data is ready
E. Get a reading when operator pushes button
F. Special function key K7 pushed
G. Instrument malfunction (select code timeout)
H. HP-85 key pressed
I. Error
J. A character received from a system device (eg part of measurement data)

NOTES:

The HP-85 responds to these events (called interrupts) by branching to the appropriate program segment (as set up by programmer), and executing this section of program. Upon completion of this "interrupt routine", the HP-85 returns to the next line it would have executed had the interrupt not occurred.

Branch Precedence Table

Branch Type	Select Code 3	4	5	6	7	8	9	10
ON ERROR	1							
ON INTR	2	3	4	5	6	7	8	9
ON TIMEOUT	10	11	12	13	14	15	16	17
ON EOT	18	19	20	21	22	23	24	25
ON KEY	26							
ON TIMER	27							

End-of-line service occurs in a specific order. That is, if more than one end-of-line branch is pending at the end of a program line, one of the branches will be taken before the other. The following table lists the types of end-of-line branches and the select codes, and gives the precedence order for combinations of branch type and select code.

NOTES:

INTERRUPT IMPLEMENTATION

```
XXX   ! MAIN PROGRAM
XXX
XXX   INTERRUPT(S) SETUP
XXX   MAIN PROGRAM EXECUTION                                  CHECK
X+1      do something                                          FOR
X+2      do something                                       INTERRUPT
X+3      do something                                          AT
X+4      do something --- if interrupted here                 THE
X+5      do something --- will return here!                   END
X+6      do something                                          OF
X+7      do something                                        EACH
X+8      do something                                        LINE
YYY   INTERRUPT SUBROUTINE OR SUBPROGRAM #1
Y+1      take action
Y+2      take action
Y+3      RE-ENABLE FOR INTERRUPT
Y+4      RETURN
ZZZ   INTERRUPT SUBROUTINE OR SUBPROGRAM #2
Z+1      take action
Z+2      take action
Z+3      RE-ENABLE INTERRUPT
Z+4      RETURN
XXX   MORE SUBROUTINES OR SUBPROGRAMS
XXX      do something
XXX
```

NOTES:

It services interrupts on an end-of-line basis. At the end of each line it checks a status register to see if an interrupt has occurred. If so and the program has been setup to service it, the computer will jump to the "programed" subroutine or program area to take the desired action. If a subroutine was used, upon completion of this action, the program returns to the next statement it would have executed had the interrupt not occurred.

HEWLETT hp PACKARD

HP-85 TIMERS FOR INTERRUPT (156 - 158)

3 Timers Available (1, 2, 3)

Activated with

```
ON TIMER #1, 1ØØØØ GOTO  2Ø5Ø
ON TIMER #2, 2450  GOSUB 5ØØØ
```

Timers are deactivated with;

```
OFF TIMER # 1
          # 2
          # 3
```

during editing, [SCRATCH] , or [RESET]

NOTES:

The number of milliseconds must be greater then .5 and less than 99999999.

Timers continue to interrupt the program after program has halted, but branching does not occur, so be sure to de-activate!

ENTER THIS PROGRAM

```
 10 ! TIMER INTERRUPTS
 20 !
 30 ON TIMER# 1,5000 GOSUB 110
 40 !
 50 I=1
 60 DISP I
 70 I=I+1
 80 GOTO 60
 90 !
100 !
110 ! INTERRUPT ROUTINE
120 BEEP
130 PRINT "INTERRUPT ON COUNT OF
    ";I
140 RETURN
```

WHAT HAPPENS?

NOTES:

Change the 5ØØØ to 7ØØ. What happens?

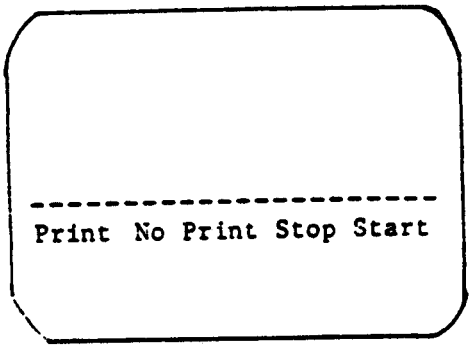
Change the GOSUB / RETURN TO GOTO's

Any changes?

HEWLETT PACKARD

SPECIAL FUNCTION KEYS

HP-85 CRT



KEY LABLE	K5 K1	K6 K2	K7 K3	K8 K4

```
... ON KEY #1,"PRINT" GOSUB 2000
... ON KEY #2,"NOPRINT" GOSUB 2040
... ON KEY #5,"START" GOTO 1040
... ON KEY #4, "STOP" GOTO 4000

1010 DISP"WAITING FOR START KEY"
1020 CLEAR
1030 GOTO 1010
1040 !
1050 !
...
...
...  IF P9=1 THEN PRINT "DATA IS"
...
...
...
...
2000 ! SET PRINT MODE
2010 P9=1
2020 RETURN
2030 !
2040 ! PRINT MODE OFF
2050 P9=0
2060 RETURN
...
...
4000 END
```

NOTES:

USE TO SET OR CLEAR CERTAIN CONDITIONS OR MODES OF OPERATION.

USE TO CONTROL PROGRAM EXECUTION ONLY USING SPECIAL FUNCTION KEYS.

INTERRUPTS CAUSED BY HP-85 KEYBOARD KEYS

A program is interrupted whenever an HP-85 key is pressed. It continues whenever the CONT Key is pressed.

Desirable Aspects:

A. Allow a mathmatical calculation

B. Allow a variable value to be examined

Undesirable Aspects:

A. Unintensional hit of key halts "Unattended" Program.

NOTES:

Try

```
1Ø I = 1
20 DISP I
30 I = I + 1
40 GOTO 20
50 END
```

?? What keys <u>do</u> <u>not</u> interrupt the computer ??

DISABLING THE KEYBOARD (I/O Pg. 75-77)

The programmer has the ability to lock out 4 classes of keys while:

A. A program is executing
B. A keyboard entry is being input

The 4 Key Classes are:

1. RESET Key
2. PAUSE Key
3. Special function and KEY LABEL Keys
4. All remaining keys not covered above.

NOTES:

1. Any and all can be masked out.
2. It can only be done in a program.
3. A stopped PROGRAM (due to error, STOP, PAUSE, END, etc.) returns complete control of the keyboard to the system (as if no ENABLE KEYBOARD was ever executed.) When the program is continued or re-run the ENABLE KEYBOARD mask comes back into effect.

ENABLE KEYBOARD STATEMENT

KEYBOARD MASK:

Bit Number	Decimal Value	Operating Mode	Keys Masked
7	128	↑	RESET
6	64	Program	PAUSE
5	32	Execution	Special Function Keys and KEYLABEL
4	16	↓	Other keys
3	8	↑	RESET
2	4	Keyboard	PAUSE
1	2	Input	Special Function Keys and KEYLABEL
0	1	↓	Other keys

Setting a bit (=1) enables the corresponding Keys

Clearing a bit (=Ø) disables the corresponding Keys

ENABLE KBD 32 + 1 + 2 allows what ??

NOTES:

1. ENABLE KBD 255 enables all keys and is done at power up.
2. ENABLE KBD BTD ("01001111") is allowable

   ?? What does the above mask do ??

ENTER THIS PROGRAM AND FIND OUT

```
10 ENABLE KBD BTD("01001111")
20 I=1
30 DISP I
40 I=I+1
50 GOTO30
60 END
```

Try other variations.

USEFUL HINT

If an HP-IB I/O transfer becomes hung and the and the keyboard is locked out, and you do not want to lose memory, you can abort the I/O process by grounding the IFC line.
(Pins 24 or 12 to 9)

IF

The following program lines are present;

```
XXX ON INTR 7 GOTO 9000
XXX ENABLE INTR 7;128 !interrupt on IFC asserted
...
...
...
...
9000 END
```

NOTES:

EXTERNAL INTERRUPTS

Key in the following program:

```
 10 ON INTR 7 GOSUB 100
 20 ENABLE INTR 7;8
 30 I=1
 40 DISP I
 50 I=I+1
 60 GOTO 40
 70 !
100 BEEP
110 STATUS 7,1;A
120 PRINT "I= ";I
130 ENABLE INTR 7;8
140 RETURN
```

?? What will happen if you ground the SRQ line (Pin 1Ø to 12 or 24) ??

Comment out line 130 eg ! 130 ENABLE INTR 7;8 so that it no longer gets executed.

?? How does this affect the Program ??

NOTES:

HEWLETT hp PACKARD

INTERRUPT STATEMENTS

ON INTR 7 GOSUB line number ON INTR 7 GOTO line number	Tells computer where to go when interrupt occurs.
ENABLE INTR 7;8 or CONTROL 7,1;8	Enable the interrupt register to interrupt on SRQ being asserted.
STATUS 7,1; Variable	Read the interrupt cause register. Do this in the interrupt routine.
OFF INTR 7	Disable interrupts on select code 7. Allow 1 interrupt to be logged in while disabled.

NOTES:

```
10 ! INTERUPT STRUCTURE
20 ! ON HP-85
30 ! USING 3497 AS EXAMPLE
40 !
50 !
60 CLEAR 709 ! RESET 3497
70 !
80 ! SET UP INTERRUPT LINKAGE
90 ON INTR 7 GOSUB 260
100 !
110 ! SET INFO TO 3497
120 OUTPUT 709 USING "K" ; "SE200" ! INTERRUPT ON FRONT PANEL SRQ
130 !
140 !
150 ! ENABLE 85 TO RECEIVE
160 ! INTERRUPTS
170 ENABLE INTR 7;8
180 !
190 I=1 !          KEEP
200 DISP I !       BUSY
210 WAIT 100 !     UNTIL
220 I=I+1 !        INTERRUPT
230 GOTO 200 !     OCCURRS
240 !
250 !
260 ! INTERRUPT ROUTINE
270 !
280 ! MUST READ INTERRUPT
290 ! CAUSE REGISTER
300 STATUS 7,1 ; A@ PRINT A
310 !
320 ! MUST SERIAL POLL TO
330 ! TO CLEAR INSTRUMENT SRQ
340 D=SPOLL(709) @ PRINT D
350 !
360 ! PERFORM SOME ACTION
370 !
380 OUTPUT 709 USING "K" ; "SA"
390 !
400 ! RE-ENABLE FOR FUTURE
410 ! INTERRUPTS
420 ENABLE INTR 7;8
430 !
440 RETURN ! TO WHERE YOU WERE
```

HEWLETT hp PACKARD

TIMEOUTS

Provides a method of aborting an interface handshake that does not occur within a specifed period of time.

SET TIMEOUT 7; number of milliseconds

If handshake does not occur within specified time abort the I/O operation and continue with the next line.

SET TIMEOUT 7; number of milliseconds

ON TIMEOUT 7 GOSUB line number
GOTO

Go to specified area if timeout ocurrs.

OFF TIMEOUT 7 Cancels the ON TIMEOUT statement.

SET TIMEOUT 7;0 - Almost infinite timeout.

```
 10 ! TIMEOUT EXAMPLE
 20 !
 30 ! 3456 IS TO BE FRONT PANEL
 40 ! TRIGGERED AND SEND READING
 50 ! TO 85.
 60 !
 70 ! IF NOT TRIGGERED IN 10 SEC
 80 ! GOSUB ROUTINE TO INDICATE
 90 ! NO TRIGGER - THEN TRY
100 ! AGAIN AND AGAIN
110 !
120 !
130 SET TIMEOUT 7;9000 ! 9 SEC
140 ON TIMEOUT 7 GOSUB 300
150 !
160 ABORTIO 7
170 CLEAR 722 ! POWER ON SETUP
180 !
190 ! FRONT PANEL TRIGGER
200 OUTPUT 722 USING "K" ; "T3"
210 LOCAL 722
220 !
230 !
240 ENTER 722 ; V
250 DISP V
260 GOTO 240
270 !
280 ! TIMEOUT ROUTINE
290 !
300 BEEP
310 DISP "3456 NOT TRIGGERED"
320 V=0
330 ABORTIO 7
340 RETURN
```

```
 3.53706
 3.53526
 3.53675
 3.53672
 3.53568
 3.53634
 3.53709
3456 NOT TRIGGERED
 0
3456 NOT TRIGGERED
 0
 3.53294
 3.53717
 3.53846
 3.53971
 3.54147
 3.54216
3456 NOT TRIGGERED
```

HEWLETT hp PACKARD

ERROR TRAPPING

ON ERROR GOSUB line number ON ERROR GOTO line number	When an error ocurrs go to this area of the program.
OFF ERROR	Cancels the ON ERROR

ERRL - Function which gives the line number of the error.

ERRN - Function which gives the error number.

I/O ROM adds;

ERROM - Function which provides the last option ROM error. It is 192 for the I/O ROM.

ERRSC - Function which provides the select code number which generated the most recent error.

NOTES:

LAB 8

DO ANY OR ALL OF THE FOLLOWING:

A) MODIFY AN EARLIER LAB TO ACQUIRE DATA PERIODICALLY AS DETERMINED BY THE INTERNAL TIMERS.

B) ACQUIRE DATA WHENEVER SPECIAL FUNCTION KEY k4 IS PRESSED.

C) PRINT THE DATA WHENEVER SPECIAL FUNCTION KEY k3 IS PRESSED - DISPLAY THE DATA WHENEVER KEY k2 IS PRESSED.

D) WRITE A PROGRAM THAT ENTERS THE DATA FROM AN "OPERATOR TRIGGERED" DEVICE (EG VOLTMETER) WHENEVER THAT DEVICE HAS DATA READY FOR THE HP-85. THE HP-85 SHOULD BE BUSY DOING OTHER THINGS WHEN NOT ENTERING THE READINGS.

E) MODIFY YOUR PROGRAM TO PROTECT AGAINST ACCIDENTAL STOPAGE THROUGH UNINTENSIONAL KEYBOARD HIT.

EXTRA CREDIT:

A) USE YOUR IMAGINATION!

NOTES:

IO BUFFERS

IO BUFFERS are areas of HP-85 memory that are allocated for the purpose of holding data received or to be sent to an external device.

Their purpose is twofold;

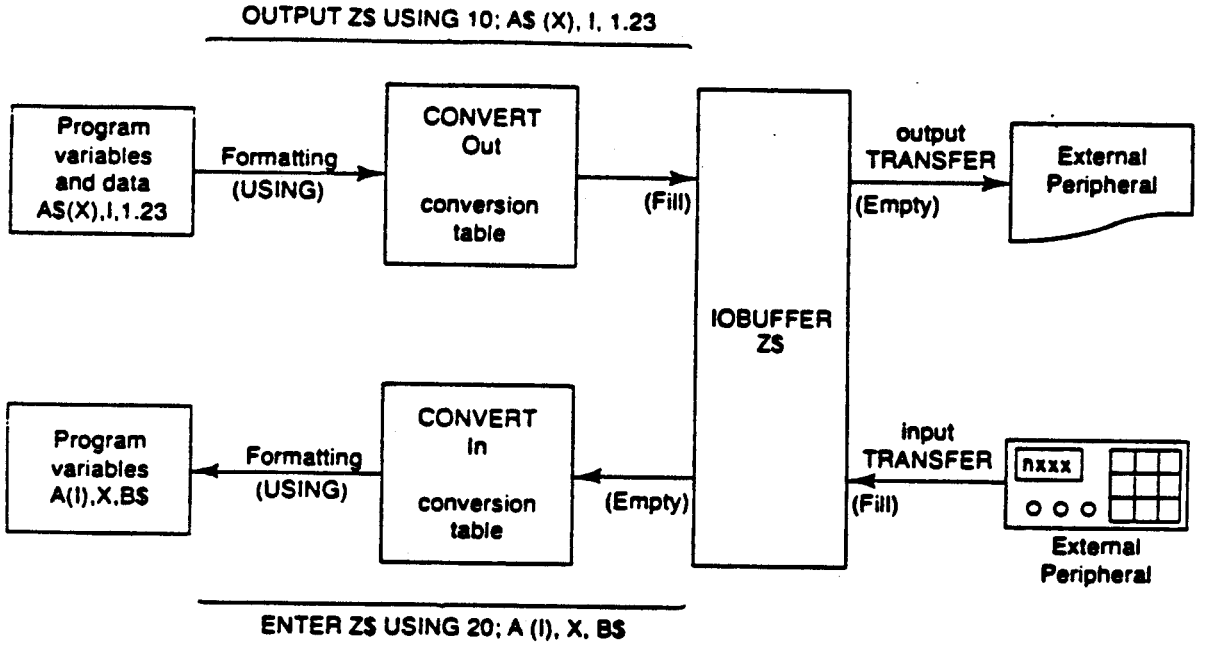
1) Some devices are a lot slower than a CPU (printers, some DVM's etc.). In these cases it is desireable to have the CPU do some useful work instead of sitting around idle, waiting on a slow IO transfer. By pre- or post-formatting your IO data and putting all the info in a buffer, an interrupt type transfer can be started, and the HP-85 will be interrupted every time the device is ready to send/receive a character.

2) Other IO devices are fast and will accept or send data as fast as the CPU can handle it. A fast handshake transfer is used in this case. On input all incoming data is put into the buffer as is. The program will later format it into useable info via the ENTER statement. On output all data is pre-formatted and put into the buffer with an OUTPUT or string operation, and then sent out as fast as the device can handle it.

NOTES:

IOBUFFER TO MEMORY CONSIDERATIONS

The ,TRANSFER statement does not do any formatting or data conversion, so what is in the buffer is what is sent to or received from the data device.



NOTES:

GIVEN: Raw data received from voltmeter in IOBUFFER Z$ in the form;

-1.235,+1.789,-3.234,+1.4,-5.678,-9.876,+3.13 CRLF

?? What would the FOR/NEXT loop needed to enter this data ??

SETTING UP AN IO BUFFER

IO BUFFER IS A STRING VARIABLE THAT HAS BEEN DECLARED AN IO BUFFER BY THE FOLLOWING STATEMENTS;

```
140 DIM A$ [10008] ! String variable of 10008 characters.
150 IOBUFFER A$    ! Buffer with 10000 character capacity.
```

* THERE IS NO WAY TO UNDECLARE AN IOBUFFER.
* 8 BYTES ARE NEEDED FOR OVERHEAD.

NOTES:

TYPES

INTERRUPT TRANSFERS - They are used for slow or random input devices or slow output devices.

The transfer can take place while the program continues running.

FAST HANDSHAKE TRANSFERS - They are used with fast input or output devices.

The HP-85 dedicated itself to the transfer and cannot continue running the program.

NOTES:

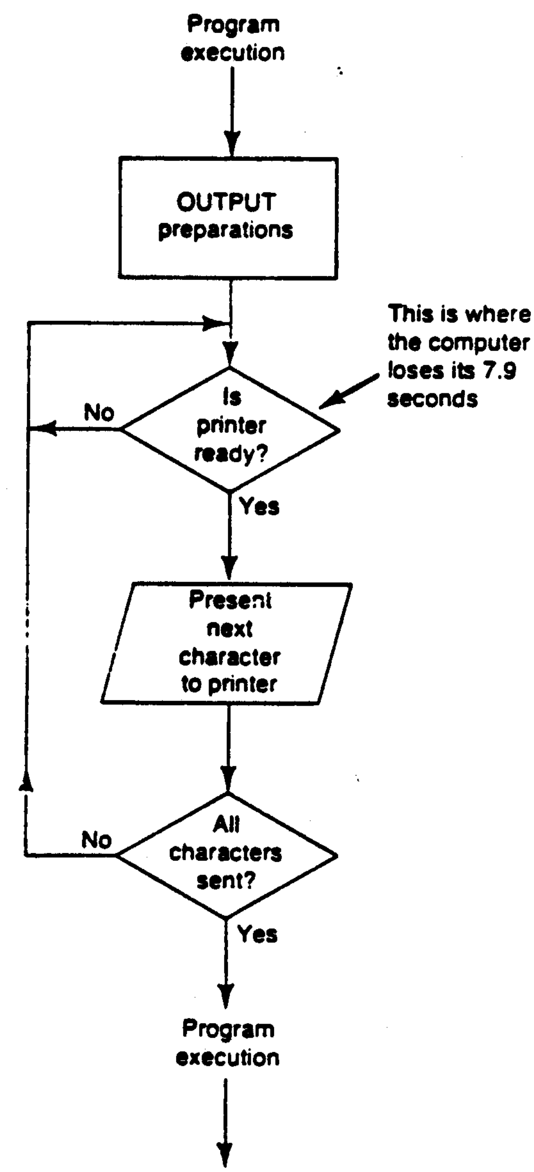
WHY USE AN INTERRUPT BUFFER?

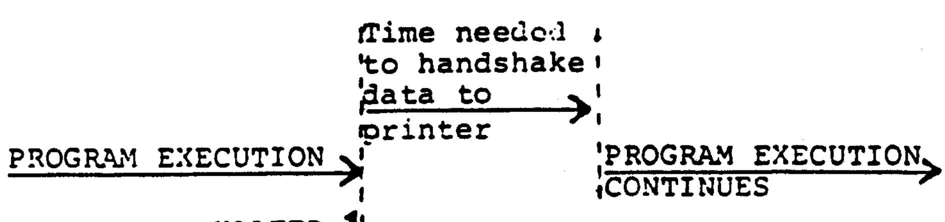
EXAMPLE: SENDING DATA TO A LINE PRINTER WITHOUT BUFFERING.

```
DIM A$ [500]
A$ = "______"

OUTPUT 705;A$
```

each character is handshaked until all of A$ is sent.

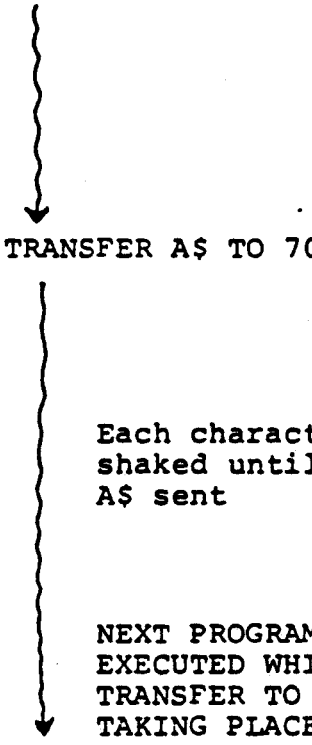
NEXT PROGRAM LINE IS NOT EXECUTED UNTIL ALL A$ SENT.





AN INTERRUPT BUFFER ALLOWS SIMULTANEOUS I/O AND PROGRAM EXECUTION

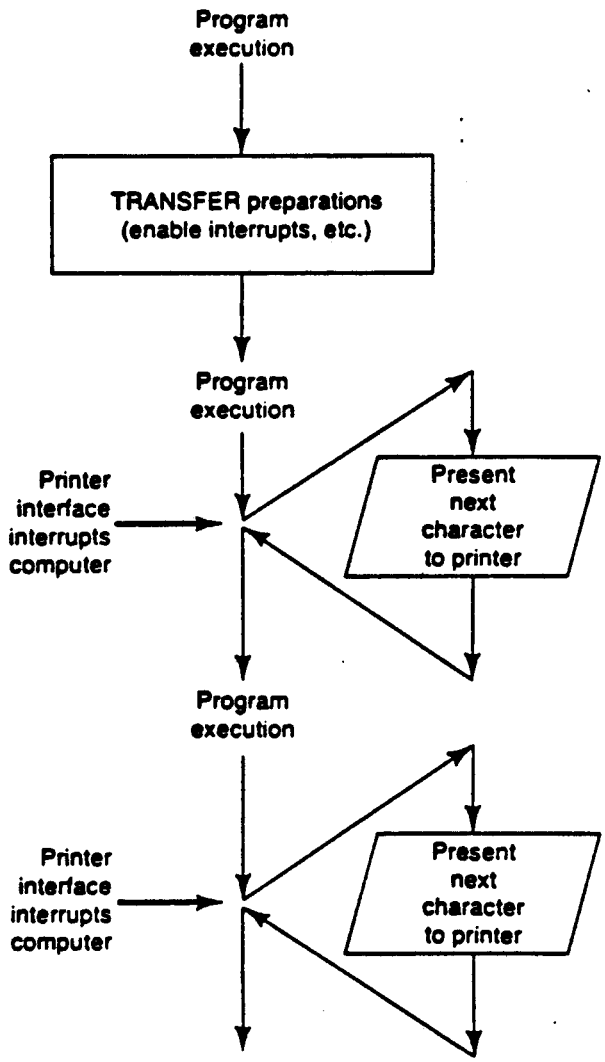
```
DIM A$ [500]
A$ = "                    "
IO BUFFER A$
```

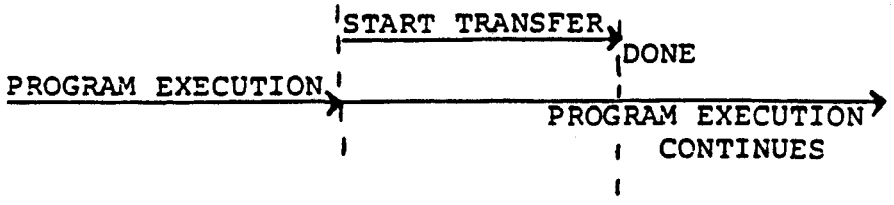


```
TRANSFER A$ TO 706 INTR
```


Each character hand shaked until all A$ sent

NEXT PROGRAM LINE(S) EXECUTED WHILE TRANSFER TO PRINTER TAKING PLACE





INTERRUPT BUFFER STATEMENTS

```
140 DIM Z$ [108]
150 IOBUFFER Z$
...
...
...
... TRANSFER 724 TO Z$  INTR ! enter until Z$ is full(100 char)
...
...
... TRANSFER 724 TO Z$ INTR; COUNT 50 ! enter 50 characters.
...
...
... TRANSFER 724 TO Z$ INTR; DELIM 10 ! terminate on LF or
                                      ! Z$ full
...
...
... TRANSFER 724 TO Z$ INTR;EOI       ! terminate on EOI or
                                      ! Z$ full
...
... TRANSFER Z$ TO 706 INTR           ! transfer all of Z$
...
...
... TRANSFER Z$ TO 706 INTR; COUNT N  ! transfer N characters
...
...
```

NOTES:

PROGRAM CONTINUES WHILE TRANSFER IS IN PROCESS.

FAST HANDSHAKE TRANSFERS

```
140 DIM X$ [1008]
150 IOBUFFER X$
...
...
...TRANSFER 724 TO X$ FHS          ! enter until X$ is full
...                                ! 1000 characters
...
...
...TRANSFER 724 TO X$ FHS;COUNT 500 ! enter 500 characters
...
...
...
...TRANSFER 724 TO X$ FHS;EOI       ! terminate on EOI
...                                 ! or after 1000 characters
...
...
...
...TRANSFER X$ TO 706 FHS          ! transfer 1000 characters
..
...
...
...TRANSFER X$ TO 708 FHS; COUNT 200 ! Transfer 200 characters
```

NOTES:

BUFFER STATUS

There are registers which contain information on the status of the buffer, such as whether it is active and how much information is in the buffer.

This example show how to obtain this info;

```
250 !
260 ! SUBROUTINE TO PRINT OUT
270 ! BUFFER STATUS
280 !
290 PRINT
300 PRINT "      BUFFER STATUS"
310 PRINT
320 STATUS Z$,0 ; S1,S2,S3,S4
330 PRINT "BUFFER EMPTY POINTER = ";S1
340 PRINT "BUFFER FULL POINTER = ";S2
350 PRINT "ACTIVE OUT S.C. = ";S3
360 PRINT "ACTIVE IN S.C. = ";S4
370 PRINT
380 PRINT
390 RETURN
 28621
```

NOTES:

INTERRUPT TERMINATION OF TRANSFER

An interrupt or fast handshake transfer can cause an END-OF-LINE interrupt after its completion.

```
...
...
... ON EOT 7 GOTO 500
...
... DIM X$   [10000]
... IOBUFFER X$
...
...
...
...
...
...TRANSFER 724 TO X$ INTR; COUNT  700
...
...! DO
...! OTHER THINGS
...
...
...
500 FORMAT DATA
```

NOTES:

```
+1.644,+1.423,+1.281,+1.172,+1.0
86.+1.015,+0.957,+0.907,+0.867,+
0.831,+0.801,+0.776,+0.755,+0.73
5,+0.720,+0.706,+0.695,+0.685,+0
.676,+0.669,+0.663,+0.656,+0.651
.+0.648,+0.644,+0.640,+0.638,+0.
635,+0.633,+0.630,+0 630.+0.628,
+0.627,+0.626,+0.625,+0.625,+0.6
24,+0.624,+0.624,+0.624.+0 623,+
0.623,+0.623
```

```
 1                          1.644
 2                          1.423
 3                          1.281
 4                          1.172
 5                          1.086
 6                          1.015
 7                           .957
 8                           .907
 9                           .867
 10                          .831
 11                          .801
 12                          .776
 13                          .755
 14                          .735
```

```
 10 ABORTIO 7
 20 CLEAR 724
 30 ON EOT 7 GOTO 150
 50 OPTION BASE 1
 60 OUTPUT 724 ;"R2T1N100SD.1SE7
    S"
 70 DIM A$[7200],V(1000)
 80 IOBUFFER A$
 90 STATUS 7,1 ; A
100 ENABLE INTR 7;8
110 TRANSFER 724 TO A$ INTR ; CO
    UNT 701
120 DISP "KEEPING BUSY"
140 GOTO 120
150 BEEP
160 PRINT A$[1,300]
170 FOR I=1 TO 100
180 ENTER A$ USING "#,SD.DDDC" ;
     V(I)
190 DISP I,V(I)
200 NEXT I
210 END
```

BUFFERS LAB

DO ANY OR ALL OF THE FOLLOWING:

A) USE AN INTERRUPT BUFFER AND PROGRAM AN INSTRUMENT TO TAKE READINGS AND TRANSFER THE READINGS TO THE 85. MEANWHILE DO OTHER THINGS.

B) USE ANINTERRUPT BUFFER TO BUFFER A LINE PRINTER OR PLOTTER SO IT WILL NOT SLOW DOWN DATA ACQUISITION WHICH REQUIRES DATA OUTPUT TO AN EXTERNAL DEVICE WHILE IT IS BEING ACQUIRED. COMPARE THE DATA ACQUISITION RATES WITH AND WITHOUT A BUFFERED LINE PRINTER.

C) USE A FAST HANDSHAKE BUFFER TO ENTER A BURST OF READINGS FROM A HIGH SPEED DVM OR COUNTER.

EXTRA CREDIT:

HAVE THE COMPUTER INTERRUPT UPON TRANSFER COMPLETION.

NOTES:

G R A P H I C S

GRAPHICS MODE IS SET WHEN:

* GRAPH STATEMENT IS EXECUTED.
  SYNTAX: GRAPH
* [GRAPH] KEY IS PRESSED.
* ANY GRAPHICS STATEMENT THAT CHANGES THE GRAPHICS DISPLAY IS EXECUTED.

NOTES:

ALPHA MODE IS SET WHEN:

* ALPHA STATEMENT IS EXECUTED
  SYNTAX: ALPHA
* ANY ALPHANUMERIC KEY IS PRESSED (EXCEPT IN GRAPHICS INPUT).
* DISP STATEMENTS ARE EXECUTED OR ALPHA DISPLAY INPUT IS REQUIRED.

G R A P H I C S

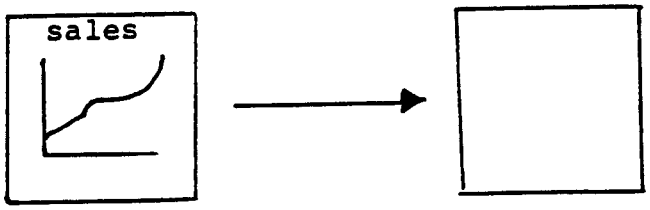
GCLEAR

PURPOSE: CLEARS THE GRAPHICS DISPLAY.

SNYTAX: GCLEAR [Y-COORDINATE]

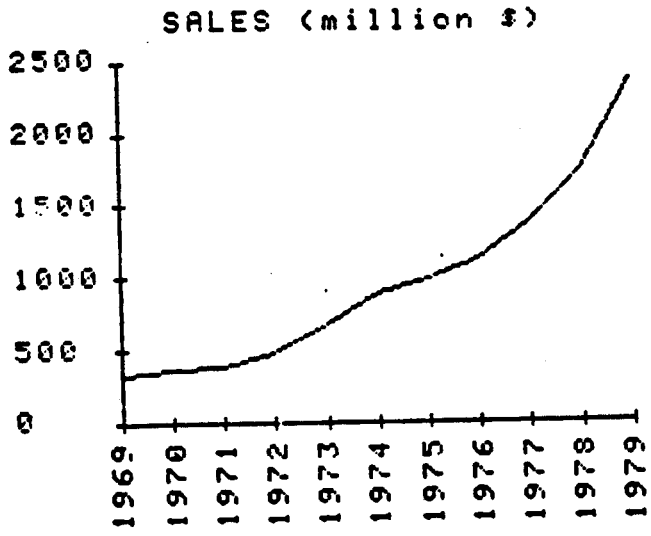
RULE: OPTIONAL PARAMETER ENABLES PARTIAL CLEARING OF GRAPHICS DISPLAY FROM INDICATED Y-VALUE TO BOTTOM OF DISPLAY.

NOTES:



HEWLETT PACKARD

GRAPHICS



NOTES:

INTRODUCTION TO GRAPHICS

EXECUTE THE FOLLOWING STATEMENTS:

```
GCLEAR
SCALE 0,2*PI, -1,1
XAXIS 0, PI/4
YAXIS 0, .5
MOVE 0,0
FOR X=0 TO 7 STEP PI/2 @ DRAW X, SIN(X) @ NEXT X
```

NOTES:

S C A L E

PURPOSE:

TO SCALE DISPLAY TO
SPECIFIED UNITS.

SYNTAX:

SCALE X MIN, X MAX, Y MIN, Y MAX

EXAMPLES:

```
250 SCALE 1966, 1980, -700, 3000
260 SCALE 413 *(-10), 413*10, -10, 10
270 SCALE 0,255,0,191
```

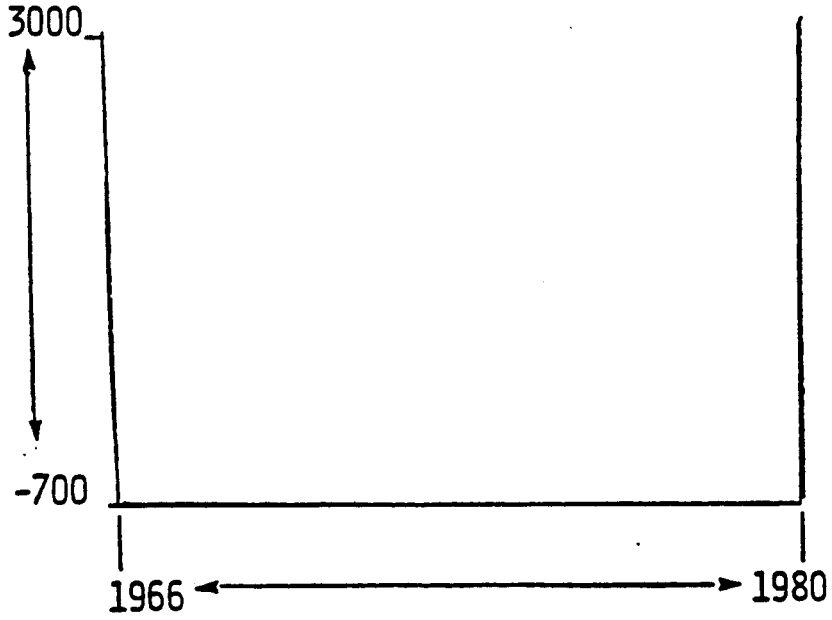
NOTES:

* DEFAULT VALUES: 0,100,0,100
* SCREEN SIZE: 256 DOTS X 192 DOTS (RATIO 4:3)
* FOR EQUAL UNIT SCALING: #OF X UNITS = 4/3 *# OF Y UNITS

HEWLETT hp PACKARD

SCALE EXAMPLE

SCALE 1966, 1980, -700, 3000



NOTES:

X A X I S / Y A X I S

PURPOSE:

DRAWS SCALED AXES WITH TIC MARKS BETWEEN SPECIFIED LIMITS

SYNTAX:

```
XAXIS Y-INTERCEPT [,TIC[, X MIN, X MAX]]
YAXIS X-INTERCEPT [,TIC[, Y MIN, Y MAX]]
```

EXAMPLES:

```
XAXIS 0, 1, 1969, 1979
YAXIS 1969, 500, 0, 2500
```

NOTES:

GRAPHICS EXAMPLE

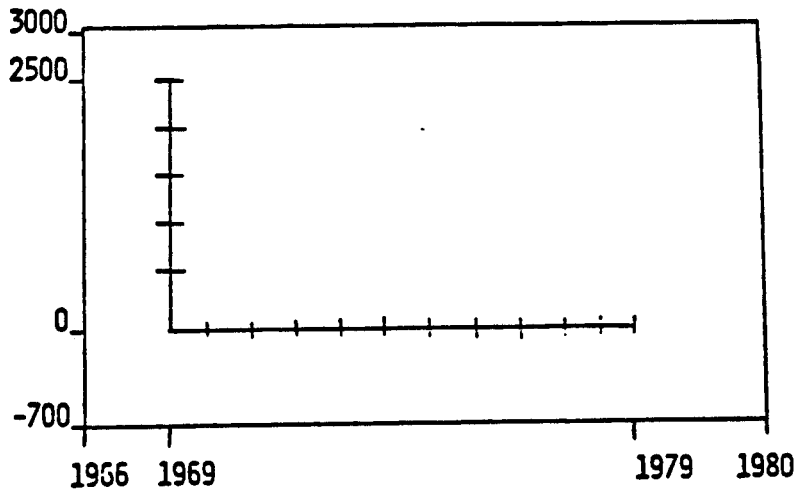
```
10  GCLEAR
20  SCALE -10,10, -2,2
30  XAXIS 0,1
40  YAXIS 0,.5
50  COPY
60  END
```

NOTES:

HEWLETT hp PACKARD

XAXIS & YAXIS EXAMPLE

```
SCALE 1966, 1980, -700, 3000
XAXIS 0,1, 1969, 1979
YAXIS 1969, 500, 0, 2500
```



NOTES:

M O V E

PURPOSE:

TO MOVE THE PEN TO THE SPECIFIED POINT ON THE GRAPH WITHOUT DRAWING A LINE.

SYNTAX:

MOVE X-COORDINATE , Y-COORDINATE

NOTES:

D R A W

PURPOSE:

TO DRAW A LINE FROM THE CURRENT PEN POSITION TO THE SPECIFIED POINT.

SYNTAX:

DRAW X-COORDINATE , Y-COORDINATE

EXAMPLES:

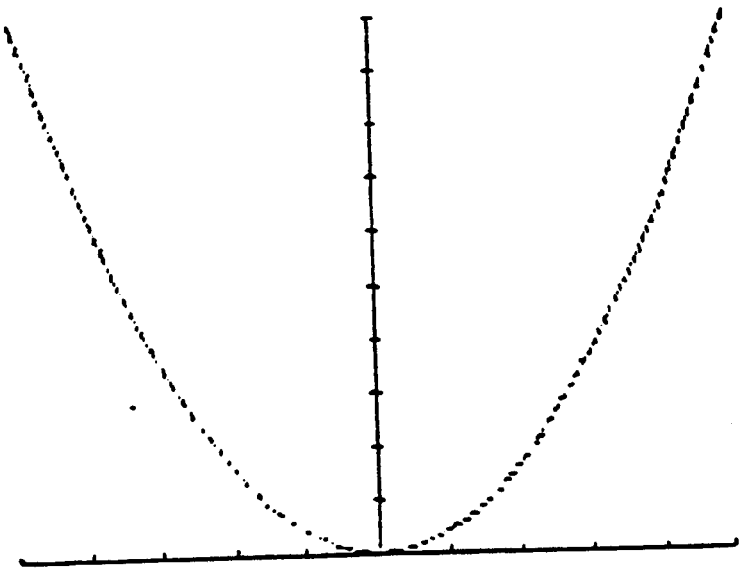
```
MOVE 1974, 1000
MOVE X, X ^ 2
DRAW 1975, 1300
DRAW X, SIN (X)
```

NOTES:

MOVE/DRAW EXAMPLE

```
10  REM X^2 FUNCTION
20  GCLEAR
30  SCALE -10,10,0,100
40  XAXIS 0,2
50  YAXIS 0,10
60  MOVE -10, (-10)^2
70  FOR X = -10 TO 10
80  DRAW X, X^2
90  NEXT X
100 END
```

NOTES:



PROBLEM

FLOWCHART AND WRITE A PROGRAM TO DRAW A GRAPH OF SIN(X) FROM X= -180° TO X= 180.° BE SURE TO INCLUDE THE DEG STATEMENT TO SET DEGREES MODE FOR THE TRIGONOMETRIC FUNCTIONS.

NOTE: THE VALUES OF SIN(X) RUN FROM -1 TO 1.

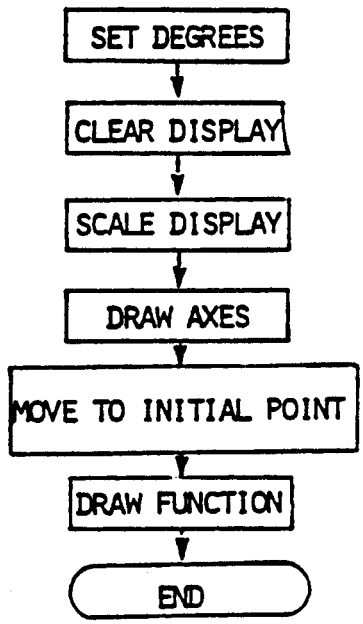
NOTES:

PROBLEM SOLUTION

SIN(X)

```
10  REM SOLUTION TO PROBLEM # 20
20  DEG
30  GCLEAR
40  SCALE -180, 180, -1, 1
50  XAXIS 0, 90
60  YAXIS 0, .5
70  MOVE -180, SIN (-180)
80  FOR X=180 TO 180
90  DRAW X, SIN (X)
100 NEXT X
110 END
```

FLOWCHART



HEWLETT hp PACKARD

LABEL

PURPOSE:

TO PUT LABELS OR TEXT ON PLOTS AT CURRENT PEN POSITION

SYNTAX:

LABEL STRING EXPRESSION

EXAMPLES:

```
LABEL "H-P SALES (MILLION $)
LABEL VAL$ (X)
```

NOTES:

L D I R

PURPOSE:

SETS ORIENTATION FOR LABEL

SYNTAX:

LDIR NUMERIC EXPRESSION

EXAMPLE:

```
80 DEG
90 LDIR 90
```

RULE:

>= 45, VERTICAL

< = 45, HORIZONTAL

NOTES:

LABEL EXAMPLE

```
10  REM
20  GCLEAR
30  SCALE-20 ,120 - 20 ,120
40  XAXIS  0,5, 0 ,100 @ YAXIS 0,5 , 0, 100
50  FOR I = 0 TO 100 STEP 10
60  LDIR 90
70  MOVE I + 3,  -10
80  LABEL VAL$ (I)
90  LDIR Ø
100 MOVE -10, I
110 LABEL VAL$(I)
120 NEXT I
130 END
```

NOTES:

PROBLEM #21

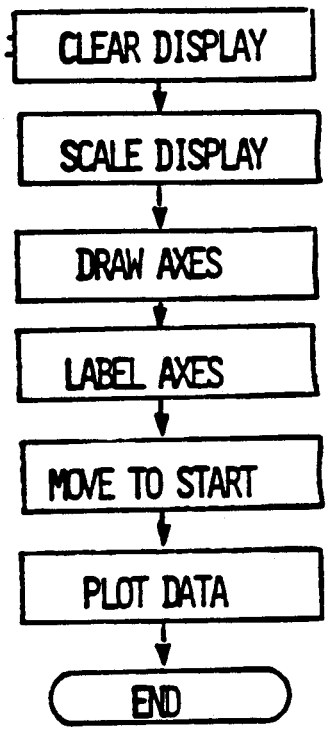
FLOWCHART AND WRITE A PROGRAM TO GRAPHICALLY REPRESENT THE FOLLOWING TABLE OF DATA. PLOT DOLLARS ON THE Y-AXIS, YEARS ON THE X-AXIS. BE SURE TO LABEL THE GRAPH APPROPRIATELY.

NOTES:

HP SALES

YEAR	MILLION$
1969	336
1970	365
1971	378
1972	483
1973	669
1974	893
1975	985
1976	1121
1977	1368
1978	1737
1979	2361

SOLUTION TO PROBLEM




```
10  REM *HP SALES (1969-1979)
20  GCLEAR
30  SCALE 1966, 1980, -700, 3000
40  XAXIS 0,1, 1969, 1979
50  YAXIS 1969, 500, 0, 2500
60  ! ********** LABEL AXES
70  DEG
80  LDIR 90
90  FOR X=1969 TO 1979
100 MOVE X + .2, -700
110 LABEL VAL$(X)
120 NEXT X
130 LDIR 0
140 FOR Y=0 TO 2500 STEP 500
150 MOVE 1967, Y-50
160 LABEL VAL$(Y)
170 NEXT Y
180 MOVE 1970, 2700
190 LABEL "SALES (MILLION $)"
200 ! ********** PLOT DATA
210 MOVE 1969, 336
220 FOR X=1970 TO 1979
230 READ Y
240 DRAW X,Y
250 NEXT X
260 DATA 365, 378, 483, 669, 893, 985
    ,1121, 1368, 1737, 2361
270 END
```

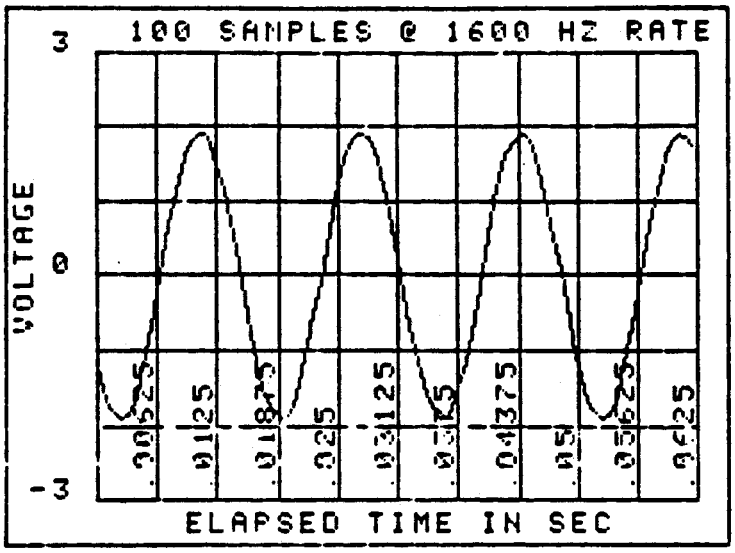
INPUT IN GRAPHICS MODE (OPTIONAL)

1. THE HP-85 WILL ACCEPT INPUT FROM THE KEYBOARD WHILE IN GRAPHICS MODE. HOWEVER, ONLY THE "BACK SPACE" KEY IS ACTIVE, NOT THE "ARROW" KEYS.
2. THE INPUT PROMPT (?) WILL OCCUR AT THE PREVIOUS PEN POSITION.
3. THE EQUIVALENT OF "DISP" IS "LABEL" (+ USE OF "GCLEAR Y").
4. FOR INPUT IN ALPHA MODE, USE 'ALPHA' AND 'GRAPH' ALTERNATLEY (OR USE 'DISP' AND 'LABEL' APPROPRIATELY).

NOTES:

Graphics Statements

ALPHA	Page 197
BPLOT character string, number of characters per line	Page 237
DRAW x-coordinate , y-coordinate	Page 211
GCLEAR [y]	Page 199
GRAPH	Page 197
IDRAW x-increment , y-increment	Page 217
IMOVE x-increment , y-increment	Page 217
LABEL character string	Page 221
LDIR numeric expression	Page 224
MOVE x-coordinate , y-coordinate	Page 211
PEN numeric expression	Page 207
PENUP	Page 207
PLOT x-coordinate , y-coordinate	Page 208
SCALE xmin , xmax , ymin , ymax	Page 199
XAXIS y-intercept [, tic length [, xmin , xmax]]	Page 202
YAXIS x-intercept [, tic length [, ymin , ymax]]	Page 202




```
130 !
140 ! FIND MAX/MIN VOLTAGES"
150 !
160 V=CEIL(AMAX(D))
165 IF V>=1 THEN V=V+1
170 !
180 !
190 GCLEAR
200 SCALE 0,1.2*N,-1.2*V,1.2*V
210 !
220 ! BORDER FRAME
230 XAXIS -1.2*V @ XAXIS 1.2*V
240 YAXIS 0 @ YAXIS 1.2*N
250 !
260 ! X AXIS SETUP
270 !
280 IF V<1 THEN B=.1 ELSE B=1
290 IF N<=200 THEN B2=10 ELSE B2=100
300 !
310 FOR I=V TO -V STEP -B
320 XAXIS I,0,.15*N,1.15*N
330 NEXT I
340 !
350 ! Y AXIS SETUP
360 !
370 FOR I=.15*N TO 1.15*N STEP B2
380 YAXIS I,B,-V,V
390 NEXT I
400 !
410 ! Y AXIS LABELING
420 !
430 LDIR 0
440 FOR I=V TO -V STEP -V
450 MOVE .04*N,I @ LABEL I
460 NEXT I
470 MOVE .05*N,-V/4
480 LDIR 90 @ LABEL "VOLTAGE"
490 !
500 ! X AXIS LABELING
510 !
520 MOVE .3*N,-1.15*V @ LDIR 0
530 LABEL "ELAPSED TIME IN SEC"
540 C=1 @ LDIR 90
550 FOR I=.25*N TO 1.15*N STEP B2
560 MOVE I,-1.05*V
570 LABEL B2*W*C
580 C=C+1 @ NEXT I
590 !
600 ! PLOTTING
610 !
620 FOR I=1 TO N
630 PLOT I-1+.15*N,D(I)
640 NEXT I
650 MOVE .17*N,1.05*V @ LDIR 0
660 LABEL N,"SAMPLES @",1/W,"HZ RATE"
670 END
```

BIT FUNCTION

BIT (<NUM EXP>,<BIT POSITION>)

Numeric expression agrument range: -32768 to 32767

Bit position argument range: 0 to 15

NOTES:

BIT FUNCTION: returns a one or zero indicating the value of the bit in the numeric expression specified by the bit position parameter.

HEXADECIMAL TO DECIMAL

HTD (<string exp>)

DECIMAL TO HEXDECIMAL

DTH$ (<num exp>)

Decimal Argument Range: -32767

Hexadecimal Argument Range: "8000" to "7FFF"

NOTES:

HEXADECIMAL TO DECIMAL: return decimal equivalent of the hexadecimal number represented by the string expression.

DECIMAL TO HEXADECIMAL: returns a string, 4 characters long, representing the hexadecimal equivelent of the decimal integer given.

Hexadecimal arguments or hexadecimal results are string expressions. Hex arguments don't have to be 4 characters long. For example:

HTD("F") returns the value 15.

Hex results on the other hand are always 4 character strings. For example:

DTH$(31) returns "001F".

BINARY TO DECIMAL

BTD(<string exp>)

DECIMAL TO BINARY

DTB$(<num exp>)

Decimal Argument Range: -32768 to 32767

Binary Argument Range: "1000000000000000" to "0111111111111111"

ie: 16 bit 2's complement

NOTES:

BINARY TO DECIMAL: returns decimal equivalent of the binary number represented by the string expression.

DECIMAL TO BINARY: returns a string, 16 characters, representing the binary equivalent of the decimal integer given.

Stress that binary arguments or binary results are string expressions. Binary arguments don't have to be 16 characters long. For example:

BTD("111") will return a value of 7.

The binary result on the other-hand is always 16 characters long. For example:

DTB$(7) gives the string "0000000000000111"

OCTAL TO DECIMAL
OTD (<string exp>)

DECIMAL TO OCTAL
DTO$ (<num exp>)

Decimal Argument Range: -32768 to 32767

Octal Argument Range: "100000" to "077777"

NOTES:

OCTAL TO DECIMAL: returns decimal equivalent of the octal number represented by the string expression.

DECIMAL TO OCTAL: returns a string, 6 characters long, representing the octal equivalent of the decimal integer given.

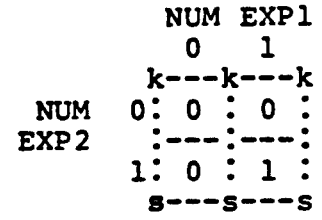
Octal arguments or octal results are string expressions. Octal arguments don't have to be 6 characters long, but octal results are always 6 characters.

BINARY ANDing

BINAND(<NUM EXP1>,<NUM EXP2>)

for example: BINAND (7,5)
yields 5

```
0...0111
0...0101
--------
0...0101
```



NOTES:

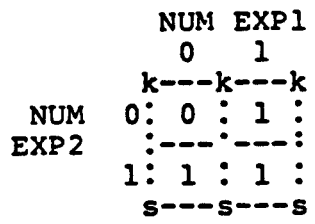
Numeric expressions 1 and 2 are ANDed bit-by-bit returning an integer result.

BINARY INCLUSIVE ORing

BINIOR (<NUM EXP1>,<NUM EXP2>)

for example: BINIOR (7,5)
yields 7

```
0...0111
0...0101
--------
0...0111
```



Numeric expression argument range: -32768 to 32767

NOTES:

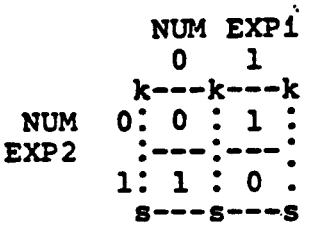
Numeric experssions 1 and 2 are inclusively ORed bit-by-bit returning an integer result.

BINARY EXCLUSIVE ORing

BINEOR (<NUM EXP1>,<NUM EXP2>)

for example: BINEOR (7,5)
yields 2

```
0...0111
0...0101
--------
0...0010
```



NOTES: Numeric expressions 1 and 2 are exclusively ORed bit-by-bit returning an integer result.

Numeric expression argument range: -32768 to 32767

BINARY COMPLEMENT

BINCMP (<NUM EXP>)

for example: BINCMP(5)
yields -6

```
0...0101
--------
1...1010
```

```
NUM EXP
 0   1
k---k---k
: 1 : 0 :
s---s---s
```

NOTES:

Returns the binary one's complement of the numeric expression argument. All zeros are changed to ones and all ones are changed to zeros.

Numeric Argument is . er from -32768 to 32767

Summary of OUTPUT Image Specifiers

Image	Meaning
A	Output one string character
B	Output number as one 8-bit byte
C	Output a comma separator in a number
D	Output one digit character; blank for leading zero
E	Output exponent information; five characters
e	Output exponent information; four characters
K	Output a variable in free-field format
M	Output number's sign if negative, blank if positive
P	Output a period separator in a number
R	Output a European radix point (comma)
S	Output number's sign, plus or minus
W	Output number as two 8-bit bytes (16-bit word)
X	Output one blank
Z	Output one digit character, including leading zeros
"..."	Output a literal
#	Suppress end-of-line sequence at end of statement
*	Output one digit character; asterick for leading zero
.	Output an American radix point (decimal point)
/	Output an end-of-line sequence

Summary of ENTER Image Specifiers

Image	Meaning
A	Demands one string character
B	Enter number as one 8-bit byte
C	Demand one character for a numeric field; allows commas to be skipped over
D	Demand one character for a numeric field
E	Demand five characters for a numeric field
e	Demand four characters for a numeric field
K	Enter a variable in free-field format
M	Demand one character for a numeric field
S	Demand one character for a numeric field
W	Enter number as two 8-bit bytes (16-bit word)
X	Skip one character
Z	Demand one character for a numeric field
#	Suppress requirement for a line-feed to terminate statement or field
%	Allow EOI to terminate statement or field
*	Demand one character for a numeric field
.	Demand one character for a numeric field
/	Demand a line-feed

ASCII CHART

HP-IB		ASCII	Decimal	Binary	Octal	Hexa-decimal
Addressed Command Group ACG	GTL	NUL	0	00 000 000	000	00
		SOH	1	00 000 001	001	01
		STX	2	00 000 010	002	02
		ETX	3	00 000 011	003	03
	SDC	EOT	4	00 000 100	004	04
	PPC	ENQ	5	00 000 101	005	05
		ACK	6	00 000 110	006	06
		BEL	7	00 000 111	007	07
	GET	BS	8	00 001 000	010	08
	TCT	HT	9	00 001 001	011	09
		LF	10	00 001 010	012	0A
		VT	11	00 001 011	013	0B
		FF	12	00 001 100	014	0C
		CR	13	00 001 101	015	0D
		SO	14	00 001 110	016	0E
		SI	15	00 001 111	017	0F
Universal Command Group UCG	LLO	DLE	16	00 010 000	020	10
		DC1	17	00 010 001	021	11
		DC2	18	00 010 010	022	12
		DC3	19	00 010 011	023	13
	DCL	DC4	20	00 010 100	024	14
	PPU	NAK	21	00 010 101	025	15
		SYN	22	00 010 110	026	16
		ETB	23	00 010 111	027	17
	SPE	CAN	24	00 011 000	030	18
	SPD	EM	25	00 011 001	031	19
		SUB	26	00 011 010	032	1A
		ESC	27	00 011 011	033	1B
		FS	28	00 011 100	034	1C
		GS	29	00 011 101	035	1D
		RS	30	00 011 110	036	1E
		US	31	00 011 111	037	1F
Listen Address Group LAG Note 1	L0	SP	32	00 100 000	040	20
	L1	!	33	00 100 001	041	21
	L2	"	34	00 100 010	042	22
	L3	#	35	00 100 011	043	23
	L4	$	36	00 100 100	044	24
	L5	%	37	00 100 101	045	25
	L6	&	38	00 100 110	046	26
	L7	'	39	00 100 111	047	27
	L8	(	40	00 101 000	050	28
	L9	)	41	00 101 001	051	29
	L10	*	42	00 101 010	052	2A
	L11	+	43	00 101 011	053	2B
	L12	,	44	00 101 100	054	2C
	L13	-	45	00 101 101	055	2D
	L14	.	46	00 101 110	056	2E
	L15	/	47	00 101 111	057	2F
	L16	0	48	00 110 000	060	30
	L17	1	49	00 110 001	061	31
	L18	2	50	00 110 010	062	32
	L19	3	51	00 110 011	063	33
	L20	4	52	00 110 100	064	34
	L21	5	53	00 110 101	065	35
	L22	6	54	00 110 110	066	36
	L23	7	55	00 110 111	067	37
	L24	8	56	00 111 000	070	38
	L25	9	57	00 111 001	071	39
	L26	:	58	00 111 010	072	3A
	L27	;	59	00 111 011	073	3B
	L28	<	60	00 111 100	074	3C
	L29	=	61	00 111 101	075	3D
	L30	>	62	00 111 110	076	3E
	UNL	?	63	00 111 111	077	3F

HP-IB		ASCII	Decimal	Binary	Octal	Hexa-decimal
Talk Address Group TAG Note 2	T0	@	64	01 000 000	100	40
	T1	A	65	01 000 001	101	41
	T2	B	66	01 000 010	102	42
	T3	C	67	01 000 011	103	43
	T4	D	68	01 000 100	104	44
	T5	E	69	01 000 101	105	45
	T6	F	70	01 000 110	106	46
	T7	G	71	01 000 111	107	47
	T8	H	72	01 001 000	110	48
	T9	I	73	01 001 001	111	49
	T10	J	74	01 001 010	112	4A
	T11	K	75	01 001 011	113	4B
	T12	L	76	01 001 100	114	4C
	T13	M	77	01 001 101	115	4D
	T14	N	78	01 001 110	116	4E
	T15	O	79	01 001 111	117	4F
	T16	P	80	01 010 000	120	50
	T17	Q	81	01 010 001	121	51
	T18	R	82	01 010 010	122	52
	T19	S	83	01 010 011	123	53
	T20	T	84	01 010 100	124	54
	T21	U	85	01 010 101	125	55
	T22	V	86	01 010 110	126	56
	T23	W	87	01 010 111	127	57
	T24	X	88	01 011 000	130	58
	T25	Y	89	01 011 001	131	59
	T26	Z	90	01 011 010	132	5A
	T27	[	91	01 011 011	133	5B
	T28	\	92	01 011 100	134	5C
	T29	]	93	01 011 101	135	5D
	T30	^	94	01 011 110	136	5E
	UNT	_	95	01 011 111	137	5F
Secondary Command Group SCG Note 3	S0	`	96	01 100 000	140	60
	S1	a	97	01 100 001	141	61
	S2	b	98	01 100 010	142	62
	S3	c	99	01 100 011	143	63
	S4	d	100	01 100 100	144	64
	S5	e	101	01 100 101	145	65
	S6	f	102	01 100 110	146	66
	S7	g	103	01 100 111	147	67
	S8	h	104	01 101 000	150	68
	S9	i	105	01 101 001	151	69
	S10	j	106	01 101 010	152	6A
	S11	k	107	01 101 011	153	6B
	S12	l	108	01 101 100	154	6C
	S13	m	109	01 101 101	155	6D
	S14	n	110	01 101 110	156	6E
	S15	o	111	01 101 111	157	6F
	S16	p	112	01 110 000	160	70
	S17	q	113	01 110 001	161	71
	S18	r	114	01 110 010	162	72
	S19	s	115	01 110 011	163	73
	S20	t	116	01 110 100	164	74
	S21	u	117	01 110 101	165	75
	S22	v	118	01 110 110	166	76
	S23	w	119	01 110 111	167	77
	S24	x	120	01 111 000	170	78
	S25	y	121	01 111 001	171	79
	S26	z	122	01 111 010	172	7A
	S27	{	123	01 111 011	173	7B
	S28	|	124	01 111 100	174	7C
	S29	}	125	01 111 101	175	7D
	S30	~	126	01 111 110	176	7E
	S31	DEL	127	01 111 111	177	7F

Commands

Non-Programmable

AUTO [beginning line number [, increment value]]	Page 80
CONT [statement number]	Page 98
DELETE first statement number [, last statement number]	Page 95
INIT	Page 99
LOAD program name	Page 179
REN [first statement number [, increment value]]	Page 96
RUN [statement number]	Page 99
SCRATCH	Page 78
STORE program name	Page 176
UNSECURE file name , security code , secure type	Page 194

Programmable

CAT	Page 175
COPY	Page 35
CTAPE	Page 282
ERASETAPE	Page 175
FLIP	Page 34
LIST [beginning statement number [, ending statement number]]	Page 97
PLIST [beginning statement number [, ending statement number]]	Page 97
PRINT ALL	Page 35
REWIND	Page 280
SECURE file name , security code , secure type	Page 193

BASIC Statements

ASSIGN# buffer number TO file name — Page 183
ASSIGN# buffer number TO * — Page 184
BEEP [tone, duration] — Page 89
CHAIN file name — Page 179
CLEAR — Page 19
COM common variable list — Page 123
CRT IS output code number — Page 169
CREATE file name , number of records [, number of bytes per record] — Page 180
DATA data list — Page 137
DEFAULT OFF — Page 70
DEFAULT ON — Page 70
DEF FN numeric variable name [(parameter)][= numeric expression] — Page 145
DEF FN string variable name [(parameter)][= string expression] — Page 145
DEG — Page 66
DIM dimension list — Page 121
DISP display list — Page 84
DISP USING image format string [; disp using list] — Page 167
DISP USING statement number [; disp using list] — Page 161
END — Page 77
FN END — Page 147
FOR loop counter = initial value TO final value [STEP increment value] — Page 111
GOSUB statement number — Page 151
GOTO statement number — Page 91
GRAD — Page 66
IF numeric expression THEN statement number [ELSE statement number] — Page 106
or — or
executable statement — executable statement
IMAGE image format string — Page 161
INPUT variable name$_1$ [, variable name$_2$ ...] — Page 87
INTEGER numeric variable [(subscripts)][, numeric variable] (subscripts)] ...] — Page 122
KEY LABEL — Page 154
[LET] numeric variable$_1$ [, numeric variable$_2$ ...] = numeric expression — Page 90
[LET] string variable$_1$ [, string variable$_2$ ...] = string expression — Page 90
[LET] FN variable name = expression — Page 147
LOAD BIN file name — Page 193
NEXT loop counter — Page 111
NORMAL — Pages 35, 80

BASIC Statements

OFF ERROR — Page 261
OFF KEY# key number — Page 156
OFF TIMER# timer number — Page 157
ON ERROR GOSUB statement number — Page 261
ON ERROR GOTO statement number — Page 261
ON numeric expression GOSUB statement number list — Page 153
ON numeric expression GOTO statement number list — Page 108
ON KEY# key number [, key label] GOSUB statement number — Page 154
ON KEY# key number [, key label] GOTO statement number — Page 154
ON TIMER# timer number , milliseconds GOSUB statement number — Page 156
ON TIMER# timer number , milliseconds GOTO statement number — Page 156
OPTION BASE 1 or 0 — Page 121
PAUSE — Page 99
PRINT [print list] — Page 86
PRINT# buffer number ; print # list — Page 185
PRINT# buffer number , record number [; print # list] — Page 188
PRINT USING image format string [; print using list] — Page 167
PRINT USING statement number [; print using list] — Page 161
PRINTER IS output code number — Page 169
PURGE file name [, purge code number] — Page 192
RAD — Page 66
RANDOMIZE [numeric expression] — Page 64
READ variable name$_1$ [, variable name$_2$ ...] — Page 137
READ# buffer number ; variable list — Page 187
READ# buffer number , record number [; variable list] — Page 190
REAL numeric variable [(subscripts)][, numeric variable [(subscripts)] ...] — Page 122
REM [any combination of characters] — Page 83
RENAME old file name TO new file name — Page 192
RESTORE [statement number] — Page 139
RETURN — Page 151
SETTIME seconds since midnight , Julian day in form yyddd — Page 56
SHORT numeric variable [(subscripts)][, numeric variable [(subscripts)] ...] — Page 122
STOP — Page 77
STORE BIN file name — Page 193
TRACE — Page 255
TRACE ALL — Page 256
TRACE VAR variable$_1$ [, variable$_2$ ...] — Page 255
WAIT number of milliseconds — Page 100

Graphics Statements

ALPHA	Page 197
BPLOT character string, number of characters per line	Page 237
DRAW x-coordinate , y-coordinate	Page 211
GCLEAR [y]	Page 199
GRAPH	Page 197
IDRAW x-increment , y-increment	Page 217
IMOVE x-increment , y-increment	Page 217
LABEL character string	Page 221
LDIR numeric expression	Page 224
MOVE x-coordinate , y-coordinate	Page 211
PEN numeric expression	Page 207
PENUP	Page 207
PLOT x-coordinate , y-coordinate	Page 208
SCALE xmin , xmax , ymin , ymax	Page 199
XAXIS y-intercept [, tic length [, xmin , xmax]]	Page 202
YAXIS x-intercept [, tic length [, ymin , ymax]]	Page 202

BASIC Predefined Functions

ABS(X)	Absolute value of X.	Page 60
ACS(X)	Arcosine of X, in 1st or 2nd quadrant.	Page 66
ASN(X)	Arcsine of X, in 1st or 4th quadrant.	Page 66
ATN(X)	Arctangent of X, in 1st or 4th quadrant.	Page 66
ATN2(Y,X)	Arctangent of Y/X, in proper quadrant.	Page 67
CEIL(X)	Smallest integer >=X.	Page 61
CHR$(X)	Character whose decimal character code is X, 0<=X <=255.	Page 131
COS(X)	Cosine of X.	Page 66
COT(X)	Cotangent of X.	Page 66
CSC(X)	Cosecant of X.	Page 66
DATE	Julian date in format yyddd (assumes system timer has been set properly).	Page 57
DTR(X)	Degree to radian conversion.	Page 67
EPS	Smallest positive machine number (1E-499).	Page 64
ERRL	Line number of latest error.	Page 261
ERRN	Number of latest error.	Page 261
EXP(X)	e^x	Page 65
FLOOR(X)	Same as INT(X) (relates to CEIL).	Page 61
FP(X)	Fractional part of X.	Page 60
INF	Largest machine number (9.99999999999E499).	Page 64
INT(X)	Largest integer <=X.	Page 61
IP(X)	Integer part of X.	Page 60
LEN(SS)	Length of string SS.	Page 128
LGT(X)	Log to the base 10 of X, X>0.	Page 65
LOG(X)	Natural logaritm, X>0.	Page 65
MAX(X,Y)	If X>Y then X, else Y.	Page 62
MIN(X,Y)	If X<Y then X, else Y.	Page 62
NUM(SS)	Decimal character code of first character of SS.	Page 132
PI	3.14159265359	Page 63
POS(S1S, S2S)	Searches string S1S for the first occurrence of string S2S. Returns starting index if found, otherwise returns 0.	Page 129
PMD(X,Y)	Remainder of X/Y: X − Y * IP(X/Y).	Page 62
RND	Next number, X, in a sequence of pseudo-random numbers, 0 < = X < 1.	Page 64
RTD(X)	Radian to degree conversion.	Page 67
SEC(X)	Secant of X.	Page 66
SGN(X)	The sign of X, −1 if X<0, 0 if X=0, and +1 if X>0.	Page 62
SIN(X)	Sine of X.	Page 66
SQR(X)	Positive square root of X.	Page 62
TAB(N)	Skips to specified column.	Page 168
TAN(X)	Tangent of X.	Page 66
TIME	Time in seconds since midnight (assumes system timer has been set properly).	Page 57
UPC$(SS)	Returns string with all lower-case alphabetic characters converted to upper-case.	Page 133
VAL(SS)	Returns the numeric equivalent of the string SS.	Page 130
VAL$(X)	String equivalent of X.	Page 131

Error Messages

Error Number	Error Condition	Default values (errors 1-8 only) with DEFAULT ON
	Math Errors (1 thru 13)	
1	Underflow: expression underflows machine	0 ±9.99999999999E499
2	Overflow: • Expression overflows machine • Attempt to store value >99999 or <−99999 in INTEGER variable. • Attempt to store value >9.9999E99 or <−9.9999E99 in SHORT variable.	±99999 ±9.9999E99
3	COT or CSC of n*180°; n=integer.	9.99999999999E499
4	TAN or SEC of n*90°; n=odd integer.	9.99999999999E499
5	Zero raised to negative power.	9.99999999999E499
6	Zero raised to zero power.	1
7	Null data: • Uninitialized string variable, or missing string function assignment. • Uninitialized numeric variable, or missing numeric function assignment.	"" 0
8	Division by zero.	±9.99999999999E499
9	Negative value raised to non-integer power.	Remaining errors are non-defaultable.
10	Square root of negative number.	
11	Argument (parameter) out of range: • ATN2(0,0). • ASN or ACSN(−1≤n≤+1). • ON expression GOTO/GOSUB; expression of range.	
12	Logarithm of zero.	
13	Logarithm of negative number.	
14	Not used.	
	System Errors (15 thru 25)	
15	System error; correct by reloading program, pressing RESET, or turning system off, then on again.	
16	Continue before run; program not allocated.	
17	FOR nesting too deep; more than 255 active FOR-NEXT loops.	
18	GOSUB nesting too deep; more than 255 nested subroutines.	

Error Number	Error Condition
19	Memory overflow: • Attempting to RUN a program that requires more than given memory. • Attempting to edit too large a program; delete a nonexisting line to deallocate program, then edit. • Attempting to load a program larger than available memory. • Attempting to open a file with no available buffer space. • Attempting any operation that requires more memory than available. • Attempting to load or run a large program after a ROM has been installed. ROMs use up a certain amount of memory. Refer to the appropriate ROM manual.
20	Not used.
21	ROM missing; attempting to RUN program that requires ROM. An attempt to edit program with missing ROM will usually SCRATCH memory.
22	Attempt to edit, list, store, or overwrite a SECUREd program.
23	Self-test error; system needs repair.
24	Too many (more than 14) ROMS.
25	Two binary programs; attempting to load a second binary program into memory (only one binary program allowed in memory at any time).
26 thru 29	Not used.
	Program Errors (30 thru 57)
30	OPTION BASE error: • Duplicate OPTION BASE declaration. • OPTION BASE after array declaration. • OPTION BASE parameter not 0 or 1.
31	CHAIN error; CHAIN to a program other than BASIC main program: e.g., CHAINing to a binary program.
32	COMmon variable mismatch.
33	DATA type mismatch: • READ variable and DATA type do not agree. • READ# found a string but required a number.
34	No DATA to read: • READ and DATA expired. • RESTORE executed with no DATA statement.
35	Dimensioned existing variable; attempt to dimension a variable that has been previously declared or used. Move DIM statement to beginning of program and try again.
36	Illegal dimension: • Illegal dimension in default array declaration. • Array dimensions don't agree; e.g., referencing A(2) when A(5,5) is dimensioned or referencing A(0) when OPTION BASE 1 declared.
37	Duplicate user-defined function.
38	Function definition within function definition; needs FN END.
39	Reference to a nonexistent user-defined function: • Finding FN END with no matching DEF FN. • Exiting a function that was not entered with a function call after branching to the middle of a multi-line function.

Error Number	Error Condition
40	Illegal function parameter; function parameter mismatch (e.g., declared as string, called as numeric).
41	FN=; user-defined function assignment. Function assignment does not occur between DEF FN and FN END.
42	Recursive user-defined function.
43	Numeric input wanted.
44	Too few inputs. Less items were given than requested by an INPUT statement.
45	Too many inputs. More items were given than requested by an INPUT statement.
46	NEXT missing; FOR with no matching NEXT.
47	FOR missing; NEXT with no matching FOR.
48	END statement necessary.
49	Null data; uninitialized data.
50	Binary program missing; attempting to RUN program that requires binary program. An attempt to edit will usually SCRATCH memory.
51	RETURN without GOSUB reference.
52	Illegal IMAGE format string; unrecognized character in IMAGE.
53	Illegal PRINT USING: • Data overflows IMAGE declaration. • Numeric data with string IMAGE. • String data with numeric IMAGE. • PRINT USING image format string is not correct.
54	Illegal TAB argument. With DEFAULT ON, an illegal TAB argument gives a warning message and defaults to TAB(1).
55	Array subscript out of range.
56	String variable overflow; string too big for variable.
57	Missing line; reference to a nonexistent statement number.
58 thru 59	Not used.
	Tape Errors (60 thru 75)
60	Tape cartridge is write-protected; RECORD slide tab is in left-most position.
61	Attempting to create/record more than 42 files on tape.
62	Cartridge out when attempting tape operations.
63	Duplicate file name for RENAME or CREATE.
64	Empty file; attempting to access file that was never recorded (e.g., tape was ejected *before* program was stored but *after* name was written in directory). Refer to PURGE.

Error Number	Error Condition
65	End of tape: • Tape run-off; check cartridge. • Tape is full. • Not enough space to CREATE data file.
66	File closed: • Attempting READ#/PRINT# to file that has not been opened with ASSIGN#. • Attempting to close a closed file (warning only). • Tape has been ejected and reinserted.
67	File name: • Name does not exist when attempt to LOAD, ASSIGN#, LOAD BIN, PURGE, RENAME, or SECURE. • Name not in quotes. • Attempt to PURGE an open file.
68	File type mismatch: • Attempting to treat program as data file, or vice versa. • Attempting to treat binary program as BASIC main program file, or vice versa. • Attempting to treat data as binary program, or vice versa.
69	Random overflow; attempting to READ#/PRINT# beyond existing number of bytes in logically-defined record with random file access.
70	READ error; system cannot read tape.
71	End-of-File; no data beyond EOF mark in data file.
72	Record: • Attempting to READ#/PRINT# to record that doesn't exist; e.g., READ# 1,120 when only 100 records in file. • Attempting to READ#/PRINT# at end of file. • Lost in record: close file to release buffer.
73	Searches and does not find: • Bad tape cartridge; may have been exposed to magnetic field. • Cannot find directory, tape may need to be initialized.
74	Stall; either bad tape cartridge or transport problem, refer to Tape Operations, appendix B.
75	Not an HP-85 file; cannot read.
76 thru 79	Not used.
	Syntax Errors (80 thru 92)
80	Right parentheses,), expected.
81	Bad BASIC statement or bad expression. If it is an expression, try it again with DISP <expression> to get a better error message.
82	String expression error; e.g., right quote missing or null string given for file name.
83	Comma missing or more parameters expected (separated by commas).
84	Excess characters; delete characters at end of good line, then press END LINE.
85	Expression too big for system to interpret.

Error Number	Error Condition
86	Illegal statement after THEN
87	Bad DIM statement.
88	Bad statement: • CCM in calculator mode. • User-defined function in calculator mode. • INPUT in calculator mode.
89	Invalid parameter: • ON KEY# less than 1 or greater than 8. • Attempt to TRACE a calculator mode variable. • PRINTER IS or CRT IS with invalid parameter. • CREATE with invalid parameters. • ASSIGN#, PRINT#, or READ# with buffer number other than 1 through 10. • Random READ# to record 0. • SETTIME with illegal time parameter. • ON TIMER#, OFF TIMER# with number other than 1, 2, or 3. • SCALE with invalid parameters. • AUTO or REN with invalid parameters. • LIST with invalid parameters. • DELETE with invalid parameters. • VAL$ with non-numeric parameter. • Any statement, command, or function for which parameters are given but they are invalid.
90	Line number too large; greater than 9999.
91	Missing parameter; e.g., DELETE with missing or invalid parameters.
92	Syntax error. Cursor returns to character where error was found.

Error No.	Message	Meaning	Possible Corrective Action
101	XFR	This is only a warning. It is issued when a program is paused with an I/O TRANSFER still active. Do not attempt to modify a program when a TRANSFER is active.	Before you modify or rerun the program, stop all active transfers with a RESET, HALT, or ABORTIO instruction; or press the RESET key.
110	I/O CARD	An interface has failed self-test. This indicates a probable hardware problem.	ERRSC can be used to determine which interface has failed. Try recycling the power (turn computer off, then back on again). If the interface still fails, contact the authorized HP-85 dealer or the HP sales and service office from which you purchased your HP-85.
111	I/O OPER	The I/O operation attempted is not valid with the type of interface being used. Some examples are: specifying a status or control register that does not exist, using a primary address with an RS-232 interface, or using an I/O statement that is not defined for the interface being used.	ERRL can be used to identify the improper statement. Check this statement in the Syntax Reference section to determine if it is defined for the interface being used. If the statement is valid, check the appropriate Interface Programming section to get details on the proper mode or configuration required for the statement used.
112	I/O ROM	The I/O ROM has failed the checksum self-test. This indicates a probable hardware problem.	Try recycling the power (turn the computer off, then back on again). If the error keeps recurring, contact the authorized HP-85 dealer or the HP sales and service office from which you purchased your HP-85.

Error No.	Message	Meaning	Possible Corrective Action
113		An interface-dependent error. HP-IB: The statement used requires the interface to be system controller. Serial: UART receiver overrun; data has been lost. BCD: Attempting to put the interface into an illegal mode. GPIO: An odd number of bytes was transferred when the interface was configured for 16-bit words.	ERRSC can be used to determine the source of the error. Refer to the appropriate Interface Programming section to get details on the error and possible corrective actions.
114		An interface-dependent error. HP-IB: The statement used requires the interface to be active controller. Serial: Receiver buffer overrun; data has been lost. BCD: Port 10 not currently available. GPIO: FHS TRANSFER aborted by ST0.	ERRSC can be used to determine the source of the error. Refer to the appropriate Interface Programming section to get details on the error and possible corrective actions.
115		An interface-dependent error. HP-IB: The statement used requires the interface to be addressed to talk. Serial: Automatic disconnect forced. BCD: FHS TRANSFER aborted by FLGB. GPIO: Interface configuration does not allow an output enable or output operation on Port A or Port B.	ERRSC can be used to determine the source of the error. Refer to the appropriate Interface Programming section to get details on the error and possible corrective actions.
116		An interface-dependent error. HP-IB: The statement used requires the interface to be addressed to listen. Serial: This error number not currently used. BCD: Data direction mismatch on current operation. GPIO: Cannot start operation because handshake CTL line is not in proper state.	ERRSC can be used to determine the source of the error. Refer to the appropriate Interface Programming section to get details on the error and possible corrective actions.

Error No.	Message	Meaning	Possible Corrective Action
117		An interface-dependent error. HP-IB: The statement used requires the interface to be non-controller. Serial: This error number not currently used. BCD: Interface command has been directed to a non-existent field. GPIO: This error number not currently used.	ERRSC can be used to determine the source of the error. Refer to the appropriate Interface Programming section to get details on the error and possible corrective actions.
118		An interface-dependent error. HP-IB: This error number not currently used. Serial: This error number not currently used. BCD: Cannot start operation because CTL line is not in the proper state. GPIO: This error number not currently used.	ERRSC can be used to determine the source of the error. Refer to the appropriate Interface Programming section to get details on the error and possible corrective actions.
119		An interface-dependent error. HP-IB: This error number not currently used. Serial: This error number not currently used. BCD: Data format does not match the mode of the interface. GPIO: This error number not currently used.	ERRSC can be used to determine the source of the error. Refer to the appropriate Interface Programming section to get details on the error and possible corrective actions.
120		An interface-dependent error. This error number not currently used.	
121		An interface-dependent error. This error number not currently used.	
122		An interface-dependent error. This error number not currently used.	
123	NO ";"	Syntax error. A semicolon delimiter was expected in the statement.	Put the semicolon where it belongs.

Error No.	Message	Meaning	Possible Corrective Action
124	BAD	Either the interface select code specified is out of range, or there is no interface present set to the specified select code. Interface select codes must be in the range of 3 thru 10. Select codes 1 (CRT) and 2 (internal printer) are allowed for OUTPUT statements only.	Be sure that the interface select code is within the proper range. Pay special attention to variables that are used to hold interface select codes. If the interface select code is OK, be sure that the interface is plugged in properly. Finally, check the switch settings on the interface. (Someone might have changed them last weekend.)
125	ADDR	The primary address specified is improper. Only addresses 00 thru 31 are allowed, but not all interfaces use this entire range.	Be sure that the primary address is within the proper range. Pay special attention to variables that are used to hold addresses or device selectors.
126	BUFFER	Four possible buffer problems: (1) The string variable specified has not been declared as an IOBUFFER. (2) Attempting to ENTER from a buffer which is out of data. (3) Attempting to OUTPUT to a buffer which is already full. (4) Attempting an output TRANSFER with an empty buffer.	Be sure you have included the necessary IOBUFFER statement. Check the logical flow of your program (in what order are the statements executed). Buffer contents can be examined at any time by simply printing or displaying the string variable being used as the buffer. If this doesn't provide enough information, the buffer pointers can be examined with the STATUS statement.
127	NUMBER	An incoming character sequence does not constitute a valid number, or a number being output requires three exponent digits and an "e" format was specified.	If the error is from an output operation, check the magnitude of the number and the format used. If the error is from an input operation, there are many possible causes. Here are some things to look for: more than 255 leading non-numeric characters, unexpected spaces in a character stream when a character-count format is used, punctuation sequences that include potentially numeric characters used in an order that is numerically meaningless.

Error No.	Message	Meaning	Possible Corrective Action
129	EARLY TERM	A buffer was emptied before all the ENTER fields were satisfied, or a field terminator was encountered before the specified character count was reached.	Check your incoming character stream, ENTER list, and image specifiers.
128	VAR TYPE	The type (string or numeric) of a variable in an ENTER list does not match with the image specified for that variable.	Check your ENTER list and image specifiers.
130	NO TERM	A required terminator was not received from an interface or buffer during an ENTER statement. Remember that there is a default requirement for a line-feed statement terminator.	Check your incoming character stream, ENTER list, and image specifiers.