HP-IB
Systems Training With The HP-85
As A Controller
HEWLETT PACKARD

TRAINING

ON

HP–IB

PROGRAMMING TECHNIQUES

FOR THE

HP–85
# AGENDA

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(8:30 - 5:00)</td>
<td>HP-85 Basic Usage</td>
</tr>
<tr>
<td>2</td>
<td>(8:30 - 5:00)</td>
<td>Basic Usage, Structured Programming</td>
</tr>
<tr>
<td>3</td>
<td>(8:30 - 5:00)</td>
<td>Introduction to I/O, Formatting</td>
</tr>
<tr>
<td>4</td>
<td>(8:30 - 5:00)</td>
<td>HP-IB (IEEE-488) Usage</td>
</tr>
</tbody>
</table>
| 5     | (8:30 - 2:30)| Advanced I/O, Interrupts, Buffering & Miscellaneous Topics*

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

☐ Class had adequate equipment
☐ Needed more controllers
☐ Needed more instrumentation

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 usage, 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:
HELPFUL HINTS

TEAM UP

SAVE YOUR PROGRAMS

USE STRUCTURED PROBLEM SOLVING AND PROGRAMMING TECHNIQUES
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.
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


A) Section 1 (pgs 17 - 3)
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: COMMANDS USED:

END-LINE CLEAR
BACK SPACE PRINT ALL
CURSOR KEYS NORMAL
CLEAR DISPLAY DISP"-------"
CLEAR TO END OF LINE PRINT"-------"
DELETE CHARACTERS/LINE COPY
INSERT CHARACTERS/LINE ERASETape
KEY LABEL AUTO
ROLL CAT
COPY LOAD"--", STORE"---"
LOAD/STORE PURGE"--"
RESET RESET
DISPLAY CONTROL KEYS (pgs 12, 19, 38, 39)

Move cursor to desired location without erasing characters.

- **UNSHIFT** = Clears line from cursor to end
- **SHIFT** = Clears the display

- **UNSHIFT** = Backup cursor while erasing
- **SHIFT** = Backspaces rapidly

TOGGLES BETWEEN INSERT AND REPLACE MODES.
- Insert mode inserts characters pushing right.
- Replace mode replaces character at cursor position.

- **UNSHIFT** = Delete character above the cursor.
- **SHIFT** = "Delete" command typing aid.

- Roll display up or down with respect to cursor.
- **UNSHIFT** = Roll down
- **SHIFT** = Roll up
- Put into graphic mode, showing any current graphic display. Push any alphanumeric key to exit mode.

NOTES:
EDITING KEYS

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

Possible listing devices

<table>
<thead>
<tr>
<th>CRT IS</th>
<th>END-OF-LINE</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>END-OF-LINE</td>
<td>HP-85 CRT</td>
</tr>
<tr>
<td>2</td>
<td>END-OF-LINE</td>
<td>HP-85 Printer</td>
</tr>
<tr>
<td>701</td>
<td>END-OF-LINE</td>
<td>External Printer</td>
</tr>
</tbody>
</table>

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

| PRINTER IS 701 | END-OF-LINE |

* 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??
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.

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

**STEP PAUSE**

**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??
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 Owner's Manual at 1st chance. This information must be thoroughly understood.

**NON PROGRAMMABLE COMMANDS**

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

**PROGRAMMABLE COMMANDS**

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

**STATEMENTS**

BEEP [tone,duration] - Make an audible beep  
CRT IS output code number - specify CRT select code  
END - Indicates end of program  
GOTO statement number - Branch to specified line number  
PAUSE - Pause a running program  
REM [any combination of characters] - Documentation Remarks  
WAIT number of milliseconds - Pause for specified time
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
  + Uninitialized variables initialized to zero
  - Warning message displayed on CRT
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 RADIANS MODE

NOTES:
HP-85 MODES OF OPERATION

1. Halted or Idle
   * Performs commands
   * Available for program modification / entry
   * In this 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
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

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 1/9
20 ! HP-95 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
.638977509773
.875440380566
.880946639384
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.

110Printer 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

Page B - 18
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:

<table>
<thead>
<tr>
<th>Number</th>
<th>Standard Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.000</td>
<td>15</td>
</tr>
<tr>
<td>00.23500</td>
<td>.235</td>
</tr>
<tr>
<td>-0547.9</td>
<td>-4.38415537301E-12</td>
</tr>
<tr>
<td>000987.5</td>
<td>987.5</td>
</tr>
<tr>
<td>10000^6</td>
<td>1.E24</td>
</tr>
<tr>
<td>.01E4</td>
<td>100</td>
</tr>
<tr>
<td>120E-4</td>
<td>.012</td>
</tr>
</tbody>
</table>
PRINT AND DISP DELIMITERS

There are two:

; which specifies free-field format

; which specifies compact free field format

<table>
<thead>
<tr>
<th></th>
<th>Numeric Data</th>
<th>String Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compact Field</td>
<td>Digits of the number are output, preceded by a space (if plus) or a minus sign (if minus), and followed by one space.</td>
<td>Characters of the string are output with no leading or trailing spaces.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free Field</td>
<td>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.</td>
<td>Characters of the string are output with no leading spaces and no more than 20 trailing spaces.</td>
</tr>
</tbody>
</table>

NOTES:
Free-field Format Practice

GIVEN  \( A\$ = "\text{NUMBER IS}" \)
\( A = 9999999 \)
\( B = -9999999 \)
\( C = -.123456789 \)

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:

\[ A$= "\text{NUMBER IS}" \]
\[ A=9999999 \]
\[ B=-9999999 \]
\[ C=-.123456789 \]

\[ 123456789\bar{9}123456789\bar{9}123456789012 \]

PRINT \( A$,A$ \)
NUMBER IS

PRINT \( A$,A$ \)
NUMBER IS NUMBER 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 \[ 9999999 \]
\[ -9999999 \]
\[ -.123456789 \]

PRINT \( A$,A,A$;B,A$;C \)
NUMBER IS \[ 9999999 \]
\[ -9999999 \]
\[ -.123456789 \]

PRINT \( A;B;C,A;B;C \)
NUMBER IS \[ 9999999 \]
\[ -9999999 \]
\[ -.123456789 \]
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.
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
### BITS, BYTES, WORDS - DATA PRECISION

<table>
<thead>
<tr>
<th>1 BIT</th>
<th>ON-OFF</th>
<th>1 or $ PRIMARY UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 BYTE</td>
<td>8 BITS</td>
<td>BASIC UNIT OF MEMORY</td>
</tr>
<tr>
<td>1 WORD</td>
<td>16 BITS</td>
<td>2 BYTES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>TYPE</th>
<th>DATA PRECISION</th>
<th>BYTES OF MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL PRECISION</td>
<td>REAL</td>
<td>$9.99999999999999E±499</td>
<td>10*</td>
</tr>
<tr>
<td>SHORT PRECISION</td>
<td>SHORT</td>
<td>$9.9999E±99</td>
<td>6*</td>
</tr>
<tr>
<td>INTEGER PRECISION</td>
<td>INTEGER</td>
<td>-99999 thru +99999</td>
<td>5*</td>
</tr>
</tbody>
</table>

*SIMPLE VARIABLES*

**NOTES:**

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

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>STANDARD FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>9876543210.1234</td>
<td>9876543210.12</td>
</tr>
<tr>
<td>15.000</td>
<td>15</td>
</tr>
<tr>
<td>00.23500</td>
<td>.235</td>
</tr>
<tr>
<td>-.05479</td>
<td>-4.38415537301E-12</td>
</tr>
<tr>
<td>000987.5</td>
<td>987.5</td>
</tr>
<tr>
<td>10000.6</td>
<td>1.E24</td>
</tr>
<tr>
<td>.01E4</td>
<td>100</td>
</tr>
<tr>
<td>120E-4</td>
<td>.012</td>
</tr>
</tbody>
</table>

*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

\[-9.99999999999 \times 10^{-499} \text{ THRU } 9.99999999999 \times 10^{-499}\]
SIMPLE NUMERIC VARIABLES (PGS. 49 - 51)

NAMES = A, A0, A1, . . , A9
   B, B0, . . , B9
   .
   .
   Z, Z0, . . , 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

116 DISP "ENTER # OF SCANS"
126 INPUT S

NOTES:

REQUIRES 10 BYTES OF MEMORY (FULL PRECISION)

a=a AS FAR AS VARIABLES ARE CONCERNED
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
ARRAY VARIABLES (PG. 119-124)

PURPOSE: USED TO STORE GROUPS OF NUMBERS.

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

TYPES: *ONE DIMENSIONAL (SCALARS)

| Eg. | A9 | 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 |

NOTES:

Simple and array variables can share the same name.

*THREE DIMENSIONAL (NOT ALLOWED)

*STRING ARRAYS (NOT ALLOWED)
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.
ARRAY VARIABLE MEMORY ALLOCATION

XXX OPTION BASE 1

| SPECIFIES WHETHER LOWEST |
| ARRAY ELEMENT IS \( \emptyset \) OR 1 |
| MUST COME FIRST |
| DEFAULT IS \( \emptyset \) |

XXX DIM V(3,100)

| 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 \( \emptyset \) WERE SPECIFIED, HOW MANY ELEMENTS WOULD ARRAY V CONTAIN ??

PROVE THE STORAGE REQUIREMENTS GIVEN EARLIER FOR ARRAY VARIABLES.
VARIABLE ASSIGNMENT

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

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

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

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!

B-32
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 10

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.
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.
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.
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<9 THEN GOTO 9000
BRANCHING INTRODUCTION (pg 103-108)

TYPES:

GOTO STATEMENT

IF numeric expression THEN execute this portion
is true
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
200  AIT 3000 @ CLEAR @ GOTO 160
210  !

990  IF V(I,S)>L2 THEN OUTPUT 709;
     "DOL,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 ! LAHC - 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."
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
.751334566861 .75133
.719220376063 .71922
.654548216962 .65454
.161648891563 .16165
.872216882948 .87222
.857017795574 .85702
.469902305241 .4699
.113434638381 .11343
AVERAGE IS ; .5555096853035
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 ??
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"

<table>
<thead>
<tr>
<th>POSITION</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>A$ contents</td>
<td>T</td>
<td>E</td>
<td>S</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

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$=""
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
\ . \ . \ .
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$
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.
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
or
date in mdd } SETS
                     } CLOCK
A = TIME
B = DATE
DISP "DATE IS ";B;"TIME IS ";A

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 will be 332 NOT 401!

B-45
# STRING FUNCTIONS

<table>
<thead>
<tr>
<th>SYNTAX</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEN (S$)</td>
<td>LENGTH OF STRING S$.</td>
</tr>
<tr>
<td>POS (S1$, S2$)</td>
<td>POSITION OF S2$ IN S1$.</td>
</tr>
<tr>
<td>UPC$ (S$)</td>
<td>CONVERTS LOWERCASE CHARACTERS IN S$ TO UPPERCASE.</td>
</tr>
<tr>
<td>VAL (S$)</td>
<td>NUMERIC EQUIVALENT OF STRING S$.</td>
</tr>
<tr>
<td>VALS(X)</td>
<td>STRING EQUIVALENT OF X.</td>
</tr>
<tr>
<td>CHR$ (X)</td>
<td>CHARACTER WHOSE DECIMAL CODE IS X: 0&lt;=X&lt;=255.</td>
</tr>
<tr>
<td>NUM (S$)</td>
<td>DECIMAL CODE OF FIRST CHARACTER IN S$.</td>
</tr>
</tbody>
</table>

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:

```plaintext
DIM A$[10]
A$ = "abc27fg"h"

LEN (A$) = 8
POS (A$, "fg") = 6
UPC$ (A$) = "ABC27FGH"
VAL ("27") = 27

VAL$ (85) = "85"
CHR$ (65) = "A"
NUM (A$) = 97
```
CODES AND NUMBER BASES

DECIMAL (BASE 10)

OCTAL (BASE 8)

BINARY (BASE 16)

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

<table>
<thead>
<tr>
<th>ASCII CHARACTER</th>
<th>DECIMAL</th>
<th>BINARY</th>
<th>OCTAL</th>
<th>HEXADECIMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>65</td>
<td>01 000 001</td>
<td>101</td>
<td>41</td>
</tr>
<tr>
<td>B</td>
<td>66</td>
<td>01 000 010</td>
<td>102</td>
<td>42</td>
</tr>
<tr>
<td>C</td>
<td>67</td>
<td>01 000 011</td>
<td>103</td>
<td>43</td>
</tr>
<tr>
<td>D</td>
<td>68</td>
<td>01 000 100</td>
<td>104</td>
<td>44</td>
</tr>
<tr>
<td>§</td>
<td>36</td>
<td>00 100 100</td>
<td>044</td>
<td>24</td>
</tr>
</tbody>
</table>

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.

16₁₀ = 1 x 10 + 6 x 1 = 16₁₀ = 10000₂
16₈ = 1 x 8 + 6 x 1 = 14₁₀ = 01110₂ (Note A)
16₁₆ = 1 x 16 + 6 x 1 = 22₁₀ = 10110₂ (Note B)

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

eg 01 110₂ = 16₈

B. BREAKING INTO GROUPS OF FOUR GIVES THE HEXADECIMAL NUMBER

1 0110₂ = 16₁₆

B-47
<table>
<thead>
<tr>
<th>HP-IB</th>
<th>ASCII</th>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
<th>HP-IB</th>
<th>ASCII</th>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Command</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Task</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group ACG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BDC</td>
<td>EOT</td>
<td>010000001</td>
<td>0101</td>
<td></td>
<td>06 37 104</td>
<td></td>
<td>F T0</td>
<td></td>
<td>01000000</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>PPC</td>
<td>ENO</td>
<td>010000001</td>
<td>0101</td>
<td></td>
<td>06 37 104</td>
<td></td>
<td>D T4</td>
<td></td>
<td>01000100</td>
<td>104</td>
<td>44</td>
</tr>
<tr>
<td>ACK</td>
<td>MNT</td>
<td>010000001</td>
<td>0101</td>
<td></td>
<td>06 37 104</td>
<td></td>
<td>E T5</td>
<td></td>
<td>01000101</td>
<td>105</td>
<td>45</td>
</tr>
<tr>
<td>GEL</td>
<td>LF</td>
<td>010000001</td>
<td>0101</td>
<td></td>
<td>06 37 104</td>
<td></td>
<td>F T6</td>
<td></td>
<td>01000110</td>
<td>106</td>
<td>46</td>
</tr>
<tr>
<td>TCT</td>
<td>VT</td>
<td>010000001</td>
<td>0101</td>
<td></td>
<td>06 37 104</td>
<td></td>
<td>G T7</td>
<td></td>
<td>01000111</td>
<td>107</td>
<td>47</td>
</tr>
<tr>
<td>GET</td>
<td>LS</td>
<td>010000001</td>
<td>0101</td>
<td></td>
<td>06 37 104</td>
<td></td>
<td>H T8</td>
<td></td>
<td>01010000</td>
<td>110</td>
<td>50</td>
</tr>
<tr>
<td>TT</td>
<td>CR</td>
<td>010000001</td>
<td>0101</td>
<td></td>
<td>06 37 104</td>
<td></td>
<td>I T9</td>
<td></td>
<td>01010001</td>
<td>111</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>010000001</td>
<td>0101</td>
<td></td>
<td>06 37 104</td>
<td></td>
<td>J T10</td>
<td></td>
<td>01010100</td>
<td>124</td>
<td>54</td>
</tr>
<tr>
<td>Universal Command LLO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>K T11</td>
<td></td>
<td>01010101</td>
<td>125</td>
<td>55</td>
</tr>
<tr>
<td>Group -UCG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L T12</td>
<td></td>
<td>01010110</td>
<td>126</td>
<td>56</td>
</tr>
<tr>
<td>DCL</td>
<td>DC2</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T16</td>
<td></td>
<td>P</td>
<td></td>
<td>01010111</td>
<td>127</td>
<td>57</td>
</tr>
<tr>
<td>PPC</td>
<td>DC3</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T17</td>
<td></td>
<td>Q</td>
<td></td>
<td>01010111</td>
<td>127</td>
<td>57</td>
</tr>
<tr>
<td>NPC</td>
<td>DC4</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T18</td>
<td></td>
<td>R</td>
<td></td>
<td>01010100</td>
<td>122</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>DC5</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T19</td>
<td></td>
<td>S_</td>
<td></td>
<td>01010100</td>
<td>122</td>
<td>53</td>
</tr>
<tr>
<td>SPE</td>
<td>CAN</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T20</td>
<td></td>
<td>T</td>
<td></td>
<td>01010110</td>
<td>126</td>
<td>56</td>
</tr>
<tr>
<td>SPD</td>
<td>EM</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T21</td>
<td></td>
<td>U</td>
<td></td>
<td>01010110</td>
<td>126</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>JB</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T22</td>
<td></td>
<td>V</td>
<td></td>
<td>01010110</td>
<td>126</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>EBC</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T23</td>
<td></td>
<td>W</td>
<td></td>
<td>01010110</td>
<td>126</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T24</td>
<td></td>
<td>X</td>
<td></td>
<td>01010110</td>
<td>126</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>GS</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T25</td>
<td></td>
<td>Y</td>
<td></td>
<td>01010110</td>
<td>126</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>RS</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T26</td>
<td></td>
<td>Z</td>
<td></td>
<td>01010110</td>
<td>126</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>US</td>
<td>010001000</td>
<td>0200</td>
<td>10</td>
<td>T27</td>
<td></td>
<td>LMT</td>
<td></td>
<td>01010111</td>
<td>127</td>
<td>57</td>
</tr>
<tr>
<td>Listen Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L0</td>
<td>SP</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S0</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S1</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S2</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S3</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L4</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S4</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L5</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S5</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L6</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S6</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L7</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S7</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L8</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S8</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L9</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>S9</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L10</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SA</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L11</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SB</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L12</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SC</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L13</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SD</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L14</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SE</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L15</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SF</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L16</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SG</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L17</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SH</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L18</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SI</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L19</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SJ</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L20</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SK</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L21</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SL</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L22</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SM</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L23</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SN</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L24</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SO</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L25</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SP</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L26</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SQ</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L27</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SR</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L28</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SS</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L29</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>ST</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L30</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SU</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L31</td>
<td>010001000</td>
<td>0400</td>
<td>20</td>
<td>SV</td>
<td></td>
<td>01000001</td>
<td>101</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Mathematical Functions

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS (X)</td>
<td>Absolute value of X</td>
</tr>
<tr>
<td>SGN (X)</td>
<td>SGN of X: +, 0 or -</td>
</tr>
<tr>
<td>CEIL (X)</td>
<td>Smallest integer ≥ X</td>
</tr>
<tr>
<td>FLOOR (X)</td>
<td>Largest integer ≤ X</td>
</tr>
<tr>
<td>INT (X)</td>
<td>Largest integer ≤ X</td>
</tr>
<tr>
<td>FP (X)</td>
<td>Fractional part of X</td>
</tr>
<tr>
<td>IP (X)</td>
<td>Integer part of X</td>
</tr>
<tr>
<td>MAX (X, Y)</td>
<td>Returns larger value of X and Y</td>
</tr>
<tr>
<td>MIN (X, Y)</td>
<td>Returns smaller value of X and Y</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP (X)</td>
<td>(e^x)</td>
</tr>
<tr>
<td>LOG (X)</td>
<td>Log (base e) of X, X &gt; 0</td>
</tr>
<tr>
<td>LGT (X)</td>
<td>Log (base 10) of X, X &gt; 0</td>
</tr>
<tr>
<td>SQR (X)</td>
<td>Square root of X</td>
</tr>
</tbody>
</table>

<p>| INF | Largest machine number: (9.9999999999999E499) |
| EPS | Smallest machine number: (1E-499) |
| PI | 3.14159265359 |
| RND | Returns a number which is next in a sequence of pseudo-random numbers, (0 ≤ \text{RND} &lt; 1) |</p>
<table>
<thead>
<tr>
<th>SYNTAX</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIN (X)</td>
<td>SINE OF X</td>
</tr>
<tr>
<td>COS (X)</td>
<td>COSINE OF X</td>
</tr>
<tr>
<td>TAN (X)</td>
<td>TANGENT OF X</td>
</tr>
<tr>
<td>CSC (X)</td>
<td>COSECANT OF X</td>
</tr>
<tr>
<td>SEC (X)</td>
<td>SECANT OF X</td>
</tr>
<tr>
<td>COT (X)</td>
<td>COTANGENT OF X</td>
</tr>
<tr>
<td>ASN (X)</td>
<td>ARC SINCE OF X - 1ST OR 4TH QUARTER</td>
</tr>
<tr>
<td>ACN (X)</td>
<td>ARC COSINE OF X - 1ST OR 2ND QUADRANT</td>
</tr>
<tr>
<td>ATN (X)</td>
<td>ARC TANGENT OF X - 1ST OR 4TH QUADRANT</td>
</tr>
<tr>
<td>ATN2 (Y,X)</td>
<td>ARC TANGENT OF Y/X - PROPER QUADRANT</td>
</tr>
<tr>
<td>DTR (X)</td>
<td>DEGREES TO RADIANS CONVERSION</td>
</tr>
<tr>
<td>RTD (X)</td>
<td>RADIANS TO DEGREES CONVERSION</td>
</tr>
</tbody>
</table>

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

- **DEG** SETS DEGREES MODE
- **RAD** SETS RADIANS MODE
- **GRAD** SETS QUAD MODE
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.

\[
2 + \frac{3 \times 6}{(7-4)^2} \Rightarrow 2 + \left( \frac{3 \times 6}{(7-4)} \right)^2
\]

Innermost parentheses 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 ! LAB 3 - 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 ! $=SUM CNT D=STRING POINTER
100 DIM R(100) ! 100 F.P. NUMBER
110 SHORT S(100) ! 100 S.P. "$"
120 DIM S$[508] ! STRING STORAGE
130 !
140 DISP "ENTER # OF READINGS"
150 DISP "TO BE AVERAGED <<101>>"
170 INPUT NO IF N<0 OR N>100 THEN N BEEP & GOTO 140
180 !
190 DISP "READY TO START (Y/N)"
200 INPUT S$
210 IF UPC$(S$[1:1])>"Y" THEN BEEP @ 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 @ $$(C)=R(C) @ S$[C]=
D+5]=VAL$(S(C))
280 DISP R(C);" ";S(C);" ";S$[D+5]
290 D=D+6 @ S=S+R(C)
300 NEXT C @ DISP
310 !
320 ! RECONVERTING STRING INTO
330 ! USEABLE DATA
340 !
350 D=1
360 FOR C=1 TO N
370 V=VAL$(S$[D+.D+5]) @ DISP V
380 D=D+6
390 NEXT C
394 !
395 DISP LEN$(S$)/6;" READINGS ST
400 ! STORED IN STRING $"
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 ! WP-85 PROGRAM
30 !
40 ! INITIALIZATION
50 !
60 PRINT 1S 2 ! PAPER
70 CRT IS 1 @ CLEAR ! CRT
80 RANDOMIZE @ OPTION BASE 1
90 S=0 @ D=1 ! S=SUM CNT D=STRING POINTER
100 DIM R(100) ! 100 F.P. NUMBERS
110 SHORT S(100) ! 100 S.P. O's
120 DIM S$(500) ! STRING STORAGE
130 !
140 DISP "ENTER # OF READINGS"
150 DISP "TO BE AVERAGED <101>"
160 INPUT N N IF N<0 OR N>100 THEN BEEP @ GOTO 140
180 !
190 DISP "READY TO START (Y/N)"
200 INPUT S$
210 IF UPCS$(S$[1,1])="Y" THEN BEEP @ GOTO 5000 @ CLEAR @ GOTO 190
220 !
230 BEGINNING OF LOOP
240 !
250 FOR C=1 TO N
260 R(C)=RND @ S$(C)=R(C) @ S$(CD
270 D+5)=VAL$(S(C))
280 DISP R(C);" ";S(C);="";S$(CD
290 D=D+6 @ S=S+R(C)
300 NEXT C @ DISP
310 !
320 ! RECONVERTING STRING INTO USEABLE DATA
330 ! USEABLE DATA
340 !
350 D=1
360 FOR C=1 TO N
370 V=VAL$(S$[CD,D+5]) @ DISP V
380 D=D+6
390 NEXT C
394 !
395 DISP LEN$(S$)/6;" READINGS STORED IN STRING S$"
400 BEEP @ DISP "DONE" @ END

HANDOUT B-52 A
10 ! LAB 3 - GYPHE 2/81
20 ! MF-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=STRING POINTER
100 DIM R(100) ! 100 F.P. NUMBER S
110 SHORT S(100) ! 100 S.P. 0's
120 DIM S$(500) ! STRING STORAGE
130 !
140 DISP "ENTER # OF READINGS"
150 DISP "TO BE AVERAGED (@101)"
160 INPUT N@ IF N<0 OR N>100 THEN
170 BEEP & GOTO 140
180 !
190 DISP "READY TO START (Y/N)"
200 INPUT S$
210 IF UPC$(S$[1,1])="Y" THEN BEEP & DISP "GET READY" & WAIT
220 @ 5000 & CLEAR @ GOTO 190
220 !
225 !
230 ! BEGINNING OF LOOP
240 !
250 FOR C=1 TO N
260 R(C)=RAND @ S$(C)=R(C) & S$(CD, D+5)=VAL$(S$(C))
280 DISP R(C);" ";S$(C);" ";S$(CD, D+5)
290 D=D+6 & S=S+R(C)
300 NEXT C & DISP
310 !
320 ! RECONVERTING STRING INTO USEABLE DATA
330 !
340 !
350 D=1
360 FOR C=1 TO N
370 V=VAL$(S$(CD, D+5)) & DISP V
380 D=D+6
390 NEXT C
394 !
395 DISP LEN(S$)/6;" READINGS STORED IN STRING S$"
400 BEEP & DISP "DONE" @ END

HANDOUT B-52 A
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
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)
Tape directory capacity: 850 data records (210K bytes)
Typical access rate (search speed): 42 files (directory entries)
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.
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:
### DATA STORE REQUIREMENTS

<table>
<thead>
<tr>
<th>TYPE OF VARIABLES</th>
<th>NUMBERS</th>
<th>STRINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINGLE VARIABLE</td>
<td>8 BYTES PER NUMBER</td>
<td>1 BYTF PER CHAR. + 3 BYTES PER STRING + 3 BYTES PER RECORD CROSSED</td>
</tr>
<tr>
<td>ARRAY VARIABLE</td>
<td>8 BYTES X NUMBER OF ARRAY ELEMENTS</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**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, i.e., 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$
### 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:

FILE "CLASS 1"

C. LAURELL, 98, 89, 95  R. KREIDEL, 86, 72, 90  K. MILLER, 90, 87, 89

ITEMS ARE ACCESSED BY RECORD NUMBER. ACCESS PORTIONS OF THE FILE BY INDIVIDUAL RECORD.
STORING AND RETRIEVING DATA

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

SYNTAX:
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

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>BYTES</th>
<th>REC</th>
<th>FS</th>
<th>FILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>DATA</td>
<td>256</td>
<td>94</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Autost</td>
<td>PROC</td>
<td>256</td>
<td>37</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>37PTSC</td>
<td>PROC</td>
<td>256</td>
<td>24</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>345TM</td>
<td>PROC</td>
<td>256</td>
<td>36</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>MERGE</td>
<td>BPIM</td>
<td>256</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>BPIM</td>
<td>256</td>
<td>1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>THERMO</td>
<td>DATA</td>
<td>256</td>
<td>32</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>REJUCT</td>
<td>DATA</td>
<td>256</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>RFTMP</td>
<td>PROC</td>
<td>256</td>
<td>28</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>PROC</td>
<td>256</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>DATA</td>
<td>256</td>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>DATA</td>
<td>256</td>
<td>1</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>BYTES</th>
<th>RECS</th>
<th>FILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>69735</td>
<td>PROG</td>
<td>256</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>69751a</td>
<td>PROG</td>
<td>256</td>
<td>36</td>
<td>2</td>
</tr>
<tr>
<td>69751b</td>
<td>PROG</td>
<td>256</td>
<td>38</td>
<td>2</td>
</tr>
<tr>
<td>97INT</td>
<td>PROG</td>
<td>256</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>790_MT</td>
<td>PROG</td>
<td>256</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>751FHS</td>
<td>PROG</td>
<td>256</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>69720a</td>
<td>PROG</td>
<td>256</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>WAVEFM</td>
<td>PROG</td>
<td>256</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>BYTES</th>
<th>RECS</th>
<th>FILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>69751a</td>
<td>PROG</td>
<td>256</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>69751b</td>
<td>PROG</td>
<td>256</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>97INT</td>
<td>PROG</td>
<td>256</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>790_MT</td>
<td>PROG</td>
<td>256</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>751FHS</td>
<td>PROG</td>
<td>256</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>69720a</td>
<td>PROG</td>
<td>256</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>WAVEFM</td>
<td>PROG</td>
<td>256</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>NULL</td>
<td></td>
<td>256</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>BYTES</th>
<th>RECS</th>
<th>FILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td></td>
<td>256</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>69751a</td>
<td>PROG</td>
<td>256</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>69751b</td>
<td>PROG</td>
<td>256</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>97INT</td>
<td>PROG</td>
<td>256</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>790_MT</td>
<td>PROG</td>
<td>256</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>751FHS</td>
<td>PROG</td>
<td>256</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>69720a</td>
<td>PROG</td>
<td>256</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>WAVEFM</td>
<td>PROG</td>
<td>256</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>69735</td>
<td>PROG</td>
<td>256</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>

T-16
MASS STORAGE OPERATION

BINARY PROGRAMS

PURPOSE:
TO STORE AND RETRIEVE BINARY PROGRAMS

SYNTAX:
STOREBIN "PROGRAM NAME"
LOADBIN "PROGRAM NAME"

NOTES:
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 program 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 program lines of the program currently in memory. This requires a little foresight 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)

<table>
<thead>
<tr>
<th>STORAGE DEVICE</th>
<th># of RECORDS</th>
<th>BYTES/RECORD</th>
<th># BUFFERS</th>
<th>TIME (SEC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tape</td>
<td>32</td>
<td>256</td>
<td>1</td>
<td>93</td>
</tr>
<tr>
<td>disc</td>
<td>32</td>
<td>256</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>tape</td>
<td>1000</td>
<td>8</td>
<td>1</td>
<td>96</td>
</tr>
<tr>
<td>disc</td>
<td>1000</td>
<td>8</td>
<td>1</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>tape</td>
<td>1</td>
<td>8008</td>
<td>1</td>
<td>91</td>
</tr>
<tr>
<td>disc</td>
<td>1</td>
<td>8008</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>26</td>
</tr>
</tbody>
</table>

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??
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:
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.
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.

1. Walk into bar
2. Catch bartender's eye
   - No: Go back
   - Yes: Order beer
3. Order beer
4. Find seat
   - No: Stand
   - Yes: Drink beer
5. Drink beer
6. Glass empty?
   - No: Go back
   - Yes:
5.1 Driving?
   - No: Enough money?
     - No: Go back
     - Yes: Bar open?
       - No: Had enough?
         - No: Go back
         - Yes: Go home
       - Yes: Go home
   - Yes: # of pints $\geq 2$
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), SS(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:
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:
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

EXECUTIVE PROGRAM

Executive program calls subroutines in order of occurrence

End or chain another program

SUBROUTINE 1
PERFORM TASK 1

SUBROUTINE 2
PERFORM TASK 2

SUBROUTINE N
PERFORM TASK N
IMPLEMENTING STRUCTURED PROGRAMMING

GOSUB STATEMENT

... ! DATA ACQUIRE  AUTHOR DATE

EXECUTIVE  PROGRAM
... GOSUB 1000 ! GIVE INSTRUCTIONS
... GOSUB 2000 ! PREPARE FOR MEASUREMENTS
... GOSUB 3000 ! TAKE MEASUREMENTS
... 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 e.g. 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
BRANCHING USING SPECIAL FUNCTION KEYS (154 - 156)

ON KEY #1, "READ V" GOSUB 9000
ON KEY #2, "READ F" GOSUB 12000 \{ GO WHERE SPECIFIED WHEN KEY PRESSED. \\
ON KEY #8, "DONE" GOTO 9000

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 number 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

... 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 4 Ø
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 FNSS$ = "10 Seconds"
20 DISP FNSS$ FUNCTION
3Ø END WITH
RUN NO
ARGUMENT

10 Seconds
MULTIPLE LINE FUNCTIONS

Used if function contains lengthy 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.
Variables can be put into common memory which allows 2 or more programs to access them.

10 ! PROGRAM A
20 OPTION BASE 1
30 COM D (2,1000)
40 COM AS [1000]
...
.. GENERATE DATA
...
.. CHAIN "PROG B"

1008BYTES

16KYBYTES

10 ! PROG B
20 ! OPTION BASE 1
30 COM F (2,1000)
40 COM BS [1000]
...
.. ANALYZE
.. DATA
...
.. CHAIN OR END

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.
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
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 ! μV 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:

I
-

ARRAY S

1

2

3

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

TEMPORARY STORAGE

To exchange the values of positions 1 & 2

1

2

3
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
FORMATTING UP TO NOW

PRINTER IS 2
PRINTER IS 701,80
CRT IS 2
PRINT A,B,C,D,E,
PRINT A;B;C;D;E
DISP "A=";A, "B=";B
DISP "A="; TAB(20);A

Know how to select printer or display device.
Know how to use , and ; to separate items to be displayed or printed.
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:

<table>
<thead>
<tr>
<th>Compact Field</th>
<th>Numeric Data</th>
<th>String Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semicolon</td>
<td>Digits of the number are output, preceded by a space (if plus) or a minus sign (if minus), and followed by one space.</td>
<td>Characters of the string are output with no leading or trailing spaces.</td>
</tr>
<tr>
<td>Comma</td>
<td>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.</td>
<td>Characters of the string are output with no leading spaces and no more than 20 trailing spaces.</td>
</tr>
</tbody>
</table>

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 = +99999999
C = -9999999   D = .0123456789  A$ = "TESTING 12345"

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1234567890</td>
<td>1234567890</td>
<td>1234567890</td>
<td>1234567890</td>
</tr>
<tr>
<td>2.34E15 99999999 -99999999 2.34E15</td>
<td>2.34E15 99999999 -99999999 2.34E15</td>
<td>2.34E15 99999999 -99999999 2.34E15</td>
<td>2.34E15 99999999 -99999999 2.34E15</td>
</tr>
<tr>
<td>.0123456789 2.34E15 99999999 -99999999</td>
<td>.0123456789 2.34E15 99999999 -99999999</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-9999999 2.34E15 99999999 -99999999</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
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
```

To this

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

.879546874885
.510432700278
.850038458669
.516683883424
.62216477247
.90864941057
.581942277323
.950857003951
.222269723105
2.90271847999E-2

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

F - 4
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.

110 IMAGE specifications HOW
120 OUTPUT 722 USING 110; data list WHERE
...
810 ENTER 709 USING 110; data list WHAT
...
920 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 Output. 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:
123456789$123456789$123456789$123456789$1
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.
More on the IMAGE statement

Alternate method of using IMAGE

110 A$=" VOLTS"
120 PRINT USING "8X,D.DDD,6A";6.324,A$

is the same as

110 IMAGE 8X,D.DDD,6A
120 PRINT USING 110;6.324,A$

Reuse of IMAGE specification

180 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.
NUMERIC OUTPUT IMAGE SPECIFIERS

MAINFRAME

nD - digit field with leading blanks
n2 - 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 < \ 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 \( T = 1234567 \), \( T1 = -1234567 \)
\( T2 = .1234 \)

<table>
<thead>
<tr>
<th>12345678901234567890123456789012</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1234567.000</td>
</tr>
<tr>
<td>+1234567.0</td>
</tr>
<tr>
<td>1234567</td>
</tr>
<tr>
<td>1.235E+06</td>
</tr>
<tr>
<td>-1,234,567.000</td>
</tr>
<tr>
<td>.1234   .1234   .1234</td>
</tr>
<tr>
<td>1234567.0000000000.12</td>
</tr>
<tr>
<td>1.235E+06</td>
</tr>
</tbody>
</table>

Use formatted PRINTs or DISPs to generate these outputs, both in 1 IMAGE STATEMENT

NOTES:
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.
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.

<table>
<thead>
<tr>
<th>BIT #</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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
<table>
<thead>
<tr>
<th>P/N-18</th>
<th>ASCII</th>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
<th>P/N-18</th>
<th>ASCII</th>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addressed Command GTL</td>
<td>NUL</td>
<td>000000000</td>
<td>00000000000</td>
<td>01</td>
<td>T0</td>
<td>60</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group ACG</td>
<td>Ethernet</td>
<td>EOT</td>
<td>00000000004</td>
<td>04</td>
<td>T4</td>
<td>D</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMA</td>
<td>ECO</td>
<td>00000010004</td>
<td>04</td>
<td>T4</td>
<td>D</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACK</td>
<td>ECO</td>
<td>00000000006</td>
<td>06</td>
<td>T6</td>
<td>F</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BUS</td>
<td>ECO</td>
<td>00000010007</td>
<td>07</td>
<td>T7</td>
<td>G</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>STS</td>
<td>ECO</td>
<td>00000000008</td>
<td>08</td>
<td>T8</td>
<td>H</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDC</td>
<td>DLE</td>
<td>000100000</td>
<td>020</td>
<td>10</td>
<td>T18</td>
<td>F</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECO</td>
<td>00000000011</td>
<td>011</td>
<td>T18</td>
<td>F</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACK</td>
<td>00000000012</td>
<td>012</td>
<td>T18</td>
<td>F</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BUS</td>
<td>00000000013</td>
<td>013</td>
<td>T18</td>
<td>F</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>STS</td>
<td>00000000014</td>
<td>014</td>
<td>T18</td>
<td>F</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DDC</td>
<td>00000010011</td>
<td>015</td>
<td>T18</td>
<td>F</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEC Level Command LLO Group UCG</td>
<td>CAN</td>
<td>001100000</td>
<td>030</td>
<td>18</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EM</td>
<td>00110000001</td>
<td>019</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SUB</td>
<td>00110000002</td>
<td>018</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ESC</td>
<td>00110000003</td>
<td>017</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>00110000004</td>
<td>016</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GS</td>
<td>00110000005</td>
<td>015</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US</td>
<td>00110000006</td>
<td>014</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SP</td>
<td>00110000007</td>
<td>013</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wire</td>
<td>00110000008</td>
<td>012</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Secondary Command Group</td>
<td>000000010101010000010000000</td>
<td>20</td>
<td>10</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Group SCG</td>
<td>000000010101000000000000000</td>
<td>20</td>
<td>10</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>000000100101010000000000000</td>
<td>20</td>
<td>10</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>000001000101010000000000000</td>
<td>20</td>
<td>10</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>000001100010100000000000000</td>
<td>20</td>
<td>10</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>000001110011000000000000000</td>
<td>20</td>
<td>10</td>
<td>T24</td>
<td>Y</td>
<td>0000000100100</td>
<td>120</td>
<td>50</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
STRING IMAGE SPECIFIERS

MAINFRA ME

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.
WRITE A PROGRAM TO OUTPUT THIS DATA EXACTLY AS SHOWN

FORMAT EXAMPLE - HP-85

DATA PRINTOUT AT TIME: 981.132

<table>
<thead>
<tr>
<th>N</th>
<th>N^2</th>
<th>1/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0.500</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>0.333</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0.250</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>0.200</td>
</tr>
<tr>
<td>6</td>
<td>36</td>
<td>0.167</td>
</tr>
<tr>
<td>7</td>
<td>49</td>
<td>0.143</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>0.125</td>
</tr>
<tr>
<td>9</td>
<td>81</td>
<td>0.111</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>0.100</td>
</tr>
</tbody>
</table>

NOTES:
Select Codes & Interface Cards

HP-85

PROCESSOR
MEMORY BUS
I/O BUS
DISPLAY TAPE
KEYBOARD PRINTER

ROMS
82903 EXTERNAL MEMORY

INTERFACE CARDS
82930 HP-IB
82940 16 BIT PARALLEL (6PID)
82939 RS232 SERIAL
82941 BCD INTERFACE
82949 PRINTER INTERFACE

SELECT CODES

CRT 1
PRINTER 2
EXTERNAL DEVICES 3-19

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 "FIR3100ST1CRLF" for 100 readings

WHICH ONE OF THESE WOULD SEND THE CORRECT INFORMATION

10 N=100
20 PRINT "FIR3",N,"ST1"
30 PRINT
40 PRINT "FIR3";N;"ST1"
50 PRINT
60 OUTPUT 2 ; "FIR3";N;"ST1"
70 PRINT
80 OUTPUT 2 USING "K,DDD,K" ; "FIR3",N,"ST1"
90 PRINT
100 END

NOTES:

Try these on your HP-85

F-14
OUT\$ PUT STATEMENT

1410 PRINTER IS 722 \@ PRINT A$ ! PRINT statement does not ! specify destination
....
....
2010 OUTPUT 722:A$ ! OUTPUT statement specifies ! destination
....
1010 OUTPUT 711;"FIR3N10ST2" ! SET UP DE\$ ICE
....
....
2520 IMAGE "FIR3N", DDDD, "STZ" ! PROGRAM DEVICE
2530 OUTPUT 722 USING 2020:N
....
5550 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.
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)
ENTER IMAGE Specifiers for STRINGS

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>n A</td>
<td>enter single character</td>
</tr>
<tr>
<td>n K</td>
<td>enter string of unknown length</td>
</tr>
</tbody>
</table>

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

F-17
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.

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
<th>NUMBER OF CHARACTERS CONTRIBUTED TO NUMERIC FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>n D</td>
<td>digit field (blanks)</td>
<td>1</td>
</tr>
<tr>
<td>n Z</td>
<td>digit field (zero's)</td>
<td>1</td>
</tr>
<tr>
<td>n *</td>
<td>digit field (asterisks)</td>
<td>-1</td>
</tr>
<tr>
<td>n .</td>
<td>decimal point</td>
<td>1</td>
</tr>
<tr>
<td>S</td>
<td>sign field (t or -)</td>
<td>1</td>
</tr>
<tr>
<td>M</td>
<td>sign field (blank or -)</td>
<td>1</td>
</tr>
<tr>
<td>( )</td>
<td>replicating fields</td>
<td>As Applicable</td>
</tr>
<tr>
<td>E</td>
<td>exponent field ($10^{99}$)</td>
<td>5</td>
</tr>
<tr>
<td>e</td>
<td>exponent field ($10^{-99}$)</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>causes ENTER to ignore commas</td>
<td>1</td>
</tr>
<tr>
<td>K</td>
<td>free-field</td>
<td></td>
</tr>
<tr>
<td>R,P,</td>
<td>not allowed</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:

Number builder terminates on seeing non numeric character.

ie any other ASCII character than;

+ - . 1 2 3 4 5 6 7 8 9 @ E e

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

All would interpret 123.4567 the same!
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:
ENTER IMAGE SPECIFIERS - BINARY

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

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Interpret 1 byte as binary data</td>
</tr>
<tr>
<td>W</td>
<td>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</td>
</tr>
</tbody>
</table>

NOTES:

Enter 6 using ";W";A

(Example of entering data from 6940B multiprogrammer on 82940A GPIO interface)
## ENTER IMAGE SPECIFIERS - TERMINATORS

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Don't hunt for anything</td>
</tr>
<tr>
<td></td>
<td>Hunt for a line-feed (LF)</td>
</tr>
<tr>
<td>@</td>
<td>Hunt for EOI or Line feed (LF)</td>
</tr>
<tr>
<td>##</td>
<td>Hunt for EOI (end or identify line on IEEE-488 interface)</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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$ *Terminate on EOI
**ENTER IMAGE SPECIFIERS - SKIP CONTROL**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>Look for a line-feed (LF) before moving on; throw away characters until a line-feed is found, then proceed to next field.</td>
</tr>
<tr>
<td>n     X</td>
<td>Skip a character</td>
</tr>
</tbody>
</table>

eg ENTER 709 USING "D,1,D";X,Y

eg ENTER 709 USING "2D,X,2D,X,2D";X,Y,Z

**NOTES:**

?? How would the latter IMAGE interpret 12345678LF

X=
Y=
Z=

??
Write the IMAGE for the following:

\[ x = \text{space} \]

\[
\text{x} \text{xx}46\,230\text{xxmsecCRLF}
\]

\[
\text{OP,} \, 7, \underbrace{100.000}_{\text{variable}}, \text{T CRLF}
\]

\[
\text{MI,} \underbrace{23.456GZ}_{\text{variable}} \text{ CRLF}
\]

\[
\text{WF,} \underbrace{8.0,2500}_{\text{variables}}, \text{T CRLF}
\]

Send the binary equivalent of the octal number 170240

Send the binary equivalent of decimal 12 (page eject)

Incoming data: \(+138.5000E-6,+0220.9000E-6,+0276.0000E-6\)
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
### Summary of OUTPUT Image Specifiers

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

### Summary of ENTER Image Specifiers

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

<table>
<thead>
<tr>
<th>SCAN</th>
<th>CHANNEL 1</th>
<th>CHANNEL 2</th>
<th>CHANNEL 3</th>
<th>CHANNEL 4</th>
<th>CHANNEL 5</th>
<th>CHANNEL 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>value</td>
<td>value</td>
<td>value</td>
<td>value</td>
<td>value</td>
<td>value</td>
</tr>
<tr>
<td>2</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>10</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
</tbody>
</table>

**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/,1
90 PRINTER IS 706
100 OUTPUT 706 USING 80 ; A$(1, 15), C,D
110 OUTPUT 706 USING "P,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,/,Z" ; "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

CHANNEL 1 CHANNEL 2 CHANNEL 3 CHANNEL 4 CHANNEL 5 CHANNEL 6
SCAN

1 .000513 .000513 .000514 .000515 .000515 .000515
2 .000517 .000520 .000522 .000525 .000525 .000522
3 .000519 .000517 .000515 .000504 .000498 .000470
4 .000458 .000441 .000427 .000417 .000407 .000395
5 -.000382 -.000238 -.000675 -.000799 -.000873 -.000996
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 .00024 -.000712 -.000672 -.000374 -.000165
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.
HP-IB BUS CONCEPT

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 appropriate 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.
OVERVIEW OF THE CONTROL LINES

HEWLETT-PACKARD INTERFACE BUS "HP-IB"

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 (e.g. 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.

H-3
COMMAND MODE VS. DATA MODE

ATN

True ←——→ False

COMMAND MODE

specify talker, listener(s)
preset / clear devices, etc.

DATA MODE

device receives setup data
sends, measurement data, etc.

CLEAR 7
OUTPUT 709;
ENTER 709;

"AI"

V(I)

NOTES:
REN MUST ALSO BE TRUE
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-1R 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 available 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

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.
SELECT CODES AND DEVICE ADDRESSES

HP-IB INTERFACE

Device 21
Select Code 7
HP-85

Device 09
INSTRUMENT

Device 15
INSTRUMENT

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.
SETTING DEVICE ADDRESSES

* Valid Address Ranges are 00 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.

---

NOTES:

<table>
<thead>
<tr>
<th>ASCII CODE CHARACTER</th>
<th>ADDRESS SWITCHES</th>
<th>DECIMAL EQUIVALENT OF BINARY SWITCH SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTEN</td>
<td>TALK</td>
<td>A₁</td>
</tr>
<tr>
<td>SP</td>
<td>@</td>
<td>0</td>
</tr>
<tr>
<td>.</td>
<td>,</td>
<td>0</td>
</tr>
<tr>
<td>#</td>
<td>$</td>
<td>0</td>
</tr>
<tr>
<td>%</td>
<td>&amp;</td>
<td>0</td>
</tr>
<tr>
<td>(</td>
<td>)</td>
<td>0</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>/</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

---

factory setting of HP-85

---

NOTE: THE TALK ONLY SWITCH SHOULD BE CHANGED ONLY WHEN THE INSTRUMENT IS OFF.
DISTINGUISHING BETWEEN TALK AND LISTEN ADDRESSES
THEIR UTILIZATION IN THE COMMAND MODE

The 5 address switch settings are pertinent 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:

<table>
<thead>
<tr>
<th>COMMAND MODE PARAMETERS (PARTIAL)</th>
<th>DATA BUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUS COMMAND</td>
<td>X00CCCCC</td>
</tr>
<tr>
<td>TALK ADDRESS</td>
<td>X10TTTTT</td>
</tr>
<tr>
<td>EXTENDED TA*</td>
<td>X10SSSSS</td>
</tr>
<tr>
<td>LISTEN ADDRESS</td>
<td>X01LLLLL</td>
</tr>
<tr>
<td>2ND LISTEN ADR*</td>
<td>X01LLLLL</td>
</tr>
</tbody>
</table>

EXAMPLE

| ALL UNLISTEN                      | X0011111 |
| DEVICE 23 TALKS                   | X1010111 |
| 2NDARY ADDRESS 10                 | X0110101 |
| DEVICE 01 LISTEN                  | X0100001 |
MECHANICAL ASPECTS

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

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.

<table>
<thead>
<tr>
<th>Interface Functions that may be included in an HP-IB device</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Talker or Extended Talker</td>
<td>T.TE</td>
<td>Capability required for a device to be a &quot;talker&quot;.</td>
</tr>
<tr>
<td>Listener or Extended Listener</td>
<td>L.LE</td>
<td>Capability required for a device to be a &quot;listener&quot;.</td>
</tr>
<tr>
<td>Source Handshake</td>
<td>SH</td>
<td>This provides a device with the capability to properly transfer a multiframe message.</td>
</tr>
<tr>
<td>Acceptor Handshake</td>
<td>AH</td>
<td>This provides a device with the capability to guarantee proper reassembly of remote multiframe messages.</td>
</tr>
<tr>
<td>Remote/Local</td>
<td>RL</td>
<td>Provides capability to select between two sources of input information. Local corresponds to front panel control and remote to the input information from the bus.</td>
</tr>
<tr>
<td>Service Request</td>
<td>SR</td>
<td>This capability permits a device to asynchronously request service from the controller.</td>
</tr>
<tr>
<td>Parallel Poll</td>
<td>PP</td>
<td>Provides capability for a device to uniquely identify itself if it requires services. The controller is requesting a response.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The capability differs from service request in that it requires a commitment of the controller to periodically conduct a parallel poll.</td>
</tr>
<tr>
<td>Device Clear</td>
<td>DC</td>
<td>This function allows a device to be initialized to a pre-defined state. A device with this capability will have the effect of the command described in its operating manual.</td>
</tr>
<tr>
<td>Device Trigger</td>
<td>DT</td>
<td>This function permits a device to have its basic operation measured by the talker on the bus.</td>
</tr>
<tr>
<td>Controller</td>
<td>C</td>
<td>The 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.</td>
</tr>
<tr>
<td>Drivers</td>
<td>E</td>
<td>This code describes the type of electrical drivers used in a device.</td>
</tr>
</tbody>
</table>
The interface functions are performed using bus commands.

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

### SUMMARY OF BUS COMMANDS THAT MOST INSTRUMENTS WILL RECOGNIZE

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>SPIS CODE</th>
<th>OCTAL CODE</th>
<th>PURPOSE</th>
<th>ASCII CHAR</th>
<th>DECIMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNADDRESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commands</td>
<td>UNL 077</td>
<td></td>
<td>Clears Bus of all listeners.</td>
<td>?</td>
<td>063</td>
</tr>
<tr>
<td></td>
<td>UNT 137</td>
<td></td>
<td>Unaddresses the current talker so that no talker remains on the Bus.*</td>
<td>-</td>
<td>137</td>
</tr>
<tr>
<td>Local Lockout</td>
<td>LLO 021</td>
<td></td>
<td>Enables front panel local-reset button on responding devices.</td>
<td>DC1</td>
<td>017</td>
</tr>
<tr>
<td>Device Clear</td>
<td>DCL 024</td>
<td></td>
<td>Returns idle devices capable of responding to pre-determined states, regardless of whether they are addressed or not.</td>
<td>DC4</td>
<td>020</td>
</tr>
<tr>
<td>UNIVERSAL</td>
<td>PPU 023</td>
<td></td>
<td>Suits all devices on the HP-IB with Parallel Poll capability to a predefined condition.</td>
<td>NAK</td>
<td>021</td>
</tr>
<tr>
<td>COMMANDS</td>
<td>SPE 030</td>
<td></td>
<td>Enables Serial Poll Mode on the Bus.</td>
<td>CAN</td>
<td>024</td>
</tr>
<tr>
<td></td>
<td>SPD 031</td>
<td></td>
<td>Enables Serial Poll Mode on the Bus.</td>
<td>EM</td>
<td>031</td>
</tr>
<tr>
<td>Selective</td>
<td>SDC 004</td>
<td></td>
<td>Returns addressed device, capable of responding to pre-determined states.</td>
<td>EDT</td>
<td>004</td>
</tr>
<tr>
<td>Device Clear</td>
<td>GTL 001</td>
<td></td>
<td>Returns responding devices to local control.</td>
<td>SOH</td>
<td>001</td>
</tr>
<tr>
<td>Go to Local</td>
<td>GET 010</td>
<td></td>
<td>Initiates a simultaneous pre-programmed action by responding devices.</td>
<td>BS</td>
<td>008</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>PPC 005</td>
<td></td>
<td>This command permits the DIO lines to be assigned to instruments on the Bus for the purpose of responding to a parallel poll.</td>
<td>ENQ</td>
<td>005</td>
</tr>
<tr>
<td>ADDRESSED</td>
<td></td>
<td></td>
<td>This command is given when the active controller on the Bus transfers control to another instrument.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMANDS</td>
<td>TCT 011</td>
<td></td>
<td></td>
<td>HT</td>
<td>009</td>
</tr>
</tbody>
</table>

NOTE: Talkers can also be unaddressed by transmitting an unused talk address on the Bus. This is not true with listeners.
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? i.e., 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 its output terminator(s)?
HP-IB PROGRAMMING STEP: KNOW PROGRAM CODES

These are the codes (usually ASCII characters) that cause the device to perform its various functions.
EXAMINING THE INTERFACE FUNCTIONS, BUS COMMANDS, AND PROGRAMMING CODES OF VARIOUS HEWLETT PACKARD INSTRUMENTS
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, TE0, L2, LE0, SR1, RL1, PP0, DC1, DT1, C0

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

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

<table>
<thead>
<tr>
<th>HP-IB MESSAGE</th>
<th>DESCRIPTION/RESPONSE</th>
<th>SAMPLE 9825A (address = 20)</th>
<th>SAMPLE 9825A/45A (address = 20)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>MEANS TO SEND COMMANDS TO 5316A AND RECEIVE MEASUREMENT DATA.</td>
<td>wrt 720, &quot;FNT&quot; red 720, A</td>
<td>OUTPUT 7.20; &quot;FNT&quot; ENTER 7.20; A</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>STARTS NEW MEASUREMENT. IF 5316A IS IN LOCAL, IT WILL REMAIN IN LOCAL AND NO TRIGGER OCCURS.</td>
<td>trg 7</td>
<td>TRIGGER 7.20</td>
</tr>
<tr>
<td></td>
<td>STARTS NEW MEASUREMENT. IF THE 5316A IS IN LOCAL THE 5316A WILL GO INTO REMOTE.</td>
<td>trg 720</td>
<td>TRIGGER 7.20</td>
</tr>
<tr>
<td>CLEAR</td>
<td>STARTS NEW MEASUREMENT (ACTS AS RESET).</td>
<td>clt 7 clt 720</td>
<td>CLEAR 7 clt 720</td>
</tr>
<tr>
<td>REMOTE</td>
<td>FRONT PANEL FUNCTION AND SLOPE SWITCHES ARE DISABLED; COUNTER DEFAULTS TO FREQUENCY A, ALL SLOPES TO POSITIVE UNLESS PREVIOUSLY PROGRAMMED.</td>
<td>rem 7 rem 720</td>
<td>REMOTE 7 rem 720</td>
</tr>
<tr>
<td>LOCAL</td>
<td>RETURNS TO LOCAL (FRONT PANEL) OPERATION.</td>
<td>lcl 7 lcl 720</td>
<td>LOCAL 7 lcl 720</td>
</tr>
<tr>
<td>LOCAL LOCKOUT</td>
<td>DISABLES FRONT PANEL; ONLY CONTROLLER CAN RETURN 5316A TO LOCAL. NOTE: IF REMOTE, FRONT PANEL FUNCTION AND SLOPE SWITCHES ARE ALSO DISABLED.</td>
<td>llo 7</td>
<td>LOCAL LOCKOUT 7</td>
</tr>
<tr>
<td>GOTO LOCAL AND CLEAR LOCAL LOCKOUT</td>
<td>5316A RETURNS TO LOCAL (FRONT PANEL) CONTROL; LOCAL LOCKOUT CLEARED.</td>
<td>lcl 7</td>
<td>LOCAL 7</td>
</tr>
<tr>
<td>SERVICE REQUEST</td>
<td>5316A WILL REQUEST SERVICE AT END OF MEASUREMENT IF SRQ AND WAIT STATE ENABLED.</td>
<td>rds (720) DEVICE STATUS</td>
<td>STATUS 7.20</td>
</tr>
<tr>
<td>STAT'S BYTE</td>
<td>PRESENTS STATUS INFORMATION. BIT 7 IS SET IF SERVICE IS REQUESTED.</td>
<td>rds (7) BUS STATUS</td>
<td>STATUS 7</td>
</tr>
<tr>
<td>STATUS BIT</td>
<td>NOT APPLICABLE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PASS CONTROL</td>
<td>NOT APPLICABLE.</td>
<td>cli 7</td>
<td>ABORTIO 7</td>
</tr>
<tr>
<td>ABORT</td>
<td>Terminates the bus communications, tells all devices to unsubscribe; 5316A ADIRD light will go off.</td>
<td>cli 7</td>
<td>ABORTIO 7</td>
</tr>
</tbody>
</table>
LEARNING SYNTAX CONDITIONS FOR THE 1980 OSCILLOSCOPE

(REFERENCE: 1980 QUICK REFERENCE GUIDE)

DATA SENT TO 1980

General Input Data Message Format:

```
<program code> | <program code> | ... | <CR> | <LF>
```

Device dependent
command

Delimiter is required
to separate multiple
commands within Data
message

EOS message

Optional characters
not required
before EOS:

Multiple device dependent
commands are allowed,
up to maximum string length
of 127 characters

Format rules:
The 1980A/B sends and receives all Data
messages except the Learn String in stand-
ard ASCII code. The Learn String consists of
80 eight-bit bytes of binary data.

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.

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.

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

Unassigned 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 per-
mitted only following program code identi-
fiers 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 tran-
sfers. 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 pro-
gram 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.
General Output Data Message Format:

```
<SP>|±|<setting or status data>|<CR>|<LF>
```

Most output Data messages are signed

Output data contains a maximum of 15 characters

EOS message

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 E01 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

<table>
<thead>
<tr>
<th>Function</th>
<th>Program Code (ASCII)</th>
<th>Function</th>
<th>Program Code (ASCII)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADVISORY MESSAGES</td>
<td>AV&lt;state&gt;</td>
<td>HORIZONTAL POSITION</td>
<td>HP&lt;value&gt;</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>state := 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTOSCOPE</td>
<td>AS&lt;state&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Execute autoscope</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SA&lt;state&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Execute selective</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>autoscope</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BANDWIDTH LIMIT</td>
<td>BW&lt;state&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>off on</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>state := 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALIBRATOR LEVEL</td>
<td>CL&lt;level&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0.2 V p-p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1 V p-p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.2 V p-p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 V p-p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 V p-p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>level := 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHARACTER GENERATOR</td>
<td>CG&lt;state&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>readout off</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>state := 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>readout on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTROL KNOB</td>
<td>RC&lt;entry&gt;[:&lt;mode&gt;]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assign control knob</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>hold</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>entry := 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>channel 1 deflection</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>channel 2 deflection</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>main sweep speed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>delayed sweep speed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>channel 1 position</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>channel 2 position</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>channel 1 position</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>channel 2 position</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>dual separation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>horizontal position</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>main trigger level</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>delayed trigger level</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>delay</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>delay</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>trace intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>character intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>panel intensity</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Select step resolution</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>coarse steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mode := 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fine steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELAY</td>
<td>DV&lt;value&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enter delay time</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(seconds)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>value := exponential</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[+a]+[d]+[e]+[a]+[d]+[d]</td>
<td>tult to 9 digits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>+0.00 to -99999999999</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-9999999999999 to</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>100 sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enter digital delay</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(delay trigger events:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>value := integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[d]+[d]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 to 999999999999</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DELTA TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>D&lt;state&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JT mode off</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>state := 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JT mode on (and zed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELTA VOLTS</td>
<td>DV&lt;channel&gt;[:&lt;mode&gt;]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>vertical channel 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>channel := 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>vertical channel 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>channel := 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JV off</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mode := 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JV on (and zed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HORIZONTAL MODE</td>
<td>HM&lt;mode&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>main</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>intensified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>delayed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>derived</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mode := 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- In volume parameters, leading zeros may be omitted.
- All parameters are for 80-byte output.
- In input parameters, all parameters are for 80-byte input.
- Control information must be transferred with UNT or UNT.
Here is the syntax used when programming the 3456A via HP-IB:

APPENDIX III. HP-IB PROGRAM CODES
HP-IB COMMANDS

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

<table>
<thead>
<tr>
<th>CONTROL LINES</th>
<th>DATA LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>REN</td>
<td>ATN</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>CONTROL LINES</th>
<th>DATA LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>REN</td>
<td>ATN</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>CONTROL LINES</th>
<th>DATA LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>REN ATN</td>
<td>OCTAL DEC</td>
</tr>
<tr>
<td>T T</td>
<td>077 63</td>
</tr>
<tr>
<td>T T</td>
<td>125 85</td>
</tr>
<tr>
<td>T T</td>
<td>051 41</td>
</tr>
<tr>
<td>T T</td>
<td>001 01</td>
</tr>
</tbody>
</table>

LOCAL 709 @ RESUME - sets ATN false

H-28
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:

<table>
<thead>
<tr>
<th>CONTROL LINES</th>
<th>DATA LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>REN</td>
<td>ATN</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

(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 110 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.
TRIGGER 7

* Must be active controller.

* All current listeners (capable of being triggered) are triggered.

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

<table>
<thead>
<tr>
<th>CONTROL LINES</th>
<th>DATA LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>REN</td>
<td>ATN</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>TRIGGER 709,708</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>T</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TRIGGER 7

| T | T | 010 | 08 | BS | GET |

H-32
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:

<table>
<thead>
<tr>
<th>CONTROL LINES</th>
<th>DATA LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>REN</td>
<td>ATN</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

OUTPUT 7

NO SEQUENCE GENERATED

H-33
NOTE ON THE FREEFIELD OUTPUT STATEMENT

eg OUTPUT 722; "F2"

The default freefield 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:

<table>
<thead>
<tr>
<th>ATN</th>
<th>OCTAL</th>
<th>DEC</th>
<th>ASCII</th>
<th>HP-IB</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>077</td>
<td>63</td>
<td>?</td>
<td>UNL unlisten</td>
</tr>
<tr>
<td>T</td>
<td>125</td>
<td>85</td>
<td>U</td>
<td>#21 talk</td>
</tr>
<tr>
<td>T</td>
<td>060</td>
<td>54</td>
<td>6</td>
<td>#22 listen</td>
</tr>
<tr>
<td>F</td>
<td>106</td>
<td>70</td>
<td>F</td>
<td>Data</td>
</tr>
<tr>
<td>F</td>
<td>062</td>
<td>71</td>
<td>2</td>
<td>Data</td>
</tr>
<tr>
<td>F</td>
<td>040</td>
<td>32</td>
<td></td>
<td>Data</td>
</tr>
</tbody>
</table>

19 ASCII BLANKS

| F   | 015   | 13   | CR    | Terminator |
| F   | 012   | 10   | LF    | Terminator |

H-34
ENTER STATEMENT

Used to get information from a device on the bus. It unlists 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:

<table>
<thead>
<tr>
<th>CONTROL LINES</th>
<th>DATA LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>REN</td>
<td>ATN</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

H-35
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.)
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 #
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

160 SEND 7; CMD"?Wn5"
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:
STATUOS OF THE BUS AND DEVICES ON IT

HP-IB

status registers
HP-IB

status registers
Device 22

status registers
Device 13

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.
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 ??

---

### HP-IB Status Registers

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

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

Q = S POLL (device selector)

\[
\begin{align*}
Q &= \text{S POLL (723)} & \text{reads the status register of device 23 on bus 7} \\
Q &= \text{S POLL (7)} & \text{read status byte of current talker}
\end{align*}
\]

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 = SPOOL (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 = SPOOL (723)
130  IF BIT (X,6) = 0 THEN GOTO 200: NOT 3455
140  IF BIT (X,6) 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 ??

**STATUS BYTE MESSAGE**

<table>
<thead>
<tr>
<th>b8</th>
<th>b7</th>
<th>b6</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Service Request (SRO)

bits 5 and 6 set high

DATA READY

SYNTAX ERROR

BINARY FUNCTION ERROR

TRIGGER TOO FAST

**NOTE**

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

<table>
<thead>
<tr>
<th>Status Byte Code</th>
<th>ASCII CHAR</th>
<th>Decimal Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>63</td>
<td>Data Ready - Indicates to the controller that measurement data is available. Applies to DATA READY Request feature.</td>
</tr>
<tr>
<td>B</td>
<td>66</td>
<td>Syntax Error - Indicates improper program code. Example - Program Code &quot;FF&quot; would cause a syntax error since the FUNCTION program set is only defined for codes F1 through F6.</td>
</tr>
<tr>
<td>D</td>
<td>68</td>
<td>BINARY FUNCTION Error - Indicates improper BINARY PROGRAM code or incomplete binary message. Similar to syntax error.</td>
</tr>
<tr>
<td>N</td>
<td>72</td>
<td>Trigger too Fast - Indicates the 3455A has been triggered while measurement data is being output to the bus. Warns of possible incorrect measurement information.</td>
</tr>
</tbody>
</table>

H-43
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.
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

<table>
<thead>
<tr>
<th>CONTROL LINES</th>
<th>DATA LINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>REN</td>
<td>ATN</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>PASS CONTROL 715</td>
<td>T</td>
</tr>
<tr>
<td>(from device #21)</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>

| PASS CONTROL 7 |     |      |      |       |      |
| T | 077 | 63 | ? | UNL unlisten |
| T | 011 | 09 | HT | TCT take control |
| F |      |      |     |       |      |

H-45
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.
HP-IB OPERATION IN GREATER DETAIL

Refer to sections of

Tutorial Description of the
Hewlett-Packard Interface Bus
(PN 5952-0156)
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 (e.g., 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!
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

<table>
<thead>
<tr>
<th>Branch Type</th>
<th>Select Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>ON ERROR</td>
<td></td>
</tr>
<tr>
<td>ON INTR</td>
<td>2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>ON TIMEOUT</td>
<td>10 11 12 13 14 15 16 17</td>
</tr>
<tr>
<td>ON EOT</td>
<td>18 19 20 21 22 23 24 25</td>
</tr>
<tr>
<td>ON KEY</td>
<td></td>
</tr>
<tr>
<td>ON TIMER</td>
<td></td>
</tr>
</tbody>
</table>

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.
INTERRUPT IMPLEMENTATION

XXX ! MAIN PROGRAM
XXX
XXX INTERRUPT(S) SETUP
XXX MAIN PROGRAM EXECUTION
X+1 do something
X+2 do something
X+3 do something
X+4 do something --- if interrupted here
X+5 do something --- will return here!
X+6 do something
X+7 do something
X+8 do something
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 set up 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.
3 Timers Available (1, 2, 3)

Activated with:

ON TIMER #1, 10000 GOTO 2050
ON TIMER #2, 2450 GOSUB 5000

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 999999999.

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 UP"
      :I
140 RETURN

WHAT HAPPENS?

NOTES:

Change the 5000 to 700. What happens?

Change the GOSUB / RETURN TO GOTO's

Any changes?
SPECIAL FUNCTION KEYS

HP-85 CRT

... 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 mathematical calculation
B. Allow a variable value to be examined

Undesirable Aspects:
A. Unintentional hit of key halts "Unattended" Program.

NOTES:

Try
10 I = 1
20 DISP I
30 I = I + 1
40 GOTO 20
50 END

?? What keys do not 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:

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Decimal Value</th>
<th>Operating Mode</th>
<th>Keys Masked</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>128</td>
<td>Program</td>
<td>RESET</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td></td>
<td>PAUSE</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>Execution</td>
<td>Special Function Keys and KEYLABEL</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td></td>
<td>Other keys</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td></td>
<td>RESET</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Keyboard Input</td>
<td>PAUSE</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td>Special Function Keys and KEYLABEL</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>Other keys</td>
</tr>
</tbody>
</table>

Setting a bit (=1) enables the corresponding Keys
Clearing a bit (=0) 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
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 10 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 ??
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 ! INTERRUPT 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 ; AE PRINT A
310 !
320 ! MUST SERIAL POLL TO
330 ! TO CLEAR INSTRUMENT SRQ
340 D=SPOLL(707) @ 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
TIMEOUTS

Provides a method of aborting an interface handshake that does not occur within a specified 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 occurs.

OFF TIMEOUT 7 Canceles the ON TIMEOUT statement.

SET TIMEOUT 7;0 - Almost infinite timeout.
10 ! TIMEOUT EXAMPLE
20 ! 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
ERROR TRAPPING

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

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:
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 UNINTENTIONAL 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 desirable to have the CPU do some useful work instead of sitting around idle, waiting on a slow IO transfer. By pre- or post-formating 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.
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.

**OUTPUT Z$ USING 10: AS (X), I, 1.23**

<table>
<thead>
<tr>
<th>Program variables and data AS(X),I,1.23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formatting (USING)</td>
</tr>
<tr>
<td>CONVERT Out conversion table</td>
</tr>
<tr>
<td>(Fill)</td>
</tr>
<tr>
<td>IOBUFFER Z$</td>
</tr>
<tr>
<td>output TRANSFER (Empty)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program variables A(I),X,B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formatting (USING)</td>
</tr>
<tr>
<td>CONVERT In conversion table</td>
</tr>
<tr>
<td>(Empty)</td>
</tr>
</tbody>
</table>

**ENTER Z$ USING 20: A (I), X, B$**

**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;

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.
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.

PROGRAM EXECUTION HALTED

PROGRAM EXECUTION CONTINUES

Buffers-5
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

START TRANSFER
PROGRAM EXECUTION
DONE
PROGRAM EXECUTION CONTINUES

Buffers-6
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.

Buffers-7
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 shows 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 XS [10000]
...  IOBUFFER XS
...  
...  TRANSFER 724 TO XS INTR; COUNT 700
...  ! DO
...  ! OTHER THINGS
...  
500 FORMAT DATA

NOTES:
3437 EXAMPLE

+1.644,+1.423,+1.281,+1.172,+1.0
+0.831,+0.801,+0.776,+0.755,+0.73
+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,+0.628,+0.627,+0.626,+0.625,+0.625,+0.6
+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     0.957
8     0.907
9     0.867
10    0.831
11    0.801
12    0.776
13    0.755
14    0.73

10 ABORT 10 7
20 CLEAR 724
30 ON EOT 7 GOTO 150
50 OPTION BASE 1
60 OUTPUT 724 ;"R2T1N100SD.1SE7S"
70 DIM A$(72000),V(1000)
80 10BUFFEP A$
90 STATUS 7,1 ; A
100 ENABLE INTR 7,8
110 TRANSFER 724 TO A$ INTR ; C0UNT 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" ;
190 V(I)
200 DISP I:V(I)
200 NEXT I
210 END

Buffers-11
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 AN INTERRUPT 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:
GRAPHICS

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.
GRAPHICS

GCLEAR

PURPOSE: CLEARS THE GRAPHICS DISPLAY.

SYNTAX: GCLEAR [Y-COORDINATE]

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

NOTES:

sales

G-2
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 a DRAW X, SIN(X) a NEXT X
SCALE

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
SCALE EXAMPLE

SCALE 1966, 1980, -700, 3000

NOTES:
XAXIS / YAXIS

PURPOSE:

DRAW 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:
XAXIS & YAXIS EXAMPLE

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

NOTEs:
MOVE

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

SYNTAX:
MOVE X-COORDINATE , Y-COORDINATE
DRAW

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)
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 $x = -180^\circ$ TO $x = 180^\circ$. 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.
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

SET DEGREES

CLEAR DISPLAY

SCALE DISPLAY

DRAW AXES

MOVE TO INITIAL POINT

DRAW FUNCTION

END

G-14
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 $ 00

NOTES:
LDI R

PURPOSE:
SETS ORIENTATION FOR LABEL

SYNTAX:
LDIR NUMERIC EXPRESSION

EXAMPLE:
80 DEG
90 LDIR 90

RULE:
\[ \geq 45, \text{ VERTICAL} \]
\[ \leq 45, \text{ HORIZONTAL} \]

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$(I)
90 LDIR 0
100 MOVE -10, I
110 LABEL VAL$(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

<table>
<thead>
<tr>
<th>YEAR</th>
<th>MILLIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1969</td>
<td>336</td>
</tr>
<tr>
<td>1970</td>
<td>365</td>
</tr>
<tr>
<td>1971</td>
<td>378</td>
</tr>
<tr>
<td>1972</td>
<td>483</td>
</tr>
<tr>
<td>1973</td>
<td>669</td>
</tr>
<tr>
<td>1974</td>
<td>893</td>
</tr>
<tr>
<td>1975</td>
<td>985</td>
</tr>
<tr>
<td>1976</td>
<td>1121</td>
</tr>
<tr>
<td>1977</td>
<td>1358</td>
</tr>
<tr>
<td>1978</td>
<td>1737</td>
</tr>
<tr>
<td>1979</td>
<td>2361</td>
</tr>
</tbody>
</table>
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$ "00"
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, 356
220 FOR X=1970 TO 1979
230 READ Y
240 DRAW X,Y
250 NEXT X
260 DATA 365, 378, 483, 669, 893, 985
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’ ALTERNATELY (OR USE ‘DISP’ AND ‘LABEL’ APPROPRIATELY).

NOTES:
Graphics Statements

ALPHA
BFLCT character string, number of characters per line
DRAW x-coordinate , y-coordinate
GCLEAR [y]
GRAPH
IDRAW x-increment, y-increment
IMOVE x-increment, y-increment
LABEL character string
LDIF numeric expression
MCUE x-coordinate, y-coordinate
FN numeric expression
FINUP
PLCT x-coordinate, y-coordinate
SCALE xmin, xmax, ymin, ymax
YAXIS y-intercept [ , tic length [ , xmin , xmax ] ]
YAXIS x-intercept [ , tic length [ , ymin , ymax ] ]
130 1
140 1 FIND MAX/MIN VOLTAGES*
150 1
160 U=CEIL(AMAX(D))
165 IF U=1 THEN U=U+1
170 1
180 1
190 CLEAR
200 SCALE 0,1.2=IN,-1.2=IN,1.2=IN
210 1
220 ! BORDER FRAME
230 XAXIS -1.2=IN @ XAXIS 1.2=IN
240 YAXIS 0 @ YAXIS 1.2=IN
250 1
260 ! X AXIS SETUP
270 1
280 IF U=1 THEN B=.1 ELSE B=.1
290 IF N=200 THEN B2=10 ELSE B2=100
300 1
310 FOR I=U TO -V STEP -B
320 XAXIS I,0,.15=IN,1.15=IN
330 NEXT I
340 1

350 ! Y AXIS SETUP
360 1
370 FOR I=.15=IN TO 1.15=IN STEP B2
380 YAXIS I,0,-V,V
390 NEXT I
400 1
410 ! Y AXIS LABELING
420 1
430 LDIR 0
440 FOR I=1 TO -V STEP -V
450 MOVE .04=IN,I @ LABEL I
460 NEXT I
470 MOVE .05=IN,-V/4
480 LDIR 90 @ LABEL "VOLTAGE"
490 1
500 ! X AXIS LABELING
510 1
520 MOVE .3=IN,-1.15=IN @ LDIR 0
530 LABEL "ELAPSED TIME IN SEC"
540 C=1 @ LDIR 90
550 FOR I=.25=IN TO 1.15=IN STEP B2
560 MOVE I,-1.05=V
570 LABEL B2=IN+C
580 C=C+1 @ NEXT I
590 1
600 ! PLOTTING
610 1
620 FOR I=1 TO N
630 PLOT I-1+.15=IN,D(I)
640 NEXT I
650 MOVE .17=IN,1.05=IN @ LDIR 0
660 LABEL N,"SAMPLES IN"/IN,"HZ RATE"
670 END

G-21
BIT FUNCTION

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

Numeric expression argument 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 HEXDECIMAL: returns a string, 4 characters long,
representing the hexadecimal equivalent 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".

M-2
BINARY TO DECIMAL
BTD(<string exp>)

DECIMAL TO BINARY
DTBS($<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:

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

for example:  BINAND (7,5)
             yields 5

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

NUM EXP1
0 1
k--k--k

NUM EXP2
0: 0: 0:
1: 0: 1:
s--s--s

NOTES:
Numeric expressions 1 and 2 are ANDed bit-by-bit returning an integer result.
BINOR (<NUM EXP1>,<NUM EXP2>)

for example: BINOR (7,5)
yields 7

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

NUM EXP1
0 1
k---k---k

NUM 0: 0 : 1 :
EXP2 :---:'---'
1: 1 : 1 :
  s---s---s

Numeric expression argument range: -32768 to 32767

NOTES:
Numeric expressions 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

NUM EXP1
0 1
k---k---k

NUM EXP2
0: 0 : 1 :
:-:-:-:
1: 1 : 0 :
s---s---s

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

Numeric expression argument range: -32768 to 32767
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

M-8
### Summary of OUTPUT Image Specifiers

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

### Summary of ENTER Image Specifiers

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

<table>
<thead>
<tr>
<th>NP-IB</th>
<th>ASCII</th>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESSED COMMAND GTL</td>
<td>NUL</td>
<td>0 00 000 000</td>
<td>000</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>Group AGC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDC</td>
<td>EOT</td>
<td>0 00 100 000</td>
<td>004</td>
<td>04</td>
<td>T4</td>
</tr>
<tr>
<td>PPC</td>
<td>ENQ</td>
<td>0 00 001 005</td>
<td>005</td>
<td>05</td>
<td>T5</td>
</tr>
<tr>
<td>ACK</td>
<td>BEL</td>
<td>0 00 110 006</td>
<td>006</td>
<td>06</td>
<td>T6</td>
</tr>
<tr>
<td>GET</td>
<td>BS</td>
<td>0 00 011 007</td>
<td>007</td>
<td>07</td>
<td>T7</td>
</tr>
<tr>
<td>TCT</td>
<td>LT</td>
<td>0 00 111 010</td>
<td>010</td>
<td>08</td>
<td>T8</td>
</tr>
<tr>
<td></td>
<td>VT</td>
<td>0 00 011 013</td>
<td>013</td>
<td>0B</td>
<td>T9</td>
</tr>
<tr>
<td></td>
<td>FF</td>
<td>0 00 100 014</td>
<td>014</td>
<td>0C</td>
<td>T12</td>
</tr>
<tr>
<td></td>
<td>CR</td>
<td>0 00 101 015</td>
<td>015</td>
<td>0D</td>
<td>T13</td>
</tr>
<tr>
<td></td>
<td>SO</td>
<td>0 00 110 018</td>
<td>018</td>
<td>0E</td>
<td>T14</td>
</tr>
<tr>
<td></td>
<td>SI</td>
<td>0 00 111 017</td>
<td>017</td>
<td>0F</td>
<td>T15</td>
</tr>
<tr>
<td></td>
<td>Universal Command LLO</td>
<td>DLE</td>
<td>0 01 000 020</td>
<td>020</td>
<td>10</td>
</tr>
<tr>
<td>Group UCG</td>
<td>DC1</td>
<td>0 01 001 021</td>
<td>021</td>
<td>11</td>
<td>T17</td>
</tr>
<tr>
<td></td>
<td>DC2</td>
<td>0 01 001 022</td>
<td>022</td>
<td>12</td>
<td>T18</td>
</tr>
<tr>
<td></td>
<td>DC3</td>
<td>0 01 001 023</td>
<td>023</td>
<td>13</td>
<td>T19</td>
</tr>
<tr>
<td></td>
<td>DCL</td>
<td>DC4</td>
<td>0 01 010 024</td>
<td>024</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>PPU</td>
<td>NAK</td>
<td>0 01 101 025</td>
<td>025</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>ETB</td>
<td>ET8</td>
<td>0 01 111 027</td>
<td>027</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CAN</td>
<td>0 01 111 030</td>
<td>030</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>SPD</td>
<td>EW</td>
<td>0 01 111 031</td>
<td>031</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUB</td>
<td>0 01 111 032</td>
<td>032</td>
<td>1A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ESC</td>
<td>0 01 111 033</td>
<td>033</td>
<td>1B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FS</td>
<td>0 01 111 034</td>
<td>034</td>
<td>1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GS</td>
<td>0 01 111 035</td>
<td>035</td>
<td>1D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RS</td>
<td>0 01 111 036</td>
<td>036</td>
<td>1E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US</td>
<td>0 01 111 037</td>
<td>037</td>
<td>1F</td>
</tr>
<tr>
<td></td>
<td>LISTEN</td>
<td>L0</td>
<td>00 100 040</td>
<td>040</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Group - LAG Note 1</td>
<td>L1</td>
<td>00 100 041</td>
<td>041</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L2</td>
<td>00 100 042</td>
<td>042</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L3</td>
<td>00 100 043</td>
<td>043</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L4</td>
<td>00 100 044</td>
<td>044</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L5</td>
<td>00 100 045</td>
<td>045</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L6</td>
<td>00 100 046</td>
<td>046</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L7</td>
<td>00 100 047</td>
<td>047</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L8</td>
<td>00 101 050</td>
<td>050</td>
<td>2A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L9</td>
<td>00 101 051</td>
<td>051</td>
<td>2B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L10</td>
<td>00 101 052</td>
<td>052</td>
<td>2C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L11</td>
<td>00 101 053</td>
<td>053</td>
<td>2D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L12</td>
<td>00 101 054</td>
<td>054</td>
<td>2E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L13</td>
<td>00 101 055</td>
<td>055</td>
<td>2F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L14</td>
<td>00 101 056</td>
<td>056</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L15</td>
<td>00 101 057</td>
<td>057</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L16</td>
<td>00 101 058</td>
<td>058</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L17</td>
<td>00 101 059</td>
<td>059</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L18</td>
<td>00 101 060</td>
<td>060</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L19</td>
<td>00 101 061</td>
<td>061</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L20</td>
<td>00 101 062</td>
<td>062</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L21</td>
<td>00 101 063</td>
<td>063</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L22</td>
<td>00 110 070</td>
<td>070</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L23</td>
<td>00 110 071</td>
<td>071</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L24</td>
<td>00 110 072</td>
<td>072</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L25</td>
<td>00 110 073</td>
<td>073</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L26</td>
<td>00 110 074</td>
<td>074</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L27</td>
<td>00 110 075</td>
<td>075</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L28</td>
<td>00 110 076</td>
<td>076</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L29</td>
<td>00 110 077</td>
<td>077</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L30</td>
<td>00 110 078</td>
<td>078</td>
<td>46</td>
</tr>
</tbody>
</table>

| Secondary Command Group SCG | SOT | 0 100 000 064 | 064 | 34 | S20 |
| | | SOT | 0 100 000 065 | 065 | 35 | S21 |
| | | SOT | 0 100 000 066 | 066 | 36 | S22 |
| | | SOT | 0 100 000 067 | 067 | 37 | S23 |
| | | SOT | 0 100 000 070 | 070 | 38 | S24 |
| | | SOT | 0 100 000 071 | 071 | 39 | S25 |
| | | SOT | 0 100 000 072 | 072 | 40 | S26 |
| | | SOT | 0 100 000 073 | 073 | 41 | S27 |
| | | SOT | 0 100 000 074 | 074 | 42 | S28 |
| | | SOT | 0 100 000 075 | 075 | 43 | S29 |
| | | SOT | 0 100 000 076 | 076 | 44 | S30 |
| | | SOT | 0 100 000 077 | 077 | 45 | S31 |

R-2
Commands

Non-Programmable

AUTO [beginning line number [, increment value]]
CCNT [statement number]
DELETE first statement number [, last statement number]
INIT
LOAD program name
REN [first statement number [, increment value]]
RUN [statement number]
SCRATCH
STCRE program name
UNSECURE file name, security code, secure type

Programmable

CAT
COPY
CTAPE
ERASETAPE
FLIP
LIST [beginning statement number [, ending statement number]]
PLIST [beginning statement number [, ending statement number]]
PRINT ALL
REWIND
SECURE file name, security code, secure type
BASIC Statements

ASSIGN# buffer number TO file name
ASSIGN# buffer number TO *
BEEP [tone, duration]
CHAIN file name
CLEAR
COMMON common variable list
CRT IS output code number
CREATE file name, number of records [ , number of bytes per record]
DATA data list
DEFAULT CFF
DEFAULT CN
DEF FN numeric variable name [ <parameter> ] [ = numeric expression ]
DEF FN string variable name [ <parameter> ] [ = string expression ]
DEG
DIM dimension list
DISPLAY display list
DISPLAY USING image format string [ ; disp using list ]
DISPLAY USING statement number [ ; disp using list ]
END
FN END
FOR loop counter = initial value TO final value [ STEP increment value ]
GOSUB statement number
GOTO statement number
GRAD
IF numeric expression THEN statement number [ ELSE statement number ]
or executable statement [ executable statement ]
IMAGE image format string
INPUT variable name, [ , variable name[, ]
INTEGER numeric variable [ <subscripts> ] [ , numeric variable <subscripts> ] [ , ]
KEY LABEL
[LET] numeric variable, [ , numeric variable, ] [ = numeric expression
[LET] string variable, [ , string variable, ] [ = string expression
[LET] FN variable name = expression
LOAD BIN file name
NEXT loop counter
NORMAL
BASIC Statements

CFF ERROR
CFF KEY# key number
CFF TIMER# timer number
ON ERROR GOSUB statement number
ON ERROR GOTO statement number
ON numeric expression GOSUB statement number list
ON numeric expression GOTO statement number list
ON KEY# key number [, key label] GOSUB statement number
ON KEY# key number [, key label] GOTO statement number
ON TIMER# timer number, milliseconds GOSUB statement number
ON TIMER# timer number, milliseconds GOTO statement number
OPTION BASE 1 or 0
PAUSE
PRINT [print list]
PRINT # buffer number ; print # list
PRINT # buffer number ; record number [ ; print # list]
PRINT USING image format string [ ; print using list]
PRINT USING statement number [ ; print using list]
PRINTER IS output code number
PURGE file name [, purge code number]
RAD
RANDOMIZE [numeric expression]
READ variable name, [ , variable name, ...]
READ # buffer number ; variable list
READ # buffer number ; record number [ ; variable list]
REAL numeric variable [ , numeric variable [ , numeric variable [ , subscripts ] ] ]
REM [any combination of characters]
RENAME old file name TO new file name
RESTORE [statement number]
RETURN
SETTIME seconds since midnight, Julian day in form yyddd
SHCFT numeric variable [ , numeric variable [ , subscripts ] ]
STCP
STCPE E IN file name
TRACE
TRACE ALL
TRACE UAP variable, [ , variable, ...]
WRITE number of milliseconds
Graphics Statements

ALPHA
EPLCT character string, number of characters per line
DPhl x-coordinate, y-coordinate
GCLEAP [y]
GRAPH
IDPhl x-increment, y-increment
IMOVE x-increment, y-increment
LABEL character string
LDIP numeric expression
MCVE x-coordinate, y-coordinate
FEN numeric expression
PENUP
PLCT x-coordinate, y-coordinate
SCALE xmin, xmax, ymin, ymax
XAXIS y-intercept [, tic length [ , xmin, xmax ]]
YAXIS x-intercept [, tic length [ , ymin, ymax ]]
### BASIC Predefined Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS(X)</td>
<td>Absolute value of X.</td>
<td>60</td>
</tr>
<tr>
<td>ACOS(X)</td>
<td>Arcosine of X, in 1st or 2nd quadrant.</td>
<td>66</td>
</tr>
<tr>
<td>ASIN(X)</td>
<td>Arccosine of X, in 1st or 4th quadrant.</td>
<td>66</td>
</tr>
<tr>
<td>ATAN(X)</td>
<td>Arctangent of X, in 1st or 4th quadrant.</td>
<td>66</td>
</tr>
<tr>
<td>ATAN2(Y,X)</td>
<td>Arctangent of Y/X, in proper quadrant.</td>
<td>67</td>
</tr>
<tr>
<td>CEIL(X)</td>
<td>Smallest integer &gt;=X.</td>
<td>66</td>
</tr>
<tr>
<td>CHR(X)</td>
<td>Character whose decimal character code is X, 0&lt;=X&lt;=255.</td>
<td>66</td>
</tr>
<tr>
<td>COS(X)</td>
<td>Cosine of X.</td>
<td>66</td>
</tr>
<tr>
<td>COT(X)</td>
<td>Cotangent of X.</td>
<td>66</td>
</tr>
<tr>
<td>CSC(X)</td>
<td>Cosecant of X.</td>
<td>66</td>
</tr>
<tr>
<td>DATE</td>
<td>Julian date in format yyddd (assumes system timer has been set properly).</td>
<td>57</td>
</tr>
<tr>
<td>DTR(X)</td>
<td>Degree to radian conversion.</td>
<td>67</td>
</tr>
<tr>
<td>EPS</td>
<td>Smallest positive machine number (1E-499).</td>
<td>64</td>
</tr>
<tr>
<td>ERRL</td>
<td>Line number of latest error.</td>
<td>64</td>
</tr>
<tr>
<td>ERRN</td>
<td>Number of latest error.</td>
<td>65</td>
</tr>
<tr>
<td>EXP(X)</td>
<td>e^a</td>
<td>65</td>
</tr>
<tr>
<td>FLOOR(X)</td>
<td>Same as INT(X) (relates to CEIL).</td>
<td>61</td>
</tr>
<tr>
<td>FP(X)</td>
<td>Fractional part of X.</td>
<td>60</td>
</tr>
<tr>
<td>INF</td>
<td>Largest machine number (9.99999999E499).</td>
<td>64</td>
</tr>
<tr>
<td>INT(X)</td>
<td>Largest integer &lt;=X.</td>
<td>61</td>
</tr>
<tr>
<td>IP(X)</td>
<td>Integer part of X.</td>
<td>60</td>
</tr>
<tr>
<td>LEN(SS)</td>
<td>Length of string SS.</td>
<td>128</td>
</tr>
<tr>
<td>LGT(X)</td>
<td>Log to the base 10 of X, X&gt;0.</td>
<td>65</td>
</tr>
<tr>
<td>LGC(X)</td>
<td>Natural logarithm, X&gt;0.</td>
<td>65</td>
</tr>
<tr>
<td>MAX(X,Y)</td>
<td>If X&gt;Y then X, else Y.</td>
<td>62</td>
</tr>
<tr>
<td>MIN(X,Y)</td>
<td>If X&lt;Y then X, else Y.</td>
<td>62</td>
</tr>
<tr>
<td>NUM(SS)</td>
<td>Decimal character code of first character of SS.</td>
<td>132</td>
</tr>
<tr>
<td>PI</td>
<td>3.14159265359</td>
<td>63</td>
</tr>
<tr>
<td>POS(S15, S25)</td>
<td>Searches string S15 for the first occurrence of string S25. Returns starting index if found, otherwise returns 0.</td>
<td>129</td>
</tr>
<tr>
<td>PMD(X,Y)</td>
<td>Remainder of X/Y: X - Y * IP(X/Y).</td>
<td>62</td>
</tr>
<tr>
<td>RND</td>
<td>Next number, X, in a sequence of pseudo-random numbers. 0 &lt;= X &lt; 1.</td>
<td>64</td>
</tr>
<tr>
<td>RTO(X)</td>
<td>Radian to degree conversion.</td>
<td>67</td>
</tr>
<tr>
<td>SEC(X)</td>
<td>Secant of X.</td>
<td>66</td>
</tr>
<tr>
<td>SGN(X)</td>
<td>The sign of X, -1 if X&lt;0, 0 if X=0, and +1 if X&gt;0.</td>
<td>62</td>
</tr>
<tr>
<td>SIN(X)</td>
<td>Sine of X.</td>
<td>66</td>
</tr>
<tr>
<td>SQRT(X)</td>
<td>Positive square root of X.</td>
<td>66</td>
</tr>
<tr>
<td>TABN</td>
<td>Skips to specified column.</td>
<td>168</td>
</tr>
<tr>
<td>TAN(X)</td>
<td>Tangent of X.</td>
<td>66</td>
</tr>
<tr>
<td>TIME</td>
<td>Time in seconds since midnight (assumes system timer has been set properly).</td>
<td>57</td>
</tr>
<tr>
<td>UCASE(SS)</td>
<td>Returns string with all lower-case alphabetic characters converted to upper-case.</td>
<td>133</td>
</tr>
<tr>
<td>VAL(SS)</td>
<td>Returns the numeric equivalent of the string SS.</td>
<td>130</td>
</tr>
<tr>
<td>VAL(X)</td>
<td>String equivalent of X.</td>
<td>131</td>
</tr>
</tbody>
</table>
## Error Messages

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Condition</th>
<th>Default values (errors 1-8 only) with DEFAULT ON</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Math Errors (1 thru 13)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Underflow: expression underflows machine</td>
<td>0</td>
</tr>
</tbody>
</table>
| 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. | ±9.99999999999E499  
±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. | 9.99999999999E499 |
| 7 | Null data:  
  - Uninitialized string variable, or missing string function assignment.  
  - Uninitialized numeric variable, or missing numeric function assignment. | 9.99999999999E499 |
| 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:  
  - ΘRTN(0,0),  
  - ASN or ACSN(−1<n<1).  
  - CN expression GOTO/GOSUB; expression of range. | | |
<p>| 12 | Logarithm of zero. | | |
| 13 | Logarithm of negative number. | | |
| 14 | Not used. | | |
| <strong>System Errors (15 thru 25)</strong> | | |
| 15 | System error: correct by reloading program, pressing [ESC], or turning system off, then on again. | | |
| 16 | Continue before run: program not allocated. | | |
| 17 | FCF nesting too deep: more than 255 active FOR-NEXT loops. | | |
| 18 | GOSUB nesting too deep: more than 255 nested subroutines. | | |</p>
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Condition</th>
</tr>
</thead>
</table>
| 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. |
| 30           | CPTICHN BASE error:  
|              | - Duplicate CPTICHN BASE declaration.  
|              | - CPTICHN BASE after array declaration.  
|              | - CPTICHN 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:  
|              | - FERD variable and DATA type do not agree.  
|              | - READ # found a string but required a number. |
| 34           | No DATA to read:  
|              | - FERD and DATA expired.  
|              | - FESTCRE 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 CPTICHN 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.  
<p>|              | - Exiting a function that was not entered with a function call after branching to the middle of a multi-line function. |</p>
<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Illegal function parameter; function parameter mismatch (e.g., declared as string, called as numeric).</td>
</tr>
<tr>
<td>41</td>
<td>FN=; user-defined function assignment. Function assignment does not occur between DEF FN and FN END.</td>
</tr>
<tr>
<td>42</td>
<td>Recursive user-defined function.</td>
</tr>
<tr>
<td>43</td>
<td>Numeric input wanted.</td>
</tr>
<tr>
<td>44</td>
<td>Too few inputs. Less items were given than requested by an INPUT statement.</td>
</tr>
<tr>
<td>45</td>
<td>Too many inputs. More items were given than requested by an INPUT statement.</td>
</tr>
<tr>
<td>46</td>
<td>NEXT missing; FCR with no matching NEXT.</td>
</tr>
<tr>
<td>47</td>
<td>FCR missing; NEXT with no matching FCR.</td>
</tr>
<tr>
<td>48</td>
<td>END statement necessary.</td>
</tr>
<tr>
<td>50</td>
<td>Null data; uninitialized data.</td>
</tr>
<tr>
<td>51</td>
<td>Binary program missing; attempting to RUN program that requires binary program. An attempt to edit will usually SCRATCH memory.</td>
</tr>
<tr>
<td>52</td>
<td>RETURN without GOSUB reference.</td>
</tr>
<tr>
<td>53</td>
<td>Illegal IMAGE format string; unrecognized character in IMAGE.</td>
</tr>
<tr>
<td>54</td>
<td>Illegal PRINT USING:</td>
</tr>
<tr>
<td></td>
<td>- Data overflows IMAGE declaration.</td>
</tr>
<tr>
<td></td>
<td>- Numeric data with string IMAGE.</td>
</tr>
<tr>
<td></td>
<td>- String data with numeric IMAGE.</td>
</tr>
<tr>
<td></td>
<td>- PRINT USING image format string is not correct.</td>
</tr>
<tr>
<td>55</td>
<td>Illegal TAB argument. With DEFAULT OH, an illegal TAB argument gives a warning message and defaults to TAB(1).</td>
</tr>
<tr>
<td>56</td>
<td>Array subscript out of range.</td>
</tr>
<tr>
<td>57</td>
<td>String variable overflow; string too big for variable.</td>
</tr>
<tr>
<td>58</td>
<td>Missing line; reference to a nonexistent statement number.</td>
</tr>
<tr>
<td>59 thru 75</td>
<td>Not used.</td>
</tr>
<tr>
<td>60</td>
<td>Tape Errors (60 thru 75)</td>
</tr>
<tr>
<td>61</td>
<td>Tape cartridge is write-protected; RECORD slide tab is in left-most position.</td>
</tr>
<tr>
<td>62</td>
<td>Attempting to create/record more than 42 files on tape.</td>
</tr>
<tr>
<td>63</td>
<td>Cartridge out when attempting tape operations.</td>
</tr>
<tr>
<td>64</td>
<td>Duplicate file name for PEND or CREATE.</td>
</tr>
<tr>
<td>65</td>
<td>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.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Condition</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>65</td>
<td>End of tape:</td>
</tr>
<tr>
<td></td>
<td>• Tape run-off; check cartridge.</td>
</tr>
<tr>
<td></td>
<td>• Tape is full.</td>
</tr>
<tr>
<td></td>
<td>• Not enough space to CREATE data file.</td>
</tr>
<tr>
<td>66</td>
<td>File closed:</td>
</tr>
<tr>
<td></td>
<td>• Attempting READ#PRINT# to file that has not been opened with ASSIGN#.</td>
</tr>
<tr>
<td></td>
<td>• Attempting to close a closed file (warning only).</td>
</tr>
<tr>
<td></td>
<td>• Tape has been ejected and reinserted.</td>
</tr>
<tr>
<td>67</td>
<td>File name:</td>
</tr>
<tr>
<td></td>
<td>• Name does not exist when attempt to LOAD, ASSIGN#, LOAD BIN, PURGE,</td>
</tr>
<tr>
<td></td>
<td>RENAME, or SECURE.</td>
</tr>
<tr>
<td></td>
<td>• Name not in quotes.</td>
</tr>
<tr>
<td></td>
<td>• Attempt to PURGE an open file.</td>
</tr>
<tr>
<td>68</td>
<td>File type mismatch:</td>
</tr>
<tr>
<td></td>
<td>• Attempting to treat program as data file, or vice versa.</td>
</tr>
<tr>
<td></td>
<td>• Attempting to treat binary program as BASIC main program file, or vice versa.</td>
</tr>
<tr>
<td></td>
<td>• Attempting to treat data as binary program, or vice versa.</td>
</tr>
<tr>
<td>69</td>
<td>Random overflow: attempting to READ#PRINT# beyond existing number of bytes in logically-defined record with random file access.</td>
</tr>
<tr>
<td>70</td>
<td>READ error: system cannot read tape.</td>
</tr>
<tr>
<td>71</td>
<td>End-of-File: no data beyond EOF mark in data file.</td>
</tr>
<tr>
<td>72</td>
<td>Record:</td>
</tr>
<tr>
<td></td>
<td>• Attempting to READ#PRINT# to record that doesn’t exist; e.g., READ# 1, 120 when only 100 records in file.</td>
</tr>
<tr>
<td></td>
<td>• Attempting to READ#PRINT# at end of file.</td>
</tr>
<tr>
<td></td>
<td>• Lost in record: close file to release buffer.</td>
</tr>
<tr>
<td>73</td>
<td>Searches and does not find:</td>
</tr>
<tr>
<td></td>
<td>• Bad tape cartridge: may have been exposed to magnetic field.</td>
</tr>
<tr>
<td></td>
<td>• Cannot find directory, tape may need to be initialized.</td>
</tr>
<tr>
<td>74</td>
<td>Stall; either bad tape cartridge or transport problem, refer to Tape Operations, appendix B.</td>
</tr>
<tr>
<td>75</td>
<td>Not an HP-85 file; cannot read.</td>
</tr>
<tr>
<td>76 thru 75</td>
<td>Not used.</td>
</tr>
<tr>
<td>80</td>
<td>Syntax Errors (80 thru 92)</td>
</tr>
<tr>
<td>81</td>
<td>Right parentheses, ), expected.</td>
</tr>
<tr>
<td>82</td>
<td>Bad BASIC statement or bad expression. If it is an expression, try it again with DISP &lt;expression&gt; to get a better error message.</td>
</tr>
<tr>
<td>83</td>
<td>String expression error; e.g., right quote missing or null string given for file name.</td>
</tr>
<tr>
<td>84</td>
<td>Comma missing or more parameters expected (separated by commas).</td>
</tr>
<tr>
<td>85</td>
<td>Excess characters: delete characters at end of good line, then press [Esc].</td>
</tr>
<tr>
<td></td>
<td>Expression too big for system to interpret.</td>
</tr>
<tr>
<td>Error Number</td>
<td>Error Condition</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>86</td>
<td>Illegal statement after THEN.</td>
</tr>
<tr>
<td>87</td>
<td>Bad DIM statement.</td>
</tr>
<tr>
<td>88</td>
<td>Bad statement:</td>
</tr>
<tr>
<td></td>
<td>• CCM in calculator mode.</td>
</tr>
<tr>
<td></td>
<td>• User-defined function in calculator mode.</td>
</tr>
<tr>
<td></td>
<td>• INPUT in calculator mode.</td>
</tr>
<tr>
<td>89</td>
<td>Invalid parameter:</td>
</tr>
<tr>
<td></td>
<td>• CN KEY# less than 1 or greater than 8.</td>
</tr>
<tr>
<td></td>
<td>• Attempt to TRAPEZ a calculator mode variable.</td>
</tr>
<tr>
<td></td>
<td>• PRINTER IS or CRT IS with invalid parameter.</td>
</tr>
<tr>
<td></td>
<td>• CREATE with invalid parameters.</td>
</tr>
<tr>
<td></td>
<td>• ASSIGN#, PRINT#, or READ# with buffer number other than 1 through 10.</td>
</tr>
<tr>
<td></td>
<td>• Random READ# to record 0.</td>
</tr>
<tr>
<td></td>
<td>• SETTIME with illegal time parameter.</td>
</tr>
<tr>
<td></td>
<td>• CN TIME#, CFF TIME# with number other than 1, 2, or 3.</td>
</tr>
<tr>
<td></td>
<td>• SCALE with invalid parameters.</td>
</tr>
<tr>
<td></td>
<td>• AUTO or REN with invalid parameters.</td>
</tr>
<tr>
<td></td>
<td>• LIST with invalid parameters.</td>
</tr>
<tr>
<td></td>
<td>• DELETE with invalid parameters.</td>
</tr>
<tr>
<td></td>
<td>• VAL# with non-numeric parameter.</td>
</tr>
<tr>
<td></td>
<td>• Any statement, command, or function for which parameters are given but they are invalid.</td>
</tr>
<tr>
<td>90</td>
<td>Line number too large; greater than 9999.</td>
</tr>
<tr>
<td>91</td>
<td>Missing parameter; e.g., DELETE with missing or invalid parameters.</td>
</tr>
<tr>
<td>92</td>
<td>Syntax error. Cursor returns to character where error was found.</td>
</tr>
<tr>
<td>Error No.</td>
<td>Message</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>101</td>
<td>IFR</td>
</tr>
<tr>
<td>110</td>
<td>ICCARD</td>
</tr>
<tr>
<td>111</td>
<td>IOOPER</td>
</tr>
<tr>
<td>112</td>
<td>IOROM</td>
</tr>
<tr>
<td>Error No.</td>
<td>Message</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>113</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td></td>
</tr>
<tr>
<td>115</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td></td>
</tr>
<tr>
<td>Error No.</td>
<td>Message</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>117</td>
<td>An interface-dependent error. HP-IB: The statement used requires the</td>
</tr>
<tr>
<td></td>
<td>interface to be non-controller. Serial: This error number not currently</td>
</tr>
<tr>
<td></td>
<td>used. BCD: Interface command has been directed to a non-existent field.</td>
</tr>
<tr>
<td></td>
<td>GPIO: This error number not currently used.</td>
</tr>
<tr>
<td>118</td>
<td>An interface-dependent error. HP-IB: This error number not currently</td>
</tr>
<tr>
<td></td>
<td>used. Serial: This error number not currently used. BCD: Cannot start</td>
</tr>
<tr>
<td></td>
<td>operation because CTL line is not in the proper state. GPIO: This error</td>
</tr>
<tr>
<td></td>
<td>number not currently used.</td>
</tr>
<tr>
<td>119</td>
<td>An interface-dependent error. HP-IB: This error number not currently</td>
</tr>
<tr>
<td></td>
<td>used. Serial: This error number not currently used. BCD: Data format</td>
</tr>
<tr>
<td></td>
<td>does not match the mode of the interface. GPIO: This error number not</td>
</tr>
<tr>
<td></td>
<td>currently used.</td>
</tr>
<tr>
<td>120</td>
<td>An interface-dependent error. This error number not currently used.</td>
</tr>
<tr>
<td>121</td>
<td>An interface-dependent error. This error number not currently used.</td>
</tr>
<tr>
<td>122</td>
<td>An interface-dependent error. This error number not currently used.</td>
</tr>
<tr>
<td>123    NO</td>
<td>Syntax error. A semicolon delimiter was expected in the statement.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Error No.</td>
<td>Message</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>124</td>
<td>I51</td>
</tr>
<tr>
<td>125</td>
<td>ACOR</td>
</tr>
<tr>
<td>126</td>
<td>BUFFER</td>
</tr>
<tr>
<td>127</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

R16
<table>
<thead>
<tr>
<th>Error No.</th>
<th>Message</th>
<th>Meaning</th>
<th>Possible Corrective Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>129</td>
<td>EARLY TERM</td>
<td>A buffer was emptied before all the ENTER fields were satisfied, or a field terminator was encountered before the specified character count was reached.</td>
<td>Check your incoming character stream, ENTER list, and image specifiers.</td>
</tr>
<tr>
<td>129</td>
<td>VAR TYPE</td>
<td>The type (string or numeric) of a variable in an ENTER list does not match with the image specified for that variable.</td>
<td>Check your ENTER list and image specifiers.</td>
</tr>
<tr>
<td>130</td>
<td>NO TERM</td>
<td>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.</td>
<td>Check your incoming character stream, ENTER list, and image specifiers.</td>
</tr>
</tbody>
</table>