Paul Walsh

CFL

Newcastle

System 45

Customer Course

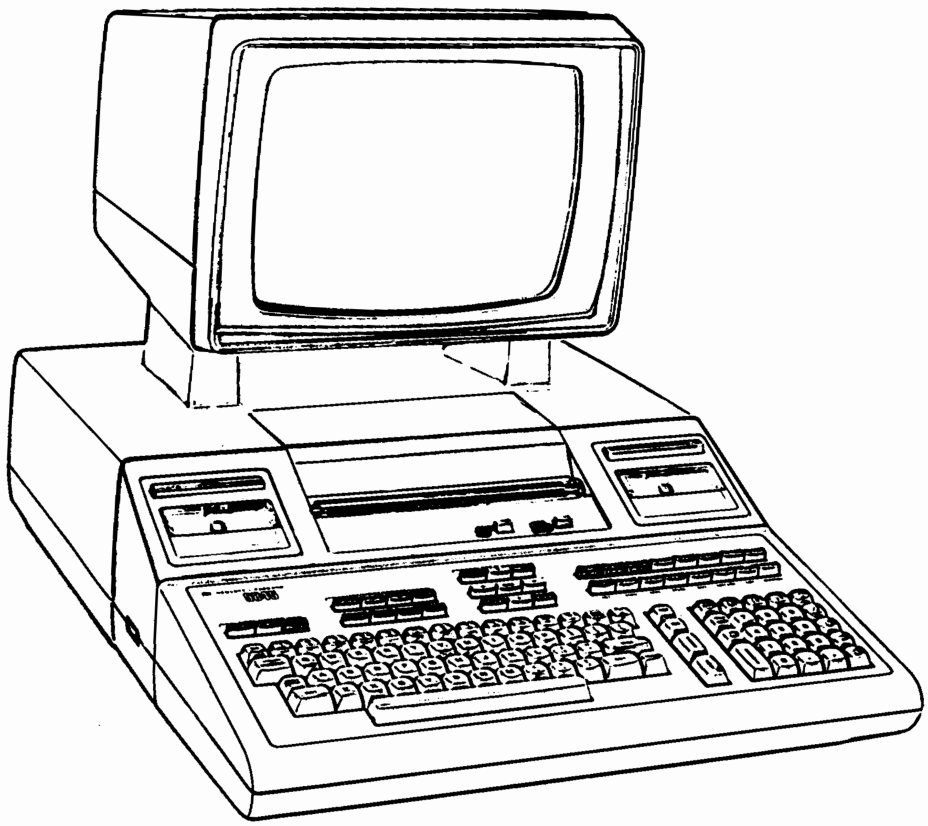
Student Notebook

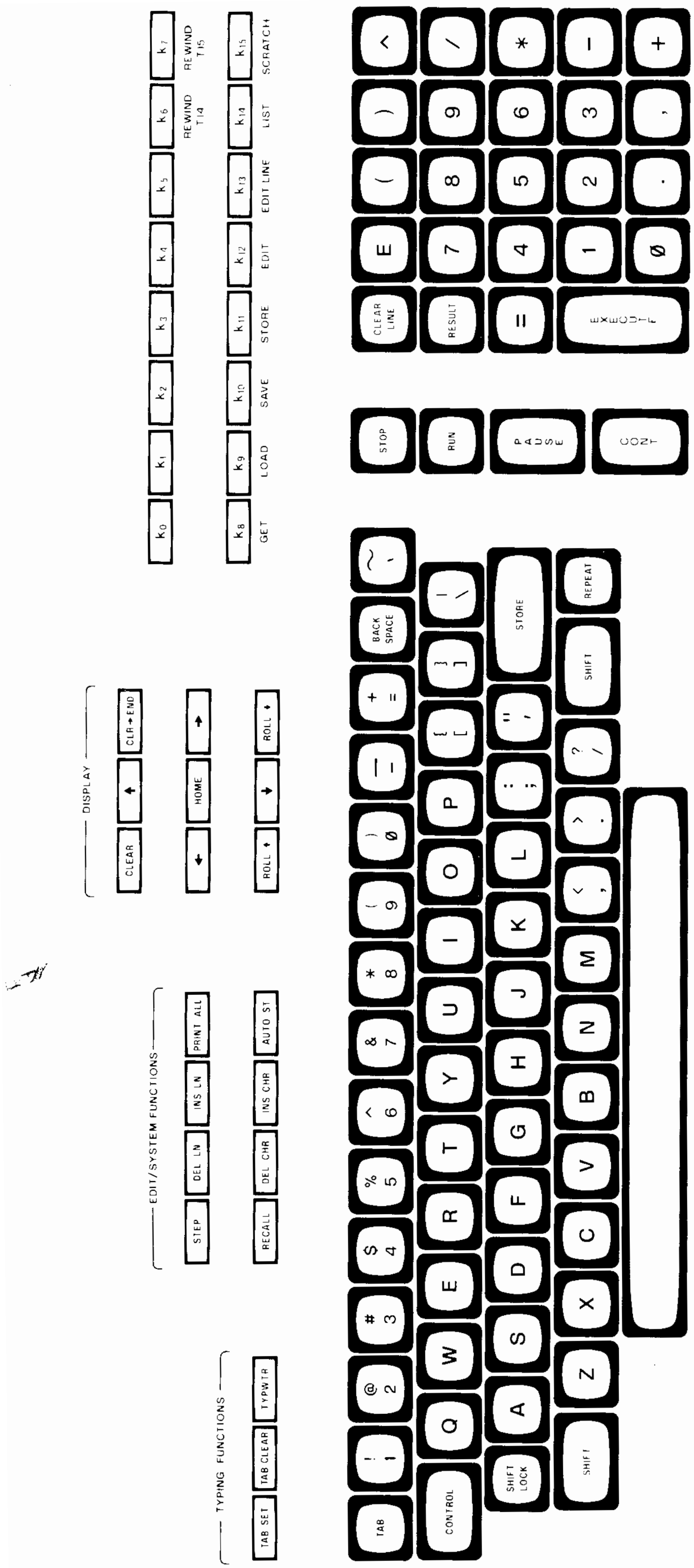
Part No. 11141-70131

Rev. B

This Page For Packaging Only

Printed in U.S.A.




DISPLAY
CLEAR
CLR→END
HOME
ROLL
EDIT/SYSTEM FUNCTIONS
STEP
DEL LN
INS LN
PRINT ALL
RECALL
DEL CHR
INS CHR
AUTO ST
TYPING FUNCTIONS
TAB SET
TAB CLEAR
TYPWTR
GET
LOAD
SAVE
STORE
EDIT
EDIT LINE
LIST
SCRATCH
REWIND T14
REWIND T15
STOP
RUN
PAUSE
CONT
CLEAR LINE
RESULT
EXECUTE
TAB
CONTROL
SHIFT LOCK
SHIFT
BACK SPACE
STORE
REPEAT

STANDARD FORMAT

of the CRT Display

LINE #	
1	120 FOR I = 1 TO 10
2	130 DISP "ELEMENT";I;
3	140 INPUT A(1,I)
4	150 FOR J = 2 TO 4
5	160 A(J,I) = A(1,I) J
6	170 NEXT J
7	180 NEXT I
8	190 PRINT A(*)
9	200 – –
10	210 – –
11	220 – –
12	230 – –
13	240 – –
14	250 – –
15	260 – –
16	270 – –
17	280 – –
18	290 – –
19	300 – –
20	310 – –
21	
22	ELEMENT 6 ?
23	47//_3.5 Blinking Cursor
24	
25	IMPROPER NUMERIC EXPRESSION

EDIT LINE FORMAT

```
LINE #

  1
  2
  3
  4
  5
  6    110   - -
  7    120   FOR I = 1 TO 10
  8    130   DISP "ELEMENT";I
  9    140   INPUT A(1,I)
 10    150   FOR J = 2 TO 4
 11
 12    160   A(J,I) = A(1,I) <J_
 13
 14                                 Blinking Cursor
 15
 16    170   NEXT J
 17    180   NEXT I
 18    190   PRINT A(*)
 19    200   - -
 20    210   - -
 21
 22
 23
 24
 25
```

FF2/M9-3

EXAMPLE PROGRAM

```
10    OPTION BASE 1
20    DIM Vector(10)
30    DATA 15,32,74,85,94,10,7,66,14,85
40    FOR I=1 TO 10                    ! This loop reads the data and prints
50       READ A                        !  it out
60       PRINT A;
70    NEXT I
80    RESTORE 30                       ! This enables re-use of DATA statement
90    FOR I=1 TO 10                    ! This loop reads the data into an
100      READ Vector(I)                !  array
110   NEXT I
120   STOP
```

MODIFIED EXAMPLE PROGRAM

```
    10   OPTION BASE 1
    20   DIM Vector(10)                 ! Lines 30 through 80 are gone.
    90   FOR I=1 TO 10                  ! This loop allows the data to be
--> 91      DISP "#";I;                 !  entered from the keyboard
--> 100     INPUT ". Please enter the next number",Vector(I)
    110  NEXT I
--> 120  PRINT TAB(32);"Here is the array"
--> 130  FIXED 4
--> 140  FOR I=1 TO 10                  ! Print the array
--> 150     PRINT TAB(35);Vector(I)
--> 160  NEXT I
--> 170  PRINT LIN(1),"The program is finished.";SPA(15),"Goodbye."
--> 180  STOP
```

FURTHER MODIFICATION OF THE EXAMPLE PROGRAM

```
    10   OPTION BASE 1
    20   DIM Vector(10)
    90   FOR I=1 TO 10              ! This loop allows the data to be
    91     DISP "#"; I;             !  entered from the keyboard
    100    INPUT ". Please enter the next number", Vector(I)
    110  NEXT I
    120  PRINT TAB(32); "Here is the array"
--> 130  FLOAT 2
--> 140  FOR I=1 TO 10 STEP 2       ! Print the array
--> 150    PRINT TAB(35); Vector(I); SPA(3); Vector(I+1)
    160  NEXT I
    170  PRINT LIN(1), "The program is finished.", SPA(15), "Goodbye."
    180  STOP
```

; space before & space after

, columns of 20

; at end of DISP or PRINT suppresses the LINE FEED

FURTHER MODIFICATIONS USING 'GOSUB'

```
     10   OPTION BASE 1
     20   DIM Vector(10)
     90   FOR I=1 TO 10               ! This loop allows the data to be
     91      DISP "#";I;              !  entered from the keyboard
     100     INPUT ".  Please enter the next number",Vector(I)
     110  NEXT I
     120  PRINT TAB(32);"Here is the array"
 --> 121  STANDARD
 --> 122  GOSUB Print                 ! Print the original array
 --> 123  FOR K=2 TO 4
 --> 124     FOR J=1 TO 10
 --> 125        Vector(J)=K*Vector(J)
 --> 126     NEXT J
 --> 127     PRINT TAB(32);"Here is the array multiplied by ";K
 --> 128     GOSUB Print
 --> 129  NEXT K
 --> 130  PRINT LIN(1);"The program is finished.",SPA(15),"Goodbye."
 --> 131  BEEP
 --> 132  STOP
 --> 133 Print: ! This routine prints the array.
     140  FOR I=1 TO 10 STEP 2        ! Print the array
     150     PRINT TAB(30);Vector(I);SPA(3);Vector(I+1)
     160  NEXT I
 --> 170  PRINT LIN(2)
 --> 180  RETURN
```

THE OPERATORS

ADDITION: +	Sum=A+B
SUBTRACTION: -	Difference=A-B
MULTIPLICATION: *	Product=A*B
DIVISION: /	Quotient=A/B
EXPONENTIATION: ^ or **	A_to_power=A^B
INTEGER DIVISION: DIV	Iquotient=A DIV B
MODULUS: MOD	Remainder=A MOD B

10

SYSTEM FUNCTIONS

```
Minimum=MIN(X,Y,Z,1.3)
Greatest=MAX(P,Vector(2),4)
Number=ABS(Number)*SGN(Number)
Number=INT(Number)+FRACT(Number)
Area_circle=PI*R^2
Hypotenuse=SQR(Side1^2+Side2^2)
Log_base_ten=LGT(Number)
Log_base_e=LOG(Number)
Naperian_e=EXP(1)
DEG
RAD
GRAD
P=SIN(X)+COS(X)-TAN(X)
Q=ASN(X)+ACS(X)+ATN(X)
```

O&P, Ch. 3, "Math Functions & Statements" 11

WHAT WE'VE SEEN SO FAR

```
OPTION BASE 0 / OPTION BASE 1
DIM X(5),Y(3),Z(237)
PRINT X,B;"Here is C";C,TAB(40);X(2),LIN(2),14^8*5
Root1=(-B-SQR(ABS(B^2-4*A*C)))/(2*A)
GOTO Compute_wages
DATA 4,5,3.141592654
READ A,B,Pi
RESTORE 10
FOR Counter=100 TO -100 STEP -4   ! Count backwards
NEXT Counter
REMark
DISP "Please enter the";I;"th value of the array";
INPUT "Enter A",A,"Now enter B",B
FIXED 5      FLOAT 4      STANDARD
GOSUB Find_time
RETURN
```

12

1. Write a program to find the mean, variance, and standard deviation of the following 1Ø numbers: 2Ø,45,13,64,85,97,59,24,72,6
The output should consist of the above numbers listed in a column which also includes the index of each number. The mean, variance, and standard deviation should be labelled, and should be separated from the list of numbers by at least two lines. The output should appear on both the CRT and the internal printer.
You may use either INPUT or READ/DATA to enter the numbers to your program.
The formula for the mean (or average) of an array of numbers is:

$$\overline{X} = \left(\sum_{i=1}^{n} X_i\right)/n;$$ where n represents the number of numbers

The formula for the variance of an array of numbers is:

$$S^2 = \left[\sum_{i=1}^{n} X_i^2 - \left(\sum_{i=1}^{n} X_i\right)^2/n\right]/(n-1);$$ where n represents the number of numbers

The formula for the standard deviation is:

$$S = \sqrt{(\text{variance})} = \sqrt{S^2}$$

2. Write a program to print out a table of numbers consisting of the sines, cosines, and tangents of all the angles (in half degree increments) between Ø and 45. There should be a total of four columns (the angle has its own column). Each column should be titled such that the title is centered over the column of numbers. The numbers should appear in fixed-point notation, with the angular column having two digits after the decimal point, while the other three columns have six digits past the decimal point. The output should appear on both the CRT and the internal printer.
HINT: Suppress cr/lf at end of PRINT with ; or ,.

'IF' STATEMENT WITH IMPLIED 'GOTO'

```
1     REM This is a sample program
2     I=Ø
1Ø    PRINT "2*I=";2*I
2Ø    I=I+1
3Ø    IF I<=1Ø THEN 1Ø
4Ø    STOP
```


Example 5

Program Cartridge 1

VARIATION ON 'IF' STATEMENT: EXECUTABLE STATEMENT

```
    1    REM This is a sample program
    2    I=0
    10   PRINT "2*I=";2*I
    20   I=I+1
--> 30   IF I>10 THEN STOP
--> 40   GOTO 10
```

VARIATION ON 'IF': NO RELATIONAL OPERATORS

```
    1    REM This is a sample program
--> 2    I=10
    10   PRINT "2*I=";2*I
--> 20   I=I-1
--> 30   IF I THEN 10
--> 40   STOP
```

ie IF $I \neq 0$ THEN 10

0 is FALSE

Everything but ϕ is TRUE.

VARIATION ON 'IF': LOGICAL OPERATORS

```
    1    REM This is a sample program
    2    I=10
--> 10   I=I-1
--> 20   IF NOT (I MOD 2) THEN PRINT I
    30   IF I THEN 10
    40   STOP
```

IF I not divisible by 2.

Remainder theorem :

remainder

$$a = bq + r$$

quotient

$\boxed{a = bq + r}$

I = 2

IMOD2 = 0

I

WHAT IS A STRING?

A GROUP OF CHARACTERS:

    "The quick brown fox"
    "12"

STRING VARIABLES:

    A$
    Name$
    String$
    Book_of_month$

ASSIGNING VALUES TO STRING VARIABLES:

    Name$="Al Toesacks"
    Street$="1310 Maple St."

WHY STRINGS?

    NAMES, ADDRESSES, PART DESCRIPTIONS, ETC.

O&P, Ch. 5, "String Variables" 18

STRINGS WITH READ

```
10   FIXED 2
20   DATA Sam Smith,3.25,Bill Jones,3.50,John Doe,4.15,Gerry Atric,3.58
30   DATA Fred Perkins,2.50,Mary Atwood,3.84,Penny Ante,4.52
40   DATA Kurt Remarque,3.40,Rhonda Campfire,4.15,Steve Smith,5.67
50   PRINT "Name";TAB(20);"Hourly Pay",LIN(1)
60   FOR I=1 TO 10
70      READ Name$,Pay
80      PRINT Name$;TAB(20);Pay
90   NEXT I
100  STOP
Name                Hourly Pay

Sam Smith            3.25
Bill Jones           3.50
John Doe             4.15
Gerry Atric          3.58
Fred Perkins         2.50
Mary Atwood          3.84
Penny Ante           4.52
Kurt Remarque        3.40
Rhonda Campfire      4.15
Steve Smith          5.67
```

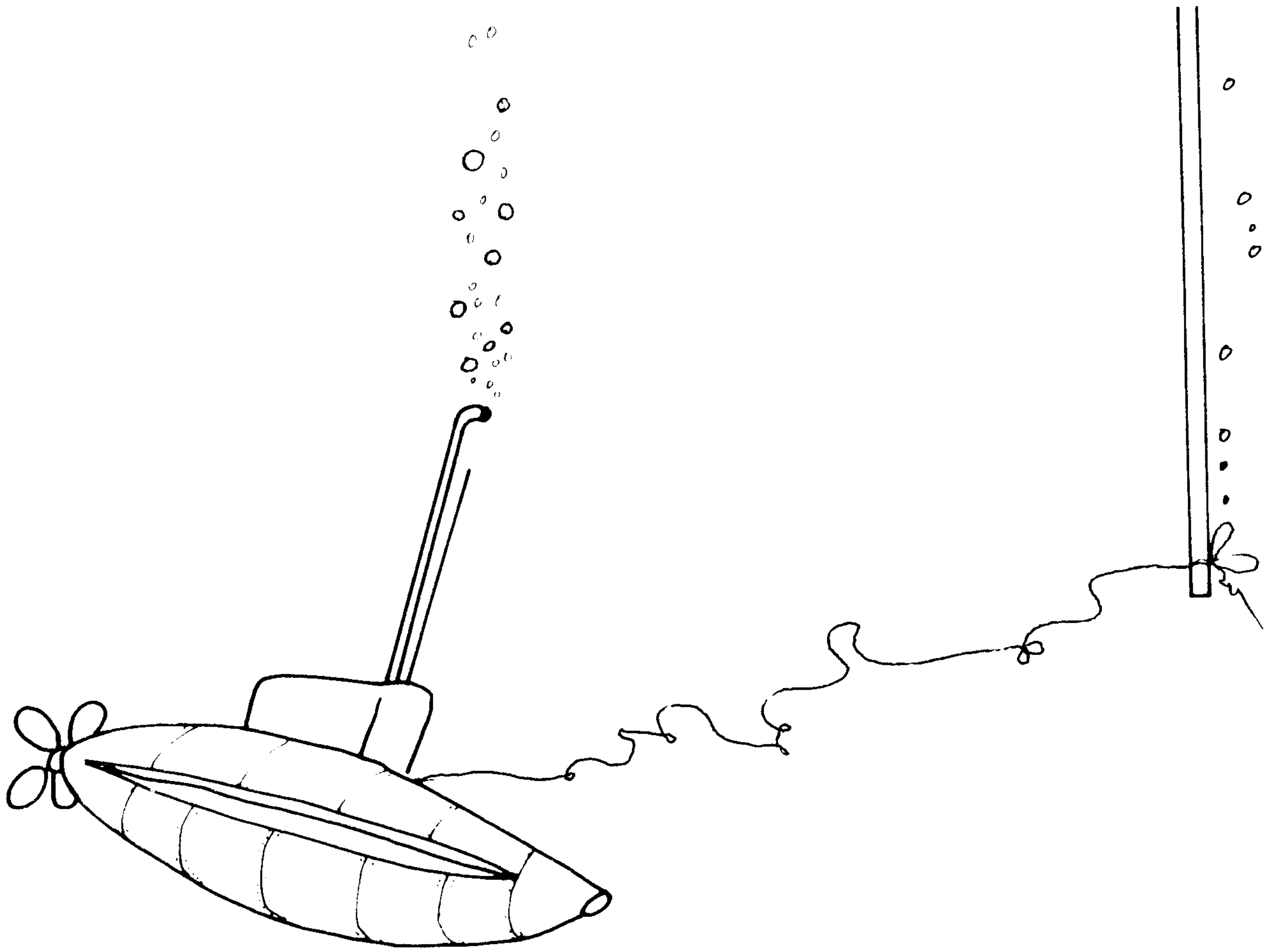
Example 9

VARIATION: STRINGS WITH INPUT

```
      60   FOR I=1 TO 10
--> 70       INPUT "Name?", Name$, "Pay?", Pay
      80       PRINT Name$; TAB(20); Pay
      90   NEXT I
      100  STOP
```

HOW MUCH INFORMATION CAN YOU PUT INTO A STRING?

```
--> 20   DIM Name$[30]          !DIMensions Name$ to contain up to 30 characters
    60   FOR I=1 TO 10
    70      INPUT "Name?",Name$,"Pay?",Pay
    80      PRINT Name$;TAB(20);Pay
    90   NEXT I
    100  STOP
```



SUBSTRING

SUBSTRINGS

EXECUTE FROM THE KEYBOARD:

A$="ABCDEFGHIJKLMNOP"

A$[5]
A$[5,10]   5th element up to & including 10th element
A$[4;3]   next 3 characters
A$[5]="x"
A$

NOW, SET A$ BACK TO "ABCDEFGHIJKLMNOP"

A$[5,5]="x"   replace from the 5th character to
A$[5;1]="p"   the 5th character

B$=A$[3;4]

O&P, Ch. 6, "String Variables"   22

SUBSTRING AND CONCATENATION EXAMPLE

```
10   DIM First$[20],Last$[20],Name$[40],City$[30],State$[3]
20   FOR I=1 TO 5
30      INPUT "First name?",First$
40      INPUT "Last name?",Last$
50      Name$=First$&" "&Last$  ! Concatenate first and last names and
60                              !  put a space between them.
70      INPUT "City?",City$
80      INPUT "State?",A$
90      State$=A$[1;2]          ! Only use the first two letters of the
100                             !  state
110     PRINT Name$
120     PRINT City$&", "&State$ ! Print the city and state together and
130                             !  put a comma and a space between them
140  NEXT I
150  STOP
```

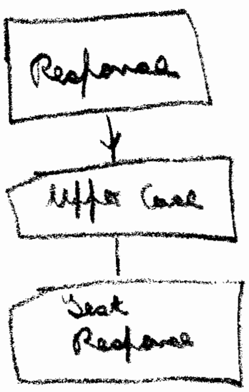
STRING FUNCTION
UPPER/LOWER CASE

CONVERTS ALPHABETIC CHARACTERS TO ALL UPPER OR ALL LOWER CASE.

EXECUTE FROM KEYBOARD:

```
UPC$("Hiram")
LWC$("Hiram")
```

USEFUL FOR STANDARDIZING OPERATOR RESPONSES.



or L.C

O&P, Ch. 5, "String Variables" 24

Ex 12

POSITION EXAMPLE

```
10    INPUT "Enter a string, any string",A$
20    Pos=POS(UPC$(A$),"E")
30    IF NOT Pos THEN Not_found
40    PRINT "E is character #";Pos;"in ";A$
50    STOP
60 Not_found: PRINT A$;" does not contain E"
70    END
E is character # 3 in THESE ARE THE DAYS
```

Obtains position in string of a given character, 1st occurrence only

LENGTH EXAMPLE

```
10   DIM A$[400]
20   INPUT "Enter a string, any string",A$
30   PRINT A$;" has";LEN(A$);"characters in it."
40   END
THESE ARE THE DAYS has 18 characters in it.
```

Returns the length of a string

STRING FUNCTION
TRIMMING OFF EXTRA BLANKS

TRIM$ TRIMS OFF LEADING AND TRAILING BLANKS FROM A GIVEN STRING.

EXECUTE FROM THE KEYBOARD:

```
A$="    abc    "
LEN(A$)
LEN(TRIM$(A$))
```

NB Not embedded blanks

O&P, Ch. 5, "String Variables" 27

STRING FUNCTIONS
NUMERIC/STRING CONVERSION

VAL CONVERTS A STRING WHICH STARTS WITH NUMERIC CHARACTERS TO A NUMBER.

```
A$="March 12, 1978"
VAL(A$)
VAL(A$[7])
VAL(A$[7;1])
```

VAL$ CONVERTS A NUMBER INTO A STRING USING THE CURRENT FIXED/FLOAT/STANDARD SETTING WITH NO LEADING/TRAILING BLANKS.

```
FLOAT 3
VAL$(PI)
FIXED 2
VAL$(PI)
```

O&P, Ch. 5, "String Variables" 28

STRING FUNCTIONS
REVERSING/REPEATING STRINGS

REV$ RETURNS THE REVERSE OF A STRING.

```
REV$("123")
REV$("abcdefg")
REV$(UPC$("xyz  "))
A$="The quick brown fox"
REV$(A$[POS(A$,"ox");2])
LEN(A$)-POS(REV$(A$),"o")+1
```

RPT$ RETURNS A STRING REPEATED AS MANY TIMES AS YOU SPECIFIED.

```
RPT$("HO ",3)
LEN(RPT$("HA",4))
```

O&P, Ch. 5, "String Variables" 29

STRINGS AND RELATIONAL OPERATORS

STRINGS ARE COMPARED ACCORDING TO THEIR ASCII REPRESENTATIONS, ELEMENT BY ELEMENT.

EXECUTE THESE FROM THE KEYBOARD (A "1" ON THE CRT INDICATES A TRUE RELATIONSHIP):

```
"BOY"<"MAN"
"MAN"<"WOMAN"
"WOMAN"<"man"
"A">"B"
"q"<="queen"
"computer "="computer"
```

O&P, Ch. 5, "String Variables"; Appx. B

30

Test carries out as follows.

" BOY " < " MAN "

character by character test

STRING FUNCTIONS
ASCII REPRESENTATIONS

CHR$ FUNCTION RETURNS THE CHARACTER WHOSE ASCII REPRESENTATION IS THE SPECIFIED INTEGER.

NUM FUNCTION RETURNS THE ASCII REPRESENTATION OF THE FIRST CHARACTER IN THE STRING GIVEN.

```
5    DIM A$[25]
10   A$=CHR$(34)&"Hello,"&CHR$(34)&" the man
said."                  ! 34 is ASCII code for ".
20   PRINT A$
30   STOP
```

RUN THE ABOVE PROGRAM.

EXECUTE FROM THE KEYBOARD: NUM(A$)

O&P, Ch. 5, "String Variables" 31

ASCII Character	Comments	Key(s) to Press☆	Octal Code	Decimal Code
NUL	Null	CONTROL space bar	00	0
SOH	Start of Header	CONTROL A	01	1
STX	Start of Text	CONTROL B	02	2
ETX	End of Text	CONTROL C	03	3
EOT	End of Transmission	CONTROL D	04	4
ENQ	Enquiry	CONTROL E	05	5
ACK	Acknowledgement	CONTROL F	06	6
BEL	Bell	CONTROL G	07	7
BS	Backspace	CONTROL H	10	8
HT	Horizontal Tab	CONTROL I	11	9
LF	Line Feed	CONTROL J	12	10
VT	Vertical Tab	CONTROL K	13	11
FF	Form Feed	CONTROL L	14	12
CR	Carriage Return	CONTROL M	15	13
SO	Shift Out	CONTROL N	16	14
SI	Shift In	CONTROL O	17	15
DLE	Data Link Escape	CONTROL P	20	16
DC1	Device Control	CONTROL Q	21	17
DC2	Device Control	CONTROL R	22	18
DC3	Device Control	CONTROL S	23	19
DC4	Device Control	CONTROL T	24	20
NAK	Negative Acknowledgement	CONTROL U	25	21
SYN	Synchronous Idle	CONTROL V	26	22
ETB	End of Text Block	CONTROL W	27	23
CAN	Cancel	CONTROL X	30	24
EM	End of Media	CONTROL Y	31	25
SUB	Substitute	CONTROL Z	32	26
ESC	Escape	CONTROL { [	33	27
FS	File Separator	CONTROL | \\	34	28
GS	Group Separator	CONTROL }]	35	29
RS	Record Separator	CONTROL SHIFT ^ 6 *	36	30
US	Unit Separator	CONTROL SHIFT ? /	37	31

use this when interfacing with ICL

☆Assumes typewriter mode; multiple keys must be pressed simultaneously.

*Also can be found among calculator keys

ASCII Character	Comments	Key(s) to Press☆	Octal Code	Decimal Code
SP	Blank	space bar	40	32
!	Exclamation Point	SHIFT !/1	41	33
"	Double Quote	SHIFT "/'	42	34
#	Pound Sign	SHIFT #/3	43	35
$	Dollar Sign	SHIFT $/4	44	36
%	Percent Sign	SHIFT %/5	45	37
&	Ampersand	SHIFT &/7	46	38
'	Apostrophe	"/'	47	39
(	Left Parenthesis	SHIFT (/9 *	50	40
)	Right Parenthesis	SHIFT)/Ø *	51	41
*	Asterisk	SHIFT */8 *	52	42
+	Plus Sign	SHIFT =/+ *	53	43
,	Comma	</, *	54	44
−	Minus Sign (Dash)	_/− *	55	45
.	Period	>/. *	56	46
/	Forward Slash	?// *	57	47
0		)/Ø *	60	48
1	N	!/1 *	61	49
2	U	@/2 *	62	50
3	M	#/3 *	63	51
4	E	$/4 *	64	52
5	R	%/5 *	65	53
6	I	^/6 *	66	54
7	C	&/7 *	67	55
8	S	*/8 *	70	56
9		(/9 *	71	57
:	Colon	SHIFT :/;	72	58
;	Semicolon	:/;	73	59
<	Less Than	SHIFT </,	74	60
=	Equal	SHIFT =/+ *	75	61
>	Greater Than	SHIFT >/.	76	62
?	Question Mark	SHIFT ?//	77	63

☆Assumes typewriter mode; multiple keys must be pressed simultaneously.

*Also can be found among calculator keys.

ASCII Character	Comments	Key(s) to Press☆	Octal Code	Decimal Code
@	Commercial At	SHIFT @/2	100	64
A		SHIFT A	101	65
B		SHIFT B	102	66
C		SHIFT C	103	67
D		SHIFT D	104	68
E		SHIFT E	105	69
F		SHIFT F	106	70
G	C	SHIFT G	107	71
H	A	SHIFT H	110	72
I	P	SHIFT I	111	73
J	I	SHIFT J	112	74
K	T	SHIFT K	113	75
L	A	SHIFT L	114	76
M	L	SHIFT M	115	77
N		SHIFT N	116	78
O		SHIFT O	117	79
P	L	SHIFT P	120	80
Q	E	SHIFT Q	121	81
R	T	SHIFT R	122	82
S	T	SHIFT S	123	83
T	E	SHIFT T	124	84
U	R	SHIFT U	125	85
V	S	SHIFT V	126	86
W		SHIFT W	127	87
X		SHIFT X	130	88
Y		SHIFT Y	131	89
Z		SHIFT Z	132	90
[	Left Bracket	[**	133	91
\	Reverse Slash	| \	134	92
]	Right Bracket	] **	135	93
↑	Up Arrow	SHIFT ^/6 *	136	94
—	Underscore	SHIFT ‾/_	137	95

☆Assumes typewriter mode; multiple keys must be pressed simultaneously.

*Also can be found among calculator keys.

**Shift (and) on calculator keys.

ASCII Character	Comments	Key(s) to Press*	Octal Code	Decimal Code
	Grave Mark	~ `	140	96
a		A	141	97
b		B	142	98
c		C	143	99
d		D	144	100
e	n	E	145	101
f	o	F	146	102
g	n	G	147	103
h	c	H	150	104
i	a	I	151	105
j	p	J	152	106
k	i	K	153	107
l	t	L	154	108
m	a	M	155	109
n	l	N	156	110
o		O	157	111
p		P	160	112
q	l	Q	161	113
r	e	R	162	114
s	t	S	163	115
t	t	T	164	116
u	e	U	165	117
v	r	V	166	118
w	s	W	167	119
x		X	170	120
y		Y	171	121
z		Z	172	122
{	Left Brace	SHIFT [	173	123
|	Vertical Line	SHIFT | \	174	124
}	Right Brace	SHIFT]	175	125
~	Tilde	SHIFT ~ `	176	126
DEL	Delete	Inaccessible from Keyboard	177	127

*Assumes typewriter mode; multiple keys must be pressed simultaneously

Nationalized and Drawing Characters

You can easily access various national and drawing characters using the CHR$ function. The characters and their corresponding decimal value for the CHR$ function are listed below.

160		161	À	162	Ì	163	Ò	164	Ù	165	Á	166	É	167	Ó
168	´	169	`	170	^	171	¨	172	~	173	Ê	174	Ô	175	£
176	¯	177	Ā	178	ª	179	°	180	Ç	181	ç	182	Ñ	183	ñ
184	¡	185	¿	186	¤	187	£	188	º	189	§	190	²	191	•
192	â	193	ê	194	ô	195	û	196	á	197	é	198	ó	199	ú
200	à	201	è	202	ò	203	ù	204	ä	205	ë	206	ö	207	ü
208	Å	209	î	210	Ø	211	Æ	212	å	213	í	214	ø	215	æ
216	Ä	217	ì	218	Ö	219	Ü	220	É	221	ï	222	ß	223	
224	↑	225	┣	226	┏	227	┳	228	━	229	┓	230	┫	231	┃
232	╋	233	├	234	┌	235	┬	236	─	237	┐	238	┤	239	╂
240	→	241	┣	242	└	243	┴	244	┼	245	┘	246	┫	247	↓
248	←	249	┻	250	┗	251	┻	252	╋	253	┛	254	┳	255	▒

1. Write a program to print out a table for the ASCII characters and their corresponding codes between 32 (blank) and 127 (del). Use the CHR$ instruction to convert ASCII decimal codes to characters.

INPUT EXAMPLE

```
10    INPUT "Enter a number, a string, and another number",A,A$,B
20    PRINT A,LIN(1),B,LIN(1),A$
30    STOP
```

Line Input : only takes a character string
remembers leading & trailing blanks

LINPUT EXAMPLE

Entered "HELLO", "GOODBYE"

```
10    DIM A$[80]
20    INPUT "Enter a string with quotes and commas",A$
30    PRINT "Here is what actually was entered:   ";A$
40    LINPUT "Now enter the same string again (press RECALL and CONT)",A$
50    PRINT "Here is what was entered using LINPUT: ";A$
60    STOP
Here is what actually was entered:   HELLO
Here is what was entered using LINPUT: "HELLO","GOODBYE"
```

NB

N8 EDIT EXAMPLE Allows a string to be altered

```
10    DIM A$[160]
20    LINPUT "Enter a long string",A$
30    PRINT "Here is the original string",LIN(1),A$
40    EDIT "Now change part of the string you just entered",A$
50    PRINT LIN(1),"Here is the edited string",LIN(1),A$
60    STOP
```

Example 17

STRING ARRAY DIMENSIONING EXAMPLE

```
10   OPTION BASE 1
20   DIM Numbers(10),Strings$(10)[20]
30   STOP
```

length of each element in the array

default is 18 characters

STRING ARRAY EXAMPLE

```
10   OPTION BASE 1
20   DIM Citystate$(5)[80],City$(5)[80],State$(5)[80]
30   FOR I=1 TO 5
40      LINPUT "Enter: City, State",Citystate$(I)
50      GOSUB Fixstring
60   NEXT I
70   INPUT "Do you want to make any changes (Y/N)?",C$
80   C$=UPC$(C$[1,1])
90   IF C$="N" THEN Print
100  IF C$="Y" THEN Edit
110  BEEP
120  GOTO 70
130 Edit: INPUT "Which string do you want to edit?",I
140  EDIT "Make your change and press CONT",Citystate$(I)
150  GOSUB Fixstring
160  GOTO 70
170 Print:  ! This part of the program prints the Cities and States
180  FOR I=1 TO 5
190     PRINT City$(I),TAB(40),State$(I)
200  NEXT I
210  STOP
220 Fixstring: ! This subroutine splits up Citystate$(I)
230     Commapos=POS(Citystate$(I),",")
240     City$(I)=TRIM$(Citystate$(I)[1,Commapos-1])
250     State$(I)=TRIM$(Citystate$(I)[Commapos+1])
260     RETURN
```

1. Write a program that will accept data for up to 30 employees. Each employee's record should have the following information:

   Name (a string field having a maximum of 30 characters)
   Address (another 30 character string field)
   City and State (again with the 30 characters)
   Phone number (a string field having 12 characters -- 303 667-5000)
   Hourly pay (a floating point number)

   The program should ask for the number of employees to be entered (with a maximum of 30), and once all the data has been entered, the program should ask the user if any changes need to be made. If there are changes to be made, the program should ask which record is to be changed. Each field in the employee's record should be displayed for modification. Once all the fields have been edited, the program should return to the question asking the user if any changes need to be made. If there are no changes to be made at this point, the program should print out each employee's record, separated from the others by at least two blank lines. When all the data has been printed, the program should stop.

   Instructor's note: Point out that in an INPUT statement, a numeric value will remain unchanged if CONT is pressed without first entering a numeric expression.

2. Write a program that allows the user to enter a set of names from the keyboard, and sort them alphabetically by last names. The names should be stored in a string array. The logical operators (<,>,=, <=,>=,<>) can be used to compare strings. The POS function may be used to separate the first names from the last names.

MULTI-DIMENSIONAL ARRAYS DIMENSIONING

```
10    OPTION BASE 1
20    DIM One_d(15),One_d$(10)[80],Two_d(3,3),Three_d(2,2,5)
30    DIM Four_d$(5,6,4,2)[3],Six_d(2,3,4,5,6,7)
40    STOP
```

SUBSCRIPTING

1 to 5

```
10    OPTION BASE 0
20    DIM One(5,2,3),Two$(5),Three(1:5),
      Four$(-5:6,-4:2,6:20)[12]
30    STOP
```

12 x 7 x 15 array of strings
of length 12

up to & including 6 dimensions allow

: used for selecting a sub-matrix
from a full matrix

INPUT, PRINT, AND READ WITH ARRAYS

```
10   OPTION BASE 1
20   DIM A(5),B(3,3)
30   DATA 1,2,3,4,5,6,7,8,9
40   READ B(*)
50   INPUT A(*)
60   PRINT "Here is B(*)",LIN(1),B(*)
70   PRINT "Here is B(*) again",LIN(1),B(*);
80   PRINT "And here is A(*):",A(*);
90   STOP
```

1. Use a three-dimensional array to keep track of the productivity of four production lines over a three-week period for two work shifts.
Hint: Let the first dimension signify which shift is being talked about (1 or 2), the second dimension can keep track of which of the four production lines is being talked about, and the third dimension will be the week number (1, 2, or 3).
The program should be able to print out the following table. Use a READ statement to enter the data (the data represents the number of "widgets" produced in a week).

Shift 1

	Week 1	Week 2	Week 3
Line 1	75	65	85
Line 2	3Ø	45	2Ø
Line 3	6	1Ø	5
Line 4	85	9Ø	6Ø

Shift 2

Line 1	8Ø	7Ø	9Ø
Line 2	25	4Ø	22
Line 3	8	11	4
Line 4	82	88	54

Titling of the table should be an approximation of the titling above, but it doesn't necessarily have to be exact. The numbers should be lined up in neat columns, but they don't have to be right-justified (however, they should be left-justified).

When this exercise has been completed, have the student store the exercise on the tape cartridge under the name PROG2 for future reference.

ARRAY ARITHMETIC

B:	12	5	C:	10	3
	9	2		8	4

```
MAT A=B
```

ELEMENT-BY-ELEMENT OPERATIONS:

`+,-,.,/,=,<,>,<=,>=,<>`

```
MAT A=B+C

MAT A=B-C

MAT A=B>C
```

O&P, Ch. 5, "Array Operations"

43

MAT $A = B > C$

1 – True

0 – False

$$A = \begin{bmatrix} 1 & 1 \\ 1 & \phi \end{bmatrix}$$

element x element comparison

To get a feel for these operators, modify the last exercise that was performed (stored under the file"PROG2") to conform to the following definition:

2. Change the three-dimensional array from the previous problem to two two-dimensional arrays (i.e., one array for each shift, rather than using the first subscript of the three-dimensional array to determine which shift is being used). Use the same data and method of entering it. Compute an array which will compare shift one and shift two to indicate which production line has produced the most "widgets" for each week. In this array, a 1 will indicate that shift one produced at least as many "widgets" as shift two, while a zero will indicate that shift one produced less widgets than shift two.

   Print each array, and title it accordingly, as well as printing and titling the original data.

ARRAY FUNCTIONS

HOW MANY WIDGETS WERE PRODUCED IN EACH LINE IN EACH SHIFT OVER THE THREE-WEEK PERIOD?

```
MAT L1=RSUM(Shift2)
MAT L2=RSUM(Shift1)
```

HOW MANY WIDGETS WERE PRODUCED EACH WEEK BY EACH SHIFT?

```
MAT S1=CSUM(Shift2)
MAT S2=CSUM(Shift1)
```

HOW MANY WIDGETS WERE MADE ALTOGETHER BY EACH SHIFT?

```
T1=SUM(Shift1)
T2=SUM(Shift2)
```

O&P, Ch. 5, "Array Operations" 45

A=DOT(B,C)
The DOT function returns a scalar and operates on two vectors. The returned value is defined to be the sum of the products of corresponding elements of the two vectors. This "dot product" is reflected in the "real world" by vector mathematics found in the area of physics, electrical engineering, and other scientific fields. It can also be made to apply to our example, as the following exercise will show.

3. Take the program as it was modified above and add this feature to it. Assume that the widgets made by each line have a different value. The widgets made by line 1 are worth $450, the widgets made by line 2 are worth $565, the widgets made by line 3 are worth $1200, and the widgets put out by line 4 sell for $375. Using the RSUM and DOT functions, find the total dollar amount produced by each shift. The output can be added to the end of the output already produced by the program.

Scalar operations:

This class of operations allows you to operate on entire arrays with a scalar value. So you can add a scalar to each element of the array, for example, or you can multiply each element of the array by a scalar. Here are all the operators that can be used with scalar values:
+,-,*,/,=,<,>,<=,>=,< > for example: MAT A = B + (6)

4. Going back to our example, let's add another feature to the program. Use the * scalar multiplication operator, find out what would happen to productivity during a flu epidemic, assuming that 40% of the work force is absent due to illness.

Functions such as INT/FRACT, ABS/SGN, SQR, LGT/LOG/EXP, SIN/COS/TAN, ASN/ACS/ATN, etc., can be used on entire arrays. So the statement MAT A=SIN(B) would cause the array A to be filled with the sines of every element of B.

5. In Exercise 3, the 40% absentee rate probably caused some fractional numbers to be produced by the scalar multiply. Since the accounting department of this fictional company will not recognize the completion of a part of a product, but only whole products, they will want this operation to be reflected properly in the program. Use the INT function on the entire arrays of Shift1(*) and Shift2(*) to eliminate fractional parts.

MATRIX ALGEBRA

IDENTITY MATRIX:

MAT A=IDN   identity matrix

MATRIX TRANSPOSITION:

MAT A=TRN(B)

MATRIX MULTIPLICATION:

MAT A=B*C

MATRIX INVERSION:

MAT A=INV(B)

DETERMINANT FUNCTION:

Det=DET(B)

O&P, Ch. 5, "Array Operations"   49

OTHER ARRAY OPERATIONS

REDIMENSIONING

```
DIM A(10,2)
REDIM A(6,3)
```

INITIALIZATION

```
MAT A=CON
MAT A=ZER
MAT A=(-9999)
```

NUMBER OF ROWS/COLUMNS FUNCTIONS

```
B=ROW(A)
C=COL(A)
```

O&P, Ch. 5, "Array Operations" 50

ADVANCED TECHNIQUES

a. Special function keys as typing aids

The special function keys can be used to represent any sequence of legal keystrokes, including a single occurrence of either EXECUTE, STORE, CONT, PAUSE, other keys, etc.

How is this done?
1) Type EDIT and press the special function key you want to define.
2) Then type any sequence of keystrokes you want to use. Notice that the left and right arrows, and the insert and delete character keys will behave normally (i.e., for local editing) unless you press them at the same time as you press the CONTROL key, in which case, they will be interpreted to mean that you want those keys pressed as part of the definition of the key.
3) To terminate and store the key definition, press the key again, to store the definition.

Exercise:
Define special function keys that will:

1. cause the statement REMARK to be entered as a program line whenever it is pressed. This key can then be used in EDITLINE mode to type in a program consisting entirely of REMARK's.

2. find the square root of the value of RES.

3. enter the value 7*8^5/(15*6) every time an INPUT statement is encountered.

$\boxed{\text{Shift}}$ gives you another 16 special function keys

ERASING MEMORY

ERASING MEMORY

SCRATCH - ERASES ALL PROGRAM LINES IN MEMORY.

SCRATCH A - ERASES ENTIRE MEMORY; POWER-UP.

SCRATCH C - ERASES VARIABLES, INCLUDING COMMON.
But Not program

SCRATCH KEY - ERASES ALL SFK DEFINITIONS.

SCRATCH KEY 1 - ERASES ONLY SFK #1.

SCRATCH P - ERASES PROGRAM LINES, BINARY
ROUTNES, VARIABLES, AND THE FILES TABLE.

SCRATCH V- ERASES ALL VARIABLES EXCEPT COMMON.

O&P, Ch. 4, "Erasing Memory" 52

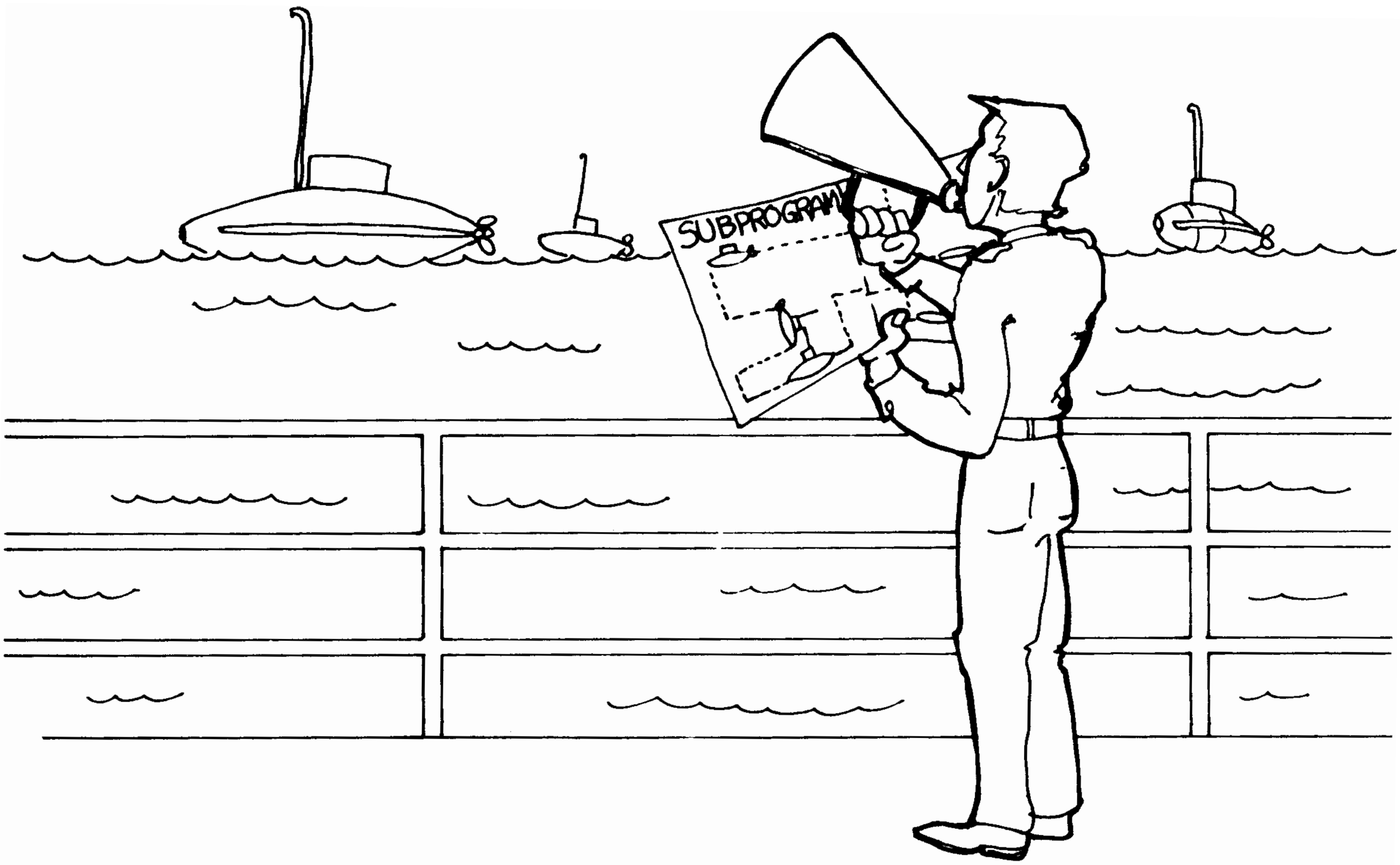
COMPUTED GOTO EXAMPLE

```
10    INPUT "Enter 1, 2, or 3",X
20    ON X GOTO One,Two,Three
30 One:   PRINT "One was entered"
40    GOTO 10
50 Two:   PRINT "Two was entered"
60    GOTO 10
70 Three:   PRINT "Three was entered"
80    GOTO 10
```

En 22.

COMPUTED GOSUB EXAMPLE

```
    10    INPUT "Enter 1, 2, or 3",X
--> 20    ON X GOSUB One,Two,Three
--> 21    PRINT "was entered"
--> 22    GOTO 10
    30 One:   PRINT "One ";
--> 40    RETURN
    50 Two:   PRINT "Two ";
--> 60    RETURN
    70 Three:   PRINT "Three ";
--> 80    RETURN
```


SUBPROGRAM

WHAT ARE SUBPROGRAMS?

* SUBPROGRAMS ARE SECTIONS OF CODE THAT ARE KEPT COMPLETELY APART FROM THE MAIN PROGRAM AND FROM OTHER SUBPROGRAMS.

WHAT ARE THEY GOOD FOR?

* INDEPENDENCE => PORTABILITY. A GIVEN SUBPROGRAM MAY BE USEFUL FOR SEVERAL APPLICATIONS. LOCAL VARIABLES.

* INDEPENDENCE => MODULARITY. EASIER TO DEVELOP, MODIFY, AND DEBUG.

* DYNAMIC MEMORY ALLOCATION.

* PARAMETER LIST ENABLES ACCESS WITH MANY DIFFERENT SETS OF VARIABLES.

O&P, Ch. 7, "Introduction" 55

SUBPROGRAM EXAMPLE

```
10   OPTION BASE 1
20   DIM Array(10),Numbers(5)
30   INPUT "Enter the first array",Array(*)
40   INPUT "Enter the second array",Numbers(*)
50   CALL Sort(Array(*),10)
60   PRINT "Here is the sorted first array",LIN(1),Array(*)
70   CALL Sort(Numbers(*),5)
80   PRINT "Here is the sorted second array",LIN(1),Numbers(*)
90   END
100  SUB Sort(A(*),N)
110  FOR I=1 TO N-1
120     FOR J=I+1 TO N
130        IF A(I)<=A(J) THEN 170
140        T=A(I)
150        A(I)=A(J)
160        A(J)=T
170     NEXT J
180  NEXT I
190  SUBEXIT
200  SUBEND
```

PASS BY REFERENCE EXAMPLE

```
     10   OPTION BASE 1
     20   DIM Array(10),Numbers(5)
-->  21   DATA 10,5
-->  22   READ Size1,Size2
     30   INPUT "Enter the first array",Array(*)
     40   INPUT "Enter the second array",Numbers(*)
-->  50   CALL Sort(Array(*),Size1)
-->  60   PRINT "Here is the first";Size1;"element array",LIN(1),Array(*)
-->  70   CALL Sort(Numbers(*),Size2)
-->  80   PRINT "Here is the second";Size2;"element array",LIN(1),Numbers(*)
     90   END
    100   SUB Sort(A(*),N)
    110   FOR I=1 TO N-1
    120      FOR J=I+1 TO N
    130         IF A(I)<=A(J) THEN 170
    140         T=A(I)
    150         A(I)=A(J)
    160         A(J)=T
    170      NEXT J
    180   NEXT I
--> 181   N=0
    190   SUBEND
```

Size1 = 0

PASS BY VALUE EXAMPLE – doesn't change value of variable

```
     10   OPTION BASE 1
     20   DIM Array(10),Numbers(5)
     21   DATA 10,5
     22   READ Size1,Size2
     30   INPUT "Enter the first array",Array(*)
     40   INPUT "Enter the second array",Numbers(*)
 --> 50   CALL Sort(Array(*),(Size1))
     60   PRINT "Here is the first";Size1;"element array",LIN(1),Array(*)
 --> 70   CALL Sort(Numbers(*),(Size2))
     80   PRINT "Here is the second";Size2;"element array",LIN(1),Numbers(*)
     90   END
     100  SUB Sort(A(*),N)
     110  FOR I=1 TO N-1
     120     FOR J=I+1 TO N
     130        IF A(I)<=A(J) THEN 170
     140        T=A(I)
     150        A(I)=A(J)
     160        A(J)=T
     170     NEXT J
     180  NEXT I
     181  N=0
     190  SUBEND
```

Here the variable SIZE1 / SIZE2 will not change their values on return from the main program

LOCAL VARIABLES EXAMPLE

```
10   FOR Count=1 TO 10
20     PRINT Count;"IN MAIN";
30     CALL Sub(Count)
40   NEXT Count
50   END
60   SUB Sub(M)
70   Count=M+2            ! What if this was M=M+2?
80   PRINT Count;"IN SUB"
90   SUBEND
```

FUNCTION SUBPROGRAMS

* USED DIRECTLY, SUCH AS A MATHEMATICAL FORMULA MIGHT USE A FUNCTION.

* RETURNS A VALUE JUST BY BEING USED, WHILE SUBROUTINE SUBPROGRAM CAN ONLY RETURN VALUES THROUGH THE PARAMETER LIST.

* TWO TYPES: SINGLE- AND MULTIPLE-LINE FUNCTIONS.

O&P, Ch. 7, "Multiple-Line Function Subprograms" 60

SINGLE- vs. MULTIPLE-LINE FUNCTIONS

* LENGTH: SINGLE-LINE FUNCTION DEFINITION IS ONE LINE LONG; MULTIPLE-LINE FUNCTIONS ARE DEFINED IN SEVERAL LINES.

* SCOPE: SINGLE-LINE FUNCTION IS LIKE A LOCAL VARIABLE; THOSE DEFINED IN MAIN ARE UNKNOWN TO SUBS, AND VICE VERSA; MULTIPLE-LINE FUNCTIONS ARE TRULY SUBPROGRAMS AND CAN BE USED BY MAIN OR OTHER SUBS.

* WHERE THEY CAN BE USED: SINGLE-LINE FUNCTIONS CAN BE USED ANYWHERE AN EXPRESSION (STRING OR NUMERIC) CAN BE USED; MULTIPLE-LINE FUNCTIONS CANNOT APPEAR IN PRINT STATEMENTS.

O&P, Ch. 7, "Multiple-Line Function Subprograms" 61

SINGLE AND MULTIPLE-LINE FUNCTION EXAMPLE

```
10   FOR I=1 TO 10
20      X=FNPoly(I)
30      PRINT I,FNSquare(I),X
40   NEXT I
50   STOP
60   DEF FNSquare(Variable)=Variable*Variable
70   END
80   DEF FNPoly(Nomial)
90   IF Nomial<=5 THEN RETURN 4*Nomial^3-2*FNSquare(Nomial)+5
100  RETURN 54.5
110  DEF FNSquare(Variable)=Variable*Variable
120  FNEND
```

Ex 28

STRING FUNCTION EXAMPLE

```
10    DIM A$[8],D$[8]
20    A$="10010011"
30    D$=FNRotateleft$((A$),1)
40    PRINT A$,LIN(1),D$
50    END
60    DEF FNRotateleft$(X$,Nopos)
70    DIM N$[1]
80    FOR I=1 TO Nopos
90       N$=X$[1,1]
100      X$=X$[2]&N$
110   NEXT I
120   RETURN X$
130   FNEND
10010011
00100111
```

RECURSION EXAMPLE

```
10    INPUT "Please enter a number less than 70",N
20    N=INT(N)
30    IF (N<0) OR (N>69) THEN 10
40    F=FNFactorial((N))
50    PRINT N;"factorial is";F
60    GOTO 10
70    DEF FNFactorial(X)
80    IF X=0 THEN RETURN 1
90    RETURN X*FNFactorial(X-1)
100   FNEND
```

Ex 30

MEMORANDUM ON 9845 SUBPROGRAMS

CALLING PROGRAM

The following are saved prior to entering a subprogram and are restored following exit from the subprogram.

1. DATA pointer
2. file table
3. OPTION BASE
4. trig units (RAD, DEG, GRAD)
5. format (STANDARD, FIXED, FLOAT)
6. ON ERROR line id or subprogram name
7. ON KEY GOTO or ON KEY GOSUB line id
   (ON KEY CALL statements are global and executed immediately.)

CALLED PROGRAM

The following are initialized at the start of a subprogram.

1. DATA pointer
2. file table
3. OPTION BASE
4. trig units (RAD, DEG, GRAD)

Do one of the following exercises on subprograms:

1. Write a program which allows the user to enter some numbers into an array. The program should then find the mean of the array by using a single-line function. (Hint: Use the SUM and ROW array operations.)

2. Write a function subprogram that will search through a vector of numbers A(*), and return the subscript of the vector element which contains a certain value. If the given value does not happen to be in the vector, the function should return Ø. (Hints: Assume the vector's lower subscript is 1. Use the ROW function to find how large the vector is. Use READ/DATA to initialize the vector in the main program. The function should be accessed by FNSearch(A(*),X).)

3. Write a subroutine subprogram that will exchange the values of two variables. The subprogram should be accessed by CALL Switch(X,Y). X and Y should be entered from the keyboard, and their values should be printed both before and after the subprogram is called.

COMMON BLOCK EXAMPLE WITH ERROR

```
10   OPTION BASE 1
20   COM A,B,C,D,X$,A(5)
30   A=1
40   B=2
50   C=D=3
60   X$="HELLO"
70   DATA 1,2,3,4,5
80   READ A(*)
90   CALL Sandwich
100  END
110  SUB Sandwich
120  COM W,X,Y,Z,R$
130  PRINT W,X,Y,Z,R$
140  PRINT A(*)
150  SUBEND
```

COMMON BLOCK EXAMPLE CORRECTED

```
     10    OPTION BASE 1
     20    COM A,B,C,D,X$,A(5)
     30    A=1
     40    B=2
     50    C=D=3
     60    X$="HELLO"
     70    DATA 1,2,3,4,5
     80    READ A(*)
     90    CALL Sandwich
     100   END
     110   SUB Sandwich
     120   COM W,X,Y,Z,R$
 --> 121   COM A(*)
     130   PRINT W,X,Y,Z,R$
     140   PRINT A(*)
     150   SUBEND
```

REAL EXAMPLE

```
10    OPTION BASE 1
20    REAL A,B,C,X,Array(4),Boogie(3,4,5,6,7)
30    END
```

SHORT EXAMPLE

```
10    OPTION BASE 1
20    SHORT A,B,C,X,Array(4),Boogie(3,4,5,6,7)
30    END
```

INTEGER EXAMPLE

```
10    OPTION BASE 1
20    INTEGER A,B,C,X,Array(4),Boogie(3,4,5,6,7)
30    END
```

DATA TYPES IN PARAMETER LISTS

* THE WORDS SHORT AND INTEGER MUST PRECEDE VARIABLES WHICH ARE OF ALTERNATE DATA TYPES.

* ANY VARIABLE FOLLOWING A STRING IS ASSUMED TO BE REAL UNLESS OTHERWISE DECLARED.

FOR EXAMPLE:

```
200  SUB Program(X,SHORT Y,A,B,X$,T,INTEGER
     N(*),I,REAL U)
```

O&P, Ch. 7, "Parameters" 69

DATA TYPES IN COM STATEMENTS

* DO NOT MIX DATA TYPES BETWEEN COM DECLARATION STATEMENTS.

FOR EXAMPLE:

```
10    COM X,SHORT Y,A,B,X$,T,INTEGER N(5),I,
      REAL U
```

IF PAIRED WITH THIS COM STATEMENT, ERROR RESULTS BECAUSE SECOND MEMBERS DON'T MATCH IN TYPE:

```
1000 COM C,D,P,Q,E$,T,INTEGER N(*),I,REAL V
```

O&P, Ch. 5, "Declaring and Dimensioning Variables" 70

PROUND FUNCTION

SYNTAX: PROUND (<expression>,<power of ten>)

EXAMPLES:

```
10   INPUT "Balance?",Balance
20   INPUT "Interest rate?",Interest_rate
30   Late_charge=PROUND(Interest_rate*Balance,-2)
40   ! Store Late_charge in the data base
```

```
10   INPUT "Enter a number",Number
20   Log2n=PROUND(LGT(Number)/LGT(2),-8)
30   PRINT Log2n
40   STOP
```

71

DROUND FUNCTION

SYNTAX: DROUND (<expression>,<# of digits>)

EXAMPLES:

```
10   INPUT "Voltage?",Volts
20   INPUT "Current?",Current
30   X=DROUND(FNResistance(Volts,Current),4)
40   PRINT "Resistance=";X
50   STOP
```

72

FORMATTED PRINT

* ENABLES OUTPUT TO BE EASILY FORMATTED TO SUIT YOUR NEEDS:
  - RIGHT-JUSTIFICATION OF NUMBERS
  - ALIGNMENT ON DECIMAL POINT
  - "HUMANIZED" DIGIT SEPARATION FOR LONG NUMBERS
  - FLOATING TEXT UP TO FIRST DIGIT OF NUMBER FOR PROTECTION (CHECKS, ETC.)

* PRINT USING STATEMENT TELLS THE SYSTEM <u>WHAT</u> NEEDS TO BE PRINTED; IMAGE STATEMENT TELLS <u>HOW</u> IT SHOULD BE PRINTED.

* SPECIFIERS: NUMERIC, STRING, CARRIAGE CONTROL.

O&P, Ch. 9, 'Formatted Output' 73

PRINT vs. PRINT USING/IMAGE

```
10    PRINT "Pi=";PI
20    END
Pi= 3.1415926536
```

```
10    PRINT USING 20;"Pi=",PI
20    IMAGE K
30    END
Pi=3.1415926536
```

or

PRINT USING "K";"Pi=",PI

PRINT USING WITH TEXT

```
10    PRINT "Pi=";PI
20    END
Pi= 3.1415926536
```



```
10    PRINT USING 20;"Pi=",PI
20    IMAGE 3A,K
30    END
Pi=3.1415926536
```

```
10    PRINT USING 20;PI
20    IMAGE "Pi=",K
30    END
Pi=3.1415926536
```

PRINT USING WITH BLANK SPACES

```
10   PRINT USING 20; "Thisisastringoftext", "So is this"
20   IMAGE 4AX2AXAX6AX2AX4A".",2X,K"."
30   STOP
This is a string of text.  So is this.
```

NUMERIC SPECIFIERS: D

```
10   DATA 1,20,548,8754
20   FOR I=1 TO 4
30      READ A
40      PRINT USING 50;A
50      IMAGE 4D
60   NEXT I
70   STOP
   1
  20
 548
8754
```

NUMERIC SPECIFIERS: DECIMAL POINT

```
--> 10   DATA 1.4,20,548.35,8754
    20   FOR I=1 TO 4
    30      READ A
    40      PRINT USING 50;A
--> 50      IMAGE 4D.2D     ! This line could have been "IMAGE DDDD.DD"
    60   NEXT I
    70   STOP
       1.40
      20.00
     548.35
    8754.00
```

replaces leading zeros with blanks

right-justified

NUMERIC SPECIFIERS: SCIENTIFIC NOTATION-

```
    10   DATA 1.4,20,548.35,8754
    20   FOR I=1 TO 4
    30      READ A
    40      PRINT USING 50;A
--> 50      IMAGE D.3DE
    60   NEXT I
    70   STOP
    1.400E+00
    2.000E+01
    5.484E+02
    8.754E+03
```

NUMERIC SPECIFIERS: Z — Zero specifier

```
--> 10    DATA 1,20,548,8754
    20    FOR I=1 TO 4
    30       READ A
    40       PRINT USING 50;A
--> 50       IMAGE 4Z
    60    NEXT I
    70    STOP
    0001
    0020
    0548
    8754
```

4 zeros where specifier shows

LEADING ZERO BEFORE DECIMAL POINT

```
--> 10    DATA 0.4,20,548.35,8754
    20    FOR I=1 TO 4
    30       READ A
    40       PRINT USING 50;A
--> 50       IMAGE 3DZ.2D
    60    NEXT I
    70    STOP
       0.40
      20.00
     548.35
    8754.00
```

NUMERIC SPECIFIERS: *

```
10   DATA 0.4,20,548.35,8754
20   FOR I=1 TO 4
30      READ A
40      PRINT USING 50;A
50      IMAGE "$"3*Z.2D
60   NEXT I
70   STOP
$***0.40
$**20.00
$*548.35
$8754.00
```

NC Cheques

NUMERIC SPECIFIERS: SIGNS - M AND S

```
--> 10   DATA -0.4,20,548.35,-8754
    20   FOR I=1 TO 4
    30      READ A
--> 40      PRINT USING 50;A,A,A,A,A
--> 50      IMAGE M4D.2D,5X,S4D.2D,5X,M4Z.2D,5X,S3DZ.DD,5X,3DZ.2DM
    60   NEXT I
    70   STOP
        -.40         -.40     -0000.40       -0.40        0.40-
       20.00       +20.00      0020.00      +20.00       20.00
      548.35      +548.35      0548.35     +548.35      548.35
    -8754.00     -8754.00     -8754.00    -8754.00     8754.00-
```

NUMERIC SPECIFIERS: DIGIT SEPARATOR - C

```
--> 10   DATA -12365487.55,-214.87,9.487E8,4369
    20   FOR I=1 TO 4
    30      READ A
--> 40      PRINT USING 50;A,A
--> 50      IMAGE M8DZ.2D,5X,M3DC3DC3D.DD
    60   NEXT I
    70   STOP
     -12365487.55       -12,365,487.55
          -214.87              -214.87
      948700000.00       948,700,000.00
           4369.00             4,369.00
```

highlight the thousands

Australian Standards

EUROPEAN NUMERIC SPECIFICATIONS

```
      10   DATA -12365487.55,-214.87,9.487E8,4369
      20   FOR I=1 TO 4
      30      READ A
      40      PRINT USING 50; A,A
  --> 50      IMAGE M8DZR2D,5X,M3DP3DP3DRDD
      60   NEXT I
      70   STOP
       -12365487,55        -12.365.487,55
            -214,87                -214,87
       948700000,00        948.700.000,00
            4369,00              4.369,00
```

P — format specifier

CARRIAGE CONTROL EXAMPLE

```
10   DATA The, quick, brown, fox, jumped, over, the, lazy, dog.
20   FOR I=1 TO 9
30      READ A$
40      PRINT USING 50; A$
50      IMAGE -, K      suppresses CR ; doesn't suppress line feed.
60   NEXT I
70   PRINT USING 80; "The"
80   IMAGE #, K
90   PRINT USING 100; " end"
100  IMAGE K, /, "Bye now"
110  END
The
   quick
        brown
             fox
                jumped
                      over
                          the
                             lazy          CR/
                                 dog.     suppressed line feed now
                                     The end
Bye now
```

IMPLIED IMAGE EXAMPLE

```
    10   DATA The,quick,brown,fox,jumped,over,the,lazy,dog.
    20   FOR I=1 TO 9
    30      READ A$
--> 40      PRINT USING "-,K";A$
    60   NEXT I
--> 70   PRINT USING "#,K";"The"
--> 80   A$="K,/,"&CHR$(34)&"Bye now"&CHR$(34)
--> 90   PRINT USING A$;" end"
    110  END
    The
       quick
            brown
                 fox
                    jumped
                          over
                              the
                                 lazy
                                     dog.
                                         The end
  "Bye now"
   end
```

Exercise:

1. Write a program to print out a table of X, X^2, X^3, and SQR(X), for X as it ranges from 51 to 100. Center appropriate headings over the columns. Print out X^2 and X^3 with digit separators, and SQR(X) to 2 decimal places. Print a blank line after every 5 values of X. Note: There is a summary of the IMAGE specifiers in the Reference Guide.

INTRODUCTION TO KEYBOARD INTERRUPTS

```
1Ø    ON KEY #Ø GOTO End
2Ø    GOTO 2Ø
3Ø End:   PRINT "Program finished"
4Ø    BEEP
5Ø    END
```

Only useable in this way when program is running

ON KEY WITH GOTO, GOSUB, AND CALL

```
10    ON KEY #0 GOTO End
20    ON KEY #1 GOSUB Print
30    ON KEY #2 CALL The_police
40    GOTO 40
50 Print:    FOR I=1 TO 10
60           PRINT "Don't worry.  I'm still running"
70           NEXT I
80           RETURN
90 End:      PRINT "Program finished"
100          BEEP
120          END
130   SUB The_police
140   FOR I=1 TO 10
150      PRINT "Help! Murder! Police!  Someone's pushing key #2!!! "
160   NEXT I
170   SUBEXIT
```

PRIORITIZED INTERRUPT FROM THE KEYBOARD

```
10    ON KEY #0,15 GOTO End
20    ON KEY #1,1 GOSUB Print1
30    ON KEY #2,2 GOSUB Print2
40    ON KEY #3,3 GOSUB Print3
50    DISP "IDLE"
60    GOTO 50
70 Print3:  FOR I=1 TO 50
80          DISP "Top priority!!! -- ";I
90          WAIT 100
100         NEXT I
110         RETURN
120 Print2: FOR J=1 TO 50
130         DISP "Second priority! -- ";J
140         WAIT 100
150         NEXT J
160         RETURN
170 Print1: FOR K=1 TO 50
180         DISP "Lowest priority! -- ";K
190         WAIT 100
200         NEXT K
210         RETURN
220 End:    PRINT "Program finished"
230         BEEP
240         END
```

Syntax:

ON KEY #<key number>[,<priority>] GOTO<line identifier>

ON KEY #<key number>[,<priority>] GOSUB<line identifier>

ON KEY #<key number>[,<priority>] CALL<subprogram name>

the higher the
value the higher
the priority

ENABLE/DISABLE

* AFFECT ALL ON KEY DECLARATIONS.

* TEMPORARILY DISABLE WITH DISABLE.

* RE-ENABLE WITH ENABLE.

* WHEN DISABLES, KEYS ARE BUFFERED UP (ONE INTERRUPT PER KEY) UNTIL ENABLED.

* ENABLE/DISABLE ARE GLOBAL, AFFECTING ALL PROGRAMS AND SUBPROGRAMS IN MEMORY.

O&P, Ch. 4, "Interrupting a Program" 91

ADDITIONAL KEYBOARD INTERRUPTS

```
SYNTAX: ON KBD [<priority>] GOTO  <line id>[,ALL]
        ON KBD [<priority>] GOSUB <line id>[,ALL]
        ON KBD [<priority>] CALL <s/p name>[,ALL]
```

EXAMPLE:

```
10 ON KBD GOTO Select
20 GOTO 20
30 Select:  Key$=KBD$           !KBD$ IS A BUFFER
40 IF Key$="A" THEN PRINT "PROGRAM A WAS SELECTED"
50 IF Key$="B" THEN PRINT "PROGRAM B WAS SELECTED"
60 GO TO 10
```

→ just waits on this statement

KBD$ can contain up to 80 characters

9845B ONLY

91A

can be used in a
menu type situation
instead

specifying ALL
contains generally any key
as a keyboard interrupt

ERROR TRAPPING

* ON ERROR STATEMENT ENABLES YOU TO BRANCH TO AN ERROR RECOVERY ROUTINE WHEN AN ERROR OCCURS.

* OFF ERROR STATEMENT DEACTIVATES ON ERROR.

* THREE SYSTEM FUNCTIONS ARE ESTABLISHED WHEN AN ON ERROR BRANCH OCCURS:

  - ERRN: RETURNS THE ERROR NUMBER.

  - ERRL: RETURNS THE LINE NUMBER IN WHICH THE ERROR OCCURRED.

  - ERRM$: RETURNS THE ENTIRE ERROR MESSAGE (ERRL AND ERRN).

O&P, Ch. 8, "ON ERROR" 92

MEMORANDUM ON 9845 ERROR TRAPPING

There are no simple, clearcut rules that define trappable errors in the 9845A. Here are some GUIDELINES concerning NONTRAPPABLE ERRORS.

1. Syntax errors.
2. Errors in statements executed from the keyboard.
3. Errors detected by the I/O processor when the computer is in OVERLAP mode.
4. Errors detected by the I/O processor on LOAD, STORE, GET, LINK, and SAVE statements.
5. Certain execution errors such as Error 5 - abnormal program termination or Error 49 resulting from ON ERROR GOTO a nonexistent label.
6. Most error numbers are processor specific, i.e., they are generated by either the computation processor or the I/O processor but not both. When exceptions occur, the error may be trappable if detected by the computation processor but not trappable if detected by the I/O processor.

ERROR TRAPPING EXAMPLE

```
10   OPTION BASE 1
20   DIM A(5)
30   INPUT A(*)
40   PRINT A(*);
50   INPUT "Do you want to make any changes (Y/N)?",A$
60   IF A$="Y" THEN Change
70   STOP
80 Change:   INPUT "Which element do you want to change?",I
90   ON ERROR GOTO Trap
100  INPUT "Enter new number",A(I)
110  OFF ERROR
120  GOTO 40
130 Trap:  BEEP
140  IF ERRN<>17 THEN 180
150  DISP "Subscript out of range -- try again."
160  WAIT 1000
170  GOTO Change
180  PRINT ERRM$
190  STOP
```

Ex. 59

DEFAULT ON/OFF

* DEFAULT ON STATEMENT CAUSES CERTAIN DEFAULT VALUES TO BE USED WHEN ILLEGAL ARITHMETIC OPERATIONS ARE PERFORMED, RATHER THAN GENERATING AN ERROR.

* DEFAULT OFF IS NORMAL, ERROR-PRODUCING CONDITION.

* SEE O&P MANUAL FOR LIST OF DEFAULT VALUES.

O&P, Ch. 3, 'Math Errors - Recovery' 94

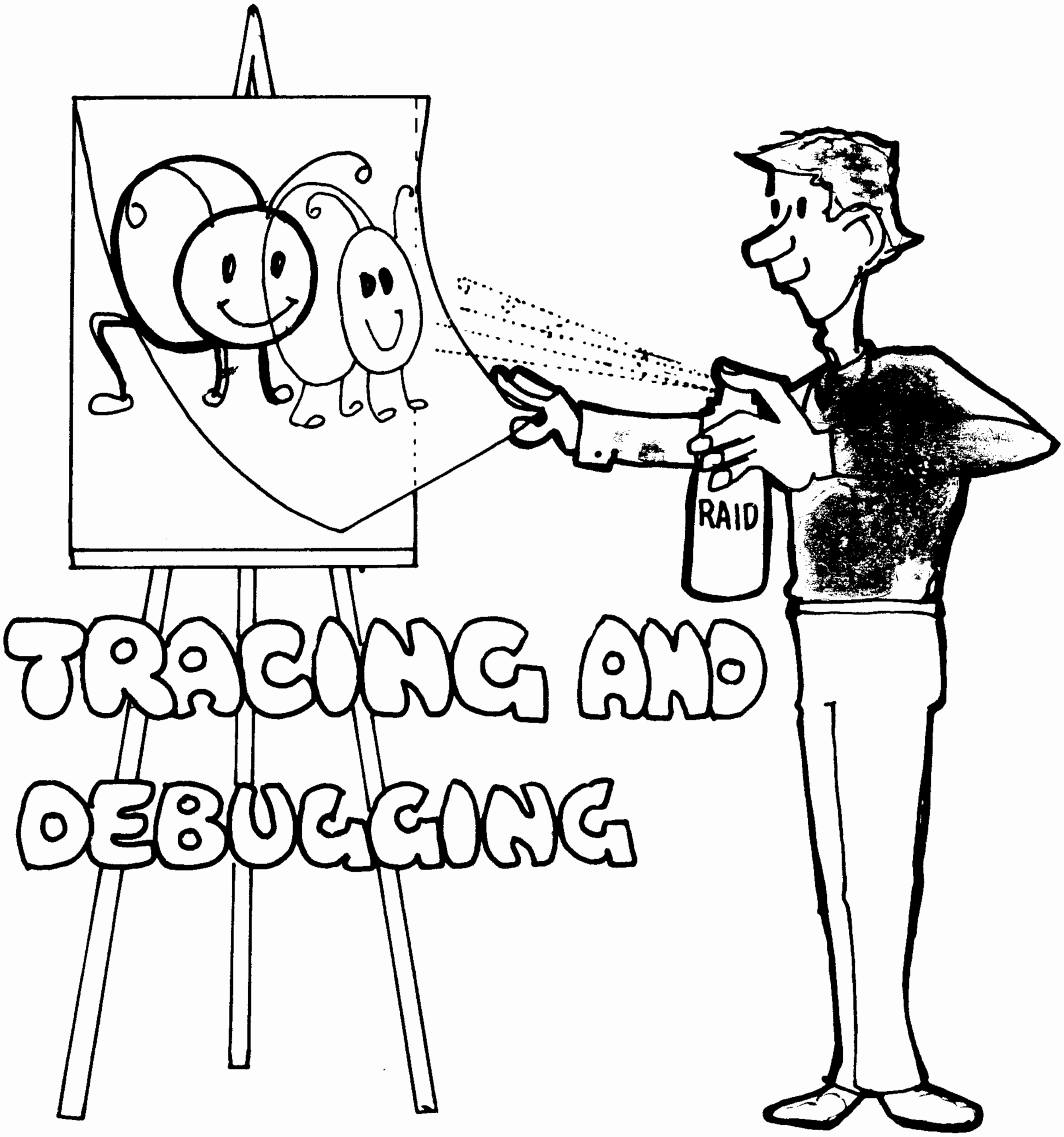
PAUSE

* ENABLES YOU TO HALT THE PROGRAM WITHOUT RESETTING THE PROGRAM COUNTER TO THE FIRST LINE OF THE PROGRAM.

* JUST LIKE PRESSING THE PAUSE KEY, EXCEPT YOU CAN CONTROL WHERE YOU ARE PAUSING.

* CONTINUE KEY CAUSES CONTINUATION OF PROGRAM.

O&P, Ch. 4, "Stopping and Resuming Execution" 95



TRACING AND DEBUGGING

TRACING AND DEBUGGING

* STEP KEY:
  - FOR SHORT SECTIONS
  - USE WITH PAUSE

* TRACE STATEMENTS:
  - LONG PROBLEMS WITH UNLOCALIZED BUGS
  - REPETITIOUS PROBLEMS
  - PRINTS MESSAGES TO BOTTOM LINE OF CRT
  - RECORD CAN BE OBTAINED WITH PRINT ALL

O&P, Ch. 8, 'Debugging a Program' 96

TRACING EXAMPLES

```
    10   TRACE
    20   DIM A(1:5)
    30   FOR I=1 TO 5
    40   INPUT A(I)
    50   NEXT I
    60   FOR I=1 TO 5
    70   PRINT A(I)
    80   NEXT I
    90   END

--> 10   TRACE 60
    20   DIM A(1:5)
    30   FOR I=1 TO 5
    40   INPUT A(I)
    50   NEXT I
    60   FOR I=1 TO 5
    70   PRINT A(I)
    80   NEXT I
    90   END

--> 10   TRACE 30,60
    20   DIM A(1:5)
    30   FOR I=1 TO 5
    40   INPUT A(I)
    50   NEXT I
    60   FOR I=1 TO 5
    70   PRINT A(I)
    80   NEXT I
    90   END
```

OTHER TRACE STATEMENTS

* TRACE VARIABLES - LIST OF UP TO 5 VARIABLES

* TRACE ALL VARIABLES

* TRACE ALL - TRACE AND TRACE ALL VARIABLES.

* TRACE WAIT - WAITS EVERY TIME A TRACE MESSAGE IS PRINTED.

* TRACE PAUSE - SET BREAK POINTS IN PROGRAM FROM WHICH YOU CAN STEP YOUR PROGRAM OR CONTINUE.

O&P, Ch. 8, "Debugging a Program" 98

To get out of TRACE mode

use NORMAL.

"SHUTTING OFF" TRACE

* NORMAL - SHUTS OFF ANY AND ALL TRACE STATEMENTS.

* TRACE - TRACE WITH DUMMY PARAMETERS

* TRACE ALL VARIABLES - TRACE ALL VARIABLES WITH LOWER AND UPPER LINE NUMBER RANGE NOT IN PROGRAM, SUCH AS LINE 1.

* TRACE, TRACE VARIABLES, TRACE PAUSE SIMILAR.

* TRACE WAIT - USE TRACE WAIT 0.

RANDOM NUMBER FUNCTION

```
10    OPTION BASE 1
20    DIM A(10)
30    FOR I=1 TO 10
40        A(I)=RND
50    NEXT I
60    PRINT A(*)
70    STOP
```

EXAMPLE GENERATING NUMBERS FROM 1 TO 1000

```
    10   OPTION BASE 1
    20   DIM A(10)
    30   FOR I=1 TO 10
--> 40      A(I)=INT(RND*1000)+1
    50   NEXT I
    60   PRINT A(*)
    70   STOP
```

RANDOMIZE

* FORCES A DIFFERENT SEQUENCE OF RANDOM NUMBERS.

* RANDOMIZE - FORCES ONE OF 116 MACHINE-CHOSEN SEEDS; NON-REPEATABLE.

* RANDOMIZE N, WHERE N IS SOME INTEGER VALUE, CAUSES RND FUNCTION TO RETURN 0.

* RANDOMIZE X, WHERE X IS SOME NON-INTEGER REAL VALUE, SETS SEED USING X IN THE CALCULATION.

* WITHOUT RANDOMIZE - PI/180 IS USED.

O&P, Ch. 3, "Math Functions and Statements" 102

SUSPEND/RESUME INTERACTIVE

* LIVE KEYBOARD ENABLES YOU TO EXECUTE NUMERIC EXPRESSIONS, CHANGE THE VALUES OF VARIABLES, CHANGE PROGRAM LINES, PRINT THE VALUES OF VARIABLES, ETC. WHILE A PROGRAM IS RUNNING.

* TO SUSPEND OPERATOR INTERACTION THROUGH LIVE KEYBOARD: SUSPEND INTERACTIVE.

* TO RESUME OPERATOR INTERACTION THROUGH LIVE KEYBOARD: RESUME INTERACTIVE.

O&P, Ch. 3, "Operating Modes" 103

OVERLAP/SERIAL

* SERIAL MODE: TWO PROCESSORS NEVER RUN SIMULTANEOUSLY.

* OVERLAP MODE: TWO PROCESSORS CAN, WITH CAREFUL PROGRAMMING, BE MADE TO RUN CONCURRENTLY.

* IT IS UP TO THE PROGRAMMER TO DISTRIBUTE I/O STATEMENTS THOUGHOUT THE PROGRAM RATHER THAN ALL AT THE END.

* THROUGHPUT WILL INCREASE UP TO A FACTOR OF TWO.

O&P, Ch. 9, "Overlapped Processing" 104

THE CONTROL KEY

* WITH STOP = RESET (SIMILAR TO POWER-UP).

* WITH HIGHLIGHTING (SFK'S Ø, 1, AND 2) FOR THE CRT. NOTE IDIOSYNCRASIES OF HIGHLIGHTS.

* WITH ORDINARY KEYS (SEE ASCII TABLE).

* WITH TYPWTR KEY = SPACE DEPENDENT MODE.

O&P, Appx. C 1Ø5

MASS STORAGE

DEFINITION:

MASS STORAGE IS A MEANS OF SAVING LARGE AMOUNTS OF INFORMATION.

USES:

TO RETAIN INFORMATION IN MACHINE-READABLE FORM WHEN THE COMPUTER IS UNAVAILABLE, AND

TO RETAIN INFORMATION WHICH EXCEEDS THE MEMORY CAPACITY OF THE COMPUTER.

APPLICATIONS:

SAVING PROGRAMS, PARTIAL PROGRAMS, MEMORY.

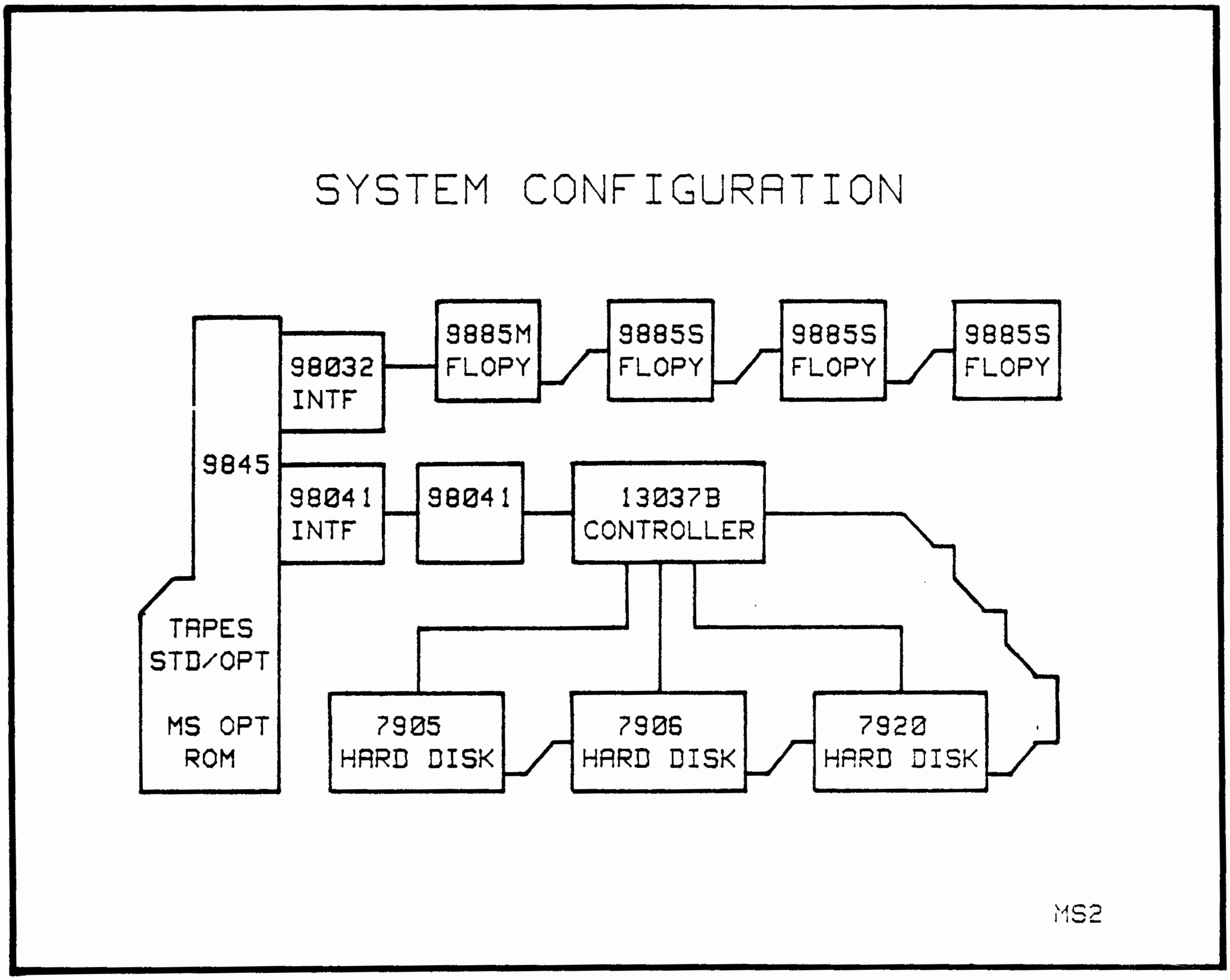
STORING DATA READ FROM A PERIPHERAL OR COMPUTED.

UNIFIED MASS STORAGE CONCEPT:

MASS STORAGE STATEMENTS ARE DEVICE INDEPENDENT!

"USES FOR MASS STORAGE", Ch.1, MST

MS1



a double floppy disc drive 9895
also exists

MASS STORAGE AGENDA

MEDIA AND DEVICES

PROGRAM STORAGE

GENERAL PURPOSE FILE STATEMENTS

FUNDAMENTALS OF DATA STORAGE

HOW TO IMPROVE DATA ACCESS SPEED

ADDITIONAL CAPABILITIES

MS3

MEDIA AND DEVICES

MASS STORAGE IS STATEMENT

INITIALIZE STATEMENT

REWIND STATEMENT

MS4

DEVICES AND MEDIA

DEVICES

MACHINERY THAT READS AND WRITES INFORMATION,
E.G.,

- TAPE DRIVES
- FLEXIBLE (FLOPPY) DISK DRIVES
- HARD DISK DRIVES

MEDIA

THE MATERIAL ON WHICH DATA IS STORED,
E.G.,

- TAPE CARTRIDGES
- FLEXIBLE (FLOPPY) DISKS
- PLATTERS

MS5

MASS STORAGE UNIT SPECIFIER (msus)

DEFINITION:

A STRING, SUBSTRING, OR STRING EXPRESSION THAT IDENTIFIES THE TYPE OF DEVICE AND ITS ADDRESS.

TAPE CARTRIDGES:

```
:T [select code]
   ":T15"        ":T"       Msus$
```

FLEXIBLE DISKS:

```
:F [select code [, 9885 unit code]]
   ":F8,0"       ":F"       ":"&Dt$&Sc$
```

HARD DISKS:

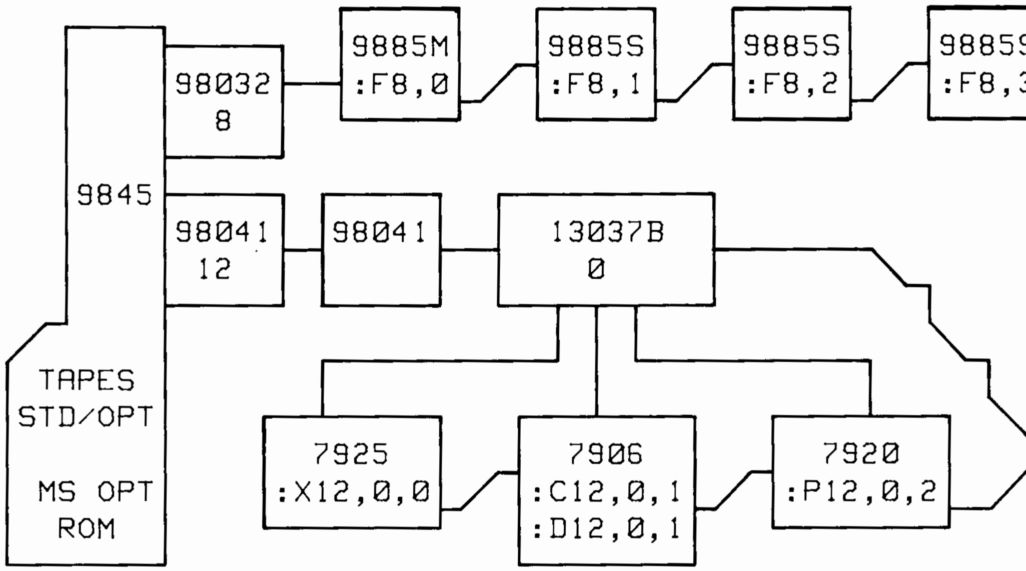
```
: device type [select code [, controller address
  [, unit code]]]
   ":P12,0,0"    ":P"       ":C"      ":Y"
```

"MASS STORAGE UNIT SPECIFIER", Ch.1, MST MSS

Medium Identifiers

ID	Medium
T	Tape Cartridge
F	Flexible Disk (HP9885)
Z	Fixed Disk (HP7905A)
Y	Removable Disk Cartridge (HP7905A)
D	Fixed Disk (HP7906A)
C	Removable Disk Cartridge (HP7906A)
P	Removable Disk Pack (HP7920)
X	Removable Disk Pack (HP7925)
H	Floppy disk unit HP 9895

MASS STORAGE UNIT SPECIFIER (msus)



MS7

MASS STORAGE IS STATEMENT

PURPOSE:

TO SELECT THE MASS STORAGE DEVICE.

SYNTAX:

```
MASS STORAGE IS msus
```

EXAMPLES:

```
MASS STORAGE IS ":F8"

File$="PARTS:P12"
MASS STORAGE IS File$[6,9]
MASS STORAGE IS File$[POS(File$,":")]

INPUT "TAPE UNIT? (14 OR 15)",Tapeunit
MASS STORAGE IS ":T"&VAL$(Tapeunit)
```

"MASS STORAGE IS STATEMENT", Ch.1, MST MS8

INITIALIZE STATEMENT

PURPOSE:

TO ESTABLISH BLOCKS OF STORAGE AND A DIRECTORY ON A MEDIUM.

RULE:

EVERY MEDIUM MUST BE INITIALIZED BEFORE IT CAN BE USED FOR STORAGE.

SYNTAX:

```
INITIALIZE msus
INITIALIZE msus [, interleave factor]
```

allows you to label tracks on discs other than standard

EXAMPLES:

```
INITIALIZE ":T15"
INITIALIZE ":F8",7
```

'INITIALIZATION' AND 'INTERLEAVING', Ch.2, MST

MS9

REWIND STATEMENT

PURPOSE:

TO REWIND A TAPE CARTRIDGE TO OBTAIN PROMPT RESPONSE ON THE NEXT ACCESS OR TO PREVENT DAMAGE FROM DUST WHILE THE TAPE IS STORED.

SYNTAX:

```
REWIND [msus]
```

EXAMPLES:

```
REWIND ":T15"

MASS STORAGE IS ":T15"
REWIND
```

"REWINDING THE TAPE", Ap.B, MST

MS10

MEDIA AND DEVICES EXERCISE 1

DEVELOP A PROGRAM THAT INITIALIZES A TAPE CARTRIDGE LOCATED IN EITHER OF THE TAPE DRIVES. ENTER THE SELECT CODE FROM THE KEYBOARD AND CHECK FOR A VALID VALUE. USE THE PROGRAM TO INITIALIZE A BLANK TAPE.

MEDIA AND DEVICES EXERCISE 2

DEVELOP A PROGRAM THAT SELECTS EITHER A TAPE CARTRIDGE OR A FLEXIBLE DISK FOR THE MASS STORAGE MEDIUM. ENTER THE DEVICE TYPE, SELECT CODE, AND IF APPLICABLE, THE 9885 UNIT CODE FROM THE KEYBOARD.

ALLOW THESE INPUTS FOR DEVICE TYPE.

T t TAPE Tape tape F f FLOPPY Floppy floppy

CHECK FOR A VALID SELECT CODE AND UNIT CODE.

TAPE SELECT CODE: 14 or 15
FLOPPY SELECT CODE: 1 THROUGH 12, INTEGER
FLOPPY UNIT CODE: 0 THROUGH 7, INTEGER

PROGRAM STORAGE

STORE STATEMENT
RE-STORE STATEMENT
STOREKEY STATEMENT
STOREALL STATEMENT
STOREBIN STATEMENT

SAVE STATEMENT
RE-SAVE STATEMENT

LOAD STATEMENT
LOADKEY STATEMENT
LOADALL STATEMENT
LOADBIN STATEMENT

GET STATEMENT
LINK STATEMENT

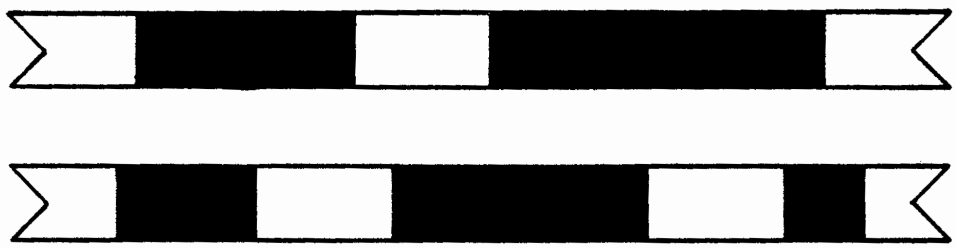
MS11

FILES

DEFINITION:

A FILE IS A BLOCK OF STORAGE THAT IS RESERVED ON A MEDIUM FOR STORING A COLLECTION OF INFORMATIONAL ITEMS SIMILAR TO ONE ANOTHER IN PURPOSE, FORM, AND CONTENT.

EXAMPLES:



"FILES", Ch.2, MST

MS12

MAXIMUM NUMBER OF FILES PER MEDIUM

CODE	MEDIUM	MAX # FILES
T	TAPE CARTRIDGE	42
F	FLEXIBLE DISK (HP9885)	352
Z	FIXED DISK (HP7905A)	1136
Y	REMOVABLE DISK CARTRIDGE (HP7905A)	2288
D	FIXED DISK (HP7906A)	2288
C	REMOVABLE DISK CARTRIDGE (HP7906A)	2288
P	REMOVABLE DISK PACK (HP7920)	8000

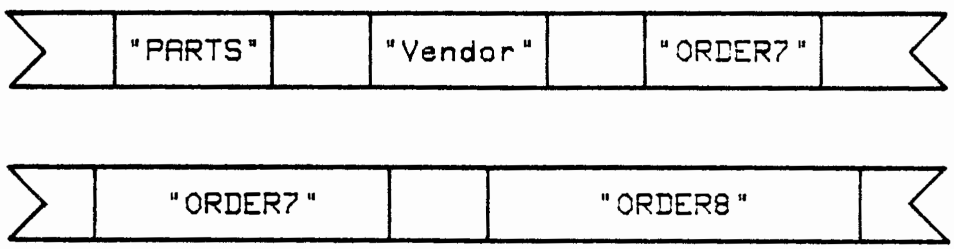
FILE NAMES

DEFINITION:

A FILE NAME IS A STRING FROM ONE TO SIX CHARACTERS LONG THAT IDENTIFIES A FILE.

EXAMPLES:

"PARTS" "Vendor" "ORDER"&VAL$(Order_no)



"FILE NAMES", Ch.1, MST MS13

FILE SPECIFIER

PURPOSE:

TO IDENTIFY A FILE ON A PARTICULAR MEDIUM.

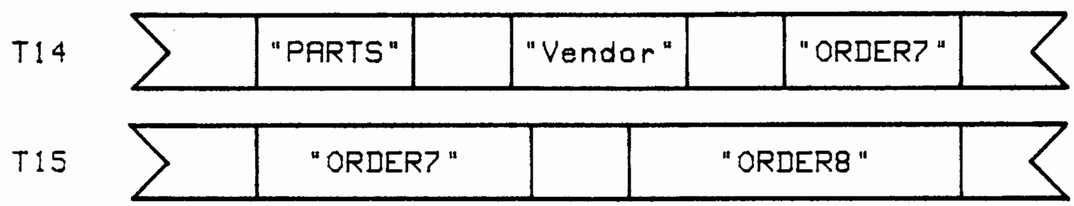
FORM AND SHORTHAND NOTATION:

file name [msus]   file specifier

EXAMPLES:

MASS STORAGE IS ":T14"

"PARTS"   "ORDER7"   "ORDER7:T15"



"FILE SPECIFIER", Ch.1, MST   MS14

FILE TYPES

TYPE	USE	TO CONSTRUCT	TO RETRIEVE
PROGRAM	QUASI-COMPILED PROGRAM	STORE RE-STORE	LOAD
BINARY PROGRAM	BINARY PROGRAM	STOREBIN	LOADBIN
KEYS	SPECIAL FUNCTION KEYS	STOREKEY	LOADKEY
STOREALL	MACHINE MEMORY	STOREALL	LOADALL
DATA	SOURCE PROGRAM	SAVE RE-SAVE	GET LINK
	DATA	CREATE PRINT#	READ#
BINARY DATA	BINARY DATA	FCREATE FPRINT	FREAD

'TYPES OF FILES', Ch.2, MST

MS15

THE STORE STATEMENTS

PURPOSE	SYNTAX	EXAMPLE
QUASI-COMPILED PROGRAM	STORE file specifier RE-STORE file specifier [, protect code]	STORE "MAIN" RE-STORE "MAIN" RE-STORE "D7","DML"
SPECIAL FUNCTION KEYS	STOREKEY file specifier	STOREKEY "KEYS"
MACHINE MEMORY	STOREALL file specifier	STOREALL "Backup"
BINARY PROGRAM	STOREBIN file specifier	STOREBIN "FBIN:T"

"NON-DATA FILES", Ch.5, MST

MS16

Protect code doesn't really do anything
— allows me to edit file in memory
but version on tape remains O.K.

SAVE AND RE-SAVE STATEMENTS

PURPOSE:

TO SAVE OR RE-SAVE IN SOURCE FORM IN THE SPECIFIED FILE ALL OR PART OF A PROGRAM.

SYNTAX:

```
SAVE file specifier [, first-line identifier
     [, final-line identifier]]
RE-SAVE file specifier [, protect code] [, first-
     line identifier [, final-line identifier]]
```

EXAMPLES:

```
SAVE "UPDATE"
RE-SAVE "UPDATE"
SAVE "DELETE",320
SAVE "INSERT",Insert,End_insert
RE-SAVE "SORT","NOWRIT"
```

"STORING PROGRAMS", Ch.5, MST — MS17

CONTRAST OF STORE AND SAVE

STORE	SAVE
STORES PROGRAM IN QUASI-COMPILED FORM ALONG WITH BINARY PROGRAMS AND SYMBOL TABLES IN PROGRAM-TYPE FILE.	STORES PROGRAM IN SOURCE FORM IN DATA-TYPE FILE.
USE WHENEVER POSSIBLE FOR ACCESS SPEED. WITH FLEXIBLE AND HARD DISKS, STORE MAY BE AS MUCH AS 5 TIMES FASTER THAN SAVE.	USE TO SAVE PART OF A PROGRAM OR TO SAVE A PROGRAM THAT WILL BE APPENDED TO ANOTHER PROGRAM.

'STORING PROGRAMS', Ch.5 & 'TIMINGS', Ap.C, MST MS18

Only LINK a saved program.
That is a saved program can only
be linked or ~~get~~ "get" "ted"

THE LOAD STATEMENTS

PURPOSE	SYNTAX	EXAMPLE
QUASI-COMPILED PROGRAM	LOAD file specifier [, execution-line identifier]	LOAD "MAIN" LOAD "MAIN:C12" LOAD "Update",130
SPECIAL FUNCTION KEYS	LOADKEY file specifier	LOADKEY "KEYS"
MACHINE MEMORY	LOADALL file specifier	LOADALL "Backup"
BINARY PROGRAM	LOADBIN file specifier	LOADBIN "FBIN:T"

"NON-DATA FILES", Ch.5, MST

MS19

GET STATEMENT

PURPOSE:

TO LOAD A PROGRAM FROM A DATA-TYPE FILE, OR
TO APPEND LINES TO THE RESIDENT PROGRAM
WHILE RETAINING THE VALUES OF COMMON
VARIABLES.

SYNTAX:

```
GET file specifier [, line identifier
    [, execution-line identifier]]
```

EXAMPLES:

```
GET "UPDATE"
GET "INSERT",End_main
GET "DELETE",End_main,Delete_part
```

'STORING PROGRAMS', Ch.5, MST MS20

LINK STATEMENT

PURPOSE:

TO LOAD A PROGRAM FROM A DATA-TYPE FILE, OR TO APPEND LINES TO THE RESIDENT PROGRAM WHILE RETAINING THE VALUES OF ALL VARIABLES.

SYNTAX:

```
LINK file specifier [, line identifier
     [, execution-line identifier]]
```

EXAMPLES:

```
LINK "4CAST"
LINK "PLOT",320
LINK "PLOT",Predict
LINK "PLOT",Predict,Plot_all
```

"STORING PROGRAMS", Ch.5, MST　　　　MS21

Note LINK retains the values of all variables while [illegible] retains only those variables in common

CONTRAST OF LOAD, GET, AND LINK

LOAD	GET	LINK
LOADS PROGRAM FROM PROGRAM-TYPE (STORED) FILE.	LOADS PROGRAM OR APPENDS PROGRAM LINES FROM DATA-TYPE (SAVED) FILE.	LOADS PROGRAM OR APPENDS PROGRAM LINES FROM DATA-TYPE (SAVED) FILE.
RETAINS VALUES OF COMMON VARIABLES.	RETAINS VALUES OF COMMON VARIABLES.	RETAINS VALUES OF ALL VARIABLES.
ACCESS SPEED IS FAST COMPARED TO GET AND LINK.	ACCESS SPEED IS SLOW COMPARED TO LOAD.	ACCESS SPEED IS SLOW COMPARED TO LOAD.

'STORING PROGRAMS', Ch.5 & 'TIMING', Ap.C, MST

MS22

CONTRAST OF ACCESS SPEEDS

	STORE/SAVE		LOAD/GET	
	MIN	MAX	MIN	MAX
HARD DISK	2	6	5	14
FLEXIBLE DISK	2	5	5	8
TAPE	1	1+	1+	2

ALL FIGURES ARE APPROXIMATE. FIGURES ARE BASED ON TRANSFER OF 5,000 TO 50,000 BYTES.

'TIMING', Ap.C, MST

MS23

PROGRAM STORAGE EXERCISE 1

THE PROGRAM SHOWN BELOW CONTAINS A MAIN PROGRAM, A SORT SUBROUTINE, AND AN EXCHANGE SUBROUTINE. THIS PROGRAM IS LOCATED IN THE PROGRAM-TYPE FILE NAMED "SORT". REORGANIZE THIS PROGRAM SO THAT THE EXCHANGE SUBROUTINE APPEARS BEFORE THE SORT SUBROUTINE. STORE THE REORGANIZED VERSION IN A PROGRAM-TYPE FILE CALLED "SORTX".

HINT: USEFUL STATEMENTS ARE LOAD, SAVE, GET, AND STORE.

```
10   !  MAIN PROGRAM
20   !  THIS PROGRAM GENERATES AND SORTS 20 RANDOM NUMBERS
30   OPTION BASE 1
40   DIM Numbers(20)
50   FOR I=1 TO 20
60   Numbers(I)=INT(90*RND)+10
70   NEXT I
80   CALL Sort(Numbers(*),20)
90   PRINT Numbers(*);
100  END
110  !
120  !  SORT SUBROUTINE
130  !  Sort SORTS IN ASCENDING ORDER THE ELEMENTS FROM 1 TO N IN
140  !  THE ONE-DIMENSIONAL ARRAY A.  OPTION BASE 1 IS ASSUMED.
150  !  SUB Sort(A(*),N)
160  OPTION BASE 1
170  FOR I=1 TO N-1
180  FOR J=I+1 TO N
190  IF A(I)>A(J) THEN CALL Exchange (A(*),I,J)
200  NEXT J
210  NEXT I
220  SUBEND
230  !
240  !  EXCHANGE SUBROUTINE
250  !  Exchange SWITCHES Z(P) AND Z(Q)
260  SUB Exchange(Z(*),P,Q)
270  OPTION BASE 1
280  Temp=Z(P)
290  Z(P)=Z(Q)
300  Z(Q)=Temp
310  SUBEND
```

PROGRAM STORAGE EXERCISE 2

DEFINE KEY 0 TO SELECT THE INTERNAL PRINTER FOR THE PRINTING DEVICE. DEFINE KEY 1 TO SELECT THE CRT FOR THE PRINTING DEVICE. RETAIN THE DEFAULT KEY DEFINITIONS. STORE THE KEY DEFINITIONS IN A FILE CALLED "KEYS". YOU MAY CHOOSE TO LOAD THESE KEY DEFINITIONS AT SOME LATER TIME.

PROGRAM STORAGE EXERCISE 3

THE FOUR PROGRAMS LISTED BELOW ARE STORED IN THE FILES NAMED "FILEA", "FILEB", "FILEC", AND "FILED", RESPECTIVELY. "FILEA" AND "FILEB" ARE PROGRAM-TYPE (STORE/LOAD) FILES AND "FILEC" AND "FILED" ARE DATE-TYPE (SAVE/GET/LINK) FILES. PREDICT WHAT THE PRINTED OUTPUT WILL BE IF "FILEA" IS LOADED AND EXECUTED. TRY IT TO CHECK YOUR RESULTS.

```
10    ! "FILEA"
20    !
30   Start: COM S
40    PRINT  LIN(3),"START1","S =";S,"T =";T
50    S=T=10
60    PRINT "START2","S =";S,"T =";T,LIN(1)
70    LOAD "FILEB",Go
80    !
90    !
100   !
110   ! "FILEB"
120   !
130  Go: COM S
140   PRINT "LOAD 1","S =";S,"T =";T
150   S=T=9
160   PRINT "LOAD 2","S =";S,"T =";T,LIN(1)
170   LINK "FILEC",Cont,Cont
180  Cont: !
190   !
200   !
210   !
220   ! "FILEC"
230   !
240  Cont: !
250   PRINT "LINK 1","S =";S,"T =";T,LIN(1)
260   GET "FILED",Cont,Cont
270   !
280   !
290   !
300   ! "FILED"
310   1
320  Cont:!
330   PRINT "GET 1","S =";S,"T =";T
340   END
```

GENERAL PURPOSE FILE STATEMENTS

PROTECT STATEMENT

CAT STATEMENT

PURGE STATEMENT

COPY STATEMENT

RENAME STATEMENT

MS24

FILE PROTECT CODE

PURPOSE:

TO HELP PREVENT THE ACCIDENTAL ERASURE OR REWRITING OF A FILE.

FORM:

protect code

RULES:

STRING EXPRESSION, AT LEAST ONE CHARACTER LONG.
UP TO SIX CHARACTERS ARE RECOGNIZED.
NO BLANKS, NO QUOTE MARKS.

EXAMPLES:

```
"WRITE"
"***"&Initials$[1,3]
"BeCareful"
```

"SPECIAL OPERATIONS", Ch.3, MST MS25

PROTECT STATEMENT

PURPOSE:

TO ASSIGN A PROTECT CODE TO A FILE TO HELP PREVENT ACCIDENTAL REWRITING OR ERASING OF THE CONTENT.

SYNTAX:

```
PROTECT file specifier, protect code
```

EXAMPLES:

```
PROTECT "Chess","Bob"

PROTECT "INVEN","WRITE"

PROTECT File_name$,Write_access$
```

"PROTECTING A FILE", Ch.3, MST

MS26

DIRECTORIES

maximum no. of entries = 43

DEFINITION:

A DIRECTORY IS A TABLE OF INFORMATION STORED ON AN INITIALIZED MEDIUM THAT CONTAINS DATA ABOUT EACH FILE ON THAT MEDIUM. THE DATA INCLUDES THE FILE NAME, PROTECTION, TYPE, LENGTH, AND LOCATION, IF ANY.

EXAMPLE:

NAME	PRO	TYPE	REC/FILE	BYTES/REC	ADDRESS
T15		2			
KEYS		KEYS	1	256	5
FLOP2		BPRG	49	256	6
SLIDES		PROG	116	256	55
NOTES		DATA	100	256	171
MEMORY	*	ALL	171	256	271

"FILES" AND "FILE DIRECTORY", Ch.2, MST

MS27

NAME PRO TYPE REC/FILE BYTES/REC

NAME	PRO	TYPE	REC/FILE	BYTES/REC	ADDRESS
T15		2			
KEYS		KEYS	1	256	5
FLOP2		BPRG	49	256	6
SLIDES		PROG	116	256	55
NOTES		DATA	100	256	171
MEMORY		ALL	171	256	271

CAT STATEMENT

PURPOSE:

TO PRINT THE DIRECTORY OF A MEDIUM.

SYNTAX:

```
        select code
          |    selective catalog specifier
          |     |     msus   heading suppression
          |     |      |        |
CAT # 0; "DATA:T15", 1
```

file name not type

EXAMPLES:

```
PRINTER IS 16
MASS STORAGE IS ":T15"
CAT
CAT #0
CAT ":F8"
```

```
9845B ONLY:
 DIM A$(42)[41]
 CAT TO A$(*)
```

MS28

PURGE STATEMENT

PURPOSE:

TO REMOVE A FILE FROM A MEDIUM.

SYNTAX:

PURGE file specifier [, protect code]

EXAMPLES:

UNPROTECTED FILES:

```
MASS STORAGE IS ":T15"
PURGE "TEST"
PURGE "ORDER7:C12"
```

PROTECTED FILES:

```
PURGE "DATA","WRITE"
```

"PURGING FILES", Ch.3, MST

MS29

COPY STATEMENT

PURPOSE:

TO BACKUP A CRITICAL FILE.
TO TRANSFER A FILE FROM ONE MEDIUM TO ANOTHER MEDIUM.

SYNTAX:

COPY source-file specifier TO destination-file specifier [, source-file protect code]

EXAMPLES:

UNPROTECTED FILES:

```
COPY "FBIN:T14" TO "FBIN:T15"
```

PROTECTED FILES:

```
COPY "SLIDES" TO "BACKUP:F8","WRITE"
```

"COPYING FILES", Ch.3, MST MS30

RENAME STATEMENT

PURPOSE:

TO CHANGE THE NAME OF A FILE.

SYNTAX:

lost

RENAME old-file specifier TO new-file name [, protect code]

EXAMPLES:

UNPROTECTED FILES:

```
RENAME "DATA:F8" TO "DATA1"
```

PROTECTED FILES:

```
RENAME "DATA:F8" TO "DATA1",Password$
```

"RENAMING FILES", Ch.3, MST

MS31

GENERAL FILES EXERCISE 1

SELECT A FILE ON THE BASICS (DAYS 1 AND 2) TAPE CARTRIDGE AND COPY IT TO ANOTHER TAPE CARTRIDGE. SELECT A SECOND FILE ON THE BASICS CARTRIDGE, PROTECT IT, AND COPY IT TO THE OTHER TAPE CARTRIDGE.

GENERAL FILES EXERCISE 2

SELECTIVELY CATALOG THE MASS STORAGE/GRAPHICS TAPE CARTRIDGE SO THAT FILES THAT START WITH THE SAME LETTER ARE ADJACENT. SEND THE OUTPUT TO THE CRT (#16) OR THE INTERNAL PRINTER (#Ø) BASED ON AN INPUT FROM THE KEYBOARD.

HINT: ALL THE FILES BEGIN WITH AN UPPER CASE LETTER. USE THE CHR$ STATEMENT WITH THE ASCII CODES FOR UPPER CASE LETTERS (DECIMAL 65 TO 9Ø) TO GENERATE THE SELECTIVE CATALOG SPECIFIER. USE HEADING SUPPRESSION.

FUNDAMENTALS OF DATA STORAGE

CREATE STATEMENT

ASSIGN STATEMENT

PRINT # STATEMENT

END DATA TYPE

READ # STATEMENT

MS32

DATA STORAGE TOPICS

- → CREATING DATA FILES
- OPENING AND CLOSING DATA FILES
- SERIAL ACCESS
- RANDOM ACCESS
- COMBINING SERIAL AND RANDOM ACCESS

MS33

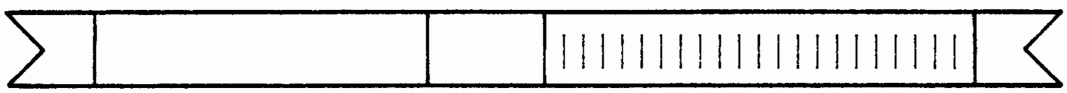
FILES AND RECORDS

FACT:

TO THE USER, A FILE OF ANY TYPE EXCEPT DATA-TYPE IS A SINGLE, UNDIFFERENTIATED BLOCK OF INFORMATION.

A DATA-TYPE FILE MAY BE TREATED AS A CONSECUTIVE SET OF SMALLER ITEMS CALLED "RECORDS."

EXAMPLE:



- STORE "SORT"
- PROGRAM FILE
- THE PROGRAM IS THE SMALLEST ACCESSIBLE UNIT.

- SAVE "SORT"
- DATA FILE
- INDIVIDUAL PROGRAM LINES ARE ACCESSIBLE.
- A PROGRAM LINE FORMS ONE "LOGICAL RECORD."

MS34

RECORDS

RECORD:
THE SMALLEST ADDRESSABLE UNIT OF STORAGE;
ACCESSIBLE TO USER IN DATA-TYPE FILES ONLY;
THREE TYPES: LOGICAL, DEFINED, AND PHYSICAL;
CONSECUTIVE GROUP OF RECORDS FORMS A FILE.

RECORD I/O:
A GENERAL TERM WHICH REFERS TO WRITING AND READING RECORDS.

END-OF-RECORD (EOR):
A SPECIAL CHARACTER THAT MARKS THE END OF A RECORD.

END-OF-FILE (EOF):
A SPECIAL CHARACTER THAT MARKS THE END OF A FILE.

"RECORDS" & "RECORD I/O", Ch.2, MST MS35

RECORDS

2 files maybe

LOGICAL RECORD:

SET OF DATA-ITEMS THAT ARE CONCEPTUALLY GROUPED.

DEFINED RECORD:

SET OF ADJACENT BYTES DEFINED AND ACCESSED BY USER.

PHYSICAL RECORD:

SET OF 256 ADJACENT BYTES ACCESSED BY COMPUTER.

(unless defined otherwise)

'RECORDS', Ch.2, MST MS36

RECORDS - EXAMPLE

DATA:
MAILING LIST OF
5 CUSTOMER NAMES
AND ADDRESSES.

LOGICAL RECORD:
ONE CUSTOMER NAME
AND ADDRESS.

DEFINED RECORD:
192 BYTES - MAX
LENGTH OF DATA
FOR ONE CUSTOMER.

≤

PHYSICAL RECORD:
256 BYTES.

```
DATA               |XXXX   XXX    XXX    XXXX   XXXXX   |
LOGICAL RECORDS    |LLLL   LLL    LLL    LLLL   LLLLL   |
DEFINED RECORDS    |DDDDD|DDDDD|DDDDD|DDDDD|DDDDD|      |
PHYSICAL RECORDS   |PPPPPPP|PPPPPPP|PPPPPPP|PPPPPPP|
```

MS37

ESTIMATING DATA STORAGE REQUIREMENTS

FULL PRECISION NUMBER	8 BYTES
SHORT PRECISION NUMBER	4 BYTES
INTEGER	4 BYTES
STRING	1 BYTE PER CHARACTER + 1 BYTE IF THE STRING LENGTH IS ODD + 4 BYTES FOR EACH DEFINED RECORD IN WHICH THE STRING RESIDES

"RECORDS", Ch.2, MST

MS38

CREATE STATEMENT

PURPOSE:

TO RESERVE, ORGANIZE, AND INITIALIZE STORAGE SPACE FOR A DATA FILE BY ENTERING IT IN THE DIRECTORY AND BY MARKING AN EOF AT THE START OF EACH DEFINED RECORD

SYNTAX:

```
CREATE file specifier, number of defined
       records [, defined-record length]
```

EXAMPLES:

```
CREATE "Names",200
CREATE "STOCK",6,30
CREATE "Cards:F",Ncards,80
```

RECOMMENDATION:

DEFAULT default-record length FOR SPEED.

"CREATING FILES", Ch.2, MST MS39

file specifier = <name><msus>

DATA STORAGE TOPICS

CREATING DATA FILES

→ OPENING AND CLOSING DATA FILES

SERIAL ACCESS

RANDOM ACCESS

COMBINING SERIAL AND RANDOM ACCESS

MS40

FILE NUMBERS AND THE FILE TABLE

FILE NUMBER:

A FILE NUMBER MUST BE ASSIGNED TO A FILE BEFORE ITS RECORDS CAN BE PRINTED OR READ. THE ASSIGN STATEMENT ASSOCIATES A FILE NUMBER TO A FILE SPECIFIER. THE FILE NUMBER HAS AN INTEGER VALUE BETWEEN 1 AND 10.

FILE TABLE:

THE FILE TABLE STORES INFORMATION ABOUT EACH FILE AVAILABLE FOR RECORD I/O. THE FILES ARE LISTED IN THE FILE TABLE BY FILE NUMBER. THE FILE TABLE IS STORED IN THE MAIN MEMORY OF THE COMPUTER.

FILE #	FILE NAME	FILE DESCRIPTION
1	STOCK	
. . .	. . .	. . .
10	INVEN	

"RECORD I/O", Ch.2, MST MS41

ASSIGN STATEMENT

PURPOSE:

- TO OPEN A DATA FILE FOR RECORD I/O, I.E.,
  - *TO ASSIGN A FILE NUMBER
  - *TO TRANSFER INFORMATION ABOUT THE FILE FROM THE DIRECTORY ON THE MEDIUM TO THE FILE TABLE
  - *TO CHECK FILE PROTECTION
- TO CLOSE A DATA FILE FOR RECORD I/O

SYNTAX:

```
ASSIGN file specifier TO # file number [, return
        variable [, protect code]]
ASSIGN # file number TO file specifier [, return
        variable [, protect code]]
```

EXAMPLES:

```
ASSIGN "Vendor:F" TO #10
ASSIGN #10 TO "Vendor:F", Error, Write_access$
ASSIGN * TO #10
```

"RECORD I/O", Ch.2, MST MS42

return variable : can interrogate its value

instead of generating errors on screen

0 – file exists & can be accessed

protected

missing table

FILE NUMBERS AND SUBPROGRAMS

FACT:

FILE NUMBERS MAY BE PARAMETERS IN SUBROUTINE SUBPROGRAMS.

EXAMPLE PROGRAMS:

```
ASSIGN #8 TO "DATA3"
CALL Sort(#8)
    . . .
END
SUB Sort(#1)
    . . .
ASSIGN #1 TO *
SUBEND
```

Loses the name it crosses the boundary

```
CALL Trend(#9)
    . . .
END
SUB Trend(#2)
ASSIGN #2 TO "TEMP"
    . . .
SUBEND
```

"SUBROUTINE SUBPROGRAMS", Ch.7, O&P

MS43

DATA STORAGE TOPICS

CREATING DATA FILES

OPENING AND CLOSING DATA FILES

→ SERIAL ACCESS

RANDOM ACCESS

COMBINING SERIAL AND RANDOM ACCESS

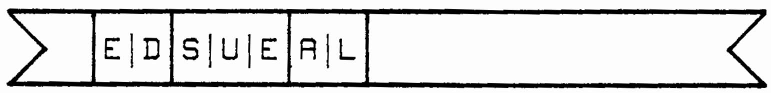
MS44

SERIAL AND RANDOM DATA STORAGE

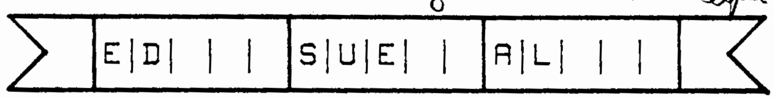
DATA:

"ED"
"SUE"
"AL"

SERIAL CONCEPT: NB most compact way.



RANDOM CONCEPT: allows us to access any record (must be of equal length)



"SERIAL ACCESS" & "RANDOM ACCESS", Ch.2, MST

MS45

SERIAL PRINTING

no. of records.

```
CREATE "NAMES",2        EOF @ start of each def. of records in a record
    |EΦF| | | | | | | | | | | | | | | | | | | | | | |      256 bytes
     2 bytes                                                 PEOF
ASSIGN #1 TO "NAMES"
    |EΦF| | | | | | | | | | | | | | | | | | | | | | |
      ↑ FILE POINTER
                         2 bytes
PRINT #1;"ED"
    |X|X|X|X|E|D|EΦR| | | | | | | | | | | | | | | |
4 bytes overhead            logical end of record
PRINT #1;"SUE","AL"
    |X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EΦR| |
                          string odd
```

"SERIAL ACCESS", Ch.2, MST — MS46

Once computer comes to first to an EOF which is not over-written then it knows where to close the file

SERIAL PRINTING

now close the file

PRINT #1;END

|X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EØF| |

ANOTHER METHOD

CREATE "NAMES",2

|EØF| | | | | | | | | | | | | | | | | | | | | | |

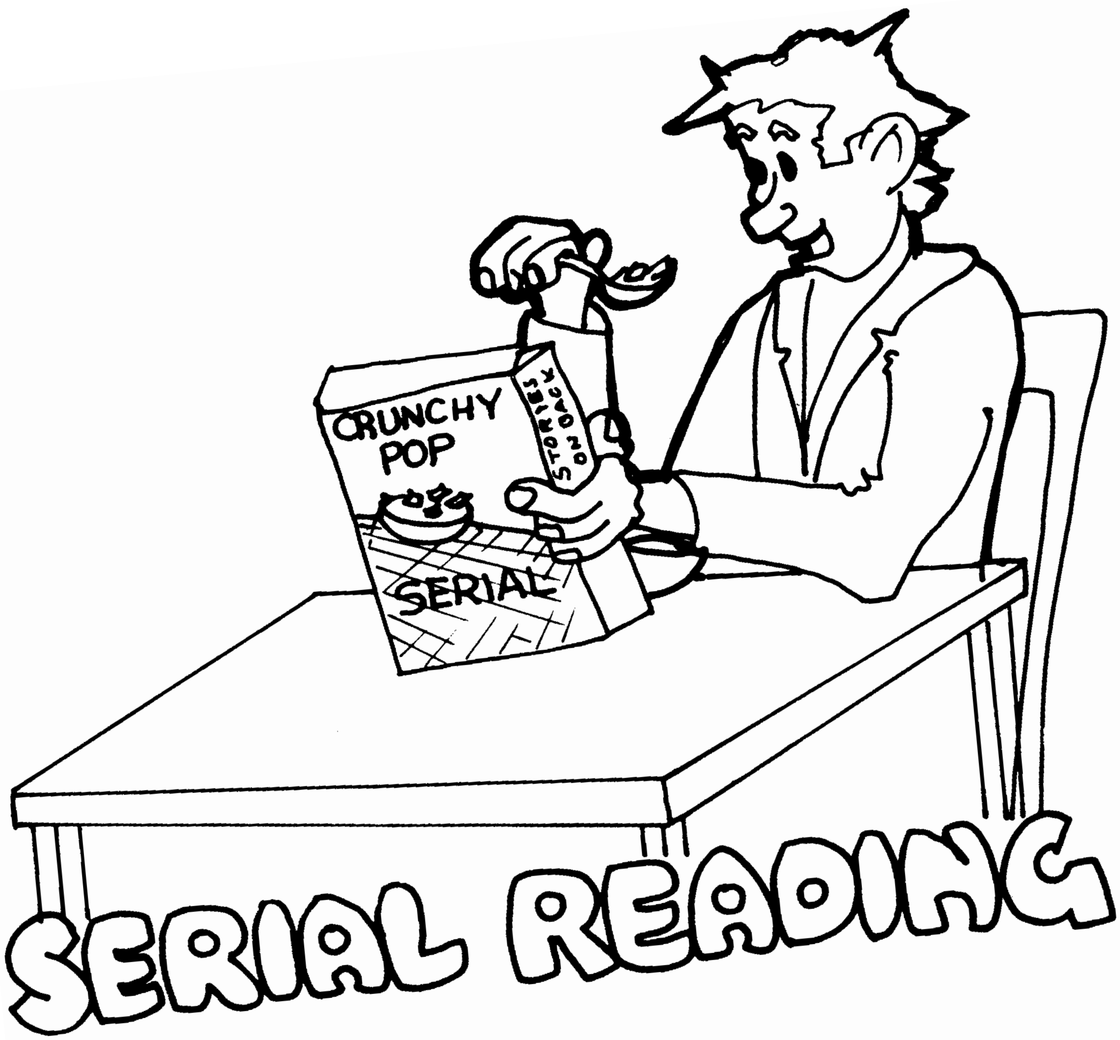
ASSIGN #1 TO "NAMES"

|EØF| | | | | | | | | | | | | | | | | | | | | | |

PRINT #1;"ED","SUE","AL",END

|X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EØF| |

'SERIAL ACCESS', Ch.2, MST MS47


CRUNCHY POP
STORIES ON BACK
SERIAL
SERIAL READING

SERIAL READING

```
ASSIGN #1 TO "NAMES"
    |X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EOF| |
     ↑

READ #1;Name$(1)
    |X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EOF| |
     ↑ ─────────→↑
```

know there is an EOR here but not transferred to vc

```
READ #1;Name$(2),Name$(3)
    |X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EOF| |
                 ↑ ──────────────────────────→↑

PRINT Name$(*)
ED
SUE
AL
```

'SERIAL ACCESS', Ch.2, MST

MS48

SERIAL READING

ANOTHER METHOD

```
ASSIGN #1 TO "NAMES"
    |X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EOF|  |
     ↑

OPTION BASE 1
DIM Name$(3)
READ #1;Name$(*)
    |X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EOF|  |
     ↑―――――――――――――――――――――――――――――――――――――――→↑

READ #1;Any$
ERROR 59 IN LINE 50
```

(tries to read pass the EOF mark)

'SERIAL ACCESS', Ch.2, MST MS49

UPDATING SERIAL FILES

CASE 1: REPLACE OLD ITEM WITH LONGER NEW ITEM.

```
ASSIGN #1 TO "NAMES"
READ #1;Dummy$
    |X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EOF| | | |

PRINT #1;"CAROL"
    |X|X|X|X|E|D|X|X|X|X|C|A|R|O|L|X|EOR|A|L|EOF| | | |

PRINT #1;"AL",END
    |X|X|X|X|E|D|X|X|X|X|C|A|R|O|L|X|X|X|X|X|A|L|EOF| |
```

"SERIAL ACCESS", Ch.2, MST MS50

UPDATING SERIAL FILES

CASE 2: REPLACE OLD ITEM WITH NEW ITEM OF EQUAL LENGTH.


```
ASSIGN #1 TO "NAMES"
READ #1;Dummy$
    |X|X|X|X|E|D|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EOF| |

PRINT #1;"BOB"
    |X|X|X|X|E|D|X|X|X|X|B|O|B|X|EOR|X|X|A|L|EOF| |

PRINT #1;"AL",END
    |X|X|X|X|E|D|X|X|X|X|B|O|B|X|X|X|X|X|A|L|EOF| |
```

MS51

UPDATING SERIAL FILES

CASE 3: REPLACE OLD ITEM WITH SHORTER NEW ITEM.

```
ASSIGN #1 TO "NAMES"
READ #1;Dummy$
   |X|X|X|X|E|D|X|X|X|X|C|A|R|O|L|X|X|X|X|X|A|L|EOF| |

PRINT #1;"JO"
   |X|X|X|X|E|D|X|X|X|X|J|O|EOR|L|X|X|X|X|X|A|L| | | |

PRINT #1;"AL",END
   |X|X|X|X|E|D|X|X|X|X|J|O|X|X|X|X|A|L|EOF|A|L|EOF| |
```

MS52

SERIAL PRINT# STATEMENT

SYNTAX:

```
[MAT]PRINT # file number [; data list [,END]]
[MAT]PRINT # file number [; END]
```

RULE:

```
IF END THEN DATA TERMINATES WITH EOF
       ELSE DATA TERMINATES WITH EOR.
```

EXAMPLES:

```
FOR N=1 TO Noparts
PRINT #1;Name$(N),Quantity(N),Price(N)
NEXT N
PRINT #1; END

PRINT #10;Fed_tax(*),State_tax(*),END
MAT PRINT #10; Fed_tax,State_tax,END
```

"WRITING RECORDS", Ch.2, MST MS53

SERIAL READ# STATEMENT

SYNTAX:

```
[MAT]READ # file number ; variable list
```

RULES:

IF EOR THEN SKIP TO NEXT DEFINED RECORD.
IF EOF OR DATA LIST FULL THEN DONE.

EXAMPLES:

```
FOR N=1 TO Noparts
READ #1;Name$(N),Quantity(N),Price(N)
NEXT N

READ #10;Fed_tax(*),State_tax(*)
MAT READ #10;Fed_tax,State_tax
```

MS54

END DATA TYPE

places an EOR at the end of the last logical record.

PURPOSE:

TO MARK THE END OF VALID DATA IN THE FILE.

EXAMPLE PROGRAM:

```
10    CREATE "TEST",2,32
20    ASSIGN #10 TO "TEST"
30    PRINT #10;1,2,3,4,5,6,7,8
40    ASSIGN #10 TO "TEST"
50    PRINT #10;9,10,END
60    ASSIGN #10 TO "TEST"
70    ON ERROR GOTO 90
80    READ #10;A,B,C,D
90    PRINT A,B,C,D
100   END
```

"USER-CONTROLLED END-OF-FILE", Ch.2, MST   MS55

```
10    CREATE "TEST",2,32
20    ASSIGN #10 TO "TEST"
30    PRINT #10;1,2,3,4,5,6,7,8
40    ASSIGN #10 TO "TEST"
50    PRINT #10;9,10
60    ASSIGN #10 TO "TEST"
70    ON ERROR GOTO 90
80    READ #10;A,B,C,D
90    PRINT A,B,C,D
100   END
```

← sets pointer at beginning of file

9   10   5   6

```
10    CREATE "TEST",2,32
20    ASSIGN #10 to "TEST"
30    PRINT #10;1,2,3,4,5,6,7,8
40    ASSIGN #10 TO "TEST"
50    PRINT #10;9,10,END
60    ASSIGN #10 to "TEST"
70    ON ERROR GOTO 90
80    READ #10;A,B,C,D
90    PRINT A,B,C,D
100   END
```

9   10   0   0

DATA STORAGE TOPICS

CREATING DATA FILES

OPENING AND CLOSING DATA FILES

SERIAL ACCESS

→ RANDOM ACCESS

COMBINING SERIAL AND RANDOM ACCESS

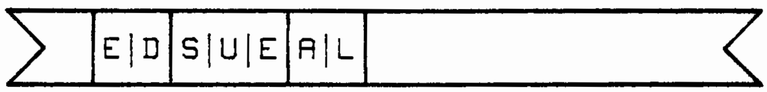
MS56

SERIAL AND RANDOM DATA STORAGE

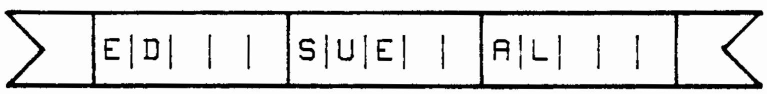
DATA:

"ED"
"SUE"
"AL"

SERIAL CONCEPT:



RANDOM CONCEPT:



MS57

RANDOM PRINTING

```
CREATE "NAMES",64,8
ASSIGN #1 TO "NAMES"
 |EØF| | | | | | |EØF| | | | | | |EØF| | | | | | |EØF|
  1               2               3               4
```

file pointer

```
PRINT #1,1;"ED"
 |X|X|X|X|E|D|EØR|EØF| | | | | | |EØF| | | | | | |EØF|
  1               2               3               4
```

file pointer now here

```
PRINT #1,2;"SUE"
PRINT #1,3;"AL"
 |X|X|X|X|E|D|EØR|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EØR|EØF|
  1               2               3               4
```

only 8 bytes long

pad

"RANDOM ACCESS", Ch.2, MST MS58

If not specifying an end on causes an EOR to be printed.

RANDOM READING

```
READ #1,3;Name$
 |X|X|X|X|E|D|EØR|X|X|X|X|S|U|E|X|X|X|X|X|A|L|EØR|EØF|
  1              2              3              4

PRINT Name$
AL
```

UPDATING RANDOM FILES

```
PRINT #1,3;"GAIL"
 |X|X|X|X|E|D|EØR|X|X|X|X|S|U|E|X|X|X|X|X|G|A|I|L|EØF|
  1              2              3              4
```

"RANDOM ACCESS", Ch.2, MST MS59

RANDOM PRINT# STATEMENT

SYNTAX:

```
[MAT]PRINT # file number, defined-record
            number [; data list [, END]]
[MAT]PRINT # file number, defined-record
            number [; END]
```

RULE:

```
IF END THEN DATA TERMINATES WITH EOF
       ELSE DATA TERMINATES WITH EOR.
```

EXAMPLES:

```
FOR N=1 TO No_employees
PRINT #1,N;Name$,Hours,Rate
NEXT N
PRINT #1,N;END

PRINT #1,3;Weights(*)
MAT PRINT #1,3;Weights
```

"WRITING RECORDS", Ch.2, MST

MS60

RANDOM READ# STATEMENT

SYNTAX:

```
[MAT]READ # file number, defined-record
            number [; variable list]
```

RULE:

READ TO EOR, EOF, OR DATA LIST FILLED.

EXAMPLES:

```
FOR N=1 TO No_employees
READ #1,N;Name$,Rate,Hours
NEXT N

READ #1,3;Weights(*)
MAT READ #1,3;Weights
```

"READING RECORDS", Ch.2, MST　　　　MS61

DATA STORAGE TOPICS

CREATING DATA TYPE FILES

OPENING AND CLOSING DATA FILES

SERIAL ACCESS

RANDOM ACCESS

→ SERIAL AND RANDOM ACCESS

MS62

COMBINING SERIAL AND RANDOM

```
CREATE "NAMES",64,16
ASSIGN #1 TO "NAMES"
|EØF| | | | | | | | | | | | | | | |EØF| | | | | | | | |
 1                                2
```

```
PRINT #1,1;"ED","M"
PRINT #1,2;"AL","W"
```

random address serial print

```
|X|X|X|X|E|D|X|X|X|X|M|X|EØR| | |X|X|X|X|A|L|X|X|X|X|
 1                                2
```

```
READ #1,1;First$,Mi$
```

random address serial read

```
|X|X|X|X|E|D|X|X|X|X|M|X|EØR| | |X|X|X|X|A|L|X|X|X|X|
 1                                2
```

"USING SERIAL AND RANDOM ACCESS TOGETHER", Ch.2, MST MS63

COMBINING SERIAL AND RANDOM UPDATING

```
READ #1,1;First$
|X|X|X|X|E|D|X|X|X|X|M|X|EOR|  |X|X|X|X|A|L|X|X|X|X|
 1                               2

PRINT #1;"J"
|X|X|X|X|E|D|X|X|X|X|J|X|EOR|  |X|X|X|X|A|L|X|X|X|X|
 1                               2
```

"USING SERIAL AND RANDOM ACCESS TOGETHER", Ch.2, MST MS64

COMBINING SERIAL AND RANDOM
SERIALLY READING A RANDOM FILE

```
READ #1,1
|X|X|X|X|E|D|X|X|X|X|M|X|EOR| | |X|X|X|X|A|L|X|X|X|X|
 1                                2
```

Can serially read a random file

```
READ #1;A$,B$,C$,D$
|X|X|X|X|E|D|X|X|X|X|M|X|EOR| | |X|X|X|X|A|L|X|X|X|X|
 1                                2
                              SKIP
```

when a serial read encounters an EOR it skips to the next available record

```
DIM Names$(2,2)
READ #1,1
READ #1;Names$(*)
PRINT USING "2(K,X,K,5X)";Names$(*)
ED M     AL W
```

MS65

Can't read a serial file randomly.

may treat a random file as a pseudo-random file

CONTRAST OF SERIAL AND RANDOM

SERIAL

- SYNTAX: PRINT #5;X
  READ #5;X
- DATA LISTS ARE STRUNG TOGETHER WITH NO MARKS SEPARATING THEM.
- WRITING AND READING OPERATIONS MAY NOT BE FREELY INTERMIXED. ITEMS ARE PRINTED SEQUENTIALLY AND READ SEQUENTIALLY FROM THE START OF THE FILE.

RANDOM

- SYNTAX: PRINT #5,3;X
  READ #5,3;X
- EACH DATA LIST STARTS AT THE BEGINNING OF A DEFINED RECORD.
- WRITING AND READING OPERATIONS MAY BE FREELY INTERMIXED.

"SERIAL ACCESS" & "RANDOM ACCESS", Ch.2, MST MS66

CONTRAST OF SERIAL AND RANDOM

SERIAL	RANDOM
- OFTEN DIFFICULT TO - UPDATE.	- EASY TO UPDATE.
- ITEMS DIFFERING IN LENGTH OCCUPY DIFFERENT AMOUNTS OF SPACE.	- ITEMS DIFFERING IN LENGTH OCCUPY EQUAL AMOUNTS OF SPACE.
- COMPACT; MINIMUM OF WASTED SPACE.	- DEFINED RECORD SIZE MUST BE CAREFULLY SELECTED TO MINIMIZE WASTED SPACE.
- IDEAL FOR TAPE.	- IDEAL FOR SPEEDY ACCESS WITH DISK.

'SERIAL ACCESS' & 'RANDOM ACCESS', Ch.2, MST MS67

DATA BASICS EXERCISE 1

USE THE ASSIGN STATEMENT TO DETERMINE THE FOLLOWING:

1. DOES THE FILE EXIST ON THE MEDIUM?
2. IF SO, IS THE FILE A DATA-TYPE FILE?
3. IF SO, IS THE FILE PROTECTED?

INPUT THE FILE NAME FROM THE KEYBOARD. FOR OUTPUT, DISPLAY ONE OF THE FOLLOWING MESSAGES:

"FILE filename IS NONEXISTENT."
"FILE filename IS NOT A DATA-TYPE FILE."
"FILE filename IS AN UNPROTECTED FILE."
"FILE filename IS A PROTECTED FILE."

HINT: THE RETURN VARIABLE IN THE ASSIGN STATEMENT IS DEFINED AS FOLLOWS:

Ø FILE EXISTS AND IS AVAILABLE FOR USE.
1 FILE WAS NOT FOUND.
2 FILE IS NOT A DATE-TYPE FILE, OR
FILE IS PROTECTED AND NO PROTECT CODE WAS GIVEN, OR
FILE IS PROTECTED AND WRONG PROTECT CODE WAS GIVEN, OR
FILE IS NOT PROTECTED AND A PROTECT CODE WAS GIVEN.

DATA BASICS EXERCISE 2

TASK A

THE LOCATION AND NAME OF EACH OF THE 28 TEAMS IN THE NATIONAL FOOTBALL LEAGUE IS SERIALLY STORED IN THE DATA FILE NAMED "SERIAL". READ THE CONTENTS OF THE FILE AND PRINT IT ON THE CRT OR INTERNAL PRINTER.

TASK B

THE DATA FILE NAMED "RANDOM" CONTAINS THE LOCATION, CONFERENCE, AND DIVISION OF EACH OF THE 28 TEAMS IN THE NATIONAL FOOTBALL LEAGUE. EACH DEFINED RECORD CONTAINS THE DATA ABOUT ONE TEAM. READ THE CONTENTS OF THE FILE USING RANDOM READ# STATEMENTS AND PRINT THE DATA ON THE CRT OR INTERNAL PRINTER.

TASK C

REPEAT TASK B USING SERIAL READ# STATEMENTS RATHER THAN RANDOM READ#'S. READ UNTIL THE DATA LIST IS FILLED OR AN EOR OR EOF IS ENCOUNTERED. SERIAL READ#'S READ UNTIL THE DATA LIST IS FILLED OR AN EOF IS ENCOUNTERED. WHEN A SERIAL READ# ENCOUNTERS AN EOR, IT SKIPS TO THE START OF THE NEXT DEFINED RECORD AND THEN CONTINUES READING.

TASK D

YOU WILL NEED A FILE NAMED "NFC" THAT CONTAINS FOUR 256-BYTE DEFINED RECORDS. DEVELOP SOME CODE THAT CHECKS TO SEE IF THE FILE ALREADY EXISTS. IF NOT, CREATE THE FILE. HINT: USE THE RETURN VARIABLE IN THE ASSIGN STATEMENT TO DETERMINE WHETHER THE FILE ALREADY EXISTS.

TASK E

YOU WILL NEED A FILE NAMED "AFC" THAT CONTAINS SIXTEEN 64-BYTE DEFINED RECORDS. AS IN TASK D, DEVELOP SOME CODE THAT CHECKS TO SEE IF THE FILE ALREADY EXISTS. IF NOT, CREATE THE FILE.

TASK F

OPEN THE "NFC" FILE FOR RECORD I/O. FOR EACH OF THE FOURTEEN TEAMS IN THE NATIONAL FOOTBALL CONFERENCE, SERIALLY PRINT THE TEAM LOCATION, NAME, AND DIVISION. TO VERIFY THE DATA, READ AND LIST THE CONTENTS OF "NFC".

TASK G

OPEN THE "AFC" FILE FOR RECORD I/O. FOR EACH OF THE FOURTEEN TEAMS IN THE AMERICAN FOOTBALL CONFERENCE, RANDOMLY PRINT ITS LOCATION AND DIVISION IN ONE DEFINED RECORD. TO VERIFY THE DATA, READ AND LIST THE CONTENTS OF "AFC".

TASK H

DURING THE 1978 SEASON, THE EAST, CENTRAL, AND WEST DIVISIONS OF THE "NFC" HAVE 5, 4 AND 5 TEAMS, RESPECTIVELY. TO PRINT THE THREE DIVISIONS IN THREE COLUMNS ACROSS ONE PAGE, IT WOULD BE CONVENIENT TO HAVE THE SAME NUMBER OF TEAMS IN EACH DIVISION. INSERT DUMMY DATA CONSISTING OF THREE BLANK (" ") STRINGS BETWEEN THE CENTRAL AND WEST DIVISIONS IN THE "NFC" FILE. THEN, READ "NFC" AND PRINT THE TEAM LOCATIONS AND NAMES BY DIVISION IN THREE COLUMNS ACROSS THE PAGE.

TASK I

FOR EACH TEAM IN THE "AFC" FILE, INSERT THE TEAM NAME BETWEEN THE LOCATION AND DIVISION. TO VERIFY, READ AND PRINT THE CONTENTS OF "AFC". PRINT THE TEAM LOCATIONS AND NAMES BY DIVISION IN THREE COLUMNS ACROSS THE PAGE.

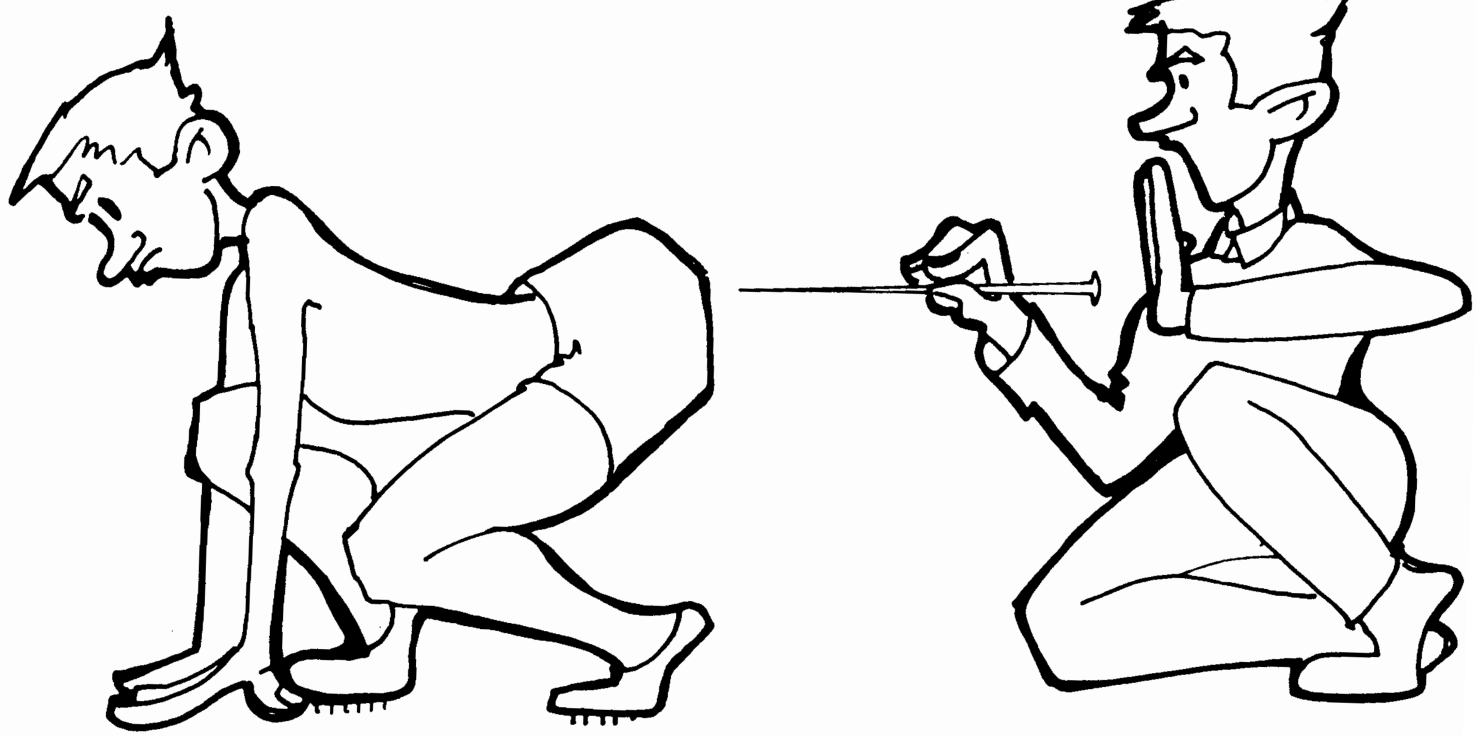
DATA BASICS EXERCISE 3

SUPPOSE YOU ARE DEBUGGING A PROGRAM THAT REQUIRES A DATA FILE CALLED "TEST" WHICH CONTAINS TEST DATA CONSISTING OF 64 FULL-PRECISION RANDOM NUMBERS FROM Ø TO 99. YOU WOULD LIKE TO HAVE THE FOLLOWING DEBUGGING CAPABILITIES BUILT INTO YOUR PROGRAM.

1. IF NO "TEST" FILE EXISTS, CREATE AND OPEN THE FILE AND WRITE 64 RANDOM NUMBERS IN IT.
2. IF THE "TEST" FILE EXISTS AND YOU WANT TO USE DIFFERENT TEST DATA, PURGE THE OLD FILE AND GENERATE A NEW ONE.
3. IF THE "TEST" FILE EXISTS AND YOU WANT TO USE THE DATA IN IT, DO NOTHING.

DEVELOP A PROGRAM TO PERFORM THE TASKS LISTED ABOVE. DECIDE WHETHER TO REPLACE THE "TEST" FILE BASED ON AN INPUT FROM THE KEYBOARD. DO NOT USE THE CAT (CATALOG) STATEMENT.

HOW TO IMPROVE



DATA ACCESS SPEED

HOW TO IMPROVE DATA ACCESS SPEED

- BUFFER STATEMENT
- FCREATE STATEMENT
- FPRINT STATEMENT
- FREAD STATEMENT

MS68

DEVICE BUFFERS

DEFINITION:

A DEVICE BUFFER IS A 256-BYTE BUFFER AUTOMATICALLY ASSIGNED TO A SELECT CODE AND USED IN I/O OPERATIONS WITH ALL DEVICES ON THAT SELECT CODE.

BENEFITS:

SHORTER EXECUTION TIME; LESS WEAR ON MEDIUM

CAUTION:

DATA IS NOT TRANSFERRED IMMEDIATELY TO THE MEDIUM.

BAD PRACTICE

```
FOR N=1 TO Max
READ #1;Rate,Hours
PRINT #2;Rate*Hours
NEXT N
```

BETTER PRACTICE

```
FOR N=1 TO Max
READ #1;Rate(N),Hours(N)
NEXT N
FOR N=1 TO Max
PRINT #2;Rate(N)*Hours(N)
NEXT N
```

"DEVICE BUFFERING", Ch.4, MST

MS69

FILE BUFFERS

DEFINITION:

A FILE BUFFER IS A 256-BYTE BUFFER ASSIGNED TO A FILE NUMBER WITH THE BUFFER # STATEMENT.

BENEFIT:

FILE BUFFERS OPERATE SIMILARLY TO DEVICE BUFFERS BUT WITHOUT THE LIMITATION THAT THEY BE FLUSHED WHEN OTHER FILES ON THE SAME SELECT CODE ARE ACCESSED.

BAD PRACTICE	BETTER PRACTICE
	BUFFER #1
	BUFFER #2
FOR N=1 TO Max	FOR N=1 TO Max
READ #1;Rate,Hours	READ #1;Rate,Hours
PRINT #2;Rate*Hours	PRINT #2;Rate*Hours
NEXT N	NEXT N

"ADDITIONAL BUFFERING", Ch.4, MST

MS70

good practice is to read 256 bytes
the print the 256 bytes of data

BUFFER # STATEMENT

PURPOSE:

TO IMPROVE SPEED WHEN ACCESSING SEVERAL FILES ON ONE SELECT CODE.

SYNTAX:

```
BUFFER # file number
```

EXAMPLE PROGRAM:

```
ASSIGN #1 TO "PAYROL"
ASSIGN #2 TO "WAGES"
BUFFER #1
BUFFER #2
```

"ADDITIONAL BUFFERING", Ch.4, MST

MS71

BUFFERS AND SUBPROGRAMS

EXAMPLE PROGRAMS:

```
ASSIGN #8 TO "DATA3"
BUFFER #8
CALL Sort(#8)
END

SUB Sort(#1)
    . . .
SUBEND
```

#1 is now Buffered.

```
ASSIGN #8 TO "DATA3"
CALL Sort(#8)
END

SUB Sort(#1)
BUFFER #1
. . .
SUBEND
```

#8 is now buffered

"SUBROUTINE SUBPROGRAMS", Ch.7, O&P

MS72

USING ARRAYS AS BUFFERS

FACT:

PRINTING OR READING ARRAYS AS A UNIT RATHER THAN AS INDIVIDUAL ELEMENTS MAXIMIZES THE EFFECTIVENESS OF DEVICE BUFFERS AND ELIMINATES THE NEED FOR FILE BUFFERS.

BENEFIT:

ARRAY SIZE CAN EXCEED 256 BYTES.

BETTER PRACTICE

```
BUFFER #1
BUFFER #2
FOR N=1 TO Max
READ #1;Rate,Hours
PRINT #2;Rate*Hours
NEXT N
```

BEST PRACTICE

```
REDIM P(Max,2)
READ #1;P(*)
FOR N=1 TO Max
Wage(N)=P(N,1)*P(N,2)
NEXT N
PRINT #2;Wage(*)
```

"USING ARRAYS AS BUFFERS", Ch.4, MST MS73

OVERLAPPED I/O

IN OVERLAP MODE THE COMPUTER CAN PERFORM I/O OPERATIONS WHILE IT PERFORMS CALCULATIONS. THIS FEATURE CAN BE USED TO ADVANTAGE IN MASS STORAGE APPLICATIONS BY PLACING AS MANY NON-I/O STATEMENTS BETWEEN PRINT# AND READ# STATEMENTS AS THE LOGIC OF THE PROGRAM PERMITS.

"OVERLAPPED I/O", Ch.4, MST MS74

DMA - Direct memory access

FCREATE STATEMENT

PURPOSE:

TO CREATE A BINARY DATA FILE FOR TRANSFERING ARRAYS AT MAXIMUM (DMA) SPEED.

RESTRICTION:

AVAILABLE THROUGH MASS STORAGE OPTION ROM FOR FLEXIBLE DISK AND HARD DISK DEVICES ONLY.

SYNTAX:

FCREATE file specifier, number of physical records

EXAMPLES:

```
INTEGER Fquantity(1:1280)
FCREATE "FSTOCK:F",12
DIM Fname$(1:1280)[18]
FCREATE "FPARTS:F",102
```

"RAPID TRANSFER OF ARRAYS", Ch.2, MST MS75

CALCULATING THE SIZE OF BINARY DATA FILES

CALCULATE THE NUMBER OF PHYSICAL RECORDS REQUIRED TO STORE AN ARRAY IN A BINARY DATA FILE AS FOLLOWS.

STRING ARRAYS:

INT [NUMBER OF ELEMENTS * (BYTES PER ELEMENT + 2)/256]+2

WHERE THE BYTES PER ELEMENT IS THE DIMENSIONED LENGTH OF EACH ELEMENT AND NOT THE CURRENT STRING LENGTH OF THE INDIVIDUAL ELEMENTS. ADD ONE ADDITIONAL BYTE IF THE DIMENSIONED LENGTH IS ODD.

NUMERIC ARRAYS:

INT [NUMBER OF ELEMENTS * K/256]+2

WHERE K=2 IF AN INTEGER ARRAY,
K=4 IF A SHORT-PRECISION ARRAY,
K=8 IF A FULL-PRECISION ARRAY.

REMEMBER THE CURRENT OPTION BASE SETTING WHEN COMPUTING THE NUMBER OF ELEMENTS IN AN ARRAY.

FPRINT AND FREAD STATEMENTS

PURPOSE:

TO TRANSFER ARRAYS TO OR FROM A BINARY DATA FILE AT MAXIMUM (DMA) SPEED.

RESTRICTION:

AVAILABLE THROUGH MASS STORAGE OPTION ROM FOR FLEXIBLE DISK AND HARD DISK DEVICES ONLY.

SYNTAX:

```
FPRINT file specifier, array identifier
FREAD  file specifier, array identifier
```

EXAMPLES:

```
FCREATE "FPARTS:F",102
FPRINT  "FPARTS:F",Fname$(*)
FREAD   "FPARTS:F",Fname$(*)
```

"RAPID TRANSFER OF ARRAYS", Ch.2, MST

MS76

IMPROVING SPEED EXERCISE 1

THE PROGRAM STORED IN THE DATE-TYPE FILE NAMED "SLOW" READS INFORMATION FROM TWO FILES. SINCE THE FILE ACCESSES ARE INTERMIXED, FILE BUFFERING WOULD SPEED THE PROGRAM CONSIDERABLY. MODIFY THE PROGRAM BY INTRODUCING FILE BUFFERS.

```
10   ! IMPROVING SPEED EXERCISE 1    PROGRAM     6OCT78
20   !
30   ASSIGN #8 TO "SERIAL",R,"WRITE" ! OPEN "SERIAL" FILE FOR RECORD I/O.
40   ASSIGN #9 TO "RANDOM",R,"WRITE" ! OPEN "RANDOM" FILE FOR RECORD I/O.
50   PRINT LIN(6);TAB(28);"NATIONAL FOOTBALL LEAGUE";LIN(2)
60   PRINT TAB(21);"TEAM";TAB(41);"CONFERENCE";TAB(56);"DIVISION";LIN(1)
70   FOR N=1 TO 28                   ! FOR EACH TEAM,
80   READ #8;Loc$,Name$              ! READ LOCATION AND NAME.
90   READ #9,N;Dummy$,Conf$,Div$     ! READ CONFERENCE AND DIVISION.
100  PRINT TAB(15);Loc$;" ";Name$;TAB(42);Conf$;TAB(57);Div$ ! PRINT.
110  NEXT N
120  ASSIGN #8 TO *                  ! CLOSE "SERIAL" FOR RECORD I/O.
130  ASSIGN #9 TO *                  ! CLOSE "RANDOM" FOR RECORD I/O.
140  END
```

NATIONAL FOOTBALL LEAGUE

TEAM	CONFERENCE	DIVISION
DALLAS COWBOYS	NATIONAL	EAST
NEW YORK GIANTS	NATIONAL	EAST
PHILADELPHIA EAGLES	NATIONAL	EAST
ST. LOUIS CARDINALS	NATIONAL	EAST
WASHINGTON REDSKINS	NATIONAL	EAST
CHICAGO BEARS	NATIONAL	CENTRAL
DETROIT LIONS	NATIONAL	CENTRAL
GREEN BAY PACKERS	NATIONAL	CENTRAL
MINNESOTA VIKINGS	NATIONAL	CENTRAL
ATLANTA FALCONS	NATIONAL	WEST
LOS ANGELES RAMS	NATIONAL	WEST
NEW ORLEANS SAINTS	NATIONAL	WEST
SAN FRANCISCO 49ERS	NATIONAL	WEST
TAMPA BAY BUCCANEERS	NATIONAL	WEST
BALTIMORE COLTS	AMERICAN	EAST
BUFFALO BILLS	AMERICAN	EAST
MIAMI DOLPHINS	AMERICAN	EAST
NEW ENGLAND PATRIOTS	AMERICAN	EAST
NEW YORK JETS	AMERICAN	EAST
CINCINNATI BENGALS	AMERICAN	CENTRAL
CLEVELAND BROWNS	AMERICAN	CENTRAL
HOUSTON OILERS	AMERICAN	CENTRAL
PITTSBURGH STEELERS	AMERICAN	CENTRAL
SEATTLE SEAHAWKS	AMERICAN	CENTRAL
DENVER BRONCOS	AMERICAN	WEST
KANSAS CITY CHIEFS	AMERICAN	WEST
OAKLAND RAIDERS	AMERICAN	WEST
SAN DIEGO CHARGERS	AMERICAN	WEST

IMPROVING SPEED EXERCISE 2

IN THE PRECEDING EXERCISE, FILE BUFFERS WERE USED TO SPEED DATA ACCESS. AN EVEN GREATER IMPROVEMENT COULD BE MADE USING ARRAY BUFFERS. DIMENSION TWO ARRAYS AS FOLLOWS:

```
DIM Team$(1:28,1:2),Conf$(1:28,1:3)
```

MODIFY THE PROGRAM TO READ THE "SERIAL" FILE INTO Team$(*) AND THE "RANDOM" FILE INTO Conf$(*). THEN, PRINT THE DATA.

ADDITIONAL CAPABILITIES

ON END # STATEMENT

OFF END # STATEMENT

TYP FUNCTION

CHECK READ STATEMENT

ON END # STATEMENT

PURPOSE:

TO TRANSFER CONTROL DURING RECORD I/O WHEN AN EOF OR EOR (RANDOM ONLY) IS ENCOUNTERED.

SYNTAX:

```
ON END # file number CALL subprogram name
ON END # file number GOSUB line identifier
ON END # file number GOTO line identifier
```

EXAMPLE PROGRAM:

```
 ON END #10 GOTO Done
Loop: READ #10;Name$,Quantity,Price
 PRINT #10; Name$,Quantity,Price
 GOTO Loop
Done:  END
```

"OVERFLOWING FILES", Ch.3, MST MS78

OFF END # STATEMENT

PURPOSE:

TO TURN OFF THE CORRESPONDING ON END #
STATEMENT.

SYNTAX:

```
OFF END # file number
```

EXAMPLE:

```
OFF END #10
```

CAUTION:

THE ON END # STATEMENT DISABLES OVERLAPPED
PROCESSING FOR THE SPECIFIED FILE NUMBER.
OFF END # ENABLES OVERLAPPED PROCESSING IF
OVERLAP MODE IS IN EFFECT.

"OVERFLOWING FILES", Ch.3, MST

MS79

TYP FUNCTION

can't use in an output dataset

PURPOSE:

TO IDENTIFY THE DATA TYPE OF THE NEXT ITEM IN THE FILE.

SYNTAX:

```
TYP (file number)
```

PARTIAL LIST OF VALUES RETURNED BY TYP FUNCTION:

0 ERROR	2 STRING	4 EOR MARK
1 F. P. NUMBER	3 EOF MARK	5 INTEGER

EXAMPLE PROGRAM:

```
READ #8;Sci_name$
IF TYP(8)=2 THEN READ #8;Common_name$
READ #8;Region
```

"PREVIEWING A DATA ITEM", Ch.2, MST   MS80

CHECK READ STATEMENT

PURPOSE:

TO VERIFY THAT INFORMATION IS CORRECTLY WRITTEN.

SYNTAX:

```
CHECK READ [OFF] [# file number]
```

EXAMPLES:

```
CHECK READ #8
ON ERROR GOTO Errorcheck
PRINT #8;Data(*)
CHECK READ OFF #8

CHECK READ
RE-SAVE "LOG"
CHECK READ OFF
```

CAUTION:

THE BUFFER STATEMENT OVERRIDES CHECKREAD.
CHECKREAD IS PERFORMED WHEN BUFFER IS DUMPED.

"PRINT VERIFICATION", Ch.2, MST MS81

ADDITIONAL CAPABILITIES EXERCISE 1

THE DATA FILE NAMED "POEM" CONTAINS AN UNKNOWN NUMBER OF SERIALLY-STORED STRINGS THAT ARE UP TO 30 CHARACTERS LONG. READ AND PRINT THE STRINGS. USE THE ON END STATEMENT TO DETECT THE END-OF-FILE.

ADDITIONAL CAPABILITIES EXERCISE 2

THE DATA FILE NAMED "MESSAG" CONTAINS AN UNKNOWN NUMBER OF SERIALLY-STORED INTEGERS AND STRINGS. THE TASK IS TO READ AND PRINT THE CONTENTS OF THE FILE. USE THE TYP FUNCTION TO IDENTIFY THE TYPE OF EACH DATUM BEFORE READING IT. THE TYP FUNCTION RETURNS 5 FOR INTEGERS, 2 FOR STRINGS, AND 3 FOR EOF MARKS. PRINT THE MESSAGE ON ONE LINE. IN A PRINT LIST A SEMICOLON FOLLOWING THE DATA LIST SUPPRESSES THE CARRIAGE RETURN AND LINE FEED.

GRAPHICS AGENDA

GETTING STARTED
ABSOLUTE PLOTTING
RELATIVE PLOTTING
LINE DIFFERENTIATION
SCALING
LIMITS
AXES
LABELING
DIGITIZING
CRT ONLY
MULTIPLE PLOTTERS

GR1

GETTING STARTED

PLOTTING SPACE

DEFAULT COORDINATE SYSTEM

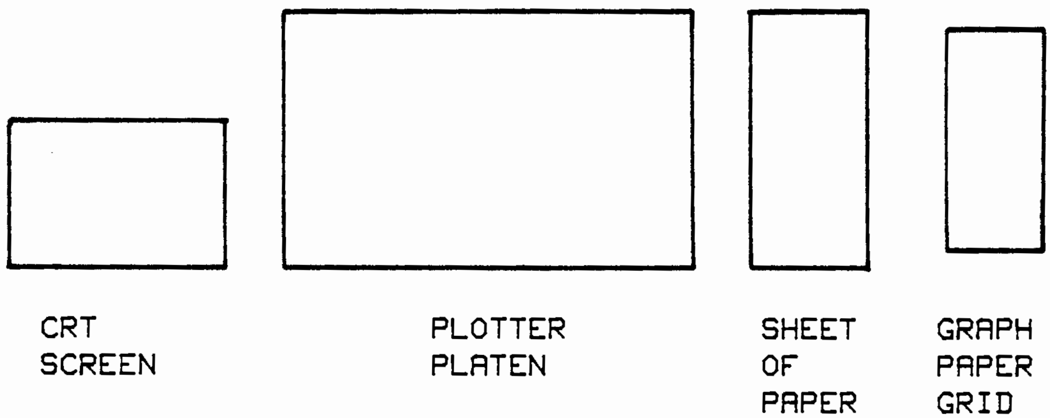
RATIO FUNCTION

FIRST PROGRAM

GR2

PLOTTING SPACE

THE PLOTTING SPACE IS THE AREA IN WHICH THE PEN CAN MOVE.



"PLOTTING SPACE", Ch.1, GPT GR3

13 graphics raster

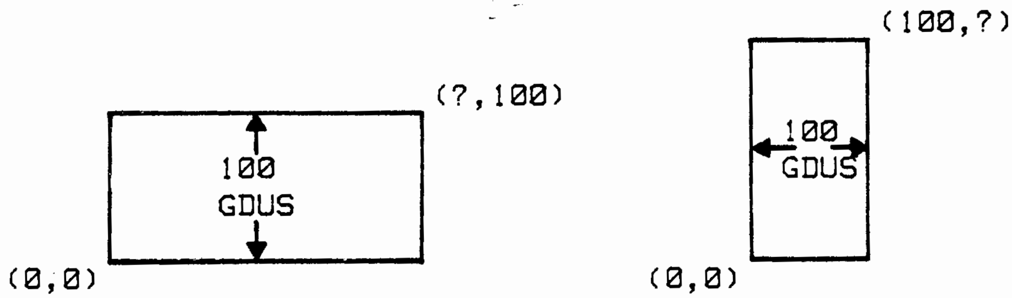
raster · physical size of area

DEFAULT COORDINATE SYSTEM

DESCRIPTION:

SHORT SIDE IS 100 "GRAPHIC DISPLAY UNITS."
LENGTH OF ONE HORIZONTAL GDU EQUALS
LENGTH OF ONE VERTICAL GDU.
ORIGIN IS AT LOWER LEFT.

EXAMPLES: ? = ~ 200



"UNIT-OF-MEASURE MODES", Ch.1, GPT

GR4

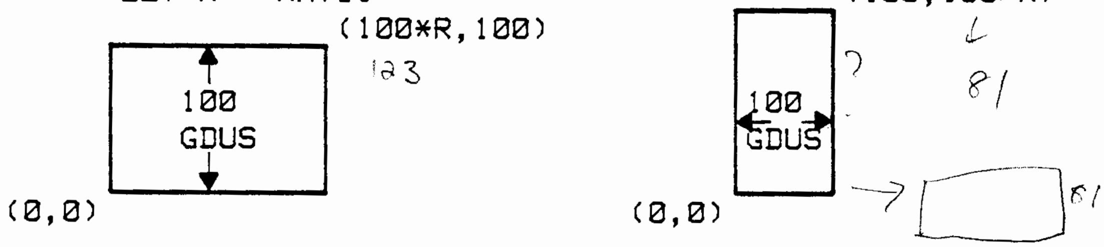
RATIO FUNCTION

DEFINITION:

RATIO = HORIZONTAL LENGTH / VERTICAL LENGTH

EXAMPLES:

LET R = RATIO = 1.23 . . .



WHAT ARE THE UPPER RIGHT COORDINATES OF THE CRT?

TO FIND OUT, TYPE THE FOLLOWING:

PLOTTER IS "GRAPHICS" $\boxed{\text{EXECUTE}}$ RATIO $\boxed{\text{EXECUTE}}$

"RATIO FUNCTION", Ch.2, GPT GR5

GDU 1 unit of shortest side

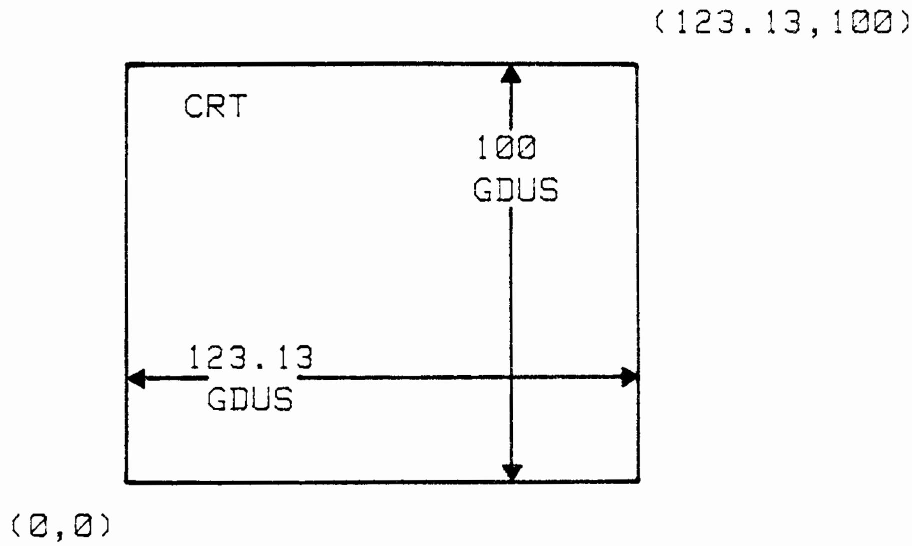
1 GDU = 1/100 shortest side

Ratio = $\frac{x}{y}$

to find x use Ratio * y

y use $\frac{x}{\text{Ratio}}$

SUPPLEMENT
CRT DEFAULT COORDINATE SYSTEM



GRS

FIRST PROGRAM

or PLOTTER IS 13, "GRAPHICS"

```
10  PLOTTER IS "GRAPHICS"
20  GRAPHICS  – involves the
30  FRAME
40  DUMP GRAPHICS
50  END
```

PLOT
MOVE
DRAW
selects

LINE 10:

SELECTS THE CRT TO BE THE PLOTTER.
INITIALIZES ALL GRAPHICS PARAMETERS.

118

LINE 20:

SELECTS GRAPHICS MODE ON THE CRT.

LINE 30:

DRAWS A BOX AROUND THE CRT SCREEN.

LINE 40:

REPRODUCES CRT GRAPHICS IMAGE ON PRINTER.

GR7

FIVE WAYS TO PLOT

THERE ARE FIVE WAYS TO MOVE THE PEN FROM ITS CURRENT LOCATION TO A NEW LOCATION.

STATEMENT	(X,Y) COORDINATES	PEN CONTROL
MOVE x,y	ABSOLUTE	PEN UP
DRAW x,y	ABSOLUTE	PEN DOWN
PLOT x,y[,p]	ABSOLUTE	DATA DIRECTED
RPLOT x,y[,p]	RELATIVE	DATA DIRECTED
IPLOT x,y[,p]	INCREMENTAL	DATA DIRECTED

absolute origin

local origin

GR8

ABSOLUTE PLOTTING

MOVE STATEMENT

DRAW STATEMENT

PLOT STATEMENT

PENUP STATEMENT

GR9

MOVE STATEMENT

PURPOSE:

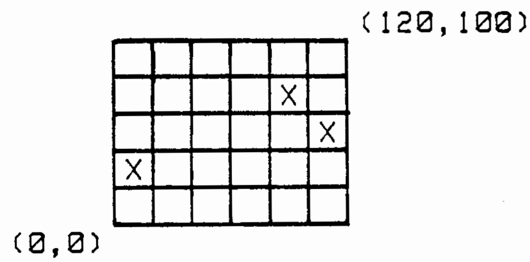
TO MOVE THE PEN TO A NEW LOCATION WITHOUT DRAWING A LINE.

SYNTAX:

MOVE x,y in current units

EXAMPLE:

```
MOVE 10,30
MOVE 90,70
MOVE 110,50
```



"MOVE STATEMENT", Ch.3, GPT GR10

DRAW STATEMENT

PURPOSE:

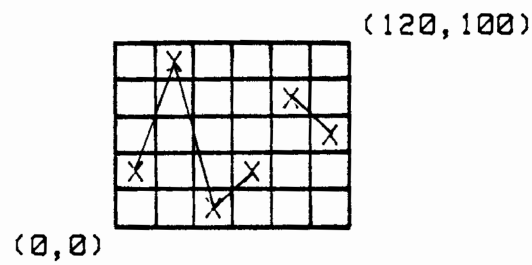
TO DRAW A LINE FROM THE CURRENT PEN LOCATION TO A NEW LOCATION.

SYNTAX:

DRAW x,y

EXAMPLE:

```
MOVE 10,30
DRAW 30,90
DRAW 50,10
DRAW 70,30
MOVE 90,70
DRAW 110,50
```

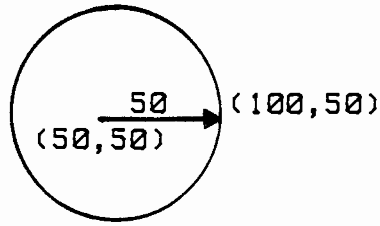


"DRAW STATEMENT", Ch.3, GPT

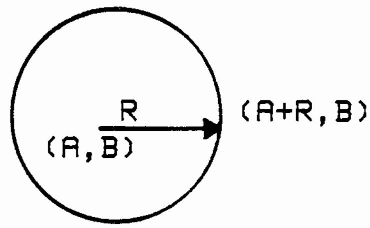
GR11

MOVE / DRAW EXAMPLE PROGRAM

```
10   PLOTTER IS "GRAPHICS"
11   GRAPHICS
12   CALL Circle (50,50,50)
13   END
```




```
20   SUB Circle (A,B,R)
21   DEG
22   MOVE A+R,B
23   FOR D=0 TO 360 STEP 5
24   DRAW A+R*COS(D),B+R*SIN(D)
25   NEXT D
26   ! LIFT PEN
27   SUBEND
```



GR12

PLOT STATEMENT

PURPOSE:

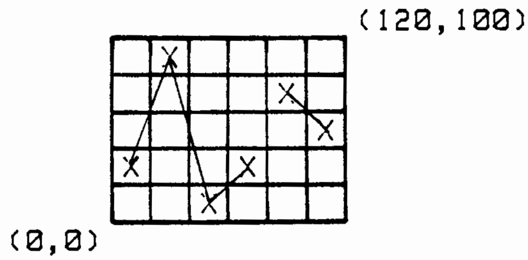
TO PERFORM ABSOLUTE DATA PLOTTING WITH OPTIONAL PEN CONTROL.

SYNTAX:

```
PLOT x, y [,p]
```

EXAMPLE:

```
PLOT 10,30,-2
PLOT 30,90,-1
PLOT 50,10,+1
PLOT 70,30,+2
PLOT 90,70
PLOT 110,50
```



"PLOT STATEMENT", Ch.3, GPT

GR13

SUPPLEMENT
PEN CONTROL

THE PEN CONTROL PARAMETER APPEARS IN THE PLOT, IPLOT, AND RPLOT STATEMENTS.

THE PEN CONTROL PARAMETER DETERMINES WHEN AND HOW THE PEN MOVES AS FOLLOWS:

ODD:	DROP PEN	- :	BEFORE MOVE
EVEN:	LIFT PEN	+ :	AFTER MOVE

E.G.,

-1:	DROP, MOVE	-2:	LIFT, MOVE
+1:	MOVE, DROP	+2:	MOVE, LIFT

DEFAULT VALUE: 1 ZERO VALUE: SAME AS +2

"PLOT STATEMENT", Ch.3, GPT GR14

PENUP STATEMENT

PURPOSE:

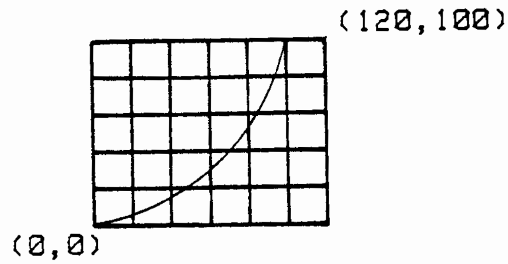
TO LIFT THE PEN.

SYNTAX:

```
PENUP
```

EXAMPLE:

```
PENUP
FOR X=0 TO 100
PLOT X,(X/10)^2
NEXT X
PENUP
```

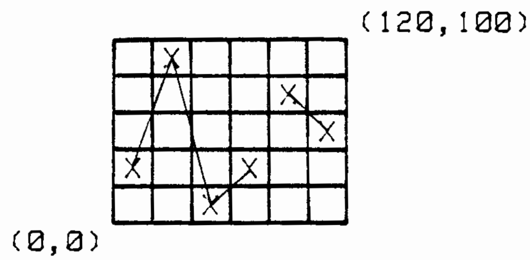


"PENUP STATEMENT", Ch.3, GPT

GR15

PLOT / PENUP EXAMPLE PROGRAM

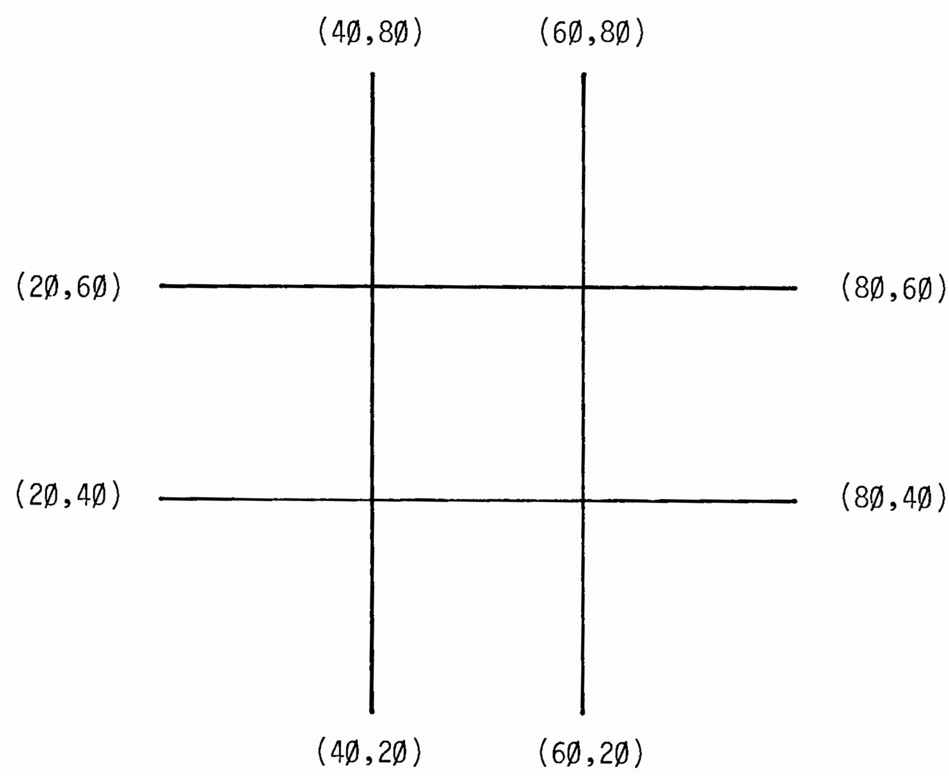
```
10  RESTORE 17
11  PENUP
12  FOR I=1 TO 6
13  READ X,Y,P
14  PLOT X,Y,P
15  NEXT I
16  PENUP
17  DATA 10,30,1,30,90,1
18  DATA 50,10,1,70,30,0
19  DATA 90,70,1,110,50,0
20  END
```



GR16

ABSOLUTE PLOTTING EXERCISE 1

DRAW THIS TIC-TAC-TOE BOARD ON THE CRT USING MOVE AND DRAW STATEMENTS.

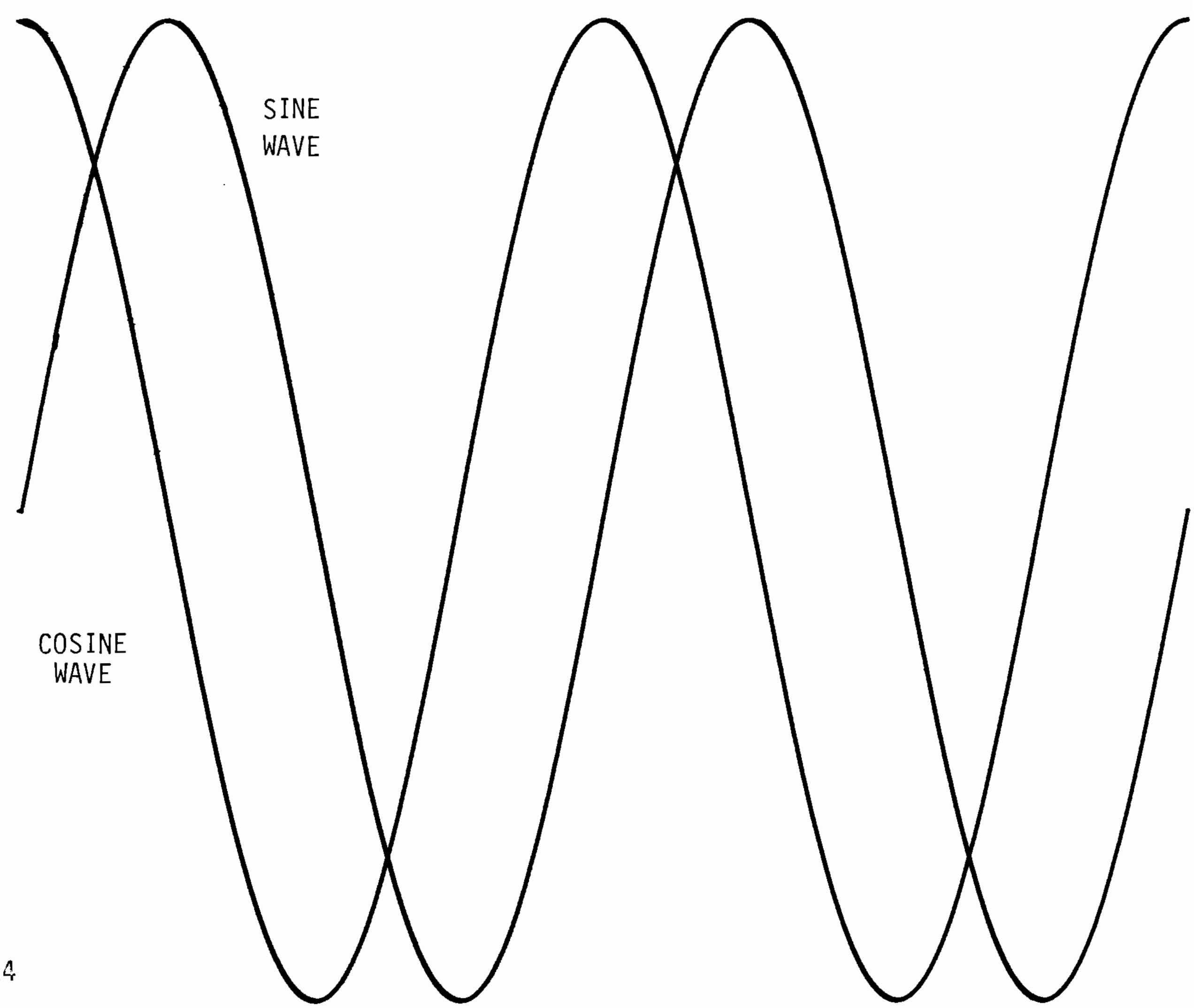


ABSOLUTE PLOTTING EXERCISE 2

WRITE A PROGRAM TO PLOT TWO CYCLES OF A SINE WAVE AND TWO CYCLES OF A COSINE WAVE.

HINT: LET Angle VARY FROM Ø TO 72Ø DEGREES.
SINE WAVE: PLOT X,Y WHERE X=Angle/6 AND Y=5Ø*(1+SIN(Angle))
COSINE WAVE: PLOT X,Y WHERE X=Angle/6 AND Y=5Ø*(1+COS(Angle))

NOTE: THE EQUATIONS ABOVE SET UP A CORRESPONENCE BETWEEN THE UNITS OF THE PROBLEM AND GRAPHIC DISPLAY UNITS (GDUS). FOR EXAMPLE, SINCE THE CRT IS ABOUT 120 GDUS WIDE, WE MUST SET UP A CORRESPONDENCE IN THE X (HORIZONTAL) DIRECTION BETWEEN Ø TO 12Ø GDUS AND Ø TO 72Ø DEGREES. THE EQUATION Angle=D/6 SETS UP THIS CORRESPONDENCE. SIMILARLY, SINCE THE CRT IS 1ØØ GDUS HIGH AND SINCE THE SINE AND COSINE FUNCTIONS RETURN VALUES FROM -1 TO 1, WE MUST SET UP A CORRESPONDENCE IN THE Y (VERTICAL) DIRECTION BETWEEN Ø TO 1ØØ GDUS AND -1 TO 1. THE EQUATIONS Y=5Ø*(1+SIN(D)) AND Y=50*(1+COS(D)) SET UP THIS CORRESPONDENCE. LATER WE WILL LEARN HOW TO TELL THE COMPUTER TO DO THESE SCALING OPERATIONS FOR US AUTOMATICALLY.



RELATIVE PLOTTING

RPLOT STATEMENT

IPLOT STATEMENT

PDIR STATEMENT

GR17

RPLOT STATEMENT

PURPOSE:

TO PLOT from where you are RELATIVE TO A LOCAL ORIGIN.

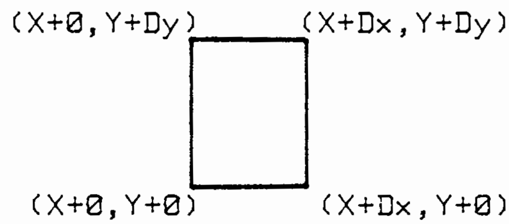
SYNTAX:

```
RPLOT x, y [, p]
```

EXAMPLE:

moves to x, y & drops pen

```
SUB Box (X,Y,Dx,Dy)
PENUP
PLOT X,Y    local origin
RPLOT Dx,0
RPLOT Dx,Dy
RPLOT 0,Dy
RPLOT 0,0
PENUP
SUBEND
```



"RPLOT STATEMENT", Ch.3, GPT

GR18

IPLOT STATEMENT

PURPOSE:

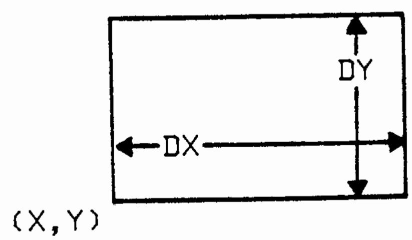
TO PLOT RELATIVE TO THE CURRENT PEN LOCATION.

SYNTAX:

IPLOT x, y [, p]

EXAMPLE:

```
SUB Box (X,Y,Dx,Dy)
PENUP
PLOT X,Y
IPLOT Dx,0
IPLOT 0,Dy
IPLOT -Dx,0
IPLOT 0,-Dy
PENUP
SUBEND
```



"IPLOT STATEMENT", Ch.3, GPT

GR19

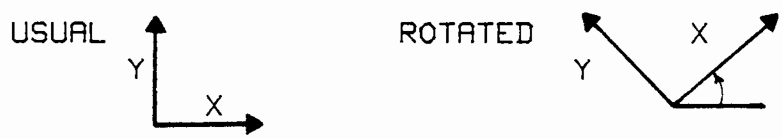
BOX SUBROUTINE

RPLOT	IPLOT
SUB Box(X,Y,Dx,Dy)	SUB Box(X,Y,Dx,Dy)
PENUP	PENUP
PLOT X,Y	PLOT X,Y
RPLOT Dx,Ø	IPLOT Dx,Ø
RPLOT Dx,Dy	IPLOT Ø,Dy
RPLOT Ø,Dy	IPLOT -Dx,Ø
RPLOT Ø,Ø	IPLOT Ø,-Dy
PENUP	PENUP
SUBEND	SUBEND

PDIR STATEMENT

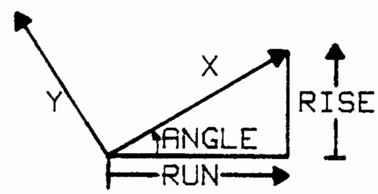
PURPOSE:

TO ROTATE THE COORDINATE SYSTEM
FOR INCREMENTAL PLOTTING (IPLOT)
AND RELATIVE PLOTTING (RPLOT).



SYNTAX:

PDIR angle
PDIR run, rise



"PDIR STATEMENT", Ch.3, GPT

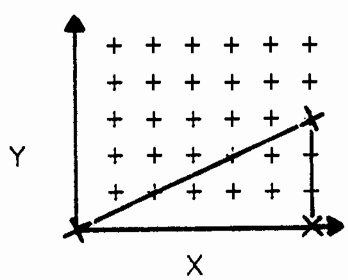
GR20

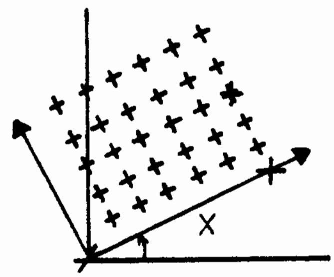
PDIR STATEMENT

GDU EXAMPLES:

THE STATEMENTS BELOW PERFORM THE SAME TASK.

```
PDIR 6,2
PDIR 3,1
PDIR ATN(1/3)
PDIR 18.4349488229      ! DEGREES
```



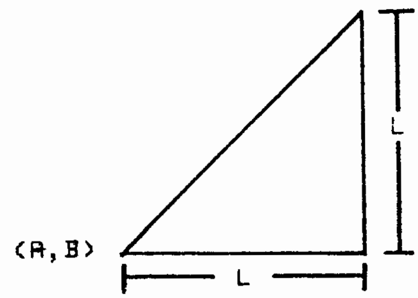



```
MOVE    0, 0
RPLOT   6, 0
RPLOT   6, 3
RPLOT   0, 0
IPLOT   6, 0
IPLOT   0, 3
IPLOT  -6,-3
```

GR21

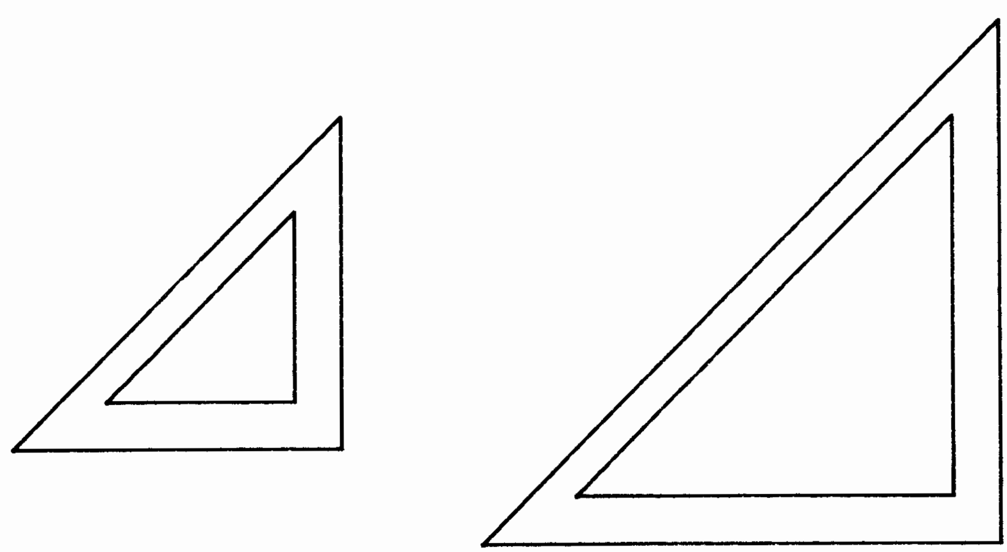
RELATIVE PLOTTING EXERCISE 1

DEVELOP A SUBROUTINE THAT DRAWS A RIGHT TRIANGLE AS SHOWN BELOW. THE ABSOLUTE COORDINATES OF THE LOWER LEFT CORNER AND THE LENGTH OF A SHORT SIDE SHOULD BE INPUTS TO THE SUBROUTINE. USE RPLOT.



WRITE A TEST PROGRAM THAT USES THE SUBROUTINE TO DRAW FOUR DIFFERENT TRIANGLES ON THE CRT SCREEN.

SAMPLE OUTPUT:



RELATIVE PLOTTING EXERCISE 2

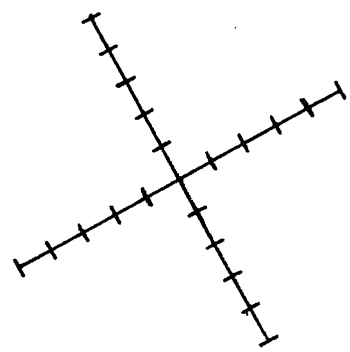
REDO RELATIVE PLOTTING EXERCISE 1 USING IPLOT INSTEAD OF RPLOT.

RELATIVE PLOTTING EXERCISE 3

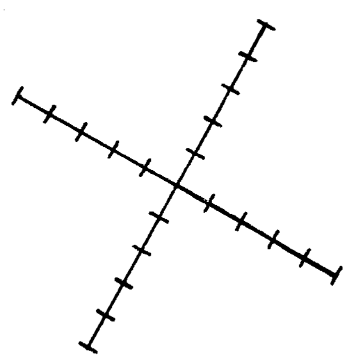
DEVELOP A PROGRAM TO DRAW A PAIR OF X-Y AXES ROTATED AT ANY ARBITRARY ANGLE. INPUT THE ANGLE FROM THE KEYBOARD. USE PDIR WITH IPLOT AND/OR RPLOT TO DRAW THE AXES.

SAMPLE OUTPUT:

```
ANGLE (DEGREES)?
3Ø
```



```
ANGLE (DEGREES)?
6Ø
```



LINE DIFFERENTIATION

LINE TYPE STATEMENT

PEN STATEMENT

GR22

LINE TYPE STATEMENT

PURPOSE:

TO SELECT ONE OF TEN LINE PATTERNS.
TO SET THE LENGTH OF REPEATED PATTERNS.

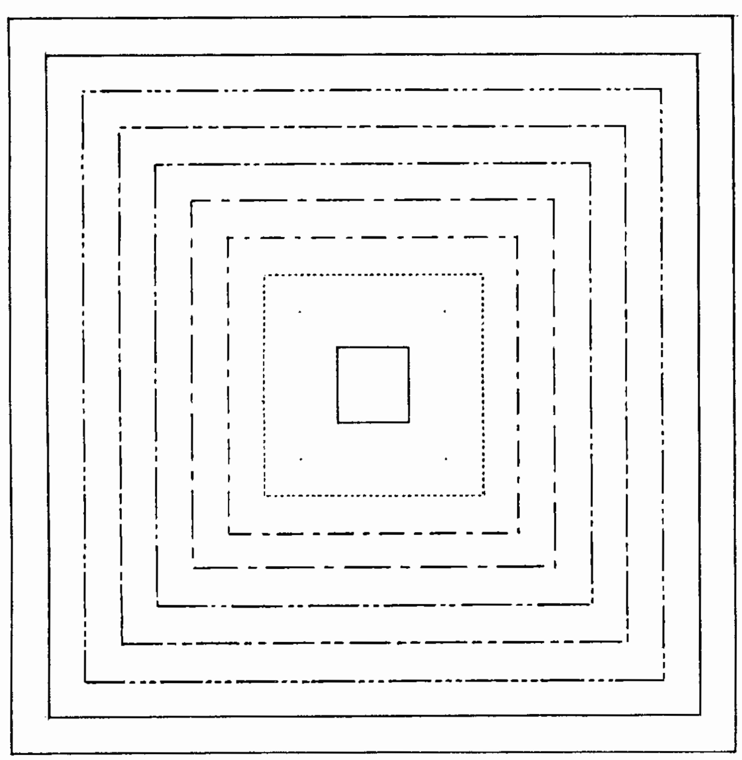
SYNTAX:

LINE TYPE id number [,length]

EXAMPLE:

```
FOR L=1 TO 10
LINE TYPE L,L+2
CALL Box(60-5*L,50-5*L,10*L,10*L)
NEXT L
```

"LINE TYPE STATEMENT", Ch.3, GPT GR23



PEN STATEMENT FOR THE CRT

PURPOSE:

TO TURN LINE GENERATION AND ERASURE ON AND OFF.

SYNTAX:

PEN pen number

EXAMPLE:

```
Loop: PEN 1                          ! TURN ON LINE GEN.
 CALL Box(50,40,20,20)               ! DRAW BOX.
 WAIT 500                            ! WAIT 500 MSEC.
 PEN -1                              ! TURN ON LINE ERASE.
 CALL Box(50,40,20,20)               ! ERASE BOX.
 WAIT 500                            ! WAIT 500 MSEC.
 GOTO Loop                           ! REPEAT.
```

"PEN STATEMENT", Ch.3, GPT

GR24

```
100  !           PEN NUMBER RULE FOR CRT
110  !
120  !   -----------------------------------------
130  !  | VALUE OF     |   LINE       |   LINE    |
140  !  | PEN NUMBER   | GENERATION   |  ERASURE  |
150  !  -------------   -------------   ---------
160  !  |    > 0       |    ON        |   OFF     |
170  !  |    = 0       |    OFF       |   OFF     |
180  !  |    < 0       |    OFF       |   ON      |
190  !   -----------------------------------------
```

PEN STATEMENT FOR MULTIPLE PEN PLOTTERS

PURPOSE:

TO SELECT, RETURN, OR REPLACE A PEN.

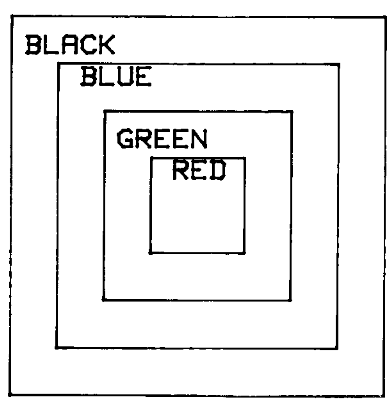
SYNTAX:

```
PEN pen number
```

EXAMPLE:

```
FOR N=1 TO 4
PEN N
CALL Box(60-5*N,50-5*N,10*N,10*N)
NEXT N
PEN 0
```

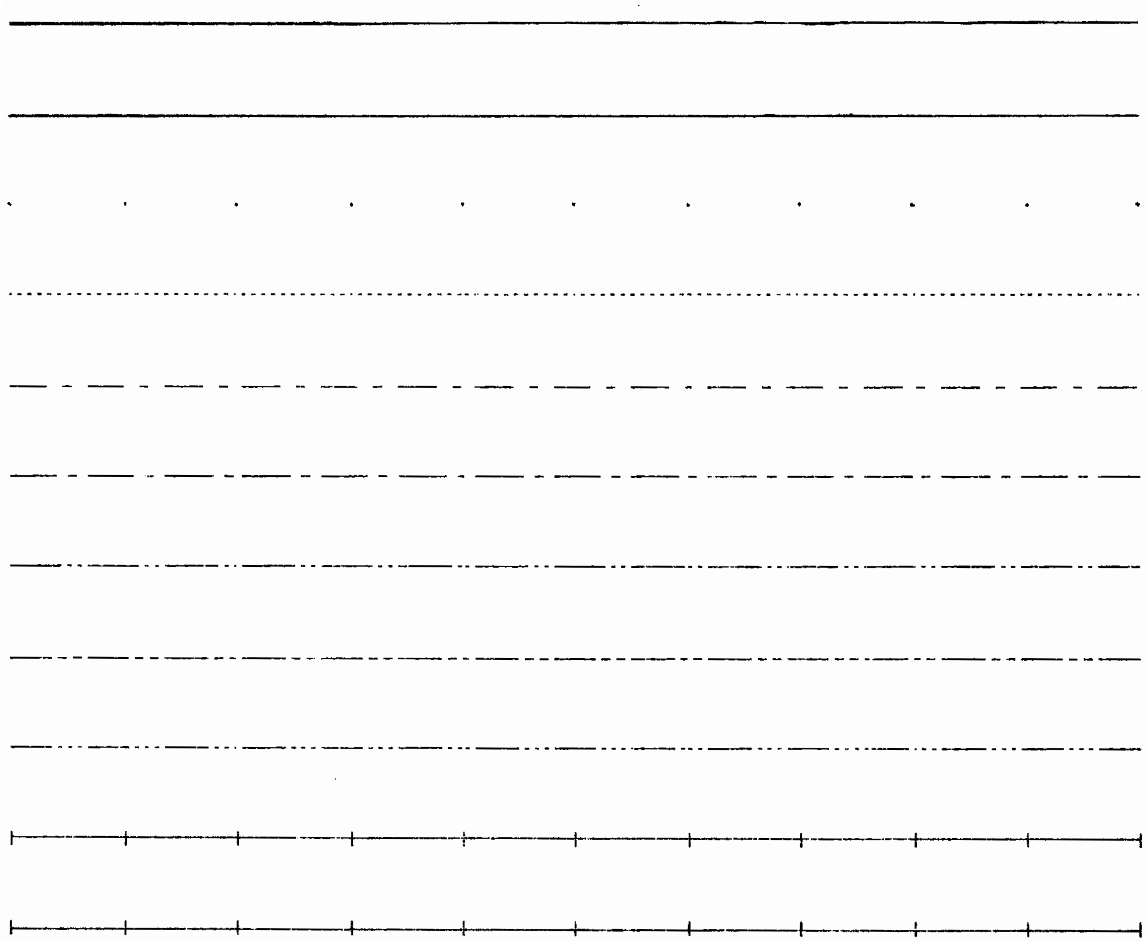
"PEN STATEMENT", Ch.3, GPT GR25



LINE DIFFERENTIATION EXERCISE 1

DRAW A TABLE OF LINE TYPES 1 THROUGH 11. EXPERIMENT WITH DIFFERENT PATTERN LENGTHS.

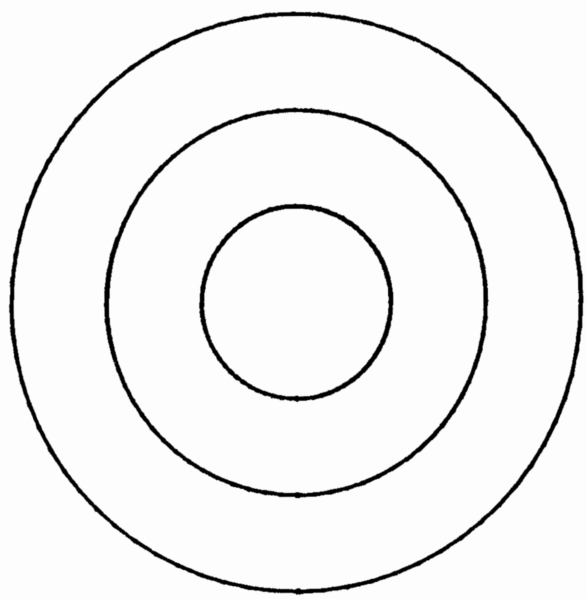
SAMPLE OUTPUT:



LINE DIFFERENTIATION EXERCISE 2

THE CIRCLE SUBROUTINE USED TO ILLUSTRATE MOVE AND DRAW IS LOCATED IN THE FILE NAMED "CIRCLE" ON THE MASS STORAGE/GRAPHICS TAPE CARTRIDGE. USE THIS SUBROUTINE TO DRAW THREE CONCENTRIC CIRCLES. THEN ERASE THE CIRCLES. REMEMBER TO TURN ON LINE GENERATION AT THE CONCLUSION OF THE PROGRAM.

SAMPLE OUTPUT:




SCALING
PREVIEW

SCALING

MSCALE STATEMENT

SHOW STATEMENT

SCALE STATEMENT

LOCATE STATEMENT

SETGU STATEMENT

SETUU STATEMENT

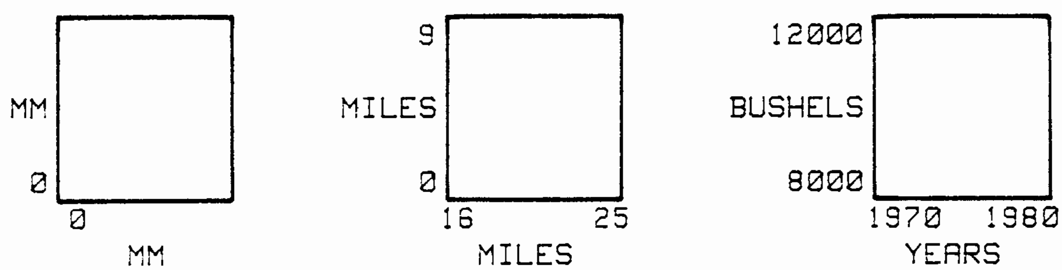
GR26

USER-DEFINED COORDINATE SYSTEMS

DESCRIPTION:

THE USER DEFINES THE COORDINATE SYSTEM VIA THE MSCALE, SHOW, OR SCALE STATEMENTS. THE PROBLEM DETERMINES THE TYPES OF UNITS AND THE RANGES OF VALUES. UNITS THUS ASSIGNED ARE CALLED "USER-DEFINED UNITS" OR "UDUS".

EXAMPLES:



GR27

4 TYPES OF COORDINATE SYSTEMS

TYPE	DEFINED BY	X UNIT= Y UNIT?	LENGTH OF ONE UNIT
DEFAULT	DEVICE	YES	1% OF SHORT SIDE
MSCALE	USER	YES	1 MILLIMETER
SHOW	USER	YES	DETERMINED BY RANGE OF VALUES
SCALE	USER	NO	DETERMINED BY RANGE OF VALUES

square grids so that a circle is not transformed into an ellipse

GR28

MSCALE STATEMENT

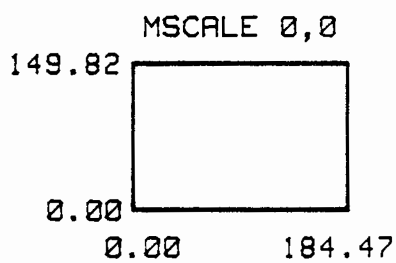
PURPOSE:

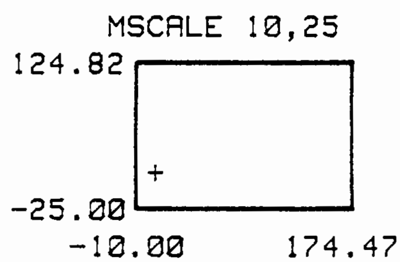
TO DEFINE A COORDINATE SYSTEM IN MILLIMETERS.

SYNTAX:

MSCALE x-offset, y-offset

EXAMPLES:





"MSCALE STATEMENT", Ch.2, GPT GR29

SHOW STATEMENT

PURPOSE:

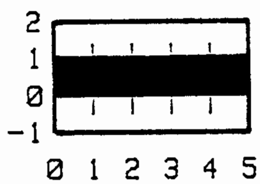
TO DEFINE A COORDINATE SYSTEM IN WHICH THE LENGTH OF ONE HORIZONTAL UNIT EQUALS THE LENGTH OF ONE VERTICAL UNIT.

SYNTAX:

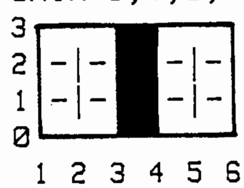
```
SHOW xmin, xmax, ymin, ymax
```

EXAMPLES:

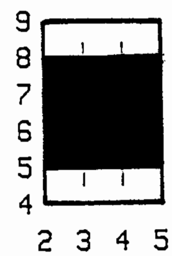
SHOW 0,5,0,1



SHOW 3,4,0,3



SHOW 2,5,5,8



"SHOW STATEMENT", Ch.2, GPT

GR30

eg SHOW 10 , 20 , 1970 , 19770

20

10

1970

1970

SCALE STATEMENT

PURPOSE:

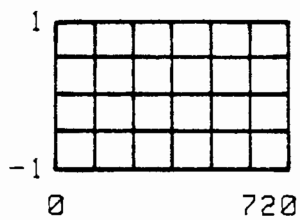
TO DEFINE A COORDINATE SYSTEM IN WHICH THE LENGTH OF ONE HORIZONTAL UNIT MAY DIFFER FROM THE LENGTH OF ONE VERTICAL UNIT.

SYNTAX:

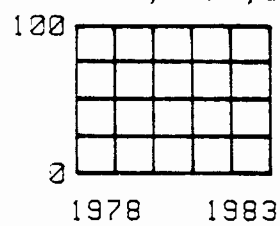
SCALE xmin, xmax, ymin, ymax

EXAMPLES:

SCALE 0,720,-1,1



SCALE 1978,1983,0,100



"SCALE STATEMENT", Ch.2, GPT

GR31

forms the [illegible] [illegible] [illegible] to
accept the given scale

LOCATE STATEMENT – Physical area which plotter uses

PURPOSE:

TO SET TWO "LOCATE" POINTS FOR SCALING.
THEY ARE (XMIN,YMIN) AND (XMAX,YMAX).

SYNTAX:

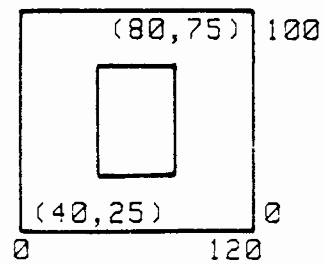
LOCATE [xmin, xmax, ymin, ymax]

RULE:

PARAMETERS MUST BE SPECIFIED IN GDUS.

EXAMPLE:

LOCATE 40,80,25,75



"LOCATE STATEMENT", Ch.2, GPT

GR32

Scale statement maps onto
Locate statement; assign a co-ordinate
system to the plotting space

LOCATE/SCALE EXAMPLE

```
120  PLOTTER IS "GRAPHICS"     ! CHOOSE CRT & INIT
130  GRAPHICS                  ! SET GRAPHICS MODE
140  LOCATE 40,80,30,70        ! SET LOCATE POINTS
150  SCALE 0,360,-1,1          ! SET PROBLEM UNITS
160  DEG                       ! SET DEGREES
170  PENUP                     ! LIFT PEN
180  FOR Angle=0 TO 360        ! PLOT POINTS AT
190  PLOT Angle,SIN(Angle)     !      ONE DEGREE
200  NEXT Angle                !      INTERVALS
210  PENUP                     ! LIFT PEN
220  END                       ! DONE
```

GR33

TWO KINDS OF UNITS

	GDUS	UDUS
NAME	GRAPHIC DISPLAY UNITS	USER DEFINED UNITS
DEFINED BY	PLOTTER IS LIMIT	MSCALE SHOW SCALE
DEFAULT	DEVICE DEPENDENT	1 UDU = 1GDU

GR34

CURRENT UNITS MODE

THE "CURRENT UNITS MODE" IS EITHER GDUS OR UDUS.

SET CURRENT UNITS MODE TO GDUS WITH:

SETGU

SET CURRENT UNITS MODE TO UDUS WITH:

SETUU	MSCALE	SHOW	SCALE
PLOTTER IS			

STATEMENTS AFFECTED BY CURRENT UNITS MODE:

MOVE	DRAW	PLOT	RPLOT
IPLOT	CLIP	AXES	GRID
WHERE	POINTER	DIGITIZE	CURSOR
DUMP GRAPHICS			
FRAME			

GR35

SETGU STATEMENT

PURPOSE:

TO SET THE CURRENT UNITS MODE TO GDUS.

SYNTAX:

```
SETGU
```

EXAMPLE:

```
! MOVE IN GDUS TO THE UPPER LEFT CORNER
! OF THE CRT TO WRITE A TITLE.
!
SETGU
MOVE 0,100
```

"SETGU AND SETUU STATEMENTS", Ch.2, GPT GR36

SETUU STATEMENT

PURPOSE:

TO SET THE CURRENT UNITS MODE TO UDUS.

SYNTAX:

```
SETUU
```

EXAMPLE:

```
! FIRST GRAPH - DRAW AXES AND PLOT DATA.
  SETUU
  . . .
! FIRST GRAPH - DRAW TITLE.
  SETGU
  . . .
! SECOND GRAPH - DRAW AXES AND PLOT DATA.
  SETUU
  . . .
```

"SETGU AND SETUU STATEMENTS", Ch.2, GPT   GR37

SCALING EXERCISE 1

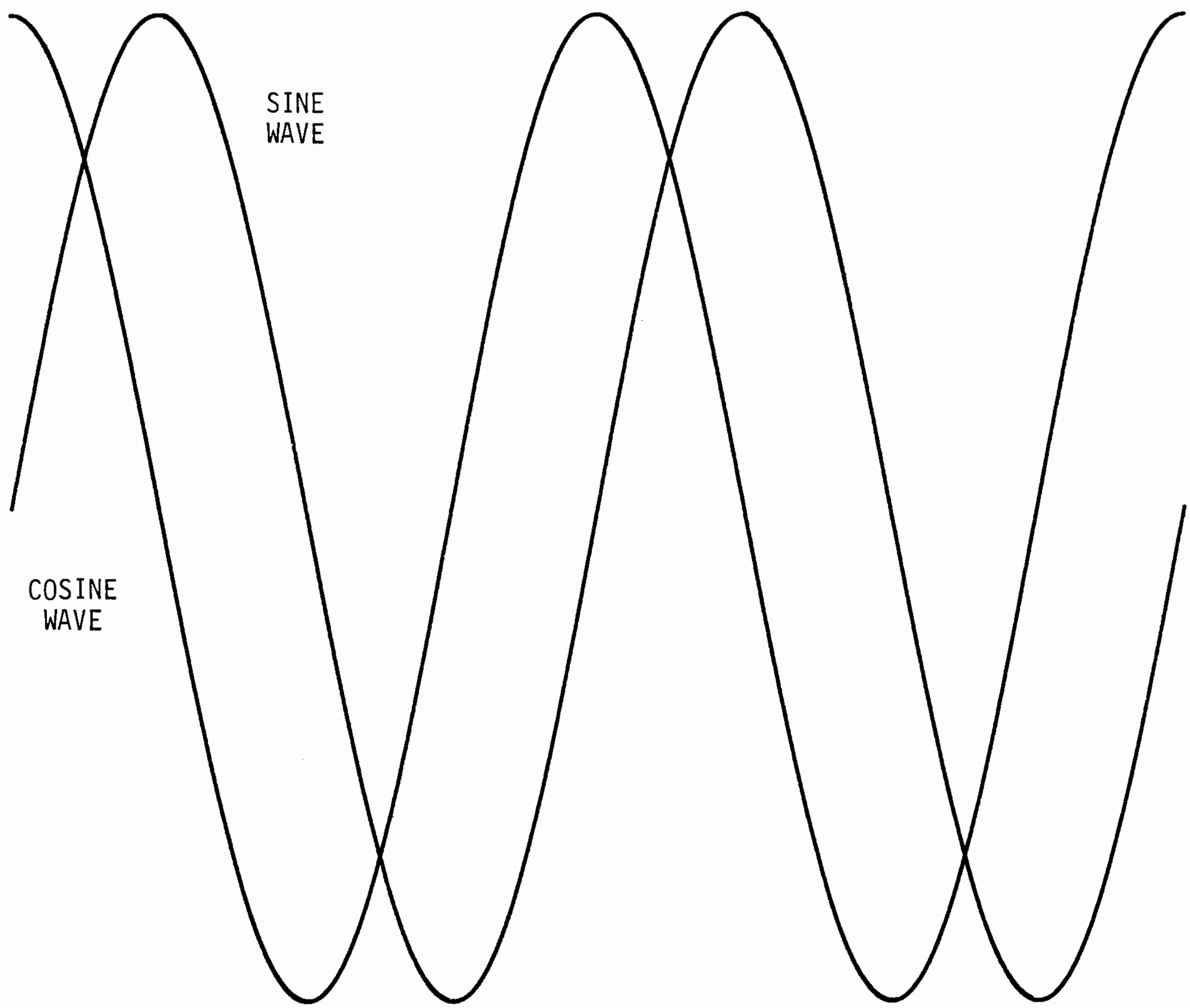
THE Circle SUBROUTINE USED TO ILLUSTRATE MOVE AND DRAW IS LOCATED IN THE FILE NAMED "CIRCLE" ON THE MASS STORAGE/GRAPHICS TAPE CARTRIDGE. THE PROGRAM BELOW PLUS THE Circle SUBROUTINE DRAWS FOUR CIRCLES. PREDICT WHAT THE OUTPUT WILL LOOK LIKE. THEN ENTER AND RUN THE PROGRAM AND SUBROUTINE TO CHECK YOURSELF.

```
10   PLOTTER IS "GRAPHICS"          ! INITIALIZE.
20   GRAPHICS                       ! SET GRAPHICS MODE.
30   CALL Circle(50,50,50)          ! DRAW CIRCLE WITH SOLID LINE.
40   MSCALE 0,0                     ! SCALE IN MM.
50   LINE TYPE 2                    ! SELECT DOTTED LINE.
60   CALL Circle(50,50,50)          ! DRAW CIRCLE WITH DOTTED LINE.
70   SHOW 0,100,0,100               ! SCALE WITH 1 X=1 Y.
80   LINE TYPE 3,80                 ! SELECT DASHED LINE.
90   CALL Circle(50,50,50)          ! DRAW CIRCLE WITH DASHED LINE.
100  SCALE 0,100,0,100              ! SCALE WITH 1 X#1 Y.
110  LINE TYPE 4                    ! SELECT LONG-SHORT LINE.
120  CALL Circle(50,50,50)          ! DRAW CIRCLE WITH LONG-SHORT LINE.
130  END
```

SCALING EXERCISE 2

IN ABSOLUTE PLOTTING EXERCISE 2, YOU WERE ASKED TO PLOT TWO CYCLES OF A SINE WAVE AND TWO CYCLES OF A COSINE WAVE. YOU WERE GIVEN EQUATIONS TO TRANSLATE THE UNITS OF THE PROBLEM (Ø TO 72Ø; -1 TO 1) TO GDUS. THE SCALE STATEMENT WILL SET UP THIS CORRESPONDNECE FOR YOU. REDO THE TWO CYCLE SINE AND COSINE WAVES USING A SCALE STATEMENT.

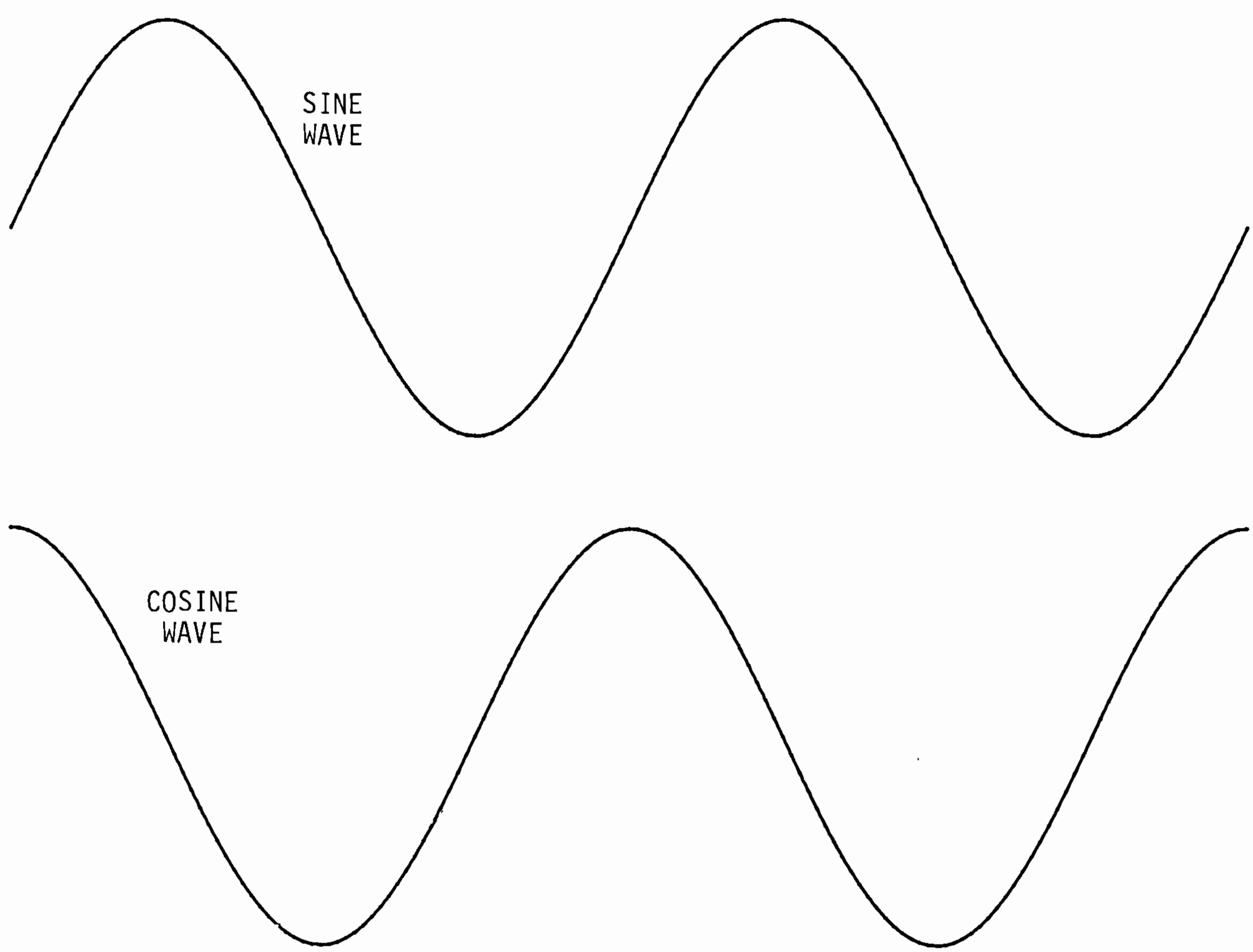
SAMPLE OUTPUT:



SCALING EXERCISE 3

MODIFY THE PROGRAM FROM SCALING EXERCISE 2 TO PLOT THE TWO CYCLE SINE WAVE IN THE UPPER PORTION OF THE SCREEN, AND THE TWO CYCLE COSINE WAVE IN THE LOWER PORTION. USE THE LOCATE STATEMENT TO POSITION EACH PLOT. SCALE FOLLOWING EACH LOCATE.

SAMPLE OUTPUT:



LIMITS

LIMIT STATEMENT

CLIP STATEMENT

UNCLIP STATEMENT

GR38

THREE TYPES OF LIMITS

TYPE	FUNCTION	HOW TO ESTABLISH
MECHANICAL LIMITS	RESTRICT PEN MOTION ABSOLUTELY	INHERENT IN THE DEVICE
HARD CLIP LIMITS	RESTRICT PEN MOTION UNDER PROGRAM CONTROL	PROGRAM WITH LIMIT STATEMENT
SOFT CLIP LIMITS	RESTRICT PLOTTING OF DATA IN UDUS	PROGRAM WITH CLIP STATEMENT

GR39

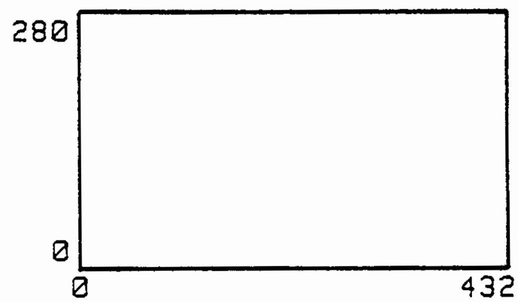
MECHANICAL LIMITS

DEFINITION:

MECHANICAL LIMITS ARE THE PHYSICAL LIMITS OF PEN MOTION INHERENT IN THE PLOTTING DEVICE.

EXAMPLE:

9872A PLOTTER PLATEN
432 MM BY 280 MM



GR40

LIMIT STATEMENT

PURPOSE:

TO ESTABLISH "HARD CLIP" LIMITS WHICH RESTRICT PROGRAMMED PEN MOTION.

SYNTAX:

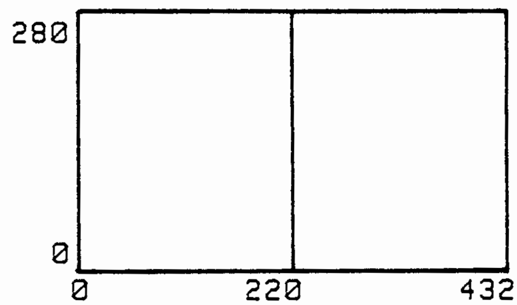
LIMIT [xmin, xmax, ymin, ymax]

RULE:

PARAMETERS MUST BE SPECIFIED IN MM.

EXAMPLE:

SHEET OF PAPER
220 MM BY 280 MM
LIMIT 0,220,0,280



"LIMIT STATEMENT", Ch.2, GPT

GR41

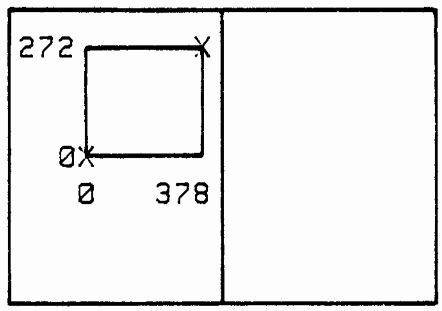
LOCATE STATEMENT REVISITED

FACTS:

THE LOCATE STATEMENT SETS TWO "LOCATE POINTS" FOR SCALING. THE LOCATE STATEMENT ALSO RESTRICTS THE PLOTTING OF DATA IN UDUS TO THE RECTANGLE DEFINED BY THESE LOCATE POINTS!

EXAMPLE:

```
COLORADO MAP
LOCATE
SHOW 0,378,0,272
```



"LOCATE STATEMENT", Ch.2, GPT

GR42

CLIP STATEMENT

PURPOSE:

TO ESTABLISH "SOFT CLIP" LIMITS WHICH RESTRICT THE PLOTTING OF DATA IN UDUS.

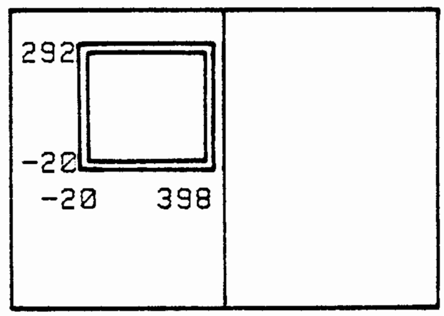
SYNTAX:

CLIP [xmin, xmax, ymin, ymax]

EXAMPLE:

COLORADO + BORDERS
CLIP -20,398,-20,292

-ve relative to last LOCATE statement



'CLIP STATEMENT', Ch.2, GPT

GR43

UNCLIP STATEMENT

PURPOSE:

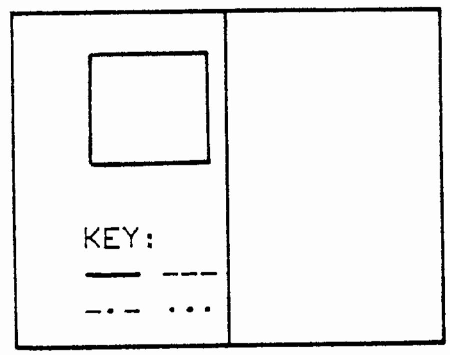
TO RELEASE THE SOFT CLIP LIMITS TO THE HARD CLIP BOUNDARIES.

SYNTAX:

UNCLIP

EXAMPLE:

ROAD KEY
UNCLIP



"UNCLIP STATEMENT", Ch.2, GPT

GR44

THREE TYPES OF LIMITS

TYPE	FUNCTION	HOW TO ESTABLISH
MECHANICAL LIMITS	RESTRICT PEN MOTION ABSOLUTELY	INHERENT IN THE DEVICE
HARD CLIP LIMITS	RESTRICT PEN MOTION UNDER PROGRAM CONTROL	PROGRAM WITH LIMIT STATEMENT
SOFT CLIP LIMITS	RESTRICT PLOTTING OF DATA IN UDUS	PROGRAM WITH CLIP STATEMENT

GR45

SUPPLEMENT

SETTING AND DEFAULTING LIMITS

	HARD CLIP	SOFT CLIP	LOCATE POINTS
PLOTTER IS	DEFAULTS -- TO PHYSICAL	DEFAULTS -- TO PHYSICAL	DEFAULTS -- TO PHYSICAL
LIMIT	DIGITIZES OR PROGRAMS --	DEFAULTS -- TO HARD CLIP	DEFAULTS -- TO HARD CLIP
LOCATE		DEFAULTS -- TO LOCATE	DIGITIZES OR PROGRAMS --
CLIP		DIGITIZES OR PROGRAMS --	
UNCLIP		SETS -- TO HARD CLIP	

GR46

To read this chart, replace the dashes with the column heading.

SUPPLEMENT
SPECIFYING PARAMETERS FOR LIMITS

STATEMENT	PARAMETER UNITS
LIMIT	MILLIMETERS
LOCATE	GDUS
CLIP	CURRENT UNITS

GR47

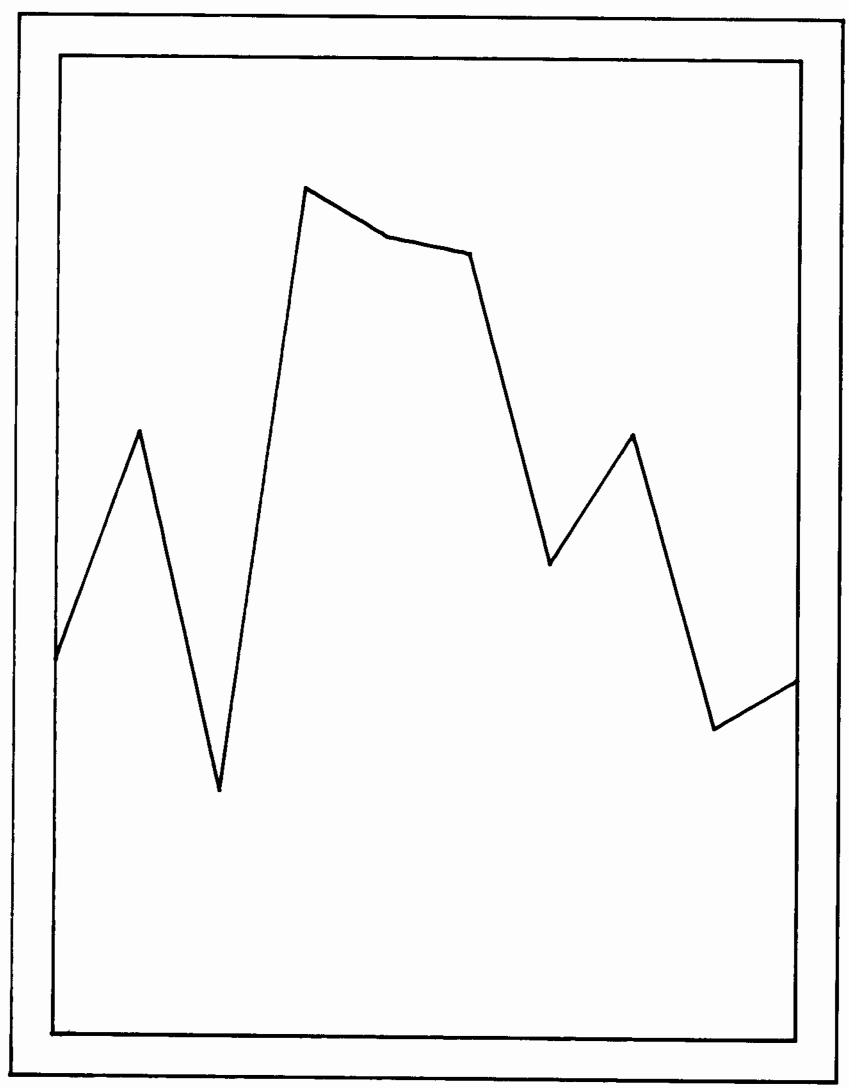
LIMITS EXERCISE 1

SUPPOSE THAT YOU WANT TO PLOT RAINFALL VERSUS YEAR FOR 1969 THROUGH 1978. EVENTUALLY, YOU WANT TO DRAW THIS GRAPH ON 22Ø MM X 28Ø MM, PREGRIDED GRAPH PAPER. FOR THE PRESENT TIME, YOU WANT TO SIMULATE THE TASK BY PLOTTING THE DATA ON THE CRT. YOU HAVE DECIDED TO SET THE HARD CLIP LIMITS TO 11Ø MM X 14Ø MM TO REPRESENT THE PAPER IN YOUR SIMULATION.

THE PREGRIDED GRAPH PAPER HAS ALL 11 MM MARGIN BETWEEN THE GRID AND THE PAPER'S EDGE. NOTE THAT 11 MM IS 5% OF THE LENGTH OF THE SHORT SIDE OF THE PAPER. THEREFORE, 11 MM CORRESPONDS TO 5 GDUS IN THE HARD COPY. YOU HAVE DECIDED TO SET THE SOFT CLIP LIMITS AT 5 TO 95 GDUS IN X AND AT 5 TO 1ØØ/RATIO-5 GDUS IN Y.

DEVELOP THE SIMULATION PROGRAM AS OUTLINED ABOVE. MAKE UP YOUR RAINFALL DATA. USE THE LIMIT STATEMENT TO SET THE HARD CLIP LIMITS AND THE LOCATE STATEMENT TO SET THE SOFT CLIP LIMITS. HOW WOULD YOU MODIFY THE LIMIT AND LOCATE STATEMENTS TO PLOT HARD COPY?

SAMPLE OUTPUT:

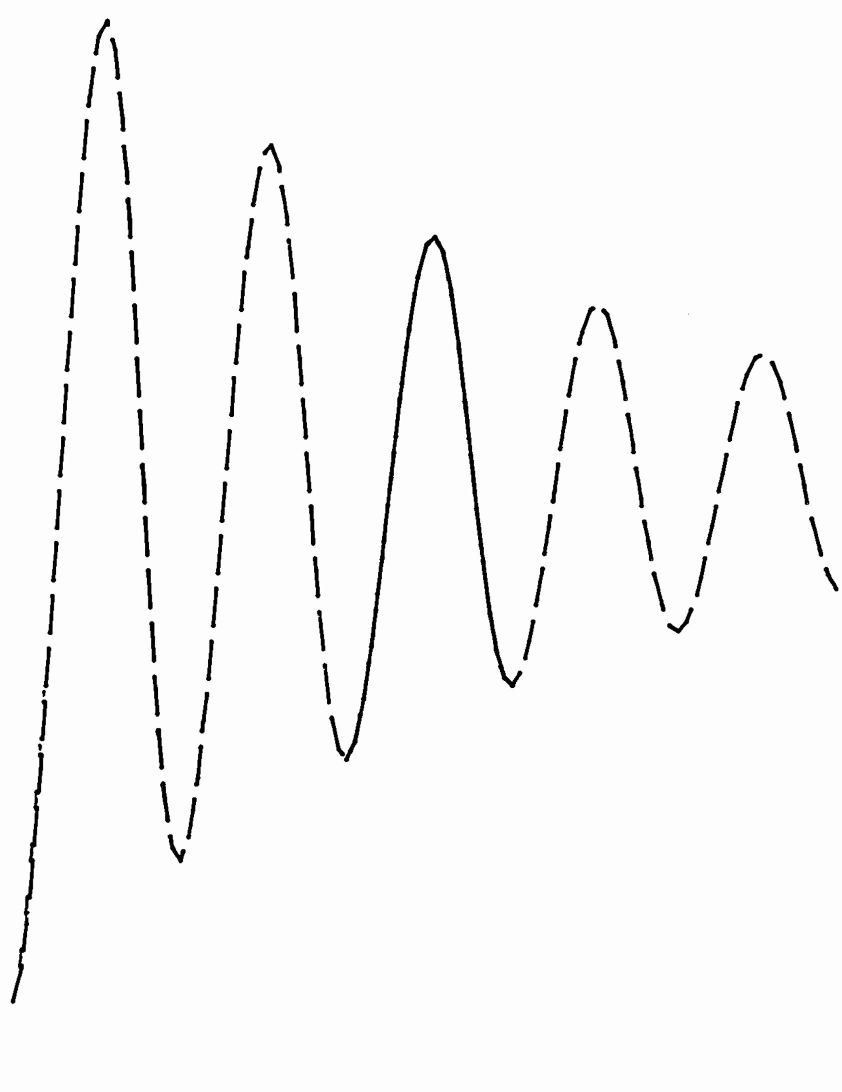


LIMITS EXERCISE 2

DEVELOP A PROGRAM TO PLOT THE FUNCTION Y=EXP(-T/2)*COS(1Ø*T-PI) WHERE T VARIES FROM Ø TO PI IN STEPS OF PI/1ØØ. SCALE FROM Ø TO PI RADIANS IN X AND FROM -1 TO 1 IN Y. DRAW THE FUNCTION USING LINE TYPE 3 (DASHED).

THEN, SET SOFT CLIP LIMITS FROM .4*PI TO .6*PI IN X AND FROM -1 TO 1 IN Y. REPLOT THE FUNCTION USING LINE TYPE 1 (SOLID).

SAMPLE OUTPUT:



AXES

AXES STATEMENT

GRID STATEMENT

FRAME STATEMENT

GR48

AXES STATEMENT

PURPOSE:

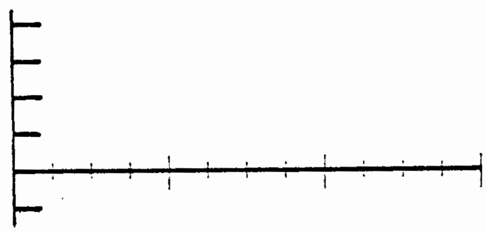
- TO DRAW HORIZONTAL AND VERTICAL AXES.

SYNTAX:

```
AXES [x tic spacing, y tic spacing
     [, x intersect, y intersect        (x origin)
     [, x major count, y major count
     [, major tic length ]]]]
```

EXAMPLE:

```
SCALE 1975,1978,-7.5,22.5
AXES .25,5,1975,0,4,1,15
```



"AXES STATEMENT", Ch.4, GPT

GR49

NB can generate log-log, log-linear plots etc

GRID STATEMENT

PURPOSE:

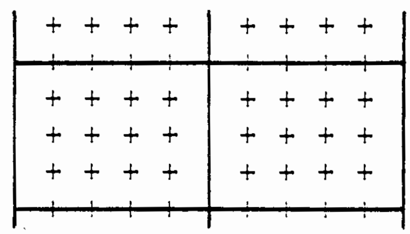
TO DRAW A GRID.

SYNTAX:

```
GRID [x tic spacing, y tic spacing
     [, x intersect, y intersect
     [, x major count, y major count
     [, minor tic length ]]]]
```

EXAMPLE:

```
SCALE 0,10,-.5,5.5
GRID 1,1,0,0,5,4,12
```



"GRID STATEMENT", Ch.4, GPT GR50

FRAME STATEMENT

PURPOSE:

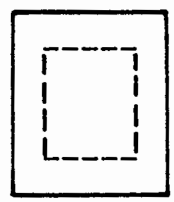
IF CURRENT UNITS ARE GDUS, TO DRAW A BOX AROUND THE HARD CLIP AREA.
IF CURRENT UNITS ARE UDUS, TO DRAW A BOX AROUND THE SOFT CLIP AREA.

SYNTAX:

```
FRAME
```

EXAMPLE:

```
LIMIT 84,108,60,84
CLIP 20,80,20,80
SETGU
FRAME
LINE TYPE 3,90
SETUU
FRAME
```



"FRAME STATEMENT", Ch.4, GPT GR51

AXES EXERCISE 1

INSERT THE AXES AND GRID STATEMENTS ONE AT A TIME INTO THE PROGRAM BELOW. PREDICT THE OUTPUT AND THEN RUN THE PROGRAM TO CHECK YOUR GUESS.

AXES/GRID STATEMENTS:

```
AXES
AXES 2,3
AXES 2,3,-12,-15
AXES 2,3,-12,-15,3,5
AXES 2,3,-12,-15,3,5,4
GRID
GRID 2,3
GRID 2,3,-12,-15
GRID 2,3,-12,-15,3,5
GRID 2,3,-12,-15,3,5,2
```

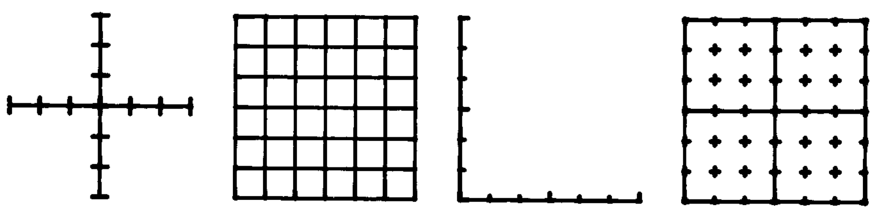
PROGRAM:

```
10   PLOTTER IS "GRAPHICS"
20   GRAPHICS
30   LOCATE 40,60,35,70
40   SCALE -12,12,-15,15
50   ! AXES OR GRID STATEMENT
60   END
```

AXES EXERCISE 2

DEVELOP A PROGRAM THAT PRODUCES PLOTTED OUTPUT SIMILAR TO THAT SHOWN BELOW.

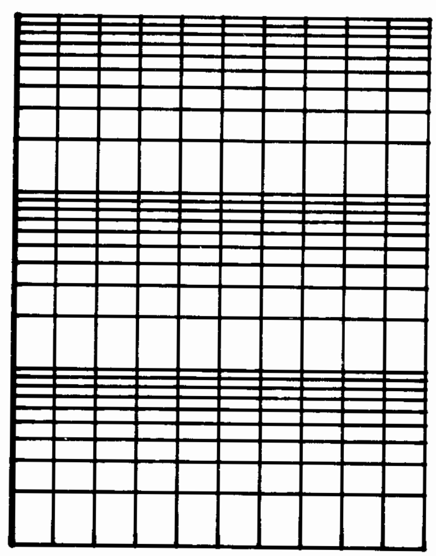
SAMPLE OUTPUT:



AXES EXERCISE 3

DEVELOP A PROGRAM TO DRAW A THREE CYCLE SEMI-LOG GRID. DIVIDE THE LINEAR AXES INTO TEN INTERVALS. A SAMPLE OUTPUT IS SHOWN BELOW.

SAMPLE OUTPUT:



LABELING

LABEL STATEMENT

LABEL USING STATEMENT

LETTER STATEMENT

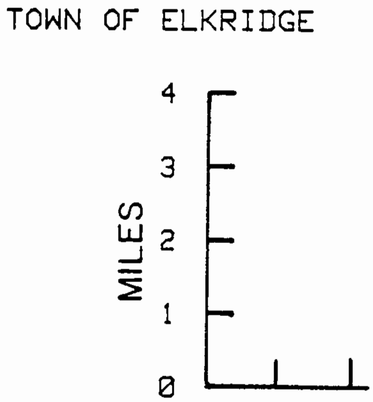
LORG STATEMENT

CSIZE STATEMENT

LDIR STATEMENT

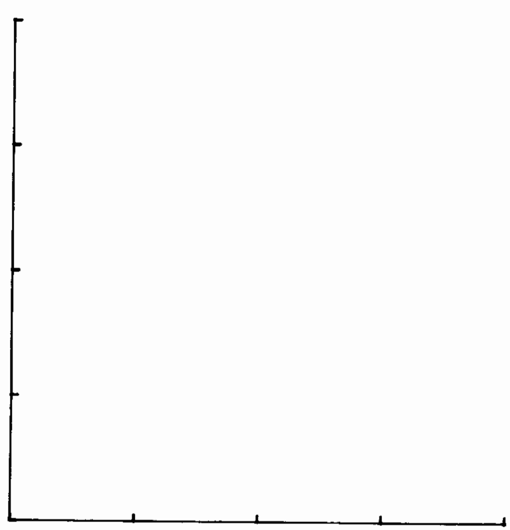
GR52

LABELING EXAMPLE




```
100  ! SETUP
110  PLOTTER IS "GRAPHICS"
120  GRAPHICS
130  !
140  ! MAP SCALE
150  LOCATE 30,110,10,90
160  SHOW 0,4,0,4
170  AXES 1,1
180  !
190  ! TITLE OF MAP
200  SETGU
280  !
290  ! Y AXIS LABELS
300  SETUU
370  END
```

GR53



LABEL STATEMENT

PURPOSE:

TO PROVIDE UNFORMATTED LABELING CAPABILITY.

SYNTAX:

LABEL list

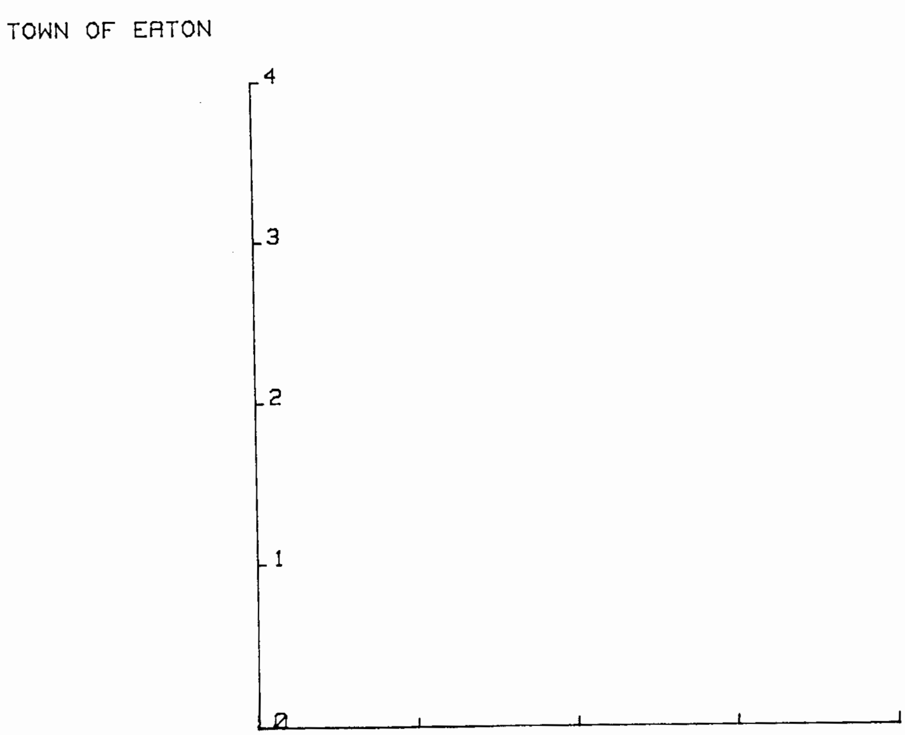
EXAMPLE PROGRAM:

```
          190  ! TITLE OF MAP
ADD       210  MOVE 0,96
ADD       240  LABEL "TOWN OF EATON"

          290  ! Y AXIS LABELS
ADD       330  FOR Y=0 TO 4
ADD       340  MOVE 0,Y
ADD       350  LABEL Y;"   "
ADD       360  NEXT Y
```

note we now set UDUs because we are graphing

"LABEL AND LABEL USING STATEMENTS", Ch.4, GPT GR54

file "LABUSE"



LABEL USING STATEMENT

PURPOSE:

TO PROVIDE FORMATTED LABELING CAPABILITY.

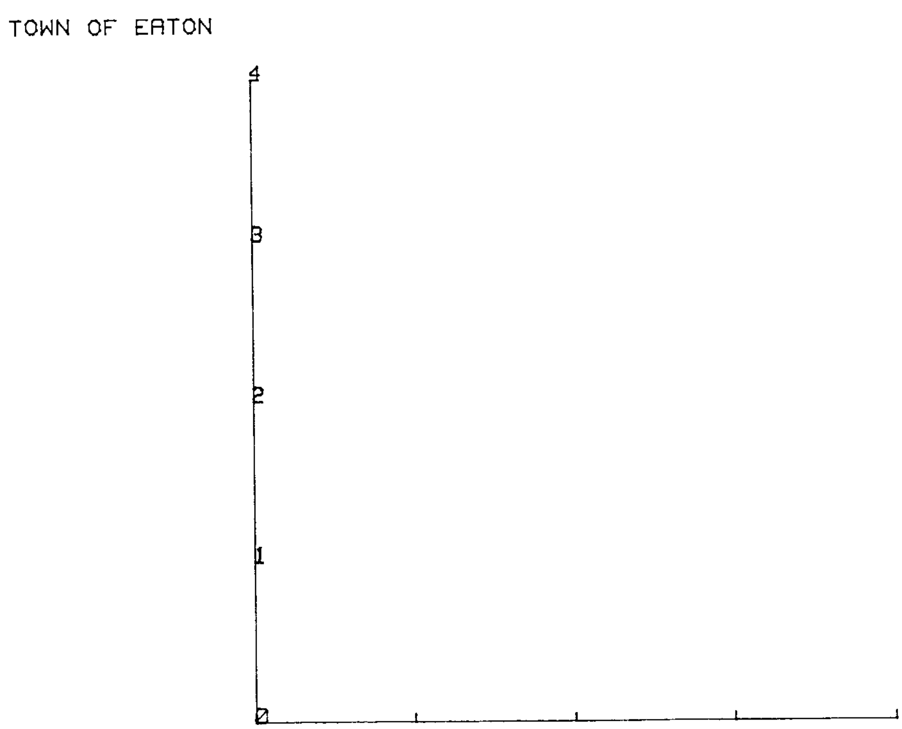
SYNTAX:

LABEL USING specifier; list

EXAMPLE PROGRAM:

```
        190  ! TITLE OF MAP
CHANGE  240  LABEL USING "#,K";"TOWN OF EATON"

        290  ! Y AXIS LABELS
CHANGE  350  LABEL USING "K,2X";Y
```

'LABEL AND LABEL USING STATEMENTS', Ch.4, GPT   GR55



LETTER STATEMENT

NB Control characters are active.

PURPOSE:

TO LOCATE AND TYPE LABELS FROM THE KEYBOARD.

SYNTAX:

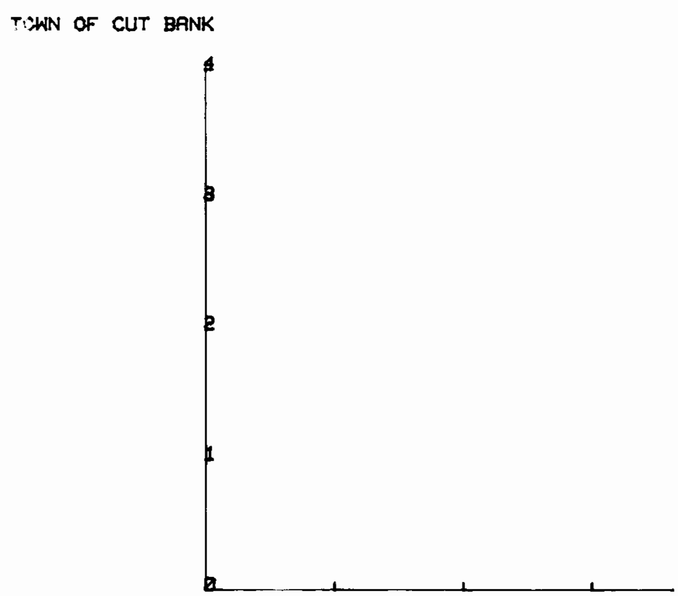
LETTER

EXAMPLE PROGRAM:

```
            190  ! TITLE OF MAP
CHANGE      240  LABEL USING "#,K";"TOWN OF  "
ADD         250  WHERE X,Y
ADD         260  POINTER X,Y,0
ADD         270  LETTER
```

"LETTER STATEMENT", Ch.4, GPT

GR56



LORG STATEMENT

PURPOSE:

TO ESTABLISH WHERE LABELS WILL BE PLACED RELATIVE TO THE CURRENT PEN POSITION.

EXAMPLES:

plot — LINE OF TEXT

TITLE — CENTERED TITLE

9.50 — Y-TIC LABEL

"LORG STATEMENT", Ch.4, GPT

GR57

LORG STATEMENT

SYNTAX:

LORG origin position

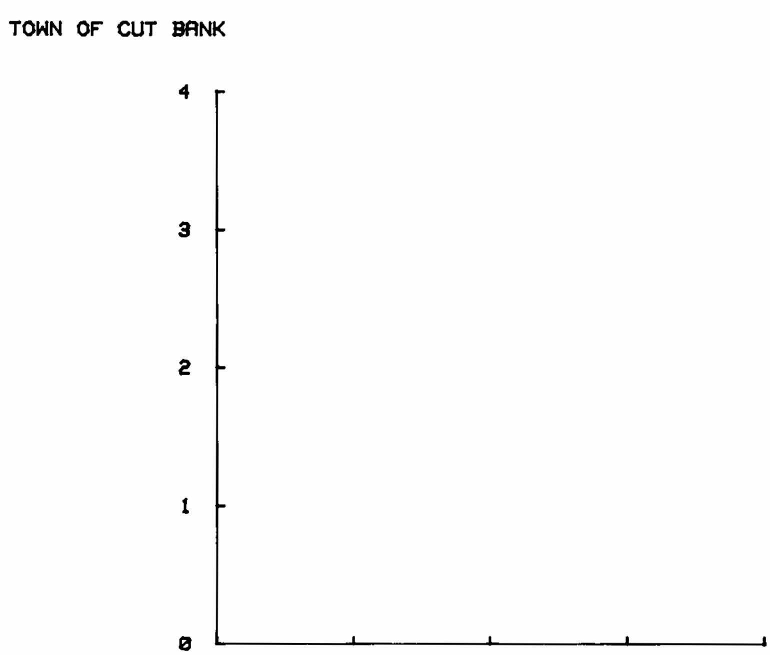
current pen position

```
3---------------6---------------9
|               |               |
2---------------5---------------8
|               |               |
1---------------4---------------7
```

EXAMPLE PROGRAM:

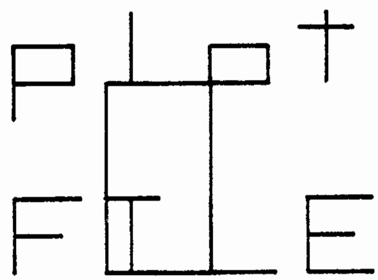
```
        190   ! TITLE OF MAP
CHANGE  210   MOVE 0,100
ADD     230   LORG 3

        290   ! Y AXIS LABELS
ADD     320   LORG 8
```

"LORG STATEMENT", Ch.4, GPT GR58



CSIZE STATEMENT

CHARACTER SPACE:



CHARACTER SPACE ASPECT RATIO:

CHARACTER SPACE WIDTH / CHARACTER SPACE HEIGHT

"CSIZE STATEMENT", Ch.4, GPT

GR59

CSIZE STATEMENT

PURPOSE:

TO SPECIFY THE SIZE AND SHAPE OF THE CHARACTER SPACE.

SYNTAX:

```
CSIZE character space height
     [,character space aspect ratio [,slant]]
```

EXAMPLE PROGRAM:

```
          190  ! TITLE OF MAP
ADD       220  CSIZE 6

          290  ! Y AXIS LABELS
ADD       310  CSIZE 3.3,.6
```

GR60

LDIR STATEMENT

PURPOSE:

TO SPECIFY LABEL DIRECTION.

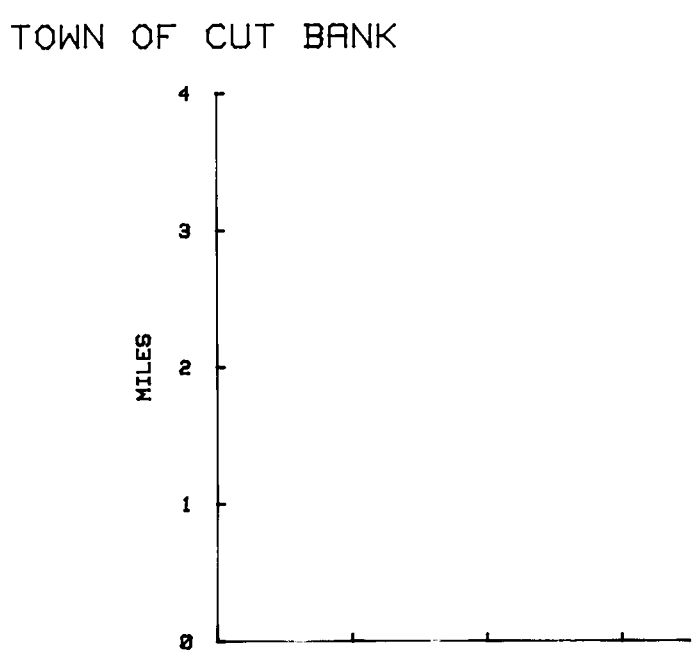
SYNTAX:

```
LDIR angle
LDIR run, rise
```

EXAMPLE PROGRAM:

```
CHANGE  370  !
ADD     380  ! Y AXIS TITLE
ADD     390  DEG
ADD     400  LDIR 90
ADD     410  LORG 4
ADD     420  MOVE -.5,2
ADD     430  LABEL "MILES";
ADD     440  END
```

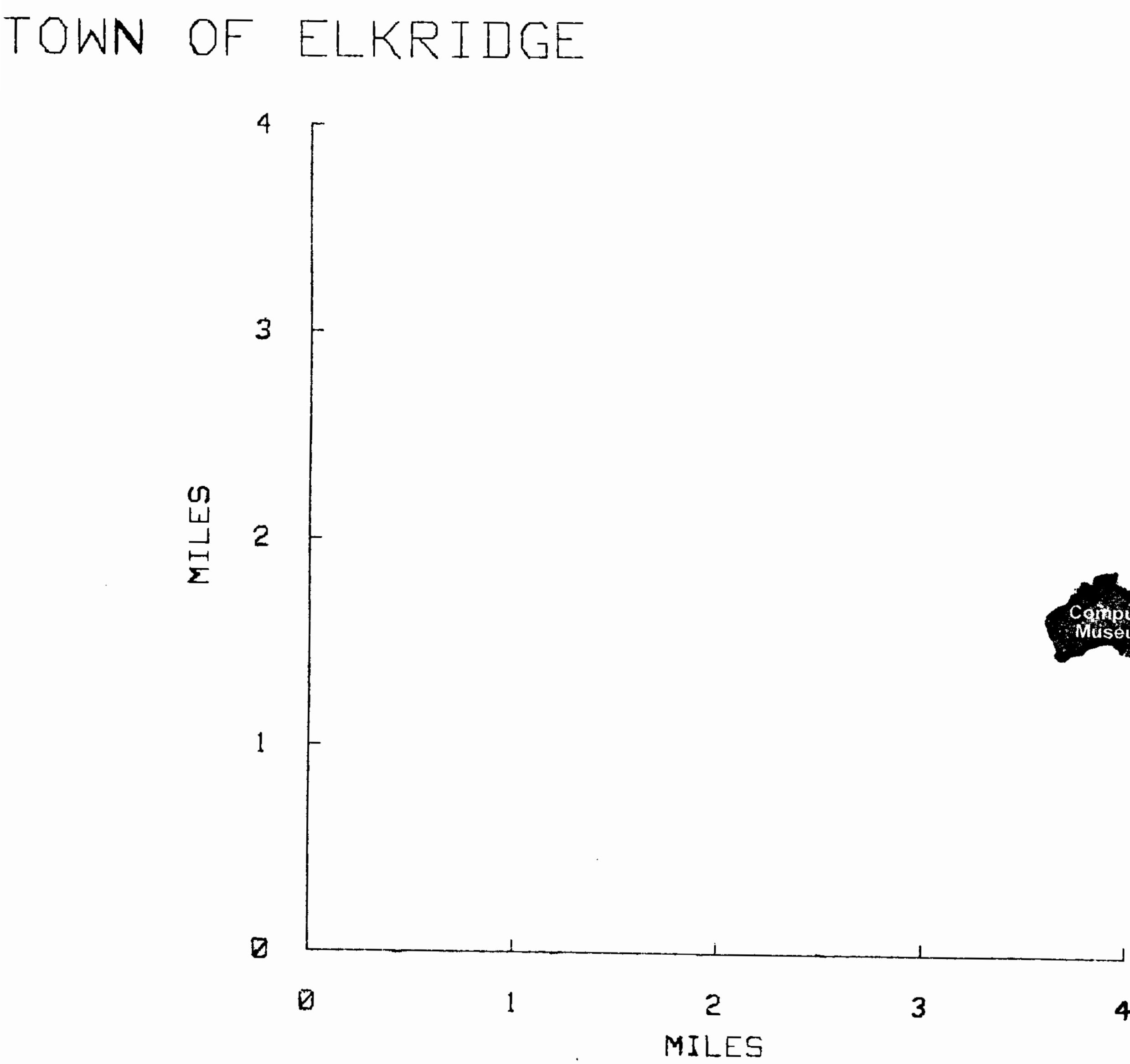
"LDIR STATEMENT", Ch.4, GPT GRS1



LABELING EXERCISE 1

COMPLETE THE EXAMPLE BEGUN IN THE SLIDES BY LABELING AND TITLING THE X AXIS. LABEL THE X AXIS TICS "Ø" THROUGH "4". TITLE THE X AXIS "MILES".

SAMPLE OUTPUT:

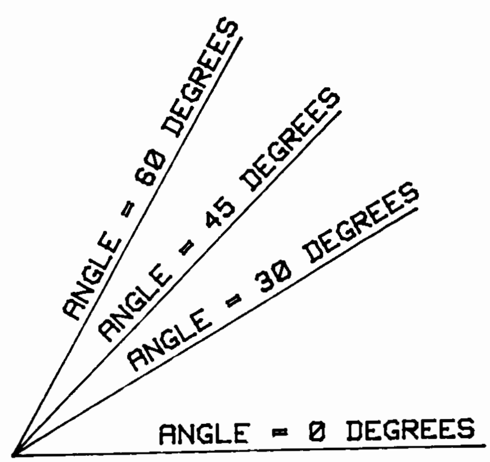


LABELING EXERCISE 2

WRITE A PROGRAM TO PERFORM THE FOLLOWING TASKS.

1. FROM THE KEYBOARD ENTER THE XY COORDINATES OF A POINT. USE THE GDU COORDINATE SYSTEM.
2. DRAW A LINE FROM THE POINT (Ø,Ø).
3. CALCULATE THE ANGLE BETWEEN THE LINE AND THE HORIZONTAL.
4. PARALLEL TO BUT SLIGHTLY OFFSET FROM THE LINE, LABEL THE VALUE OF THE ANGLE.

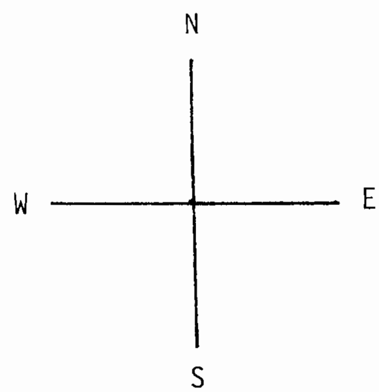
SAMPLE OUTPUT:



LABELING EXERCISE 3

WRITE A PROGRAM THAT REPRODUCES THE OUTPUT SHOWN BELOW.

SAMPLE OUTPUT:



DIGITIZING

POINTER STATEMENT

DIGITIZE STATEMENT

WHERE STATEMENT

CURSOR STATEMENT

GR62

DIGITIZING

WHAT IS DIGITIZING?

DIGITIZING IS THE PROCESS OF MOVING THE CURSOR AND THEN ENTERING THE CURSOR LOCATION.

WHY DIGITIZE?

TO READ DATA OFF A PLOT	DIGITIZE STATEMENT
TO SET HARD CLIP LIMITS	LIMIT STATEMENT
TO SET LOCATE POINTS	LOCATE STATEMENT
TO SET SOFT CLIP LIMITS	CLIP STATEMENT

HOW TO DIGITIZE

EXECUTE THE RELEVANT STATEMENT. MOVE THE CURSOR TO THE DESIRED LOCATION. PRESS THE APPROPRIATE KEY TO ENTER THE COORDINATES.

"INTRODUCTION", Ch.5, GPT GR63

Jo

DIGITIZING CURSORS

TWO KINDS:

FULL-SCREEN CROSS (DEFAULT)
SMALL, FLASHING CROSS

SET BY:

POINTER STATEMENT

USED BY:

DIGITIZE, LIMIT, LOCATE, AND CLIP STATEMENTS

REMARK:

LOGICALLY, THE CURSOR AND THE PEN ARE SEPARATE ENTITIES. ON SOME DEVICES SUCH AS THE 9872A PLOTTER, THE CURSOR IS THE PEN.

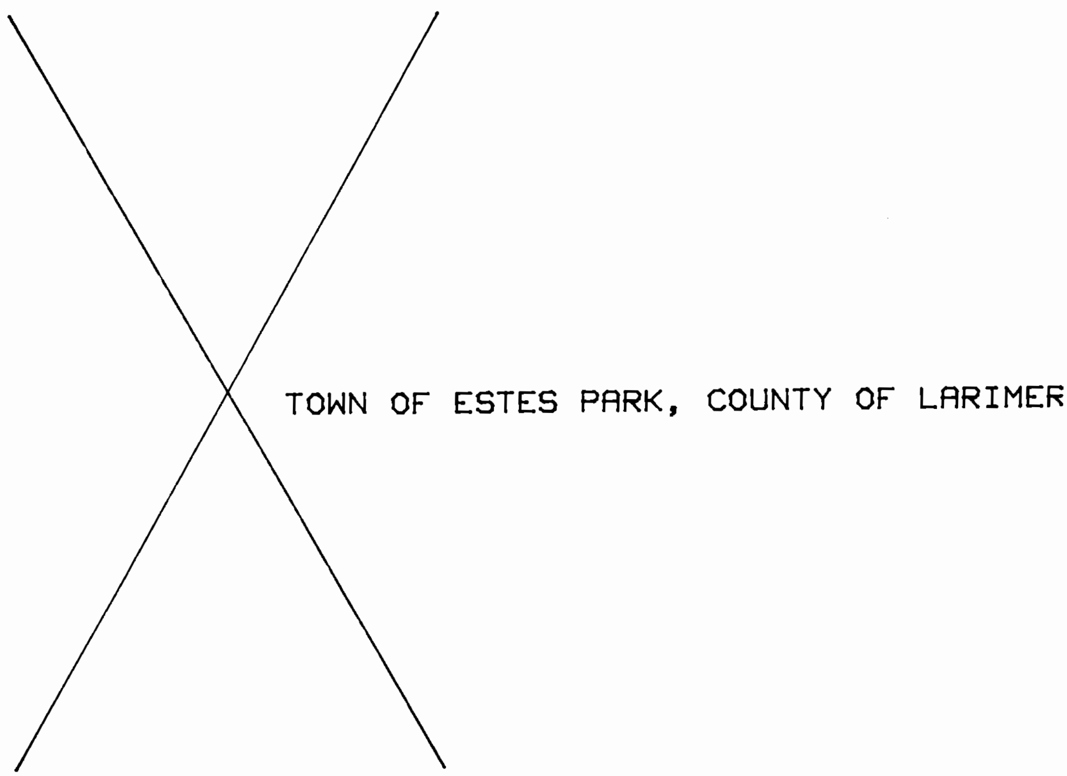
"POINTER STATEMENT", Ch.5, GPT

GR64

DIGITIZING EXERCISE 1

THE PROGRAM FILE NAMED "DIGITI" ON YOUR MASS STORAGE/GRAPHICS TAPE CARTRIDGE CONTAINS ALL THE EXAMPLES IN THIS SECTION. LOAD AND RUN THIS PROGRAM. FOLLOW THE INSTRUCTIONS.

SAMPLE OUTPUT:



DONE

COMPARISON OF CURSOR TO PEN

PEN		CURSOR	
MOVE	MOVES PEN.	POINTER	MOVES CURSOR. SETS CURSOR TYPE.
		some but allows choice of cursor	
WHERE	READS PEN LOCATION.	DIGITIZE	PERMITS OPERATOR TO MOVE CURSOR. THEN READS CURSOR LOCATION.
		CURSOR	READS CURSOR LOCATION.

GR65

POINTER STATEMENT

PURPOSE:

TO MOVE THE CURSOR TO LOCATION (x,y)
TO SELECT THE CURSOR TYPE FOR DIGITIZING
OPERATIONS (LARGE-ODD, SMALL-EVEN)

SYNTAX:

```
POINTER x, y [, cursor type]
```

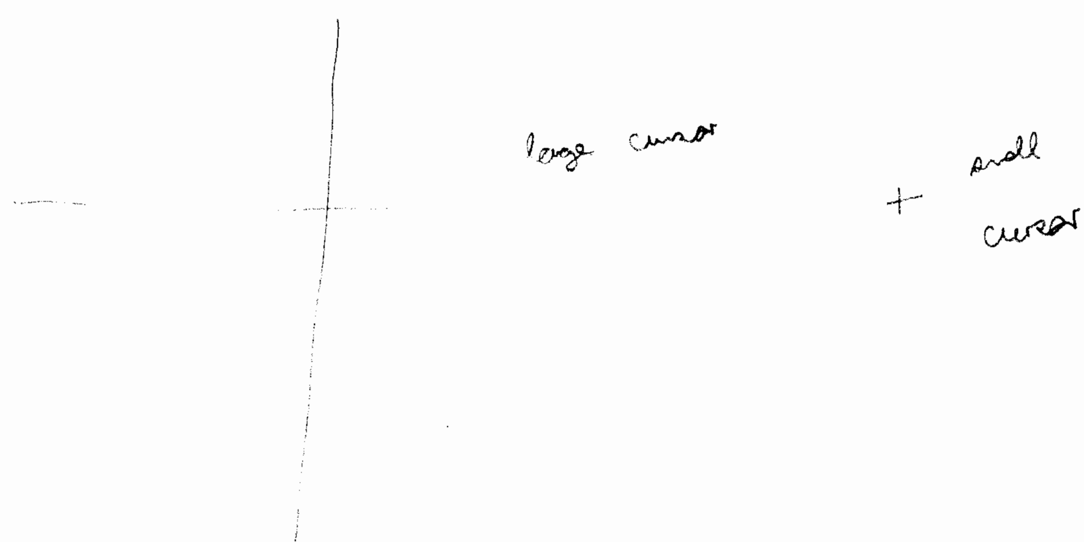
EXAMPLE:

```
POINTER Latitude,Longitude,1
```

EXAMPLE PROGRAM:

```
120  PLOTTER IS "GRAPHICS"
130  GRAPHICS
140  POINTER 30,50,0
270  END
```

"POINTER STATEMENT", Ch.5, GPT GRS6



DIGITIZE STATEMENT

PURPOSE:

TO ENTER THE COORDINATES OF A POINT IN THE PLOTTING SPACE

SYNTAX:

```
DIGITIZE x-variable, y-variable [, pen-variable]
```

EXAMPLE:

```
DIGITIZE Xloc,Yloc,Position$
```

EXAMPLE PROGRAM:

```
ADD      150  DIGITIZE X,Y    recebe a cursor posição
ADD      160  MOVE X,Y
ADD      170  LORG 5
ADD      180  LABEL "X";
```

"DIGITIZE STATEMENT", Ch.5, GPT GR67

X

WHERE STATEMENT

PURPOSE:

TO READ THE PEN LOCATION

SYNTAX:

WHERE x-variable, y-variable [, pen-variable]

EXAMPLE:

WHERE A,B,P$

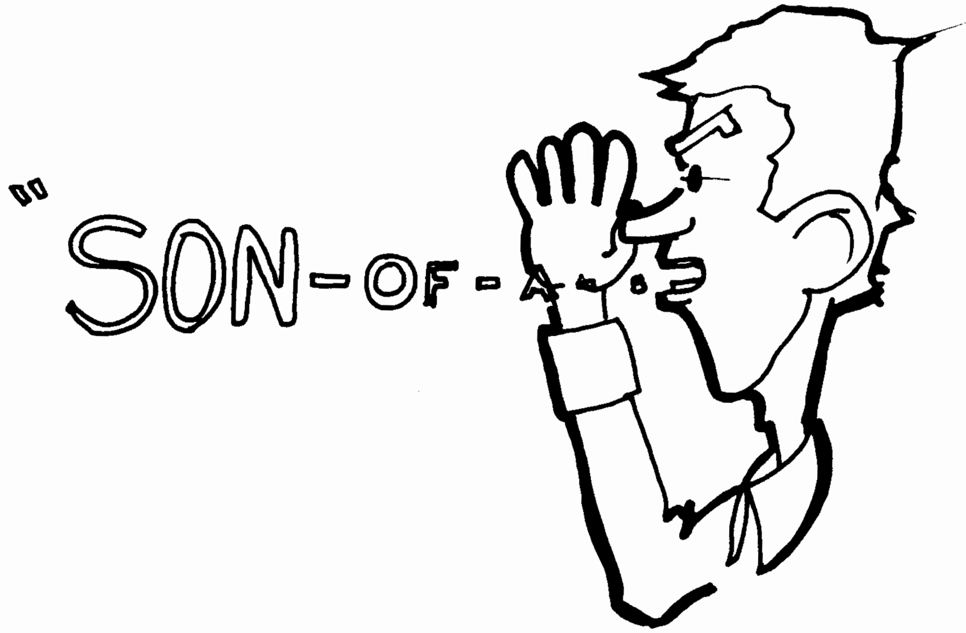
EXAMPLE PROGRAM:

```
ADD     190  LORG 1
ADD     200  LABEL " TOWN OF ";
ADD     210  WHERE X,Y
ADD     220  POINTER X,Y,0
ADD     230  LETTER
```

"WHERE STATEMENT", Ch.5, GPT GRS8

X TOWN OF ESTES PARK

Pen & cursor inseparable when using LETTER



CURSOR STATEMENT

CURSOR STATEMENT

Plotter doesn't change cursor

PURPOSE:

TO READ THE LOCATION OF THE CURSOR

SYNTAX:

CURSOR x-variable, y-variable [, pen-variable]

EXAMPLE:

CURSOR Horizontal,Vertical,Pen$

EXAMPLE PROGRAM:

```
ADD      240  CURSOR X,Y
ADD      250  MOVE X,Y
ADD      260  LABEL ", COUNTY OF LARIMER"
```

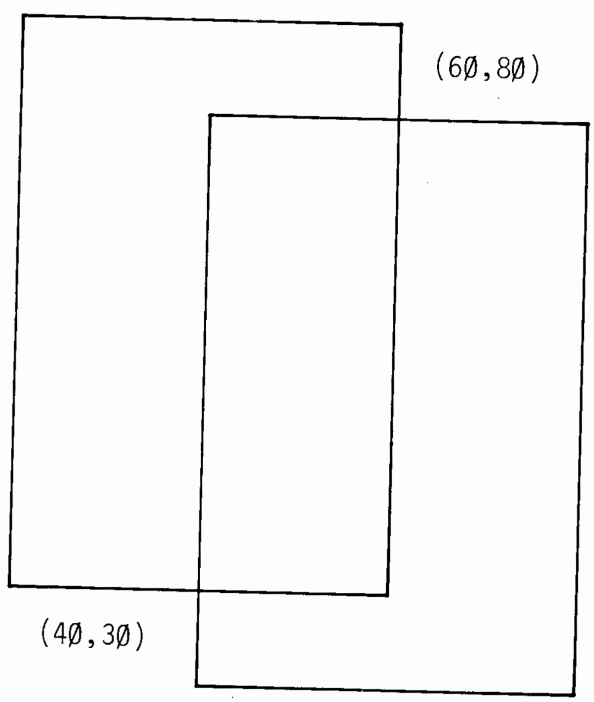
"CURSOR STATEMENT", Ch.5, GPT   GR69

X TOWN OF ESTES PARK, COUNTY OF LARIMER

DIGITIZING EXERCISE 2

DEVELOP A PROGRAM THAT DRAWS TWO OVERLAPPING BOXES. DIGITIZE AND LABEL THE INTERSECTION POINTS.

SAMPLE OUTPUT:



CRT ONLY

GRAPHICS STATEMENT

EXIT GRAPHICS STATEMENT

DUMP GRAPHICS STATEMENT

GSTORE STATEMENT

GLOAD STATEMENT

GR70

CRT MODES

ALPHANUMERIC MODE

STANDARD
ASCII CHARACTERS
160 CHARS X 25 LINES

TO KEY IN DATA
TO DISPLAY PRINTED OUTPUT
TO KEY IN AND LIST PROGRAMS
TO KEY IN COMMANDS
TO LIST SYSTEM MESSAGES

GRAPHICS MODE

OPTIONAL
DOTS
560 X 455 DOTS

TO DIGITIZE DATA
TO DISPLAY PLOTTED OUTPUT

GR71

GRAPHICS STATEMENT

PURPOSE:

TO SELECT GRAPHICS MODE ON THE CRT.

SYNTAX:

GRAPHICS

EXAMPLE PROGRAM:

```
100  PRINTER IS 16
110  PLOTTER IS "GRAPHICS"
120  PRINT PAGE,"ALPHANUMERIC MODE"
130  LABEL "GRAPHICS.";  — wait [illegible]
140  GRAPHICS
150  WAIT 1000  [illegible] see it
160  LABEL "  REPEAT.  GRAPHICS MODE."
190  END
```

"GRAPHICS AND EXIT GRAPHICS STATEMENTS", Ch.6, GPT   GR72

BYE EXIT GRAPHICS STATEMENT

EXIT GRAPHICS STATEMENT

PURPOSE:

TO SELECT ALPHANUMERIC MODE FOR THE CRT.

SYNTAX:

EXIT GRAPHICS

EXAMPLE PROGRAM:

```
ADD     170  WAIT 1000
ADD     180  EXIT GRAPHICS
```

"GRAPHICS AND EXIT GRAPHICS STATEMENTS", Ch.6, GPT   GR73

DUMP GRAPHICS STATEMENT

PURPOSE:

TO COPY PART OR ALL OF PICTURE PLOTTED ON THE CRT TO THE INTERNAL PRINTER.

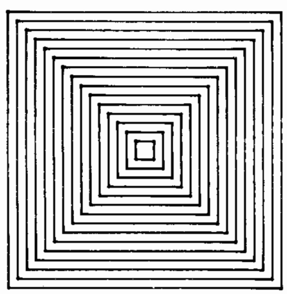
SYNTAX:

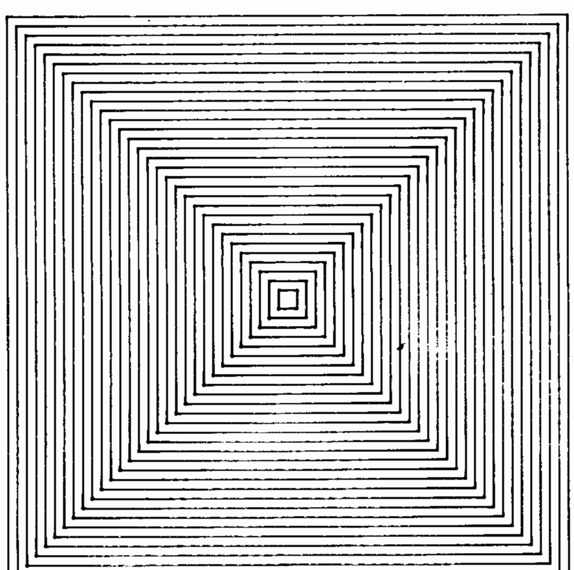
```
DUMP GRAPHICS [lower bound [, upper bound]]
```

EXAMPLE PROGRAM:

```
100  PLOTTER IS "GRAPHICS"
110  GRAPHICS
120  FOR Z=1 TO 30
130  CLIP 60-Z,60+Z,50-Z,50+Z
140  FRAME
150  IF Z MOD 10=0 THEN DUMPGRAPHICS 48-Z,52+Z
160  NEXT Z
170  END
```

'DUMP GRAPHICS STATEMENT', Ch.6, GPT GR74





GSTORE STATEMENT

PURPOSE:

TO TRANSFER A PICTURE RAPIDLY FROM THE CRT TO MAINFRAME STORAGE.

need a mass storage — ROM

SYNTAX:

```
GSTORE integer array name (*)
```

EXAMPLE:

```
INTEGER Form(0:16384)
GSTORE Form(*)
FCREATE "FORM67",130
FPRINT "FORM67",Form(*)
```

"GLOAD AND GSTORE STATEMENTS", Ch.6, GPT

GR75

GLOAD STATEMENT

PURPOSE:

TO TRANSFER A PICTURE RAPIDLY FROM MAINFRAME STORAGE TO THE CRT.

SYNTAX:

```
GLOAD integer array name (*)
```

EXAMPLE:

```
GCLEAR
WAIT 1000
FREAD "FORM67",Form(*)
GLOAD Form(*)
```

"GLOAD AND GSTORE STATEMENTS", Ch.6, GPT GR76

CRT ONLY EXERCISE 1

WITHIN ONE FOR/NEXT LOOP, OUTPUT A TABLE OF SQUARE ROOTS TO THE CRT IN ALPHA MODE, THE INTERNAL PRINTER, AND THE CRT IN GRAPHICS MODE. PLACE THE INTEGERS FROM ONE TO TEN AND THEIR SQUARE ROOTS IN THE TABLE. LOOK AT THE OUTPUT FOR EACH OF THE CRT MODES AND NOTE ANY DIFFERENCES. COPY THE GRAPHICS MODE OUTPUT TO THE INTERNAL PRINTER.

GSTORE & GLOAD

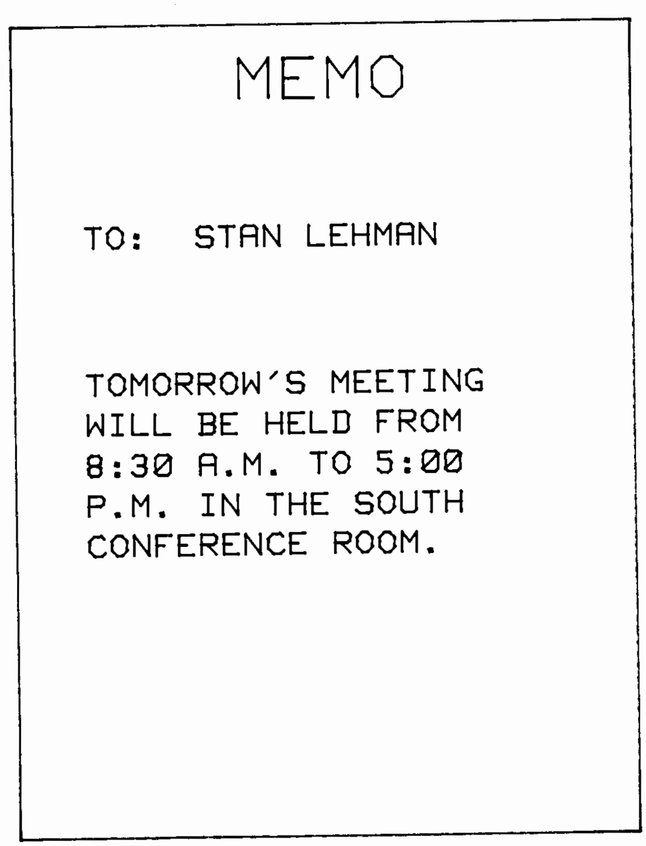
only for mass storage ROM

CRT ONLY EXERCISE 2

THE FILE NAMED "MEMO" ON YOUR MASS STORAGE/GRAPHICS TAPE CARTRIDGE CONTAINS A PROGRAM TO DRAW A MEMO AND INSERT THE NAME OF AN ADDRESSEE. THE MAIN PROGRAM CALLS THE Form SUBROUTINE TO DRAW THE MEMO FOR EACH ADDRESSEE. MODIFY THE MAIN PROGRAM SO THAT THE PROGRAM DRAWS THE FORM ONLY ONCE, STORES IT, AND REUSES THE STORED COPY FOR EACH ADDRESSEE.

NOTE: IF YOUR COMPUTER DOES NOT HAVE A MASS STORAGE OPTION ROM, DO A SCRATCH A AND LOAD THE BINARY PROGRAM "FLOP2" BEFORE ATTEMPTING THIS EXERCISE.

SAMPLE OUTPUT:



MULTIPLE PLOTTERS

PLOTTER IS STATEMENT

PLOTTER IS ON STATEMENT

PLOTTER IS OFF STATEMENT

GCLEAR STATEMENT

GR77

PLOTTER IS STATEMENT

PURPOSE:

TO DESIGNATE AND INITIALIZE THE PLOTTING DEVICE.

SYNTAX:

```
PLOTTER IS [select code [, HP-IB device
            address],] "plotter id string"
```

EXAMPLE:

```
PLOTTER IS 7,5, "9872A"
SCALE -1,1,-1,1
PLOTTER IS 13, "GRAPHICS"
SCALE 0,1,0,1
GRAPHICS
PLOTTER IS 5, "INCREMENTAL"
SCALE 0,10,0,100
```

"PLOTTER IS STATEMENT", Ch.1, GPT

GR78

PLOTTER IS ON STATEMENT

PURPOSE:

TO ACTIVATE THE SPECIFIED PLOTTER AND
TO DEACTIVATE ALL OTHER PLOTTERS.

SYNTAX:

```
PLOTTER select code [, HP-IB
        device address] IS ON
```

EXAMPLE:

```
OVERLAP
FOR Plotter=1 TO 3
IF Plotter=1 THEN PLOTTER 13 IS ON
IF Plotter=2 THEN PLOTTER 7,5 IS ON
IF Plotter=3 THEN PLOTTER 5 IS ON
GOSUB Draw
NEXT Plotter
```

"PLOTTER...IS ON OR IS OFF STATEMENTS", Ch.1, GPT GR79

PLOTTER IS OFF STATEMENT

PURPOSE:

TO DEACTIVATE THE SPECIFIED PLOTTER.

SYNTAX:

```
PLOTTER select code [, HP-IB
        device address] IS OFF
```

EXAMPLE:

```
PLOTTER 7,5 IS ON
        . . .
PLOTTER 7,5 IS OFF


PLOTTER 7,5 IS ON
        . . .
PLOTTER 13 IS ON
```

"PLOTTER...IS ON OR IS OFF STATEMENTS", Ch.1, GPT GR80

GCLEAR STATEMENT

PURPOSE:

TO CLEAR THE PLOTTING SPACE.

SYNTAX:

```
GCLEAR
```

EXAMPLE:

```
PLOTTER IS "GRAPHICS"
GRAPHICS
FRAME
GCLEAR
```

"GCLEAR STATEMENT", Ch.1, GPT

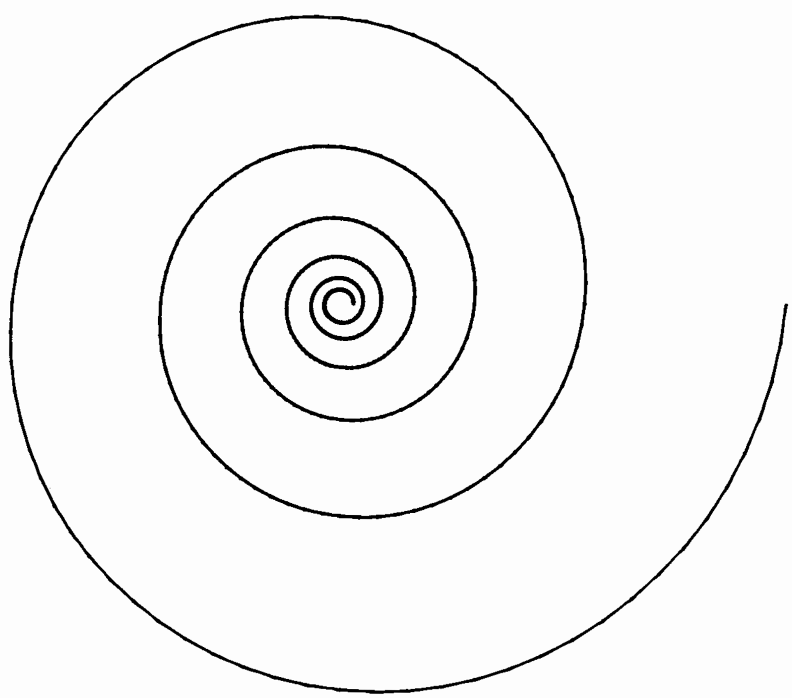
GR81

MULTIPLE PLOTTERS EXERCISE 1

THE PROGRAM BELOW DRAWS SIX CYCLES OF A SPIRAL. AS Angle VARIES FROM A TO A+180, THE PROGRAM DRAWS THE UPPER HALF OF EACH CYCLE. AS Angle VARIES FROM A+180 TO A+360, THE PROGRAM DRAWS THE LOWER HALF OF EACH CYCLE. MODIFY THE PROGRAM TO DRAW ONLY THE LOWER PORTION USING PLOTTER _ IS ON AND PLOTTER _ IS OFF.

```
10   PLOTTER IS "GRAPHICS"            ! INITIALIZE.
20   GRAPHICS                         ! SELECT GRAPHICS MODE.
30   SHOW -35,35,-40,30               ! SCALE.
40   DEG                              ! SELECT DEGREES.
50   FOR Cycle=1 TO 6                 ! SET CYCLE NUMBER.
60   A=360*(Cycle-1)                  ! COMPUTE INITIAL ANGLE.
70   B=360*Cycle                      ! COMPUTE FINAL ANGLE.
80   FOR Angle=A TO B STEP 18         ! DRAW ONE CYCLE.
90   Radius+EXP(Angle/600)            ! USE EXPONENTIAL RADIUS.
100  X=Radius*COS(Angle)              ! COMPUTE X COORDINATE.
110  Y=Radius*SIN(Angle)              ! COMPUTE Y COORDINATE.
120  PLOT X,Y                         ! PLOT SPIRAL.
130  NEXT Angle                       ! CONTINUE CURRENT CYCLE.
140  NEXT Cycle                       ! CONTINUE WITH NEXT CYCLE.
150  PENUP                            ! LIFT PEN WHEN DONE.
160  END
```

SAMPLE OUTPUT FROM PROGRAM ABOVE:

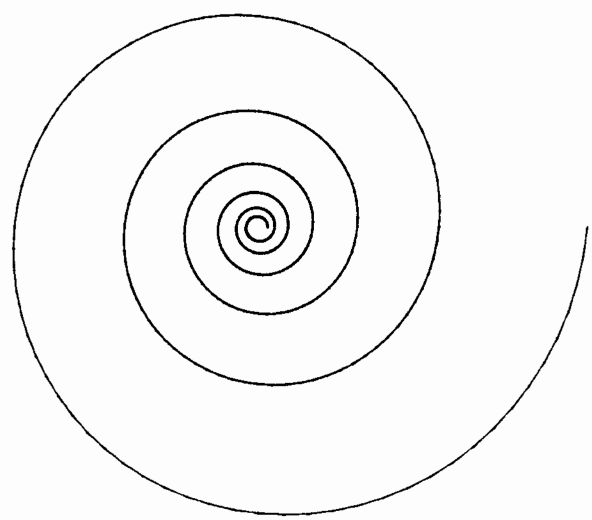


MULTIPLE PLOTTERS EXERCISE 2

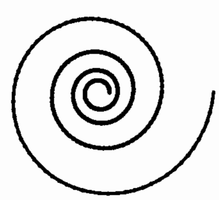
MODIFY THE PROGRAM GIVEN IN THE PREVIOUS EXERCISE TO REPEATEDLY DRAW THE SPIRAL. CLEAR THE SCREEN AND INPUT THE NUMBER OF CYCLES FROM THE KEYBOARD EACH TIME THROUGH THE LOOP. TO CHECK YOUR PROGRAM, TRY INPUTS 6 AND 3.

SAMPLE OUTPUTS:

NUMBER OF CYCLES?
6



NUMBER OF CYCLES?
3



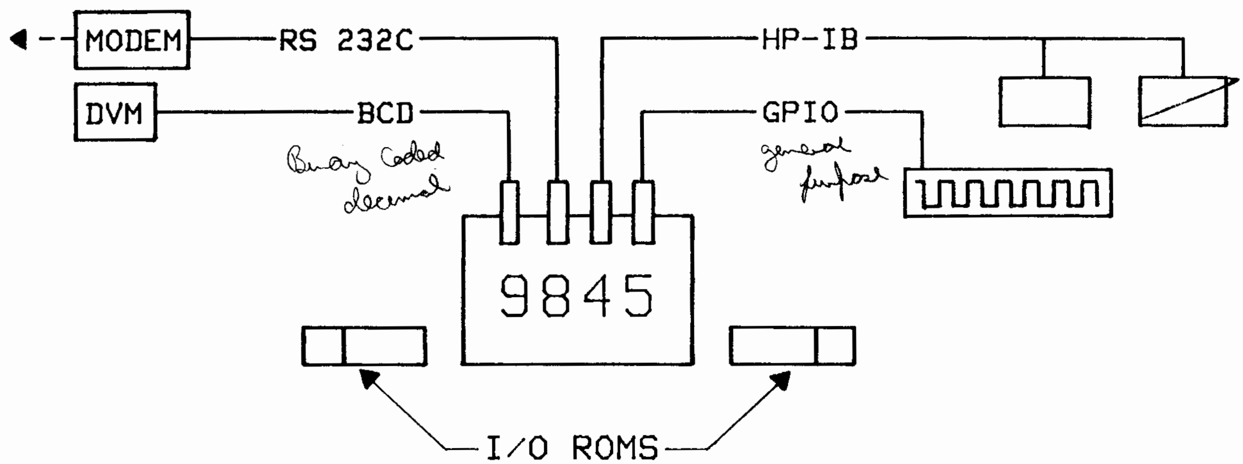
I/O AGENDA

--------> I/O Architecture
--------> Interface Cards
Formatted I/O
HP-IB - the stand IEEE interface
Interrupts

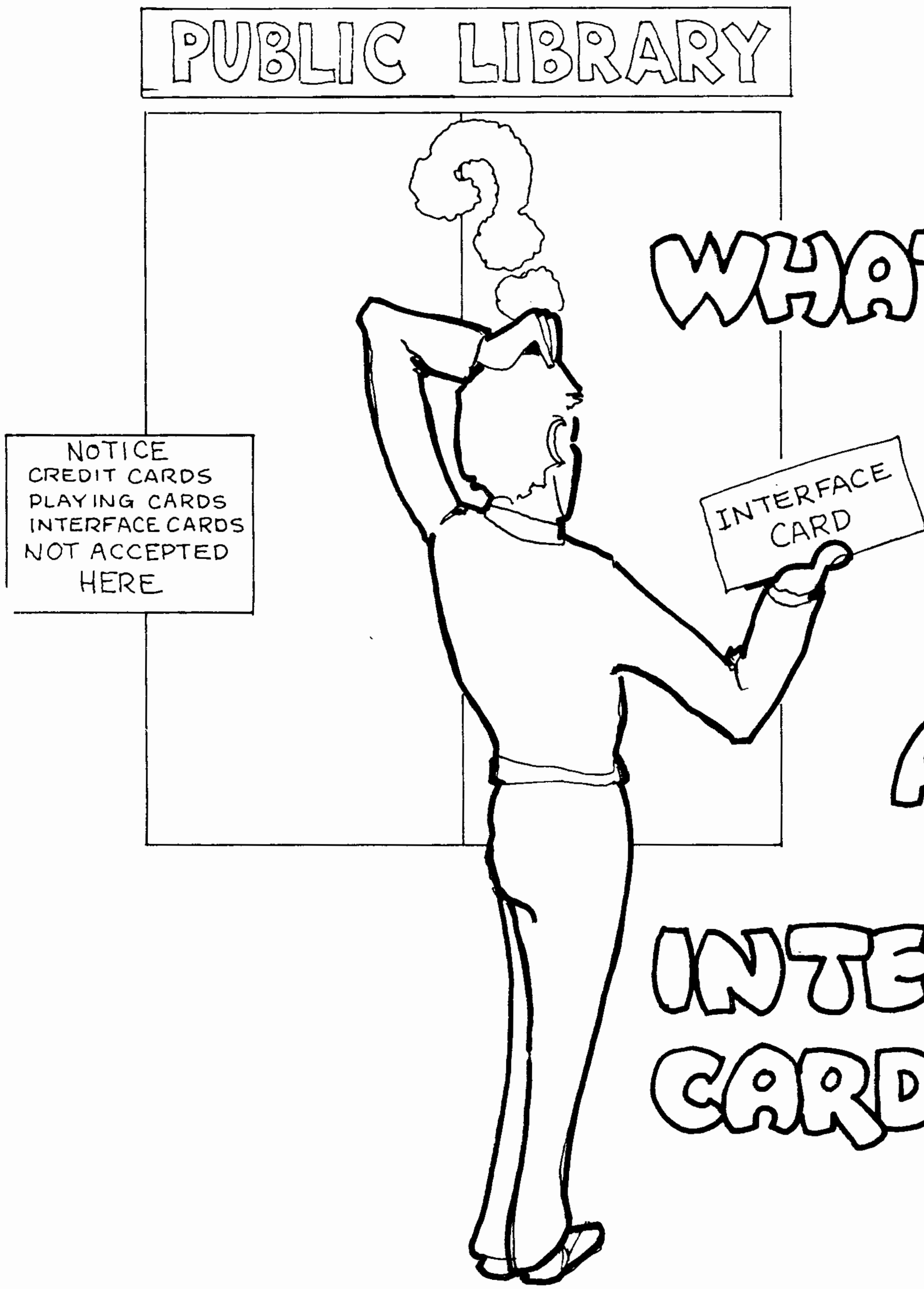
I01

HARDWARE CONFIGURATION

RS - Recommended Standard



I02



WHAT IS AN INTERFACE CARD ?

WHAT IS AN INTERFACE CARD?

PURPOSE:

ELECTRICAL AND MECHANICAL CONNECTION BETWEEN THE SYSTEM 45 AND THE PERIPHERAL

USES:

A LINK WITH THE EXTERNAL WORLD FOR TRANSFER OF INFORMATION

EXAMPLES:

98032A 16-BIT PARALLEL INTERFACE
98033A BCD INTERFACE
98034A HP-IB INTERFACE
98036A SERIAL I/O INTERFACE

I03

98032A 16-BIT PARALLEL

PURPOSE:

8 OR 16 BIT PARALLEL
ASCII OR BINARY CODES

USES:

GENERAL PURPOSE

EXAMPLES:

PRINTERS (HP9866)
TAPE PUNCH
TAPE READER
FLOPPY DISK (HP9885)

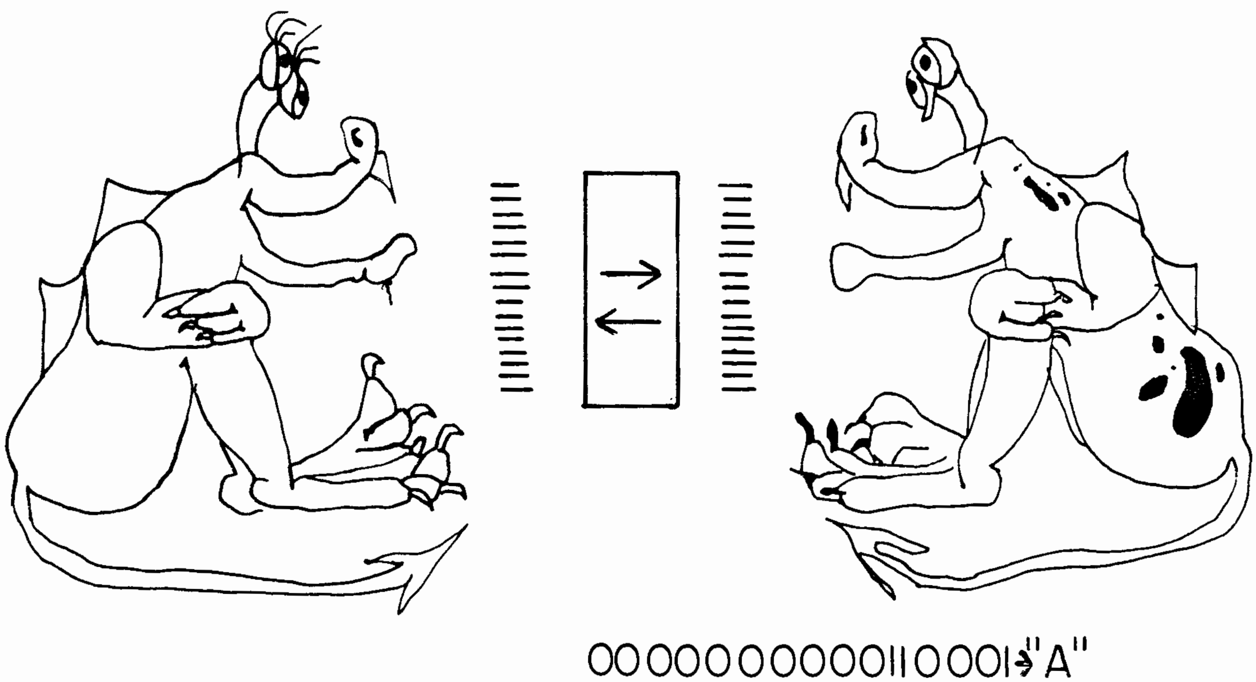
I04

[illegible]

8 bits of wire
in one burst

PRINTER

16-BIT PARALLEL



98Ø33A BCD

PURPOSE:

BINARY CODED DECIMAL
4 BITS PER DIGIT

USES:

BCD DEVICES (INPUT ONLY)

EXAMPLES:

DIGITAL VOLTMETERS
COUNTERS
PANEL METERS

1 2 3 4 factors

IO5

Several Good

Doc an

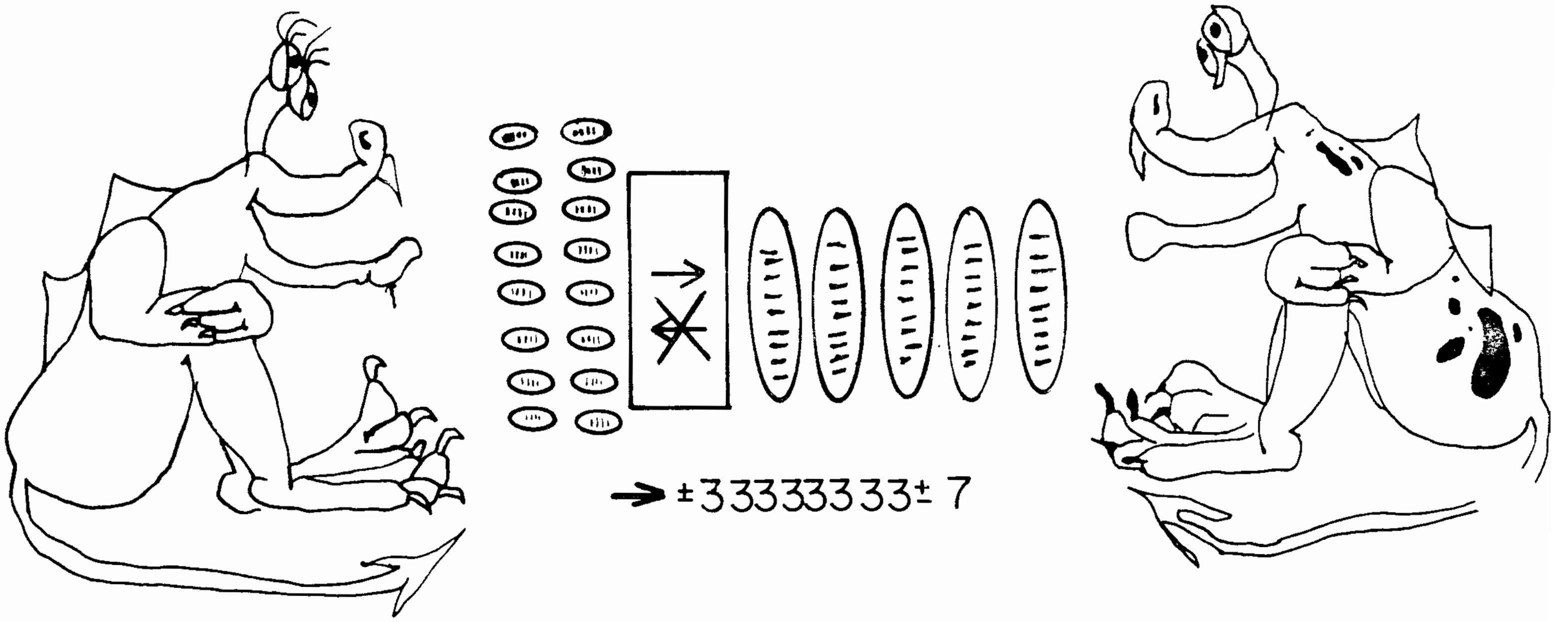
extensive

background

in interfacing

[illegible]

BINARY CODED DECIMAL



98034A HP-IB

PURPOSE:

IEEE 488-1978 STANDARD
8 BIT PARALLEL ASCII

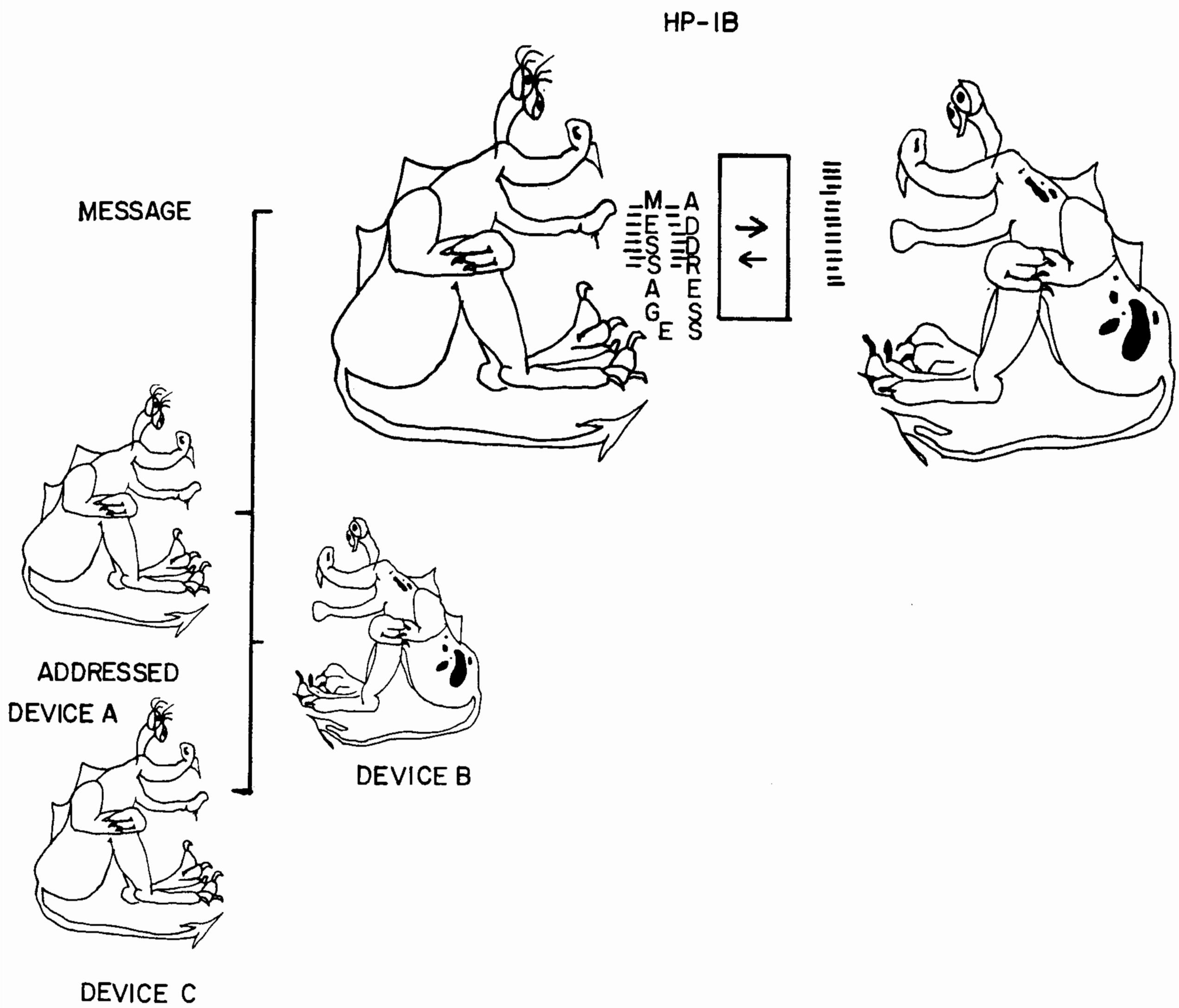
USES:

SIMPLE INTERCONNECTION
GENERAL INSTRUMENTATION

EXAMPLES:

DVM's, Printers, Plotters, Counters,
Signal generators, Synthesizers,
Power supplies...
OVER 100 MANUFACTURERS

IO6


HP-IB
MESSAGE
ADDRESS
ADDRESSED
DEVICE A
DEVICE B
DEVICE C

98036A SERIAL I/O

PURPOSE:

RS 232C STANDARD
BIT SERIAL
SWITCH SELECT BIT RATE

USES:

DATA COMMUNICATIONS

EXAMPLES:

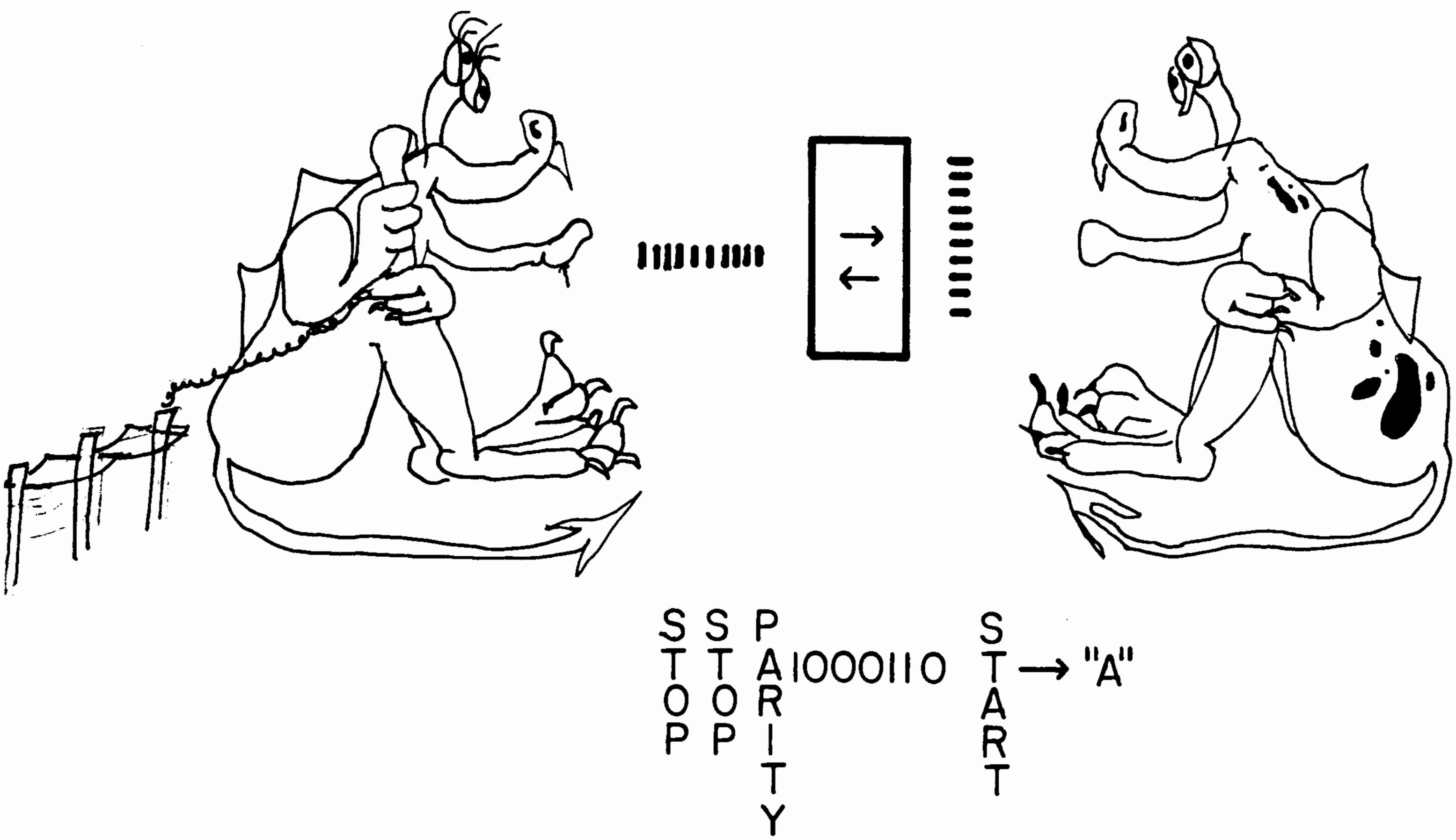
TERMINALS
MODEMS

I07

Peripherals

IEE 488 - 1978 Standard

BIT SERIAL



98036A OPTIONS

DTE: Terminal Equip.	DCE: Computer or Modem Equip.

STANDARD 98036A has female connector and assumes the System 45 is DCE.

OPTION 001 FOR 98036A has male connector to make the System 45 the DTE.

IO8

OTHER INTERFACE CARDS

98035A REAL TIME CLOCK

98040A INCREMENTAL PLOTTER
INTERFACE Calcomp & Houston

98041A MASS STORAGE INTERFACE

I09

SELECT CODES

0 KEYBOARD/PRINTER
1 |
2 |
3 | select codes
. | 1 through 12 — 7 HP-IB
. | reserved for
. | interfacing
. | peripherals
. |
10 |
11 |
12 |
13 (GRAPHICS)
14 (TAPE)
15 (TAPE)
16 DISPLAY/CRT

I010

INTERFACES FOR THE 9845A

Interface	Option	ROM Needed	Comments
98032A 16 Bit Parallel	445	I/O	Unterminated cable 4.5m (15FT) long
98033A BCD	445	I/O	Unterminated cable 4.5m (15FT) long. Input only.
98034A HP-IB	445	I/O	Standard HP-IB cable 4m (13FT) long.
98035A Real Time Clock	045	I/O	
98036A Bit Serial	445	I/O	Standard 98036A with 2m (6FT) cable. Female RS232 Connector. This interface simulates a modem.
	401	I/O	2m (6FT) cable with make RS232 connector. This interface simulates a terminal.
98040A Incremental Plotter	445	Graphics	
98041A	—	Mass Storage	Specialized High Speed Disk Interface.

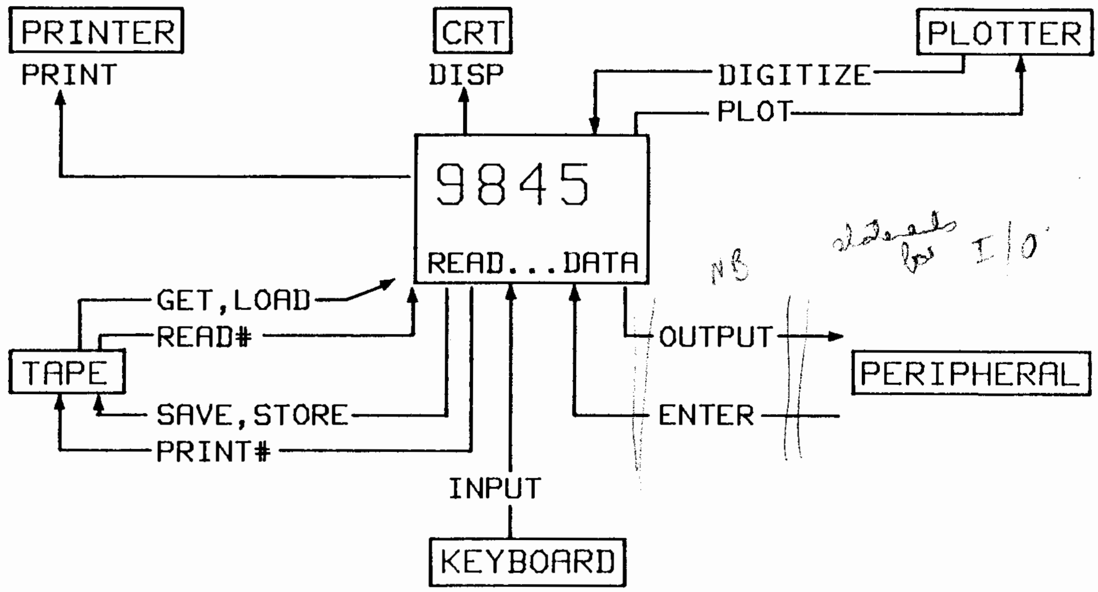
The choice of interface is usually dictated by the peripheral device. If the device has an 8 bit or 16 bit parallel output, the 98032A is the best choice. There have been instances of people trying to design HP-IB type interfaces for bit parallel devices; this is **not** a good idea, as the HP-IB uses a rather complex protocol. Using a 98032 is much more straightforward and will save you and the customer a lot of grief.

BCD devices can be interfaced via the 98033A; remember that the 98033A is an input only interface. Devices that conform to the HP-IB (IEEE standard 488-1975) can be interfaced via the 98034A. Bit serial devices that conform to RS232C or that use a 20ma current loop can be controlled by the 98036A. Remember that there are two versions of the 98036A; make sure that you know whether the 9845 is to function as a modem or as a terminal so that you order the correct option. There is an extensive discussion of all of these interfaces in the I/O Guide; this is an excellent reference for detailed interface information.

PERIPHERALS FOR THE 9845

Peripheral	Peripheral Option	Interface	Interface Option	ROM	Comments
9864A Digitizer	025	98032A	064	I/O	No special 9845 option (or manual) — use 9825 option
9866B Printer	045	98032A	466	—	
9869A Card Reader	045	98032A	469	I/O	
9871A Printer	045	98032A	471	—	
9872A Plotter	045	98034A	—	Graphics	Interface **not** included with option 045
9874A Digitizer	045	98034A	—	Graphics I/O	Interface **not** included with option 045
9875A Cartridge Tape Unit	045	98034A	—	I/O	Interface **not** included with option 045
9878A I/O Expander	045	—	—	—	
9881A Line Printer	045	98032A	481	—	
9883A Paper Tape Reader	045	98032A	483	I/O	
9884A Paper Tape Punch	045	98032A	484	—	
9885A Flexible Disk	045	98032A	485	Mass Storage	
264X Terminal	—	98036A	445	I/O	13232A cable required for terminal (male RS232)
2631A Printer	845	98034A	—	—	Interface **not** included with option 845

INPUTS AND OUTPUTS



I011

I/O AGENDA

I/O Architecture
Interface Cards
--------> Formatted I/O
HP-IB
Interrupts

I012

OUTPUT STATEMENT

PURPOSE:

To transfer information from memory to the peripheral.

SYNTAX:

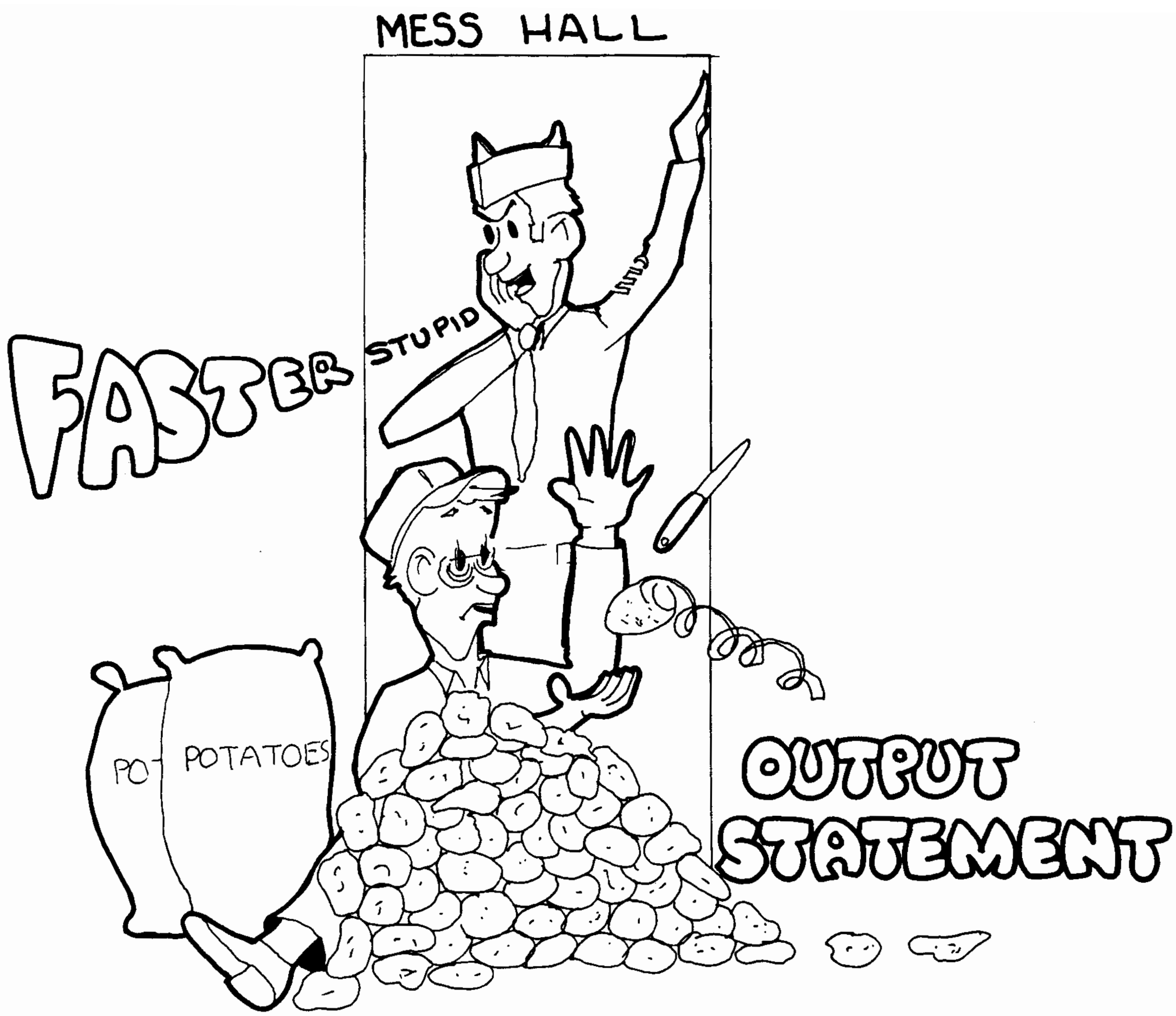
```
OUTPUT <s.c.> [USING<image ref.>];<data>
```

EXAMPLES:

```
OUTPUT 6; A,B,T$
OUTPUT 3 USING "2(D.3D)";X,Y
```

<s.c> select code

I013


MESS HALL
FASTER
STUPID
POT
POTATOES
OUTPUT STATEMENT

ENTER STATEMENT

PURPOSE:

To transfer information from the peripheral into memory.

SYNTAX:

ENTER <s.c.> [USING<image ref.>];<list>

EXAMPLES:

ENTER 3;X,Y
ENTER 9 USING "9N";A

RULES:

The ENTER statement must be terminated by a linefeed unless otherwise specified.

I014

GENERAL OUTPUT IMAGE SPECIFIERS

"text"	TEXT
X	BLANKS
K	COMPACT NUMBERS AND STRINGS
D,Z,*	DIGITS, FILL W/ BLANKS, ZEROS OR *
S,M	SIGN "+" OR BLANK WITH "-"
E	EXPONENT
R,.	RADIX, COMMA OR DECIMAL
C,P	CONDITIONAL COMMAS & DECIMAL
A	CHARACTERS
#,+,-	END OF LINE SEQUENCE
/,@	LINE, PAGE DELIMITERS

I015

GENERAL ENTER IMAGE SPECIFIERS

F,H	FREE FIELD NUMERIC, DECIMAL OR COMMA RADIX
[n]N	n NUMERIC DIGITS, DECIMAL RADIX
[n]G	n NUMERIC DIGITS, COMMA RADIX
T	FREE FIELD STRING, LF DELIMITER
[n]A	n ALPHANUMERIC CHARACTERS
[n]X	SKIP n CHARACTERS
[n]/	SKIP n LINEFEEDS
#,+,%	CANCEL END OF LINE SEQUENCE

IO16

FOR OUTPUT

IMAGE SPECIFIERS IN I/O ROM

B	8-BIT BYTE – write binary representation
W	16-BIT WORD
Y	2 8-BIT BYTES PACKED IN A WORD
L	END OF LINE SEQUENCE SPECIFIED BY THE EOL STATEMENT

I/O ROM ADDS BINARY FORMATS FOR INPUT AND OUTPUT

I017

FORMATTED I/O

OUTPUT IMAGE SPECIFIERS

Image Specifier	Replication Allowed?	Purpose	Comments
X	Yes	Blank	Can go anywhere
“ ”	No	Text	Can go anywhere
D	Yes	Digit	Fill=Blanks
Z	Yes	Digit	Fill=Zeroes
*	Yes	Digit	Fill=Asterisks
S	No	Sign	“+ or “–”
M	No	Sign	“ ” or “–”
E	No	Exponent	Format=ESDD
.	No	Radix	Output “.”
C	No	Comma	Conditional Number Separator
R	No	Radix	Output “,”
P	No	Decimal Point	Conditional Number Separator
A	Yes	Characters	Strings
L	Yes	Carriage Control	Output EOL Sequence
#	No	Carriage Control	Suppress CR-LF
+	No	Carriage Control	Suppress LF
–	No	Carriage Control	Suppress CR
K	No	Compact	Strings or Numerics
,	No	Delimiter	
/	Yes	Delimiter	Output CR-LF
@	No	Delimiter	Output FF
Y	No	Binary	Output Two Bytes
W	No	Binary	Output One 16 Bit Word
B	No	Binary	Output One 8 Bit Byte
(...)	Yes	Replicate	

ENTER IMAGE SPECIFIERS

Image Specifier	Replication Allowed?	Purpose	Comments
F	No	Freefield numeric	Decimal point radix
H	No	Freefield numeric	Comma radix
N	Yes	Digit	Decimal point radix
G	Yes	Digit	Comma radix
T	No	Freefield string	Line-feed delimiter
A	Yes	Character	String
B	No	Byte input	Input One Byte
Y	No	Word input	Input Two Bytes
W	No	Word input	Input One 16 Bit Word
X	Yes	Skip character	String or Numeric
/	Yes	Skip record	Records delimited by “LF”
+	No	Cancel “LF” delimiter	Must be first specifier
#	No	Cancel “LF” and EOI	Must be first specifier
%	No	Cancel EOI delimiter	Must be first specifier
(...)	Yes	Replicate	

PROGRAM EXERCISES USING OUTPUT

```
OUTPUT 16;"Message",1,"and",2

OUTPUT 16 USING "K"; "Message",1,"and",2

OUTPUT 16 USING "KX"; "Message",1,"and",2

OUTPUT 16 USING "K,3DX"; "Message",1,"and",2

OUTPUT 16 USING "8A,3DX"; "Message",1,"and",2
```

I078

SELECT CODE ACTIVE/INACTIVE

debug program

PURPOSE:

The peripheral does not have to be present in order to run the program

EXAMPLES:

```
SELECT CODE 6 INACTIVE
SELECT CODE 12 ACTIVE
```

RULES:

On output, no data is sent to an inactive select code.

On input, input variables from an inactive select code remain unchanged.

I018

WHY CODE CONVERSION?

TODAY'S CRYPTOGRAM:

"TA HATXY XWESE TF FXSEAGXW...
TA EGG FNLND XWESE TF MNYRAANTFE."

TRANSLATION:

"IN UNITY THERE IS STRENGTH...
IN EGG SALAD THERE IS MAYONNAISE."

EXAMPLES:

```
CONVERT 6; "IO","S" TO "R"
CONVERT 6; "IO", A$ TO B$
```

IO for conversion on both
inputs & outputs

I019

DATA COLLECTION

RULES:

Numeric data must be stored in numeric variables in order to be used in computations

Numeric and character data can be stored in string variables

WHAT IF THE DATA IS A MIXTURE OF NUMBERS AND CHARACTER DATA, AND YOU CANNOT PREDICT WHETHER THE NEXT ITEM IS NUMERIC OR CHARACTER DATA?

EXAMPLE:

```
TEMP 60.5, 62.3
ANGLE 61.05, 62.22, 63.00, 64.15
HUMIDITY 71.5
ANGLE 62.33, 65.24, 68.87, 68.95
```

I020

TEMPORARY STORAGE OF ENTERED DATA

TO ENTER DATA INTO VARIABLES:

```
ENTER 6; Comment$,X,Y
```

TO ENTER THE SAME DATA INTO A TEMPORARY AREA:

```
ENTER 6; A$
```

CHECK THE DATA OR STORE THE INPUT FOR LATER USE

TO MOVE THE DATA FROM THIS AREA INTO VARIABLES:

```
ENTER A$; Comment$,X,Y
```

RULES:

The temporary area set aside in memory for I/O with ENTER and OUTPUT statements can be either a string or a numeric array, and is called a VALUE AREA

Computer Museum

I021

could output to tape as a string

input from tape

A$[256]

output on CRT

maybe use CONVERT ?

VALUE AREA EXAMPLE

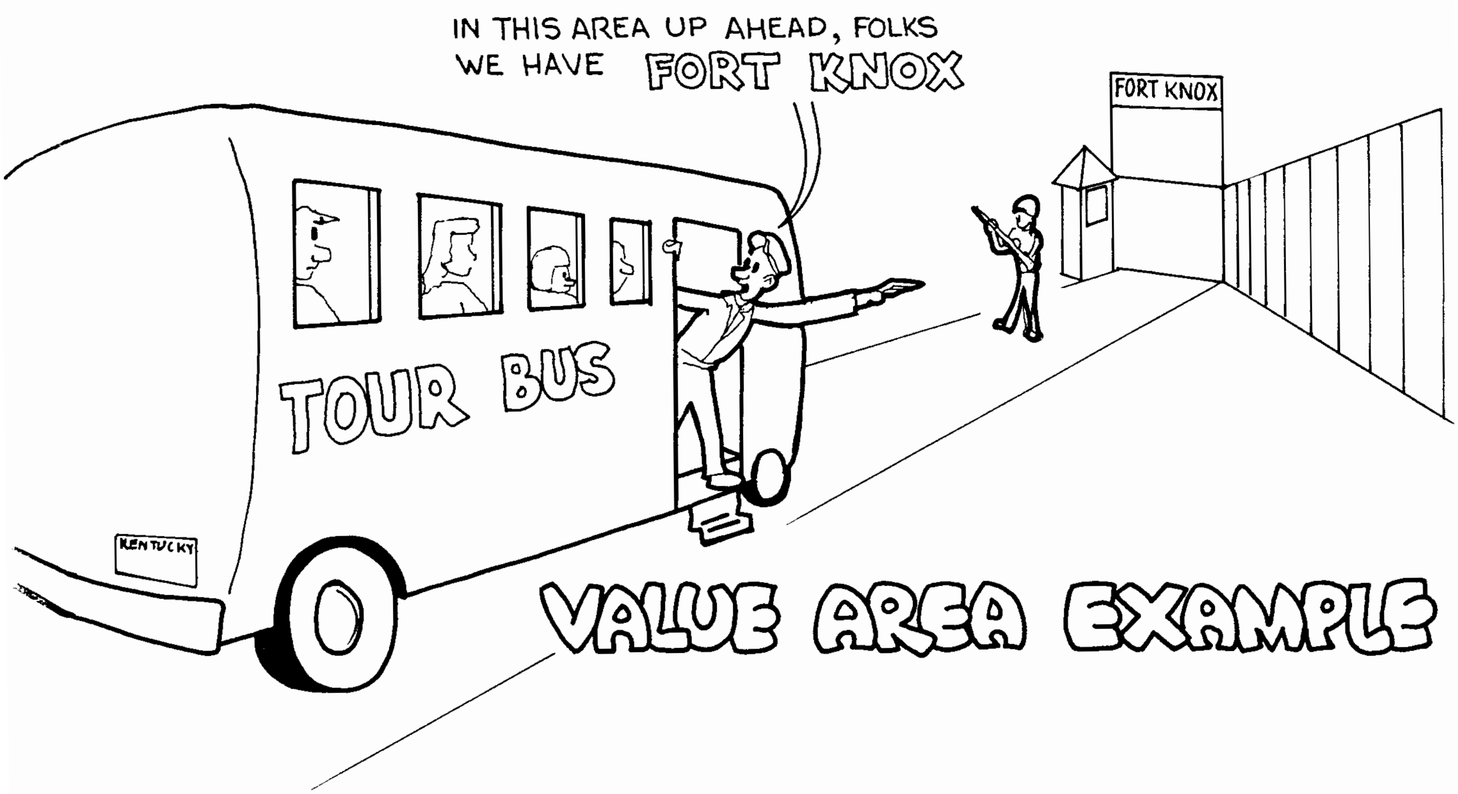
DATA:

```
TEMP 60.5, 62.3
ANGLE 61.05, 62.22, 63.00, 64.15
HUMIDITY 71.5
ANGLE 62.33, 65.24, 68.87, 68.95
ANGLE 64.34, 68.63, 71.82, 70.85
```

PROGRAM:

```
10 DIM A$[80]
20 ENTER 6; A$
30          IF A$[1,1]="H" THEN Humidity
40          IF A$[1,1]="T" THEN Temp
50 Angle: ENTER A$ USING "#,5A,4(F)";Type$,Aa,Ab,Ac,Ad
60          GOSUB Printangles
70          GOTO 20
80 Temp:  ENTER A$ USING "#,4A,2(F)";Type$,Temp1,Temp2
```

I022


IN THIS AREA UP AHEAD, FOLKS
WE HAVE FORT KNOX
FORT KNOX
TOUR BUS
KENTUCKY
VALUE AREA EXAMPLE

VALUE AREA FOR I/O

form of buffering

PURPOSE:

To assign an area in memory where the results of ENTER or OUTPUT I/O operations can be temporarily stored.

FORM:

Replace <s.c.> with
- * Simple String Variable
- * Numeric Array

EXAMPLES:

```
ENTER A$;X,Y,Z
ENTER A$ USING 3F;X,Y,Z
OUTPUT A(1);A,B$,C
```

USES:

FAST I/O; SPECIAL PROCESSING OF DATA LATER

IO23

eg A$ = 14
15

(ie :T14, :T15)

store temporarily on tape?

ENTER STATEMENT EXERCISES

A$ buffer which simulates the peripheral

First, set up a string to simulate peripheral:

```
10 DIM A$[100]
20 OUTPUT A$;5,10,15     ! string will contain the
30 PRINT A$              ! data and cr/lf
40 END
```

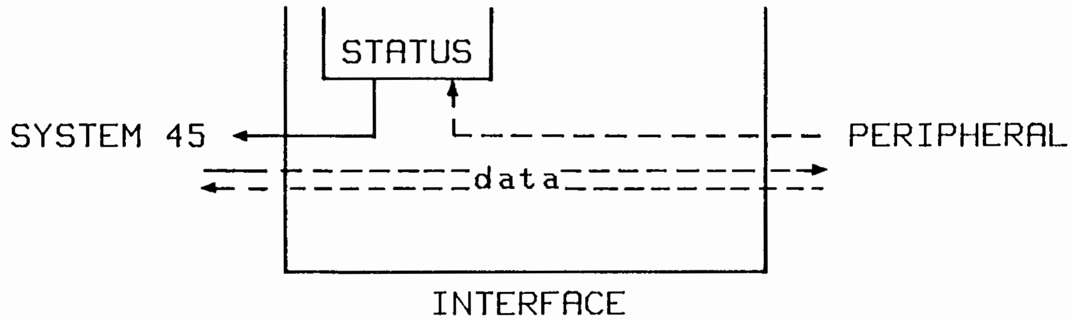
ENTER data from the 'peripheral' into variables then check the contents of those variables:

formatted input

```
ENTER A$; A

ENTER A$ USING "15N"; A,B,C

ENTER A$ USING "25N"; A,B
```

I023.5

INTERFACE STATUS DIAGRAM



I024

INTERFACE STATUS

RULES:

STATUS is an INTERFACE function.

The peripheral is not affected when the System 45 reads status.

The meaning of status codes is device dependent.

EXAMPLES:

Printer out of paper, printer busy, overflow on DVM reading...

I025

I.. AM THE GREATEST

STATUS STATEMENT

STATUS STATEMENT

PURPOSE:

To read the status byte on the interface card.

SYNTAX:

```
STATUS <s.c.>;<variable>
```

EXAMPLE:

```
STATUS 3;A
```

RULES:

Status from select codes 14, 15 and 16 is not meaningful and cannot be read.

I026

EXAMPLE STATUS BYTE

FOURTH STATUS BYTE:

7	6	5	4	3	2	1	0
Service Request	Controller Active	Talker Active	Listener Active	System Controller Set	1	Serial Poll Set	End of Record

BIT OPERATIONS

PURPOSE:

Binary (bit) operations treat the individual bits within a variable.

EXAMPLE:

true is one.

```
STATUS 7;A
IF BIT (A,3) THEN PRINT "SYSTEM";
PRINT "CONTROLLER"
```

SYSTEM or ______
CONTROLLER Controller

USES:

Binary AND	Rotate
Inclusive OR	Shift
Exclusive OR	Last bit
Complement	Bit

convert A to binary to looks at the status of each bit

I027

EXERCISE

1. Read the STATUS byte for the HP-IB interface card.

2. Find out if the System 45 is a System Controller.

INTERPRETING THE STATUS BYTE

```
10 STATUS 7;A                  ! Put status byte in A
20 PRINT "Interface status:"
30 FOR I=7 TO Ø STEP -1        ! Read bits left to right
40 PRINT BIT(A,I);             ! Suppress cr/lf and
50 NEXT I                      ! print bits in a row
60 PRINT
70 END
```

now look at particular interface over page

I028.4

INTERFACE STATUS VALUES

(returned by the STATUS statement)

98032A 16 Bit Parallel

8	7	6	5	4	3	2	1	Ø
			$\overbrace{\hspace{3em}}^{\text{Interface I.D.}}$					
STS	INT	DMA	1	Ø	IID	IOD	STI1	STIØ

STIØ = Status bit Ø
STI1 = Status bit 1
IOD = Invert Output Data
IID = Invert Input Data
DMA = Direct Memory Access Enable
INT = Interrupt Enable
STS = Status (Peripheral)

98033A BCD

8	7	6	5	4	3	2	1	Ø
			$\overbrace{\hspace{3em}}^{\text{Interface I.D.}}$					
STS	Interrupt Enable	Ø	1	Ø	Ø	Ø	Ø	Ø

98034A HPIB

First Value Returned

8	7	6	5	4	3	2	1	Ø
STS	Service Request	Controller Active	Talker Active	Listener Active	System Controller Set	1	Serial Poll Set	End of Record

Bit 8: Is 1 when STS line is true.
Bit 7: Is 1 when the SRQ signal line is true.
Bit 6: Is 1 when the calculator is the active controller.
Bit 5: Is 1 when the calculator is the active talker.
Bit 4: Is 1 when the calculator is an active listener.
Bit 3: Is 1 when the calculator is the active system controller.
Bit 2: Is always 1.
Bit 1: Is 1 when a serial poll is in process.
Bit Ø: Is 1 when the EOI (end of record) line is true.

98034 HPIB

Second Value Returned

7	6	5	4	3	2	1	Ø
Ø	Ø	Ø	Ø	Ø	Device Clear	Ø	Error

Bit Ø: Is 1 when error detected.
Bit 2: Is 1 when Device Clear received.

Third Value Returned

7	6	5	4	3	2	1	Ø
1	1	Ø	(MSB)		HP-IB Address		(LSB)

Fourth Value Returned

7	6	5	4	3	2	1	Ø
EOI	REN	SRQ	ATN	IFC	NDAC	NRFD	DAV

Logical 1 indicates corresponding signal line is true.

98035A Real Time Clock

8	7	6	5	4	3	2	1	Ø
			Interface I.D.					
STS	Current Interrupt Status	Ø	1	Ø	Ø	Ø	Interrupt Flag	Error Flag

Bit 8: Is 1 when the interface is operational.
Bit 7: Is 1 when interrupts are enabled by CARD ENABLE
Bit 1: Is 1 when interrupts are enabled; not reset by servicing an interrupt request.
Bit Ø: Is 1 when an error condition exists.

98036A Bit Serial

8	7	6	5	4	3	2	1	Ø
STS	Interface Interrupt Enable Status	Ø	Interface I.D. Ø	Interface I.D. 1	Ø	Ø	Control Status 2 Receiver Mode	Control Status 1 Transmitter Mode

Bit 8: 1 when interface operational.
Bit 7: 1 when interface enabled for interrupt.
Bit 1: 1 when interface enabled for interrupt on received character.
Bit Ø: 1 when interface enabled for interrupt when ready to transmit character.

98040A Incremental Plotter

8	7	6	5	4	3	2	1	Ø
STS	INT	DMA	ID 1	ID Ø	Ø	Ø	1	1

Bit 8: 1 when interface operational.
Bit 7: 1 when interface enabled for interrupt
Bit 6: 1 when interface enabled for DMA.

RS232-C

HP2640B CRT TERMINAL PROGRAMS

linking up terminals

To read strings from the terminal:

```
10 DIM A$[100]
20 OUTPUT 10 USING "#,B,K";27,"E"
30 ENTER 10; A$
40 PRINT A$
50 GOTO 30
```

To send strings to the terminal: escape code?

```
10 DIM A$[100]
20 OUTPUT 10 USING "#,B,K";27,"E"
30 INPUT "Message to send:",A$
40 OUTPUT 10; A$
50 GOTO 30
```


output to printer & can check STATUS

I028.6

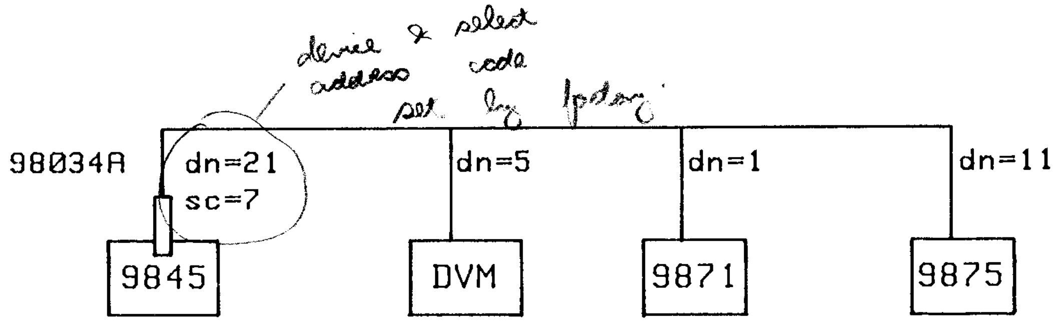
I/O AGENDA

I/O Architecture
Interface Cards
Formatted I/O
--------> HP-IB
Interrupts

I029

dn ≙ device { number / address }

HP-IB DIAGRAM



I030

Interface
Instrument
Bus

HP-IB ADDRESSING

PURPOSE:

To address peripherals connected to the HP-IB Interface.

FORM:

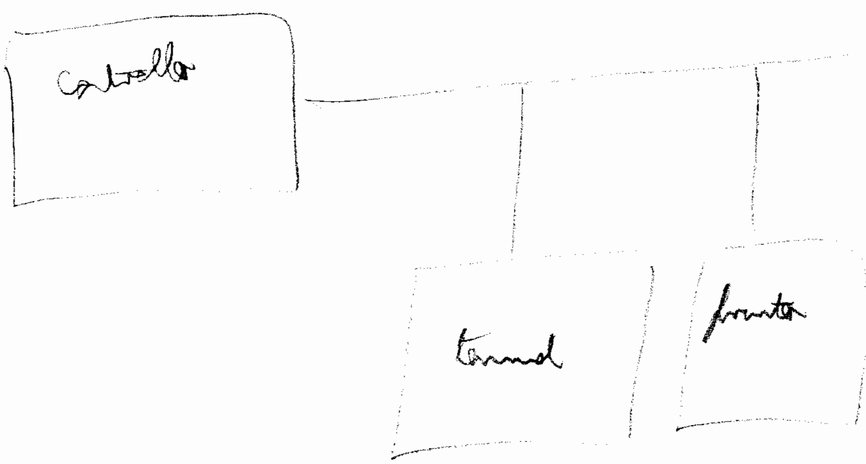
Replace <s.c.> with <s.c.><d.n.>

EXAMPLES:

```
OUTPUT 701;A$
Dvm=705
ENTER Dvm;X,Y
```

ONLY one
system controller allowed

I031



HP-IB DEVICE NUMBERS dn

RULES:

Each device on the HP-IB has its own device number.

There are switches on each device so you can assign a unique device number to each peripheral.

I032

HP-IB SYSTEM CONTROLLER

RULES:

Each bus system can have only one System Controller, although it can have several devices CAPABLE of being controllers.

NB

A switch on the HP-IB Interface Card can be manually set to make the 9845 the System Controller.

The System Controller can take priority control of the bus when it is not the active controller.

I033

HP-IB ACTIVE CONTROLLER

u devices
must eith
talk or
listen

RULES:

The current Active Controller can pass control to another device.

There can be only one Active Controller in the bus system at any time.

The Active Controller addresses devices as talkers and listeners.

I034

HP-IB TALKERS AND LISTENERS

RULES:

A device is a TALKER if it can send information on the bus when it has been addressed.

A device is a LISTENER if it can receive information from the bus when it has been addressed.

EXAMPLES:

TALKERS	LISTENERS	TALKERS/ LISTENERS
-Signal generators	-Printers	-Plotter/Digitizers
-Voltmeters	-Plotters	-Mass storage media
-Digitizers	-Tape punches	-Programmable DVM's

I035

MULTIPLE LISTENERS

PURPOSE:

To send the same information to multiple devices simultaneously.

FORM:

Replace <s.c.> with <s.c.>,<s.c.>[,<s.c.>...]

EXAMPLE:

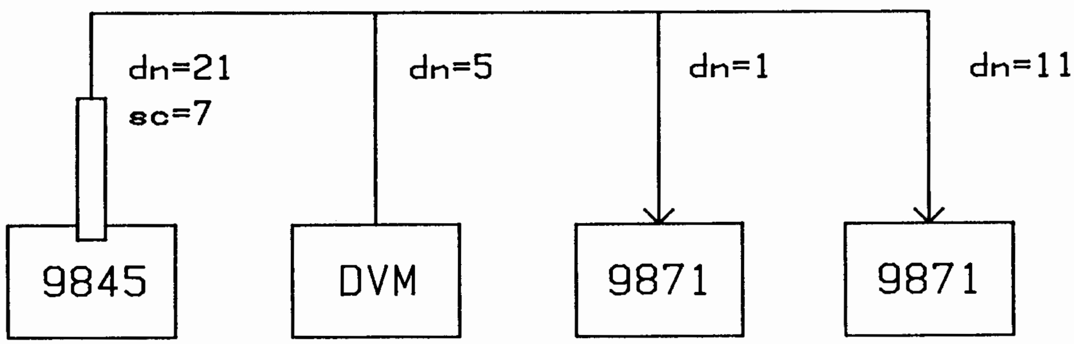
```
OUTPUT 701,711;A$
```

USES:

2 OR 3 Printers producing the same outputs.

I036

MULTIPLE LISTENERS DIAGRAM



AUTOMATIC HP-IB CONTROL

EXAMPLES:

```
ENTER 704;X,Y
OUTPUT 711;A$,B$
```

RULES:

The System 45 as System Controller automatically sends all of the necessary information to the bus to declare the current talker and listener.

Automatic Control is the easiest way to use the HP-IB.

When the System 45 is NOT the only controller in the system, special statements are needed.

DEVICE DEPENDENT SETTINGS

PURPOSE:

Some devices on the bus may need special set-up instructions before they can participate in the bus system.

EXAMPLE:

Set DVM range before taking readings.

```
OUTPUT 711 USING "#,K";"R1"
ENTER  711;X,Y
```

RULES:

The user must set the switches on the front panel of the instrument or write OUTPUT statements to get the instrument ready.

I039

HP9874A DIGITIZER PROGRAM

To digitize points as fast as possible and display the coordinates on the digitizer display:

```
10 OUTPUT 706; "IN"                        !Initialize the 9874
20 Digitize: OUTPUT 706; "OC"              !Outputs coordinates
30 ENTER 706;Xcoord,Ycoord,Pen,Annot       !Enter a point
40 IMAGE "LB",DDDDDD,",",DDDDDD            !Format for display
50 OUTPUT 706 USING 40;Xcoord,Ycoord !Write to display
60 GOTO Digitize
70 END
```

FRAME   default to 2-letter

REWIND   memonics

etc

I038.4

HP9875A TAPE DRIVE PROGRAM

To store string data in file 2 on the tape:

```
10 DIM A$[100]
20 OUTPUT 704; "RW"          ! Command to rewind tape
30 OUTPUT 704; "MF5,4"       ! Mark 5 files of 1024 bytes
40 INPUT "Enter data:",A$ ! Input string from keyboard
50 OUTPUT 704; "SF2"         ! Command to store in file 2
60 OUTPUT 704; A$            ! Sends data to the 9875A
70 END
```

2 letter mnemonics are only understood on HP-IB

I038.6

HP-IB INTERFACE STATUS

EXAMPLES:

```
STATUS 7;A
STATUS 7;A,B,C,D
```

RULE:

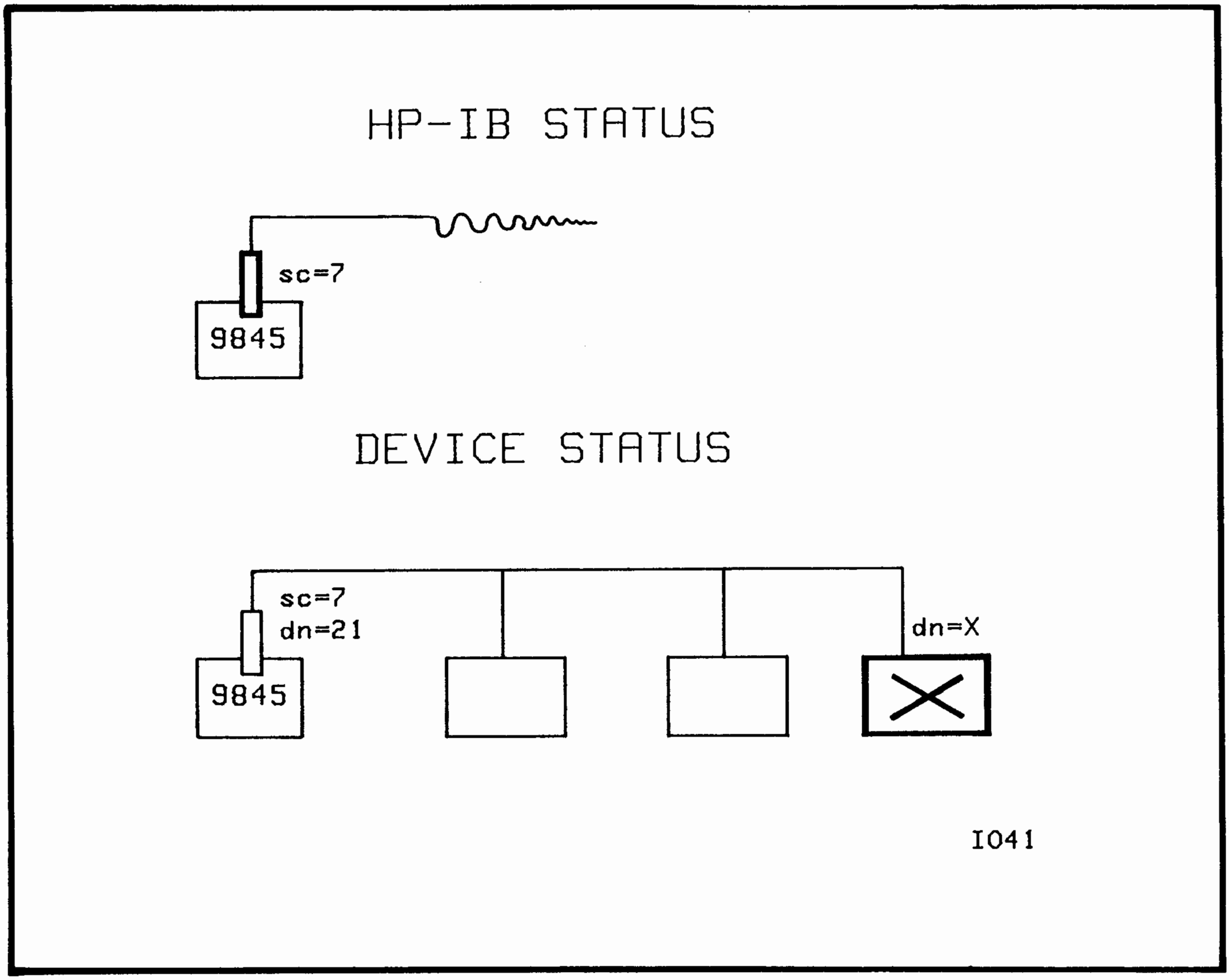
Variables returned for HP-IB Interface Card represent status bytes 4,1,2 and 3, respectively.

HP-IB DEVICE STATUS

EXAMPLE:

```
STATUS 711;A
```


I040


HP-IB STATUS
sc=7
9845
DEVICE STATUS
sc=7
dn=21
9845
dn=X
I041

HP-ID case
8 parallel lines for data
5 interface management
3 lines for handshaking

First Status Byte:

7	6	5	4	3	2	1	∅
∅	∅	∅	∅	∅	Device Clear	∅	Error

Bit 0: Is 1 when error detected.
Bit 2: Is 1 when Device Clear received.

Second Status Byte:

7	6	5	4	3	2	1	0
1	1	0	(MSB)		HP-IB Address		(LSB)

Third Status Byte:

7	6	5	4	3	2	1	0
EOI	REN	SRQ	ATN	IFC	NDAC	NRFD	DAV

Logical 1 indicates corresponding signal line is true.

Fourth Status Byte:

7	6	5	4	3	2	1	0
Service Request	Controller Active	Talker Active	Listener Active	System Controller Set	1	Serial Poll Set	End of Record

Bit 7: Is 1 when the SRQ signal line is true.
Bit 6: Is 1 when the calculator is the active controller.
Bit 5: Is 1 when the calculator is the active talker.
Bit 4: Is 1 when the calculator is an active listener.
Bit 3: Is 1 when the calculator is the active system controller.
Bit 2: Is always 1.
Bit 1: Is 1 when a serial poll is in process.
Bit 0: Is 1 when the EOI (end of record) line is true.

Figure 7. Interface Status Bytes

STATUS 7; A, B, C, D,
4th 1st 2nd 3rd

PROGRAM TO INTERPRET THE HP-IB INTERFACE DEVICE ADDRESS

```
10 STATUS 7;A,B,C,D
61 PRINT "Device address is ";BINEOR(2^7+2^6,C)
```

```
BINEOR:
11000000      2^7+2^6
11010101      Status byte C
---------
   10101   --->  21      Device address
       2           10
```

Binary or function

decimal equivalent

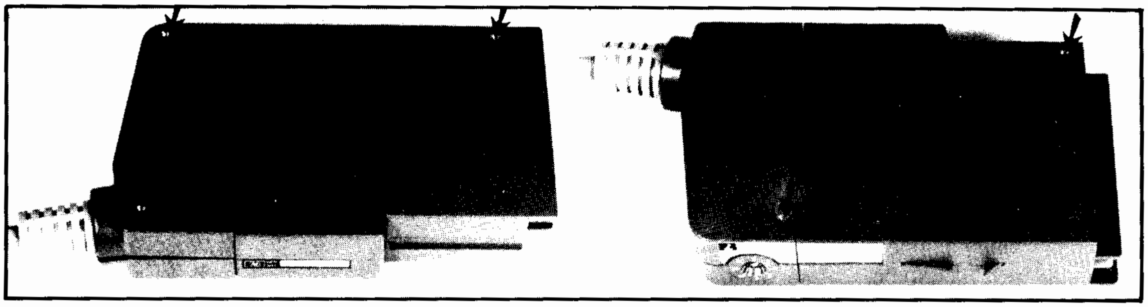
I043

EXERCISE

1. Read the STATUS bytes for the HP-IB Interface Card.

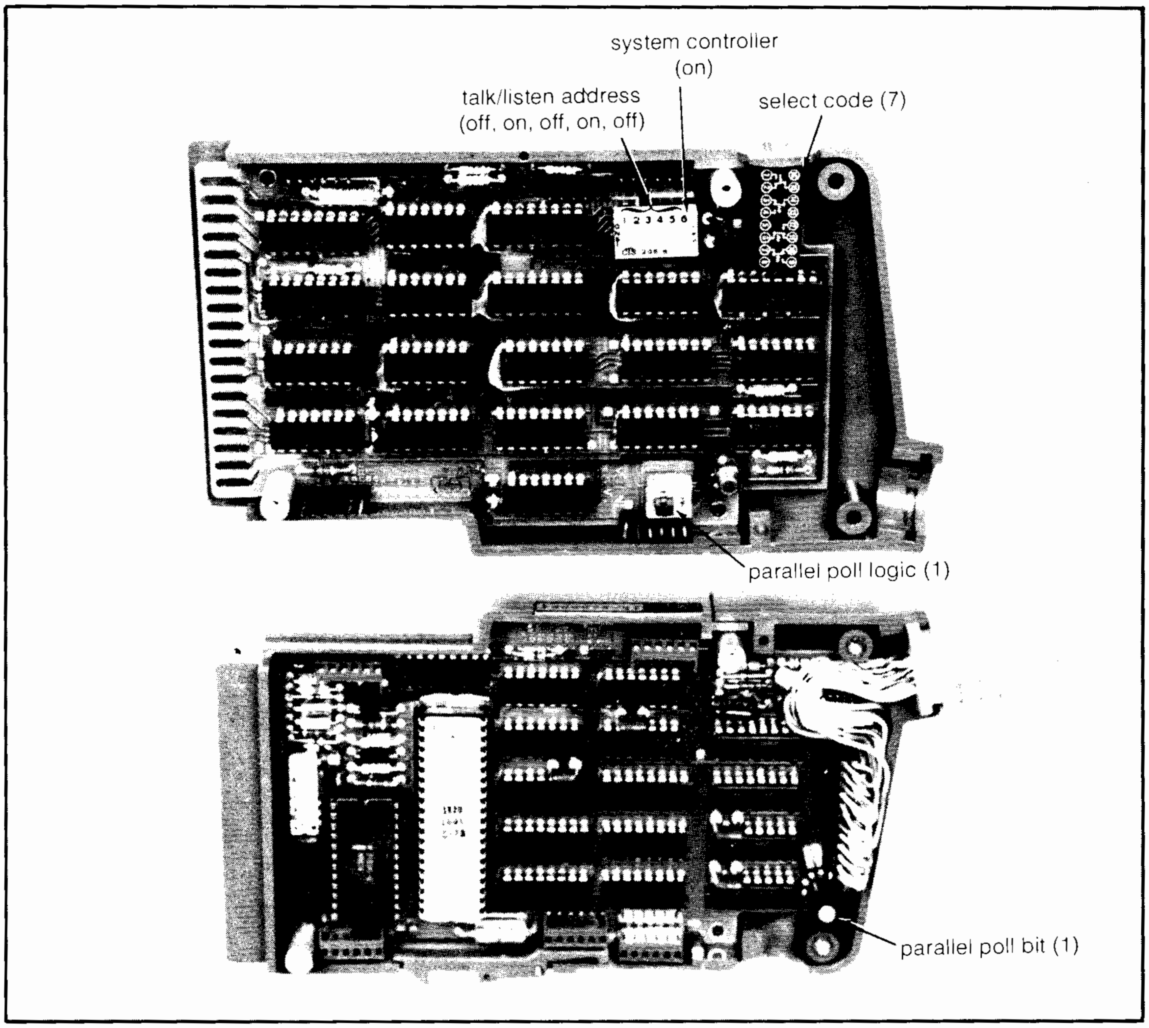
2. Compute the HP-IB Device Address assigned to the System 45.

I042



A. Remove only the four screws shown above.

B. Flip the card over and remove these two screws.



C. Seperate the case halves and position them as shown.

Figure 4. Opening the Interface Case

SWITCHES INSIDE THE HP-IB CARD

For System Controller on Device Address 21:

Switch settings

(1)	(2)	(3)	(4)	(5)	(6)
OFF	ON	OFF	ON	OFF	ON

For NON-Controller on Device Address 20:

Switch settings

(1)	(2)	(3)	(4)	(5)	(6)
ON	ON	OFF	ON	OFF	OFF

IO44.6

EXERCISE

Connect two System 45's in an HP-IB system: one must be System Controller and the other is the non-controller. The Device Addresses for the two systems must be different. Therefore, we'll assign the System Controller to Device Address 21, and the non-controller to Device Address 20.

1. Following the diagram, open the interface card with a screwdriver.

2. Set the switches for the talk/listen address and system controller as needed.

3. Re-check the device address and the system controller information after re-connecting the interface to the System 45.

I044

Changing Talk/Listen Addresses

The bus interface is set to talk address "U" and listen address "5" at the factory. These may be changed to any talk/listen pair of characters listed in the next table by setting switch S3 (1 through 5) on the A1 circuit board. Setting each slide to the "ON" position corresponds to a "0" in the table.

Table 5. Available Bus Addresses and Codes

Address Characters		Address Switch Settings					Address Codes	
Listen	Talk	(5)	(4)	(3)	(2)	(1)	decimal	octal
SP	@	0	0	0	0	0	0	0
!	A	0	0	0	0	1	1	1
"	B	0	0	0	1	0	2	2
#	C	0	0	0	1	1	3	3
$	D	0	0	1	0	0	4	4
%	E	0	0	1	0	1	5	5
&	F	0	0	1	1	0	6	6
'	G	0	0	1	1	1	7	7
(	H	0	1	0	0	0	8	10
)	I	0	1	0	0	1	9	11
*	J	0	1	0	1	0	10	12
+	K	0	1	0	1	1	11	13
,	L	0	1	1	0	0	12	14
–	M	0	1	1	0	1	13	15
.	N	0	1	1	1	0	14	16
/	O	0	1	1	1	1	15	17
0	P	1	0	0	0	0	16	20
1	Q	1	0	0	0	1	17	21
2	R	1	0	0	1	0	18	22
3	S	1	0	0	1	1	19	23
4	T	1	0	1	0	0	20	24
5	U	1	0	1	0	1	21	25 ← preset
6	V	1	0	1	1	0	22	26
7	W	1	0	1	1	1	23	27
8	X	1	1	0	0	0	24	30
9	Y	1	1	0	0	1	25	31
:	Z	1	1	0	1	0	26	32
;	[	1	1	0	1	1	27	33
<	/	1	1	1	0	0	28	34
=	]	1	1	1	0	1	29	35
>	^	1	1	1	1	0	30	36

SETTING HP-IB CARD SWITCHES

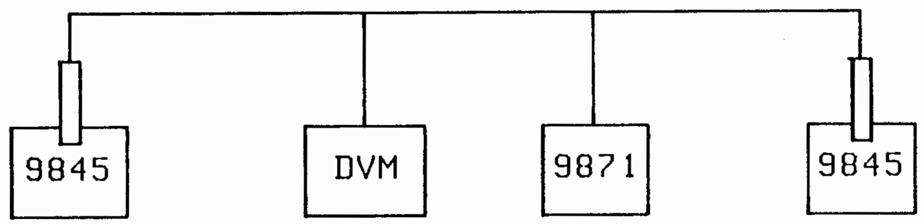
For NON-Controller on Device Address 20:

```
a.)  System Controller (Switch 6) = OFF
b.)  Device Address Switch (5) (4) (3) (2) (1)
                            1   0   1   0   0
c.)  1=OFF;0=ON            OFF  ON OFF  ON  ON
d.)  Switch settings as they appear on the card:
     (1) (2) (3) (4) (5) (6)
      ON  ON OFF  ON OFF OFF  (left to right)
```

I045.6

NON-ACTIVE CONTROLLERS ON THE BUS

NON-ACTIVE CONTROLLER DIAGRAM



I045

NON-ACTIVE CONTROLLERS ON THE BUS

PURPOSE:

To talk or listen on the bus after it has been configured by the active controller.

FORM:

Address the HP-IB interface select code only (no device number).

EXAMPLES:

```
OUTPUT 7;A$
Hpib=7
OUTPUT Hpib;A$
```

RULES:

The non-active controller cannot address a specific device on the bus.

I046

BUS ADDRESSING EXAMPLES

ACTIVE CONTROLLER:

```
10 FOR I=1 TO 20
20 ENTER 720;A$(I)
30 NEXT I
```

NON-ACTIVE CONTROLLER:

```
10 FOR I=1 TO 20
20 ENTER 7;A$(I)
30 NEXT I
```

I047

HP-IB
7 20
p.c. device address

SYSTEM CONTROLLER PROGRAMS

```
10 DIM A$[80]
20 Send:   INPUT "Message to send:",A$
30         OUTPUT 720;A$
40         GOTO Send
50 Take:   ENTER 720;A$
60         PRINT A$
70         GOTO Take
```

5,7.

NON-CONTROLLER PROGRAMS

```
10 DIM A$[80]
20 Take:   ENTER 7;A$
30         PRINT A$
40         GOTO Take
50 Send:   INPUT "Message to send:",A$
60         OUTPUT 7;A$
70         GOTO Send
```

I048.5

To go the other way
stop both programs
at system controller end
CONT Take:
at other end
CONT Send:

EXERCISE

1. Send a message from the Active Controller to the Non-Active Controller.

2. Send a return message from the Non-Active Controller to the Active Controller.

I048

OFF-LINE DATA TRANSFER

PURPOSE:

To be able to let the System 45 do something else while data is transferred on the bus.

EXAMPLE:

Send readings directly from the DVM to the HP-IB Printer.

RULES:

As Active Controller, the System 45 configures the bus, initiating the data transfer, then goes away to do something else.

I049

CONFIGURE STATEMENT

[illegible]

PURPOSE:

To configure the bus without sending data.

EXAMPLES:

```
CONFIGURE 7 TALK=21 LISTEN=1,9,5
CONFIGURE 7 TALK=7
CONFIGURE 7 LISTEN=1,9,5
```

RULES:

Only the Active Controller can configure the bus.

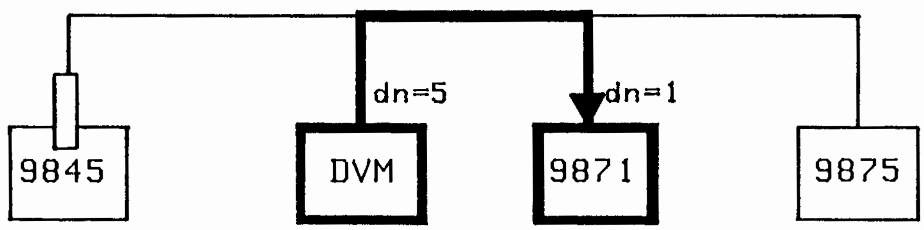
If no talker or listener is specified, the System 45 is the default.

CONFIGURE can start a data transfer.

I05Ø

EXAMPLE OF OFF-LINE DATA TRANSFER

```
Set DVM Range:       10 OUTPUT 705; "R1T"
Start transfer:      20 CONFIGURE 7  TALK=5  LISTEN=1
Do something else:   30 GOTO Star_wars
```



I051

SYSTEM MONITORING

PURPOSE:

If there is a problem during off-line data transfer, the System 45 as Active Controller needs to know.

EXAMPLE:

The printer runs out of paper.

RULES:

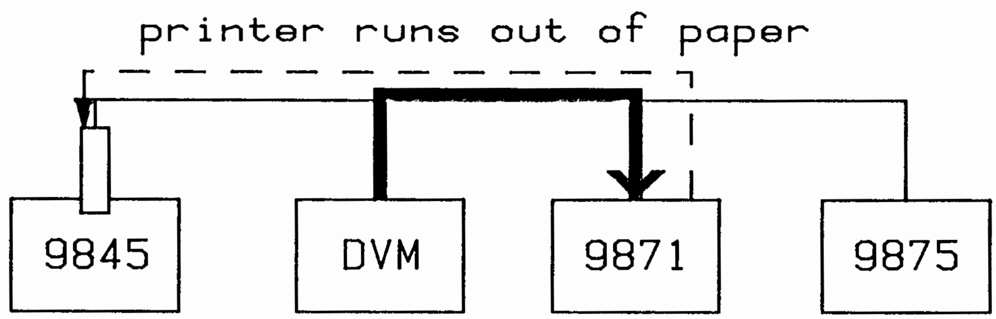
A Service Request (SRQ) is the peripheral's way of getting the Controller's attention.

The STATUS byte contains the Service Request (SRQ) bit.

Parallel Poll tells WHICH device is calling.

I052

SYSTEM MONITORING DIAGRAM



PARALLEL POLL

PURPOSE:

To check the summary information (PPOLL) byte which contains a single status bit for each of 8 devices on the bus.

SYNTAX:

PPOLL <s.c.>

EXAMPLE:

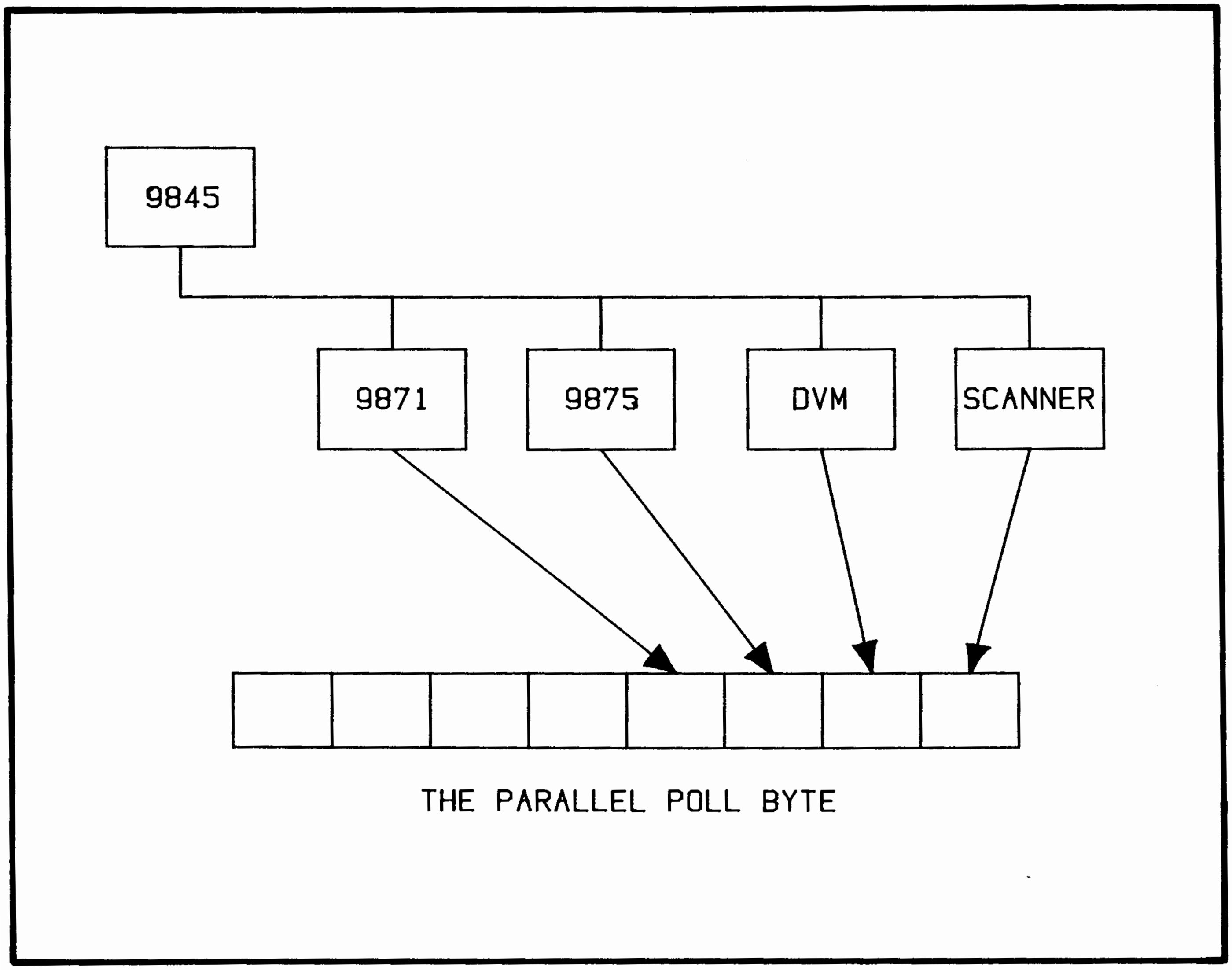
```
IF PPOLL 7 THEN GOTO Service
```

RULES:

PPOLL is a function.

The meaning of the single status bit is device dependent, but often signals SRQ.

I054



Book a "Tutorial Description of HP-IB"
write to HP Melbourne
FREE!

SERIAL POLL

PURPOSE:

To check all of the status bits for a single device after the parallel poll indicates the device requested service.

SYNTAX:

STATUS <s.c.>;<variable>

RULES:

A Serial Poll is merely a status check on a single device, giving detailed information.

A Parallel Poll is a QUICK status check on 8 devices and gives OK/NOT OK information.

Use Binary Operations to look at the bits.

I055

HP-IB STATUS BYTE

SRQ							

1 - a problem somewhere on the bus

PARALLEL POLL BYTE

dn1	dn2	dn3	dn4.	.			

1 - a problem on dn2

SERIAL POLL BYTE

1 - printer is out of paper

I056

REQUEST FOR SERVICE

PURPOSE:

To allow the System 45 as Non-Active Controller to request service from the Active Controller.

SYNTAX:

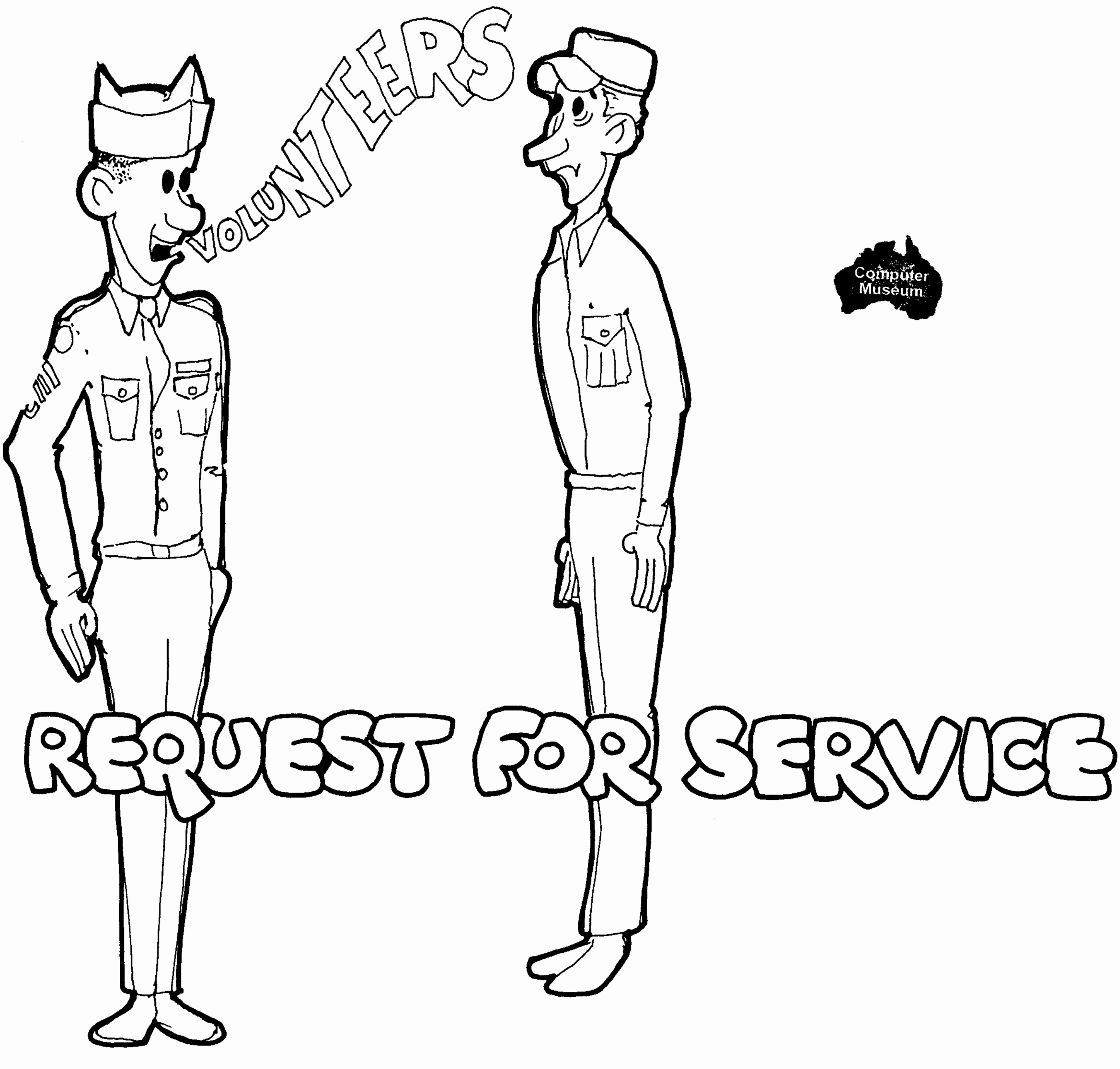
REQUEST <s.c.>;<response byte>

<response byte>: ØSxxxxxx
S: 1=Set SRQ Ø=Clear SRQ
xxxxxx: any bit pattern agreed upon by the System 45 and Controller to be the response to a serial poll.

RULES:

REQUEST only used by Non-Active Controllers.

I057


VOLUNTEERS
REQUEST FOR SERVICE

EXERCISE

1. Have the Non-Active Controller request service from the Active Controller.

2. Have the Active Controller check the status byte and answer the service request.

I058

ACTING ON A SERVICE REQUEST

To periodically check the status bit for SRQ (service request), and to service the remote System 45 if service is requested:

```
10 FOR I=1 TO 100
20 DISP I,"counting"
30 NEXT I
40 STATUS 7;A
50 IF NOT BIT(A,7) THEN 10
60 PRINT "Service requested"
70 STATUS 720;X
80 PRINT "Serial poll byte=";X
90 OUTPUT 720;"You were in trouble, but you're ok now"
100 PRINT "Device was serviced and is now working"
110 END
```

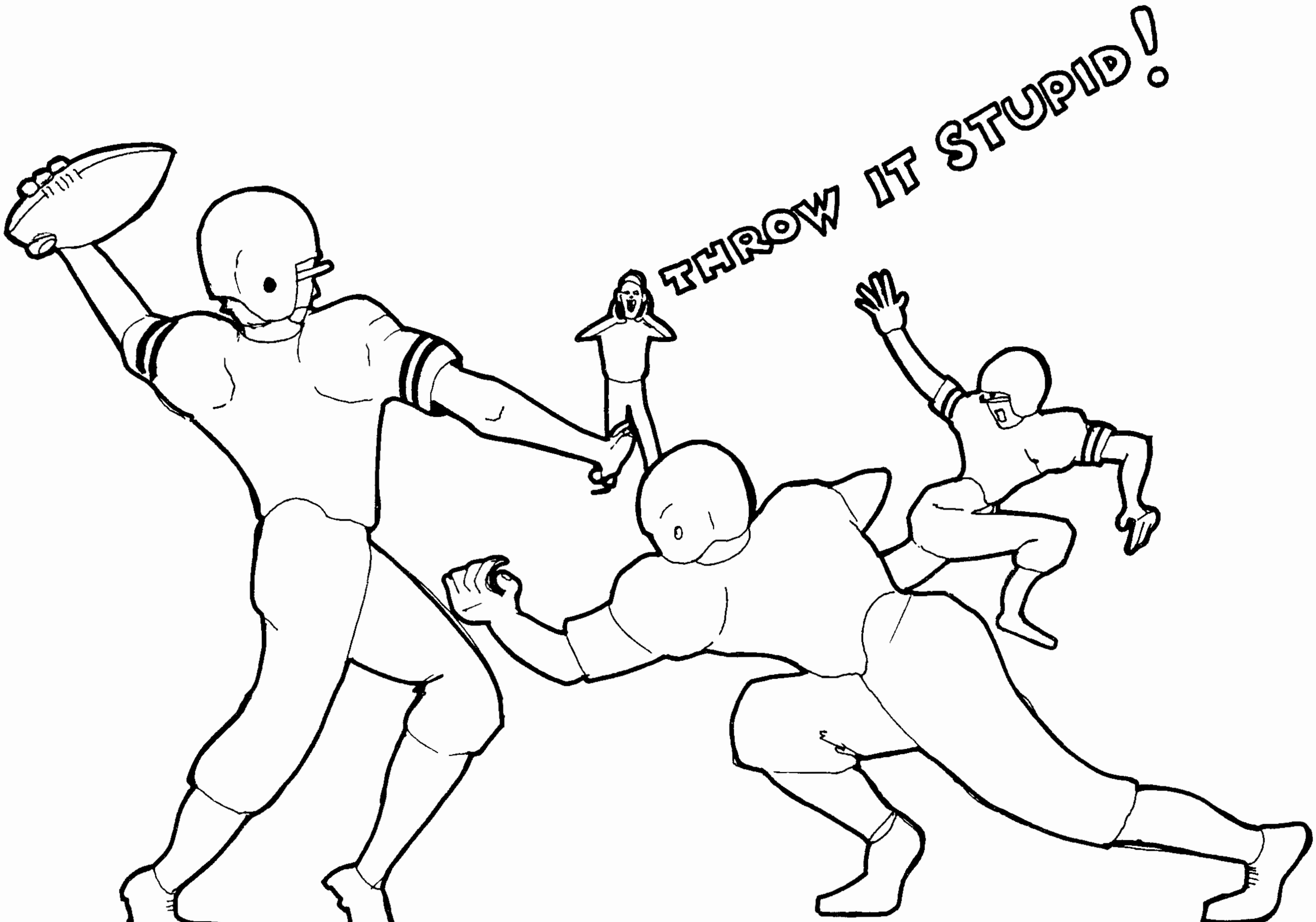
I058.4

REMOTE REQUESTING HELP

To request service from the Controlling System 45 and to receive a message back at the remote 45.

```
10 DIM A$[100]
20 DISP "Press CONTINUE to get help from your master"
30 PAUSE
40 X=2^6+2^3+2^2          ! Set serial poll bits 6,3 and 2
50 REQUEST 7;X            ! Send out service request
60 PRINT "Service requested; serial poll message=";X
70 ENTER 7; A$            ! Wait for message from master
80 PRINT A$.              ! Message indicates help arrived
90 END
```

I058.6



PASS CONTROL STATEMENT

PASS CONTROL STATEMENT

PURPOSE:

To pass active control to another device in the bus system when that device is capable of controlling the bus.

SYNTAX:

PASS CONTROL <s.c.><d.n.>

EXAMPLE:

PASS CONTROL 720

RULES:

Only the Active Controller can pass control to another device.

USES:

9815A/9845S Data Collection System

I059

ABORT I/O STATEMENT

PURPOSE:

To allow the System Controller take control of the bus UNCONDITIONALLY.

SYNTAX:

ABORTIO <s.c.>

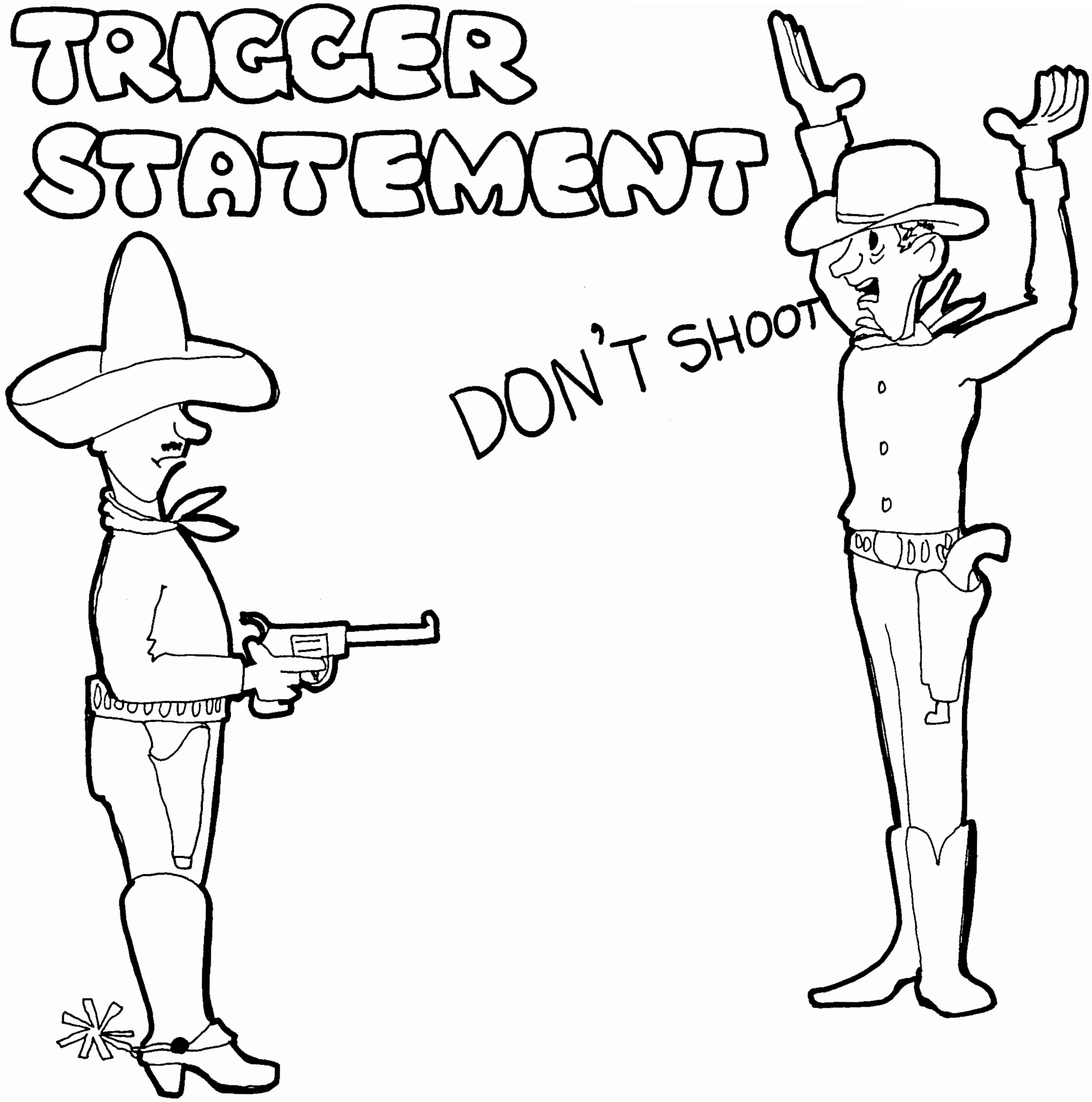
EXAMPLE:

ABORTIO 7

RULES:

- Only the System Controller can unconditionally take control of the bus.

I060


TRIGGER STATEMENT
DON'T SHOOT

TRIGGER STATEMENT

PURPOSE:

To send the bus Trigger message.

SYNTAX:

TRIGGER <s.c.>[<d.n.>]

EXAMPLE:

TRIGGER 701,711,704	for specific devices	(SDT)
TRIGGER 7	for all listeners	(GET)

USES:

To set up real-time events and measure their results.

IO61

SDT - Specific Device Trigger

GET - Group Execute Trigger

SENDBUS STATEMENT

PURPOSE:

To send commands and data to the bus at the lowest level of control.

SYNTAX:

SENDBUS <s.c.>;<command list>[;<data list>]

EXAMPLE:

SENDBUS 7;"?U6";"Hello"

RULES:

The command list is sent with ATN true and the data list is sent with ATN false.

USES:

COMMAND statements; special protocols.

I062

I/O AGENDA

I/O Architecture
Interface Cards
Formatted I/O
HP-IB
-------> Interrupts

I063

DEVICE SPEED COMPARISONS

MEDIUM SPEED DEVICES

- Thermal Printers
- Medium-speed DVM's
- 1,000 to 10,000 characters/second

9845

SLOW SPEED DEVICES

- Paper Tape Readers
- Card Readers
- Teletypes
- Digitizers
- Plotters
- Below 1,000 characters/second

FAST DEVICES

- High-speed DVM's
- Magnetic Tapes
- Disks
- A/D Converters
- Above 10,000 characters/second

INTERRUPT DATA TRANSFER

PURPOSE:

To allow the System 45 to do other things during data transfer to/from a SLOW device.

RULES:

The peripheral device is not aware of interrupt mode. The device interrupts when it is ready for each byte of information.

When the device interrupts, the I/O ROM gets the next byte of information ready and sends it to the device. While waiting for the interrupt, the System 45 continues with the main program.

IO64

internal buffer

PROGRAMMING INTERRUPT MODE

FORM:

Put the interrupt specifier in the I/O statement after the select code.

```
ENTER 3    BINT      ;A$
           ^^^^
```

EXAMPLES:

```
ENTER 3 BINT;A$
OUTPUT 5 BINT USING "#";A,B,C
```

RULES:

No other changes to your program are needed.

I065

HIGH SPEED DATA TRANSFER

PURPOSE:

To allow the System 45 to be COMPLETELY DEDICATED to a high speed data transfer to or from a FAST device.

RULES:

The System 45 can take in all of the data from a fast device without taking time to format it and store it into variables in memory.

If the data is already formatted and stored in a value area, then the System 45 can send data to a fast device without taking time to format the data during output.

I066

PROGRAMMING FOR HIGH SPEED I/O

FORM:

Substitute the appropriate transfer type in the I/O statement after the select code.

```
ENTER 3     BFHS       ;A$
            BDMA
            WFHS
            WDMA
            ^^^^
```

EXAMPLES:

```
ENTER 3 BDMA Count;A$
OUTPUT 5 BFHS USING "#";A$,B$,C$
```

I067

NOTES ON FAST I/O

FHS (Fast Handshake)

During fast handshake, the System 45 is fully dedicated to the data transfer. During this time, no other I/O operation can occur.

DMA (Direct Memory Access)

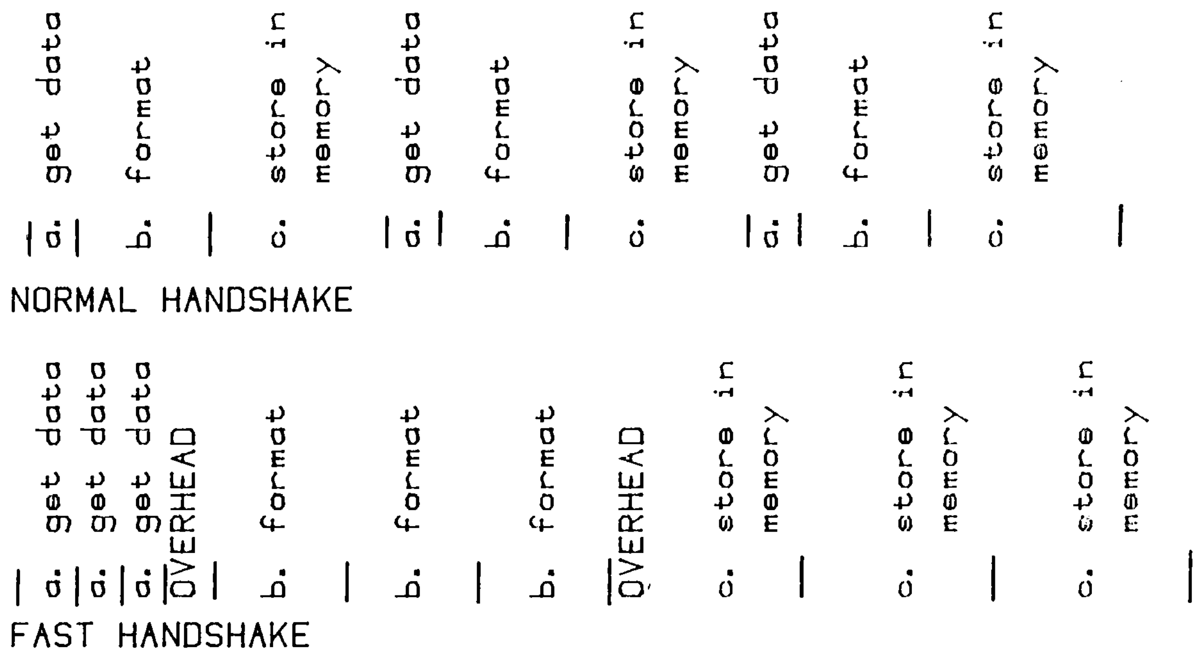
During DMA transfer, the System 45 is dedicated to the data transfer which takes place mainly through hardware, but also requires use of the I/O ROM.

DMA is available ONLY for use with the HP 98032A Interface. All others use Fast Handshake.

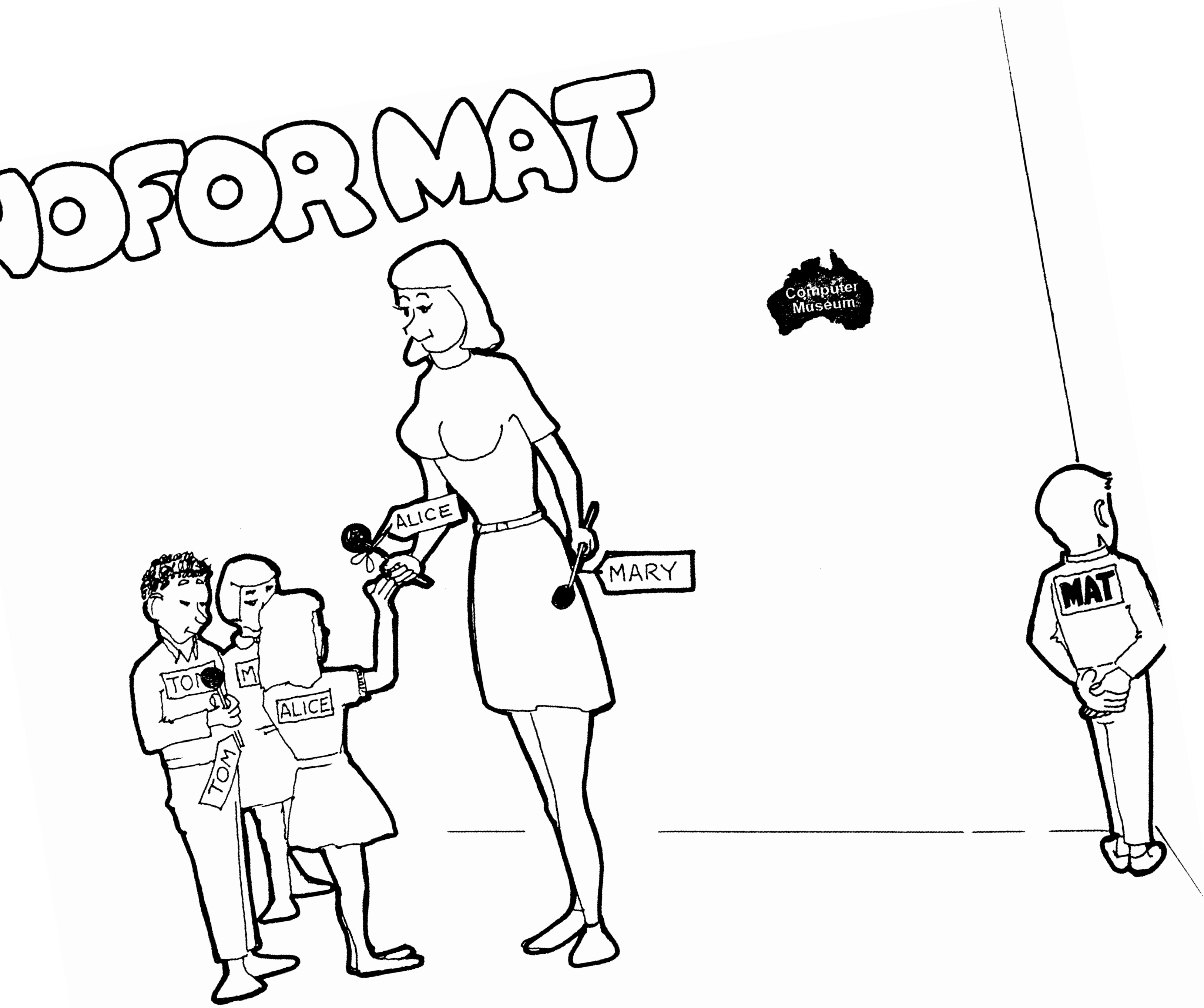
Fast I/O does not speed up THROUGHPUT.

I068

NORMAL AND FAST HANDSHAKE COMPARISON



NOTE: Input is faster, but throughput is actually slower because of increased overhead.


OFORMAT
ALICE
MARY
TOM
M
ALICE
TOM
MAT

NOFORMAT

PURPOSE:

To get the fastest speed during fast I/O.

FORM:

Substitute NOFORMAT for USING <image ref.>

EXAMPLE:

```
ENTER 6 BFHS NOFORMAT;A$
```

RULES:

NOFORMAT completely eliminates formatting during I/O.

Use I/O Value Areas for formatting off-line.

I069

TYPES OF DATA TRANSFER

TRANSFER TYPE	BYTE	WORD
NORMAL HANDSHAKE	Default	WHS
INTERRUPT	BINT	WINT
FAST HANDSHAKE	BFHS	WFHS
DIRECT MEMORY ACCESS	BDMA	WDMA

I070

ON INTERRUPT

PURPOSE:

To allow the System 45 to do something else during a data transfer, and then to take a special action when transfer is complete.

SYNTAX:

```
                               |GOTO |<line id>
ON INT#<s.c.>[,<priority>]|GOSUB|<label>
                               |CALL |
```

EXAMPLES:

```
ON INT#3 GOTO 120
ON INT#5,7 GOSUB Service
```

Branch to ON INT# is generated at the end of the data transfer. Priority can have any value 1 to 15.

I071

ON INTERRUPT EXAMPLE

```
5  OVERLAP
10 ON INT#3 GOTO Digitized
20 ENTER 3 BINT;A,B,C
30 I=1
40 I=I+1
50 GOTO40
60 Digitized: PRINT "Digitized points";A;B;C
70 PRINT "Also counted to";I;"while waiting"
80 GOTO 20
```

I072

USER SERVICE ROUTINES

PURPOSE:

The user can write routines to "service" a peripheral at completion of a data transfer.

FORM:

A service routine is usually a subroutine that provides more data for output or stores the entered data away and does whatever is required to complete the data transfer, i.e., "service".

RULES:

The line id or label following GOTO, GOSUB or CALL in the ON INT# statement references the user service routine.

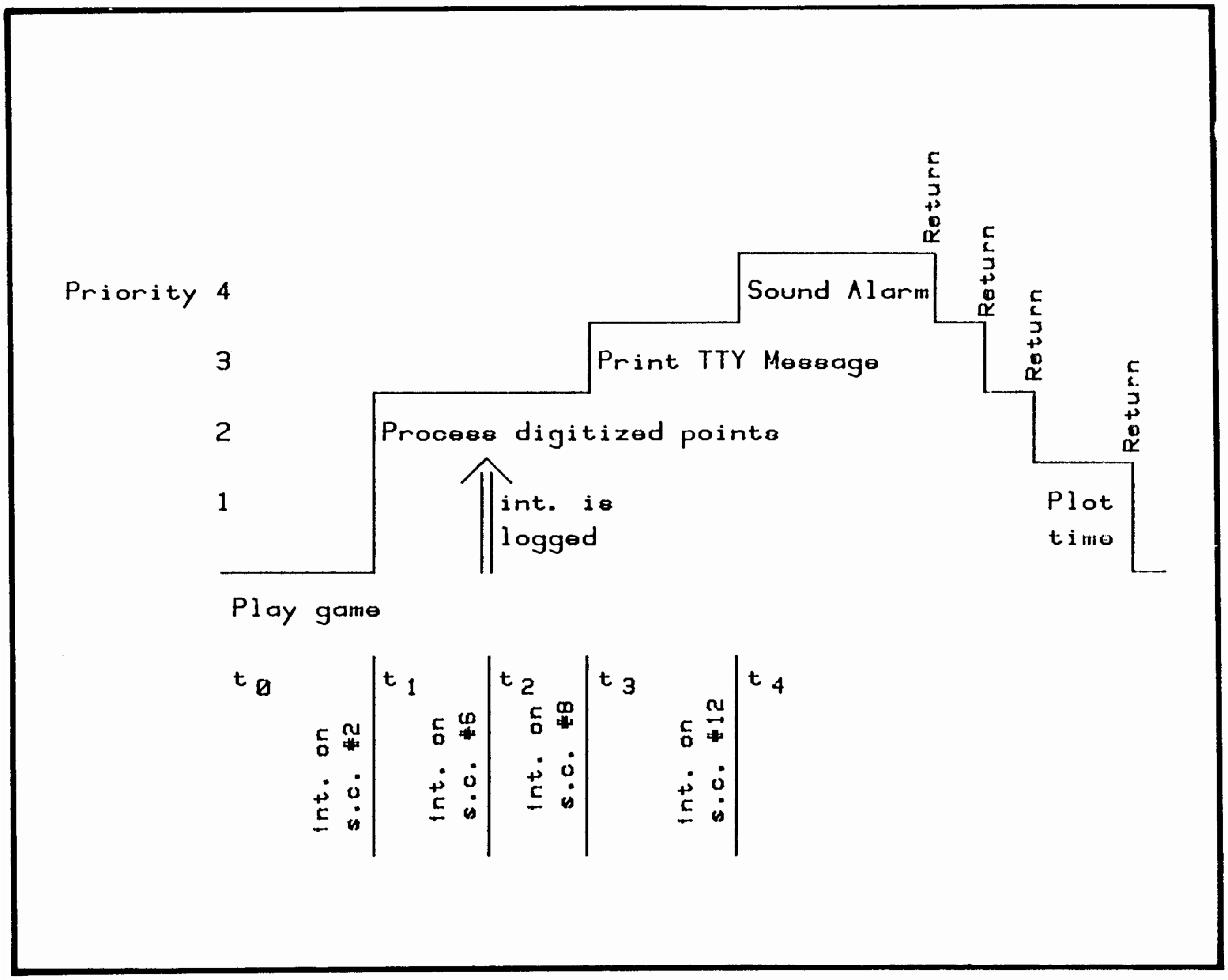
I073

enters into the buffer ready for fast action

PRIORITY LEVELS

```
OVERLAP
ON INT #12,4 GOSUB Alarm        !Heat sensor on s.c. 12
ENTER 12 BINT; Alarm$
ON INT #8,3 GOSUB Print_message     !TTY on s.c. 8
OUTPUT #8; Message$
ON INT #2,2 GOSUB Digitized     !Digitizer on s.c. 2
ENTER 2 BINT; X,Y
ON INT #6 GOSUB Plot_time     !Clock on s.c. 6
ENTER 6; Month,Day,Year,Hrs,Mins
Space_game:     !Play a game
```

11B


Priority 4
3
2
1
Sound Alarm
Print TTY Message
Process digitized points
int. is logged
Plot time
Return
Return
Return
Return
Play game
t 0
t 1
t 2
t 3
t 4
int. on s.c. #2
int. on s.c. #6
int. on s.c. #8
int. on s.c. #12

TIMEOUT STATEMENT

PURPOSE:

To allow a period of time for a peripheral to respond and then to quit waiting for the response.

SYNTAX:

SET TIMEOUT <s.c.>;<time>

RULES:

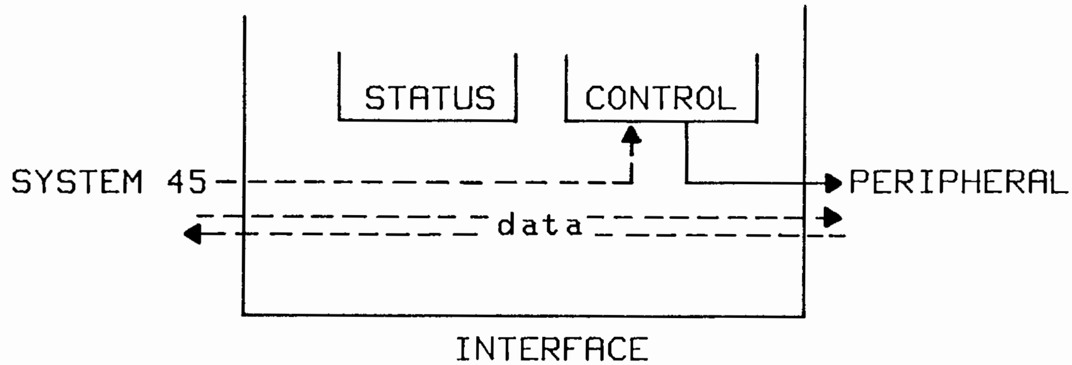
Time is in milliseconds; 0 cancels timeout.

Program branches to ON INT# routine when a timeout occurs.

TIMEOUT <s.c.> is a function which returns 1 if timeout caused entry to ON INT# routine.

I074

INTERFACE CONTROL DIAGRAM



I075

PROGRAMMING INTERFACE CONTROL

RULES:

The control byte is written to the interface card in two steps:

First, the control mask is set up in the memory of the System 45, ready to send to the interface. The CONTROL MASK statement is used to set up the control byte in memory.

When the card is to be enabled, the control mask is sent to the interface card by using the CARD ENABLE statement. The control mask in memory can also be sent automatically to the interface card if an interrupt data transfer is executed.

The control byte can be set to generate an interrupt on a character-by-character basis.

I076

CONTROL MASK & CARD ENABLE

PURPOSE:

To set up a control byte in memory and then to write the control byte to the interface card.

SYNTAX:

```
CONTROL MASK <s.c.>;<value>
CARD ENABLE <s.c.>
```

EXAMPLE:

```
CONTROL MASK 6;2^5+2^2
CARD ENABLE 6
```

I077

INTERFACE CONTROL BYTES

(set by CONTROL MASK, sent by CARD ENABLE)

98032A 16 Bit Parallel

7	6	5	4	3	2	1	Ø
INT	DMA	RESET	AHS	X	X	CTL1	CTLØ

INT: Interrupt enable on FLG = Ready
DMA: Direct Memory Access Enable
Reset: Reset to power on state
AHS: Auto Handshake Enable
X: Don't care
CTLØ,1: User definable bits

98033A BCD

7	6	5	4	3	2	1	Ø
INT	X	RESET	X	X	X	X	X

INT: Interrupt Enable on FLG = Ready
Reset: Reset to Power on state
X: Don't care

98034 HPIB

7	6	5	4	3	2	1	Ø
Service Request	Controller Active	Talker Active	Listener Active	Input Register Full	Output Register Empty	Other Interrupt Conditions	Enable EOI

Bit 7: Logical 1 enables interrupt on SRQ.
Bit 6: Logical 1 enables interrupt on active controller.
Bit 5: Logical 1 enables interrupt on active talker.
Bit 4: Logical 1 enables interrupt on active listener.
Bit 3: Logical 1 enables interrupt on input register full.
Bit 2: Logical 1 enables interrupt on output register empty.
Bit 1: Logical 1 enables interrupt when error detected, device clear or selective device clear/received (when not active controller)
Bit Ø: Enable EOI to clear status line (STS).

98035 Real Time Clock

7	6	5	4	3	2	1	Ø
INT	X	Reset	X	X	X	X	Pulse Mode

INT: Interrupt enable
Reset: Interface reset
Pulse Mode: Set pulse mode for external trigger

98036 Bit Serial

7	6	5	4	3	2	1	Ø
Interface Interrupt Enable	X	Programmed Interface Reset	X	X	Interrupt Control 2 Receiver Control	Interrupt Control 1 Transmitted Control	R4 Control 0 = Data IN OUT 1 = Control Status

Bit 7: Interrupt enable
Bit 2: Interrupt when ready with **received** character (if bit 7 set)
Bit 1: Interrupt when ready to **transmit** new character (if bit 7 set)
Bit Ø: Sets data/control mode of interface.

98040A Incremental Plotter

7	6	5	4	3	2	1	Ø
INT	DMA	Reset	AHS	X	X	X	X

INT: Enable interrupt
DMA: Direct Memory Access Enable
Reset: Interface Reset
AHS: Auto Handshake Enable
X: Don't care

```
10   ! PROGRAM TO PERMIT BRANCH TO SERVICE ROUTINE WHEN THE 9845
20   ! IS PASSED CONTROL OF BUS BY ACTIVE CONTROLLER.
30   ON INT #7 GOSUB Isr
40   CONTROL MASK 7;64                  !MASK FOR 98034 TO INTERRUPT
50   PASS CONTROL Alt_controller
60   CARD ENABLE 7                      !ENABLE INTERRUPT
70   GOTO 70                            !BACKGROUND JOB
80   END
90   Isr:    PRINT "ACTIVE CONTROLLER AGAIN"
100  ! TAKE APPROPRIATE ACTION
110  RETURN
```

Note here that you do not enable the interface to interrupt until **after** control is passed to the alternate controller (line 50). When the alternate controller passes control back to the 9845, the 98034 card will interrupt indicating that this has occurred.

Solution to Problem 1: MEAN

```
10   !  MEAN, VARIANCE, AND STANDARD DEVIATION
20   OPTION BASE 1
30   DIM X(10)
40   DATA 20,45,13,64,85,97,59,34,72,6
50   READ X(*)
60   Sum=Sumsq=0
70   N=10
80   FOR I=1 TO N
90      Sum=Sum+X(I)
100     Sumsq=Sumsq+X(I)*X(I)
110  NEXT I
120  Mean=Sum/N
130  Variance=(Sumsq-Sum*Sum/N)/(N-1)
140  Std=SQR(Variance)
150  PRINTER IS 0
160  GOSUB Print
170  PRINTER IS 16
180  GOSUB Print
190  END
200 Print:  PRINT "Index";TAB(10);"Array"
210  FOR I=1 TO 10
220     PRINT TAB(2);I;TAB(11);X(I)
230  NEXT I
240  PRINT LIN(2),"Mean=";Mean
250  PRINT "Variance=";Variance
260  PRINT "Standard Deviation=";Std
270  RETURN
```

Solution to Problem 2: TRIG

```
10   ! TRIG TABLE
20   DEG
30   PRINT "    X";TAB(22);"COS(X)";TAB(42);"SIN(X)";TAB(62);"TAN(X)"
40   PRINTER IS 0
50   GOSUB Loop
60   PRINTER IS 16
70   GOSUB Loop
80   BEEP
90   END
100 Loop:  FOR X=0 TO 45 STEP .5
110       FIXED 2
120       PRINT X;
130       FIXED 6
140       PRINT TAB(20);COS(X);TAB(40);SIN(X);TAB(60);TAN(X)
150   NEXT X
160   RETURN
```

Solution to Problem 1: ASCII TABLE

```
10    ! ASCII TABLE
20    PRINTER IS 0
30    PRINT "ASCII";TAB(10);"ASCII"
40    PRINT "CODE";TAB(8);"CHARACTER"
50    FOR I=32 TO 127
60       PRINT I;TAB(13);CHR$(I)
70    NEXT I
80    PRINTER IS 16
90    END
```

Solution to Problem 1: EMPLOYEE RECORDS

```
10   !  EMPLOYEE RECORDS
20   OPTION BASE 1
30   DIM Names$(30)[30],Address$(30)[30],Citystate$(30)[30],Phone$(30)[12]
40   DIM Pay(30)
50   PRINTER IS 16
60   INPUT "How many employees do you want to enter?",Max
70   IF (Max<1) OR (Max>30) THEN 60
80   FOR I=1 TO Max
90      DISP "Please enter name #";I;
100     INPUT Names$(I)
110     PRINT LIN(2),I,LIN(1)Names$(I)
120     INPUT "Address?",Address$(I)
130     PRINT Address$(I)
140     LINPUT "City, State?",Citystate$(I)
150     PRINT Citystate$(I)
160     INPUT "Telephone?",Phone$(I)
170     PRINT "Phone: ";Phone$(I)
180     INPUT "Hourly pay?",Pay(I)
190     PRINT "Hourly pay: ";Pay(I)
200  NEXT I
210  INPUT "Do you want to make any changes?",A$
220  A$=UPC$(A$[1,1])
230  IF A$="N" THEN Print
240  IF A$="Y" THEN Edit
250  BEEP
260  GOTO 210
270 Edit:   INPUT "Which name do you want to change?",I
280  IF (I<1) OR (I>Max) THEN 270
290     EDIT "Change the name",Names$(I)
300     PRINT LIN(2),I,LIN(1),Names$(I)
310     EDIT "Address?",Address$(I)
320     PRINT Address$(I)
330     EDIT "City, State?",Citystate$(I)
340     PRINT Citystate$(I)
350     EDIT "Telephone?",Phone$(I)
360     PRINT "Phone: ";Phone$(I)
370     INPUT "Hourly pay?",Pay(I)
380     PRINT "Hourly pay: ";Pay(I)
390  GOTO 210
400 Print:  !  This routine prints the data
410  PRINTER IS 0
420  FOR I=1 TO Max
430     PRINT LIN(2);Names$(I)
440     PRINT Address$(I)
450     PRINT Citystate$(I)
460     PRINT "Phone: ";Phone$(I)
470     PRINT "Hourly pay: ";Pay(I)
480  NEXT I
490  PRINTER IS 16
500  END
```

Solution to Problem 2: STRING SORT

```
10    ! STRING SORT
20    DIM Names$(10)[60],Temp$[60]
30    PRINTER IS 16
40    Max=10
50    FOR I=1 TO Max
60       DISP "Please enter name #";I;
70       INPUT Names$(I)
80       PRINT I,Names$(I)
90    NEXT I
100   INPUT "Do you want to make any changes?",A$
110   A$=UPC$(A$[1,1])
120   IF A$="N" THEN Sort
130   IF A$="Y" THEN Edit
140   BEEP
150   GOTO 100
160  Edit:   INPUT "Which name do you want to change?",I
170   IF (I<1) OR (I>Max) THEN 160
180   EDIT "Make your change",Names$(I)
190   PRINT I,Names$(I)
200   GOTO 100
210  Sort:  !  Bubble sort algorithm
220   FOR I=1 TO Max-1
230      FOR J=I+1 TO Max
240         If Names$(I)[POS(Names$(I)," ")+1]<=Names$(J)[POS(Names$(J)," ")+1] T
HEN J
250         Temp$=Names$(I)
260         Names$(I)=Names$(J)
270         Names$(J)=Temp$
280 J    NEXT J
290   NEXT I
300   PRINT LIN(2),"First name";TAB(20);"Last Name",LIN(1)
310   FOR I=1 TO Max
320      Pos=POS(Names$(I)," ")
330      PRINT Names$(I)[1,Pos-1];TAB(20);Names$(I)[Pos+1]
340   NEXT I
350   END
```

MULTIDIMENSIONAL ARRAYS DIMENSIONS

```
10   OPTION BASE 1
20   DIM One_d(15),One_d$(10)[80],Two_d(3,3),Three_d(2,2,5)
30   DIM Four_d$(5,6,4,2)[3],Six_d(2,3,4,5,6,7)
40   STOP
```

Solution to Problem 1: PRODUCTION LINE

```
10   OPTION BASE 1
20   DIM Shift(2,4,3)
30   DATA 75,65,85,30,45,20,6,10,5,85,90,60
40   DATA 80,70,90,25,40,22,8,11,4,82,88,54
50   READ Shift(*)
60   FOR I=1 TO 2
70      PRINT LIN(2);"Shift";I,LIN(1)
80      PRINT TAB(11);"Week 1   Week 2   Week 3"
90      FOR J=1 TO 4
91         PRINT "  Line";J;
100        FOR K=1 TO 3
110           PRINT TAB(2+K*9);Shift(I,J,K);
120        NEXT K
121        PRINT
130     NEXT J
140  NEXT I
150  END
```

Solution to Problem 2: PRODUCTION LINE

```
10   OPTION BASE 1
20   DIM Shift1(4,3),Shift2(4,3),Total(4,3),Compare(4,3),Print(4,3)
30   DATA 75,65,85,30,45,20,6,10,5,85,90,60
40   DATA 80,70,90,25,40,22,8,11,4,82,88,54
50   READ Shift1(*),Shift2(*)
60   MAT Total=Shift1+Shift2
70   MAT Compare=Shift1>=Shift2
80   PRINTER IS 0
90   MAT Print=Shift1
100  PRINT LIN(2),"Shift 1",LIN(1)
110  GOSUB Print
120  MAT Print=Shift2
130  PRINT LIN(2),"Shift 2",LIN(1)
140  GOSUB Print
150  MAT Print=Total
160  PRINT LIN(2),"Total",LIN(1)
170  GOSUB Print
180  MAT Print=Compare
190  PRINT LIN(2),"Shift 1 > Shift 2",LIN(1)
200  GOSUB Print
210  PRINTER IS 16
220  END
230 Print:  !  This routine prints the array Print(*)
240     PRINT TAB(11);"Week 1   Week 2   Week 3"
250     FOR J=1 TO 4
260        PRINT "  Line";J;
270        FOR K=1 TO 3
280           PRINT TAB(2+K*9);Print(J,K);
290        NEXT K
300        PRINT
310     NEXT J
320  RETURN
```

Solution to Problem 3: PRODUCTION LINE

```
10   OPTION BASE 1
20   DIM Shift1(4,3),Shift2(4,3),Total(4,3),Compare(4,3),Print(4,3)
30   DIM Value(4),Trows(4)
40   DATA 75,65,95,30,45,20,6,10,5,85,90,60
50   DATA 80,70,90,25,40,22,8,11,4,82,88,54
60   DATA 450,565,1200,375
70   READ Shift1(*),Shift2(*),Value(*)
80   MAT Total=Shift1+Shift2
90   MAT Compare=Shift1>=Shift2
100  PRINTER IS 0
110  MAT Print=Shift1
120  PRINT LIN(2),"Shift 1",LIN(1)
130  GOSUB Print
140  MAT Print=Shift2
150  PRINT LIN(2),"Shift 2",LIN(1)
160  GOSUB Print
170  MAT Print=Total
180  PRINT LIN(2),"Total",LIN(1)
190  GOSUB Print
200  MAT Print=Compare
210  PRINT LIN(2),"Shift 1 > Shift 2",LIN(1)
220  GOSUB Print
230  MAT Trows=RSUM(Shift1)
240  Dollars=DOT(Trows,Value)
250  PRINT "Total worth of products turned out within 3 week period by
shift 1 = S";VAL$(Dollars)
260  MAT Trows=RSUM(Shift2)
270  Dollars=DOT(Trows,Value)
280  PRINT "Total worth of products turned out within 3 week period by shift
2 = S";VAL$(Dollars)
290  PRINTER IS 16
300  END
310 Print:  !  This routine prints the array Print(*)
320     PRINT TAB(11);"Week 1   Week 2   Week 3"
330     FOR J=1 TO 4
340        PRINT " Line";J;
350        FOR K=1 TO 3
360           PRINT TAB(2+K*9);Print(J,K);
370        NEXT K
380        PRINT
390     NEXT J
400  RETURN
```

Solution to Problem 4: PRODUCTION LINE

```
1Ø   OPTION BASE 1
2Ø   DIM Shift1(4,3),Shift2(4,3),Total(4,3),Compare(4,3),Print(4,3)
3Ø   DIM Value(4),Trows(4)
4Ø   DATA 75,65,85,3Ø,45,2Ø,6,1Ø,5,85,9Ø,6Ø
5Ø   DATA 8Ø,7Ø,9Ø,25,4Ø,22,8,11,4,82,88,54
6Ø   DATA 45Ø,565,12ØØ,375
7Ø   READ Shift1(*),Shift2(*),Value(*)
71   MAT Shift1=Shift1*(.6)
72   MAT Shift2=Shift2*(.6)
8Ø   MAT Total=Shift1+Shift2
9Ø   MAT Compare=Shift1>=Shift2
1ØØ  PRINTER IS Ø
11Ø  MAT Print=Shift
12Ø  PRINT LIN(2),"Shift 1",LIN(1)
13Ø  GOSUB Print
14Ø  MAT Print=Shift2
15Ø  PRINT LIN(2),"Shift 2",LIN(1)
16Ø  GOSUB Print
17Ø  MAT Print=Total
18Ø  PRINT LIN(2),"Total",LIN(1)
19Ø  GOSUB Print
2ØØ  MAT Print=Compare
21Ø  PRINT LIN(2),"Shift 1 > Shift 2",LIN(1)
22Ø  GOSUB Print
23Ø  MAT Trows=RSUM(Shift1)
24Ø  Dollars=DOT(Trows,Value)
25Ø  PRINT LIN(2),"Total worth of products turned out within 3 week period
by shift 1 = $";VAL$(Dollars)
26Ø  MAT Trows=RSUM(Shift2)
27Ø  Dollars=DOT(Trows,Value)
28Ø  PRINT "Total worth of products turned out within 3 week period by
shift 1 = $";VAL$(Dollars)
29Ø  PRINTER IS 16
3ØØ  END
31Ø Print: !  This routine prints the array Print(*)
32Ø     PRINT TAB(11);"Week 1   Week 2   Week 3"
33Ø     FOR J=1 TO 4
34Ø        PRINT "  Line";J;
35Ø        FOR K=1 TO 3
36Ø           PRINT TAB(2+K*9);Print(J,K);
37Ø        NEXT K
38Ø        PRINT
39Ø     NEXT J
4ØØ  RETURN
```

Solution to Problem 5: PRODUCTION LINE

```
10   OPTION BASE 1
20   DIM Shift1(4,3),Shift2(4,3),Total(4,3),Compare(4,3),Print(4,3)
30   DIM Value(4),Trows(4)
40   DATA 75,65,85,30,45,20,6,10,5,85,90,60
50   DATA 80,70,90,25,40,22,8,11,4,82,88,54
60   DATA 450,565,1200,375
70   READ Shift1(*),Shift2(*),Value(*)
71   MAT Shift1=Shift1*(.6)
72   MAT Shift2=Shift2*(.6)
73   MAT Shift1=INT(Shift1)
74   MAT Shift2=INT(Shift2)
80   MAT Total=Shift1+Shift2
90   MAT Compare=Shift1>=Shift2
100  PRINTER IS 0
110  MAT Print=Shift1
120  PRINT LIN(2),"Shift 1",LIN(1)
130  GOSUB Print
140  MAT Print=Shift2
150  PRINT LIN(2),"Shift 2",LIN(1)
160  Gosub Print
170  MAT Print=Total
180  PRINT LIN(2),"Total",LIN(1)
190  GOSUB Print
200  MAT Print=Compare
210  PRINT LIN(2),"Shift 1 > Shift 2",LIN(1)
220  GOSUB Print
230  MAT Trows=RSUM(Shift1)
240  Dollars=DOT(Trows,Value)
250  PRINT LIN(2),"Total worth of products turned out within 3 week period
by shift 1 = $";VAL$(Dollars)
260  MAT Trows=RSUM(Shift2)
270  Dollars=DOT(Trows,Value)
280  PRINT "Total worth of products turned out within 3 week period by
shift 2 = $";VAL$(Dollars)
290  PRINTER IS 16
300  END
310 Print:  ! This routine prints the array Print(*)
320     PRINT TAB(11);"Week 1   Week 2   Week 3"
330     FOR J=1 TO 4
340        PRINT "  Line";J;
350        FOR K=1 TO 3
360           PRINT TAB(2+K*9);Print(J,K)
370        NEXT K
380        PRINT
390     NEXT J
400  RETURN
```

```
KEY Ø
    REMARK
   -Store

KEY 1
   -Clear line
    SQR(RES)
   -Execute

KEY 2
    7*8^5/(15*6)
   -Continue

KEY 3 -Undefined
KEY 4 -Undefined
KEY 5 -Undefined
KEY 6 -Undefined
KEY 7 -Undefined
KEY 8 -Undefined
KEY 9 -Undefined
KEY 10-Undefined
KEY 11-Undefined
KEY 12-Undefined
KEY 13-Undefined
KEY 14-Undefined
KEY 15-Undefined
KEY 16-Undefined
KEY 17-Undefined
KEY 18-Undefined
KEY 19-Undefined
KEY 20-Undefined
KEY 21-Undefined
KEY 22-Undefined
KEY 23-Undefined
KEY 24-Undefined
KEY 25-Undefined
KEY 26-Undefined
KEY 27-Undefined
KEY 28-Undefined
KEY 29-Undefined
KEY 30-Undefined
KEY 31-Undefined
```

Solution to Problem 1: AVERAGE ARRAY

```
10   INPUT "How big an array do you want?",N
20   CALL Input(N)
30   END
40   SUB Input(N)
50   OPTION BASE 1
60   DIM Array(N)
70   INPUT "Please enter the array?",Array(*)
71   PRINT Array(*);
80   PRINT "The mean of the array is";FNMean(Array(*))
90   SUBEXIT
100  DEF FNMean(X(*))=SUM(X)/ROW(X)
110  SUBEND
 7  5  3  9  2

The mean of the array is 5.2
```

Solution to Problem 2: SEARCH

```
10   OPTION BASE 1
20   DIM A(10)
30   DATA 45,87,35,96,10,45,25,20,17,99
40   READ A(*)
50   PRINT A(*)
60   INPUT "Enter a number",X
70   S=FNSearch(A(*),X)
80   PRINT X;"is array element #";S
90   GOTO 60
100  DEF FNSearch(A(*),X)
110  FOR I=1 TO ROW(A)
120     IF A(I)=X THEN RETURN I
130  NEXT I
140  RETURN 0
150  FNEND
 45  87  35  96  10  45  25  20  17  99


5 is array element # 0
20 is array element # 8
```

Solution to Problem 3: SWITCH

```
10   INPUT "ENTER X",X
20   INPUT "ENTER Y",Y
30   PRINT "X=";X,"Y=";Y
40   CALL Switch(X,Y)
50   PRINT "X=";X,"Y=";Y,LIN(1)
60   END
70   SUB Switch(A,B)
80   T=A
90   A=B
100  B=T
110  SUBEND
X= 4                Y= 9
X= 9                Y= 4
```

Solution to Problem 1: TABLE

```
10   PRINT " "&RPT$("_",75)
20   PRINT USING 30;"X","X^2","X^3","SQR(X)"
30   IMAGE "|",8X,A,9X"|",2(7X,3A,8X,"|"),6X,6A,6X,"|"
40   PRINT "|"&RPT$(RPT$("_",18)&"|",4)
50   FOR X=51 TO 100
60   PRINT USING 90;X,X*X,X^3,SQR(X)
70   IF NOT (X MOD 5) THEN PRINT USING 100
80   NEXT X
90   IMAGE "|",7X,39,8X"|",2(3X,DC3DC3D,6X,"|"),6X,2D,2D,7X,"|"
100  IMAGE "|",8X,X,9X"|",2(7X,3X,8X,"|"),6X,6X,6X,"|"
110  PRINT "|"&RPT$(RPT$("_",18)&"|",4)
120  END
```

```
100  !  MEDIA AND DEVICES EXERCISE 1    SOLUTION    24APR78
110  !
120  INPUT "INSERT TAPE.  THEN ENTER SELECT CODE OF DRIVE.  (14 OR 15)",S
130  IF (S=14) OR (S=15) THEN GOTO Initialize
140  INPUT "IMPROPER ENTRY.  TRY AGAIN.  SELECT CODE?  (14 OR 15)",S
150  GOTO 130
160 Initialize: !
170  DISP "TO INITIALIZE THE TAPE IN DRIVE T";VAL$(S);", PRESS CONTINUE."
180  PAUSE
190  INITIALIZE ":T"&VAL$(S)
210  END
```

```
INSERT TAPE.  THEN ENTER SELECT CODE OF DRIVE.  (14 OR 15)
15
TO INITALIZE THE TAPE IN DRIVE T15.  PRESS CONTINUE.
```

```
100   !  MEDIA AND DEVICES EXERCISE 2    SOLUTION   24APR78
110   !
120  Device:  !
130   DISP "DEVICE?  (T FOR TAPE, F FOR FLOPPY)";
140   INPUT D$                          !  INPUT DEVICE TYPE.
150   D$=UPC$(D$[1,1])                  !  TAKE FIRST CHAR.  GUARANTEE UPPER CASE
160   IF D$="T" THEN GOTO Tape          !  IF TAPE THEN GET TAPE SELECT CODE.
170   IF D$="F" THEN GOTO Floppy        !  IF FLOPPY THEN GET FLOPPY SELECT CODE.
180   DISP "IMPROPER ENTRY.  TRY AGAIN.  "; !  IF NEITHER THEN RE-ENTER
200   GOTO Device
210  Tape:  !
220   CALL Get("TAPE SELECT CODE",14,15,S) ! GET VALID TAPE SELECT CODE.
230   Msus$=":T"&VAL$(S)                !  CONSTRUCT MASS STORAGE UNIT SPECIFIER.
240   GOTO Declare                      !  GO TO DECLARE MASS STORAGE DEVICE.
250  Floppy:  !
260   CALL Get("FLOPPY SELECT CODE",1,12,S)  !  GET VALID FLOPPY SELECT CODE.
270   CALL Get("FLOPPY UNIT CODE",0,7,U)  !  GET VALID FLOPPY UNIT CODE.
280   Msus$=":F"&VAL$(S)&","&VAL$(U)  !  CONSTRUCT MASS STORAGE UNIT SPECIFIER
290  Declare:  !
300   MASS STORAGE IS Msus$             !  DECLARE MASS STORAGE DEVICE
310   END
320   !
330   SUB Get(Prompt$,Min,Max,Ret)      !  INPUT INTEGER BETWEEN Min AND Max.
340   Range$="?  ("&VAL$(Min)&"  TO  "&VAL$(Max)&"  !  CONSTRUCT PART OF PROMPT.
350   DISP Prompt$;Range$;              !  DISPLAY PROMPT.
360   INPUT Ret                         !  INPUT VALUE.
370   IF Ret<>INT(Ret)  THEN GOTO 390 !  GUARANTEE INTEGER.
380   IF (Min<=Ret) AND (Ret<=Max) THEN SUBEXIT!  IF VALID INPUT THEN RETURN.
390   DISP "IMPROPER ENTRY.  TRY AGAIN.  "; ! IF INVALID THEN RE-ENTER.
410   GOTO 350
420   SUBEND
```

```
DEVICE?  (T FOR TAPE, F FOR FLOPPY)
?
T
TAPE SELECT CODE?  (14 TO 15)
?
15
```

```
100  !  PROGRAM STORAGE EXERCISE 1    SOLUTION    25APR78
110  !
120  !  EXECUTE THE FOLLOWING FROM THE KEYBOARD:
130  !          LOAD "SORT"
140  !          SAVE "Sort",200,310
150  !          SAVE "Exch",320
160  !          GET "Exch",200
170  !          GET "Sort",290
180  !          STORE "SORTX"
```

```
LOAD "SORT"
SAVE "Sort",200,310
SAVE "Exch",320
GET "Exch",200
GET "Sort",290
STORE "SORTX"
```

```
LIST #0
100  ! MAIN PROGRAM
110  ! THIS PROGRAM GENERATES AND SORTS 20 RANDOM NUMBERS.
120  OPTION BASE 1
130  DIM Numbers(20)
140  FOR I=1 TO 20
150  Numbers(I)=INT(90*RND)+10
160  NEXT I
170  CALL Sort(Numbers(*),20)
180  PRINT Numbers(*);
190  END
200  !
210  ! EXCHANGE SUBROUTINE
220  ! Exchange SWITCHES Z(P) AND Z(Q).
230  SUB Exchange(Z(*),P,Q)
240  OPTION BASE 1
250  Temp=Z(P)
260  Z(P)=Z(Q)
270  Z(Q)=Temp
280  SUBEND
290  !
300  ! SORT SUBROUTINE
310  ! Sort SORTS IN ASCENDING ORDER THE ELEMENTS FROM 1 TO N IN
320  ! THE ONE-DIMENSIONAL ARRAY A.  OPTION BASE 1 IS ASSUMED.
330  SUB Sort(A(*),N)
340  OPTION BASE 1
350  FOR I=1 TO N-1
360  FOR J=I+1 TO N
370  IF A(I)>A(J) THEN CALL Exchange(A(*),I,J)
380  NEXT J
390  NEXT I
400  SUBEND
```

```
100  ! PROGRAM STORAGE EXERCISE 2    SOLUTION    27APR78
110  !
120  !
130  ! EXECUTE:
140  !              SCRATCH A
150  !
160  ! DEFINE KEY 0 AS FOLLOWS:
170  !                                   -Clear line
180  !                                    PRINTER IS 0
190  !                                   -Execute
200  !
210  ! DEFINE KEY 1 AS FOLLOWS:
220  !                                   -Clear line
230  !                                    PRINTER IS 16
240  !                                   -Execute
250  !
260  ! EXECUTE:
270  !              STOREKEY "KEYS"
280  !
290  ! THAT'S IT FOLKS!
```

```
KEY 0
   -Clear line
    PRINTER IS 0
   -Execute

KEY 1
   -Clear line
    PRINTER IS 16
   -Execute

STOREKEY "KEYS"
```

```
10   ! PROGRAM STORAGE EXERCISE 3   SOLUTION   27APR78
20   !
30   LOAD "FILEA",Start
40   END
```

START1	S = 0	T = 0
START2	S = 10	T = 10
LOAD 1	S = 10	T = 0
LOAD 2	S = 9	T = 9
LINK 1	S = 9	T = 9
GET 1	S = 9	T = 0

```
100   ! GENERAL FILE EXERCISE 2    SOLUTION     5MAY78
110   !
120   INPUT "PRINTER SELECT CODE? (16 FOR CRT, 0 FOR PRINTER)",S
130   IF S=16 THEN PRINT PAGE            ! IF CRT THEN CLEAR SCREEN.
140   CAT #S;","                         ! GET HEADING USING NONEXISTENT SELECTIVE
150                                      ! CATALOG SPECIFIER.
160   FOR Letter=65 TO 90                ! FOR EACH UPPER CASE LETTER
170   CAT #S;CHR$(Letter),1              ! CATALOG THE FILES BEGINNING WITH THAT LETTER
180   NEXT Letter
190   END
```

```
NAME PRO TYPE REC/FILE   BYTES/REC    ADDRESS
T15        2

AFC       DATA       32        64          35
BOX       DATA        1       256         148
CIRCLE    DATA        2       256         146
CSIZE     DATA        6       256         170
COPY      DATA        5       256         253
DIGITI    PROG       59       256         182
FILEA     RPOG        2       256           9
FILEB     RPOG        2       256          11
FILEC     DATA        1       256          13
FILED     DATA        1       256          14
FLOP2     BPRG       97       256          49
LABELX    DATA        3       256         149
LABEL     DATA        4       256         152
LABUSE    DATA        4       256         156
LETTER    DATA        5       256         160
LORG      DATA        5       256         165
LDIR      DATA        6       256         176
MESSAG    DATA        1       256          48
MEMO      DATA        7       256         246
NFC       DATA        4       256          31
POEM      DATA        1       256          47
PRTPLT    PROG        5       256         241
RANDOM*   DATA       32        64          23
SORT      PROG        4       256           5
```

```
100  ! DATA BASICS EXERCISE 1    SOLUTION    5MAY78
110  !
120  INPUT "FILE NAME?",File$                     ! INPUT FILE NAME.
130  Message$="FILE "&CHR$(34)&File$&CHR$(34)&" IS "
140  ASSIGN #1 TO File$,R                         ! ATTEMPT TO OPEN FILE UNPROTECTED.
150  IF R=2 THEN Reassign  ! 2 MEANS INCORRECT PROTECTION OR WRONG FILE TYPE.
160  IF R=1 THEN DISP Message$;"NONEXISTENT." ! 1 MEANS FILE DOES NOT EXIST.
170  IF R=0 THEN DISP Message$;"AN UNPROTECTED FILE." ! 0  MEANS ALL OK.
180  STOP                                         ! DONE.
190 Reassign: ASSIGN #1 TO File$,R,"WRITE" ! ATTEMPT TO OPEN FILE PROTECTED.
200  IF R=0 THEN DISP Message$;"A PROTECTED FILE."! 0 MEANS ALL OK.
210  IF R<>0 THEN DISP Message$;"NOT A DATE-TYPE FILE." ! FILE WON'T OPEN.
220  END                                          ! DONE.
```

```
FILE NAME?
FILEC
FILE "FILEC" IS AN UNPROTECTED FILE.
```



```
FILE NAME?
SERIAL
FILE "SERIAL" IS A PROTECTED FILE.
```

```
FILE NAME?
FILEA
FILE "FILEA" IS NOT A DATA-TYPE FILE.
```

```
FILE NAME?
TEST
FILE "TEST" IS NONEXISTENT.
```

```
100  ! DATA BASICS EXERCISE 2    SOLUTION    4MAY78
110  !
120  OPTION BASE 1                     ! ARRAY LOWER BOUND IS ONE.
130  DIM Serial$(28,2)                 ! Serial$ HOLDS CONTENTS OF "SERIAL" FILE.
140  DIM Random$(28,3)                 ! Random$ HOLDS CONTENTS OF "RANDOM" FILE.
150  DIM Nfc$(15,3)                    ! Nfc$ HOLDS CONTENTS OF "NFC" FILE.
160  DIM Afc$(14,3)                    ! Afc$ HOLDS CONTENTS OF "AFC" FILE.
170  Quote$=CHR$(34)                   ! SET Quotes TO ".
180  !
190 Taska:                             ! TASK A
200  ASSIGN #1 TO "SERIAL",R,"WRITE"   ! OPEN "SERIAL" FOR RECORD I/O.
210  READ #1;Serial$(*)                ! SERIALLY READ "SERIAL" INTO 28X2 ARRAY.
220  PRINT PAGE;Quote$;"SERIAL";Quote$;" FILE, SERIALLY READ",LIN(1) ! TITLE.
230  PRINT USING "K,X,K/";Serial$(*)   ! PRINT ARRAY.
240  ASSIGN * TO #1                    ! CLOSE "SERIAL".
250  !
260 Taskb:                             ! TASK B
270  ASSIGN #2 TO "RANDOM",R,"WRITE"   ! OPEN "RANDOM" FOR RECORD I/O.
280  FOR N=1 TO 28                     ! RANDOMLY READ "RANDOM" INTO 28X3 ARRAY.
290  READ #2,N;Random$(N,1),Random$(N,2),Random$(N,3)
300  NEXT N
310  PRINT PAGE,Quote$;"RANDOM";Quote$;" FILE, RANDOMLY READ";LIN(1) ! TITLE.
320  PRINT USING "20A,15A,10A,/";Random$(*) ! PRINT ARRAY.
330  !
340 Taskc:                             ! TASK C
350  READ #2,1                         ! RESET FILE POINTER TO FIRST BYTE OF FILE.
360  READ #2;Random$(*)                ! SERIALLY READ "RANDOM" INTO 28X3 ARRAY.
370  PRINT PAGE,Quote$;"RANDOM";Quote$;" FILE, SERIALLY READ",LIN(1) ! TITLE.
380  PRINT USING "20A,15A,10A,/";Random$(*) ! PRINT ARRAY.
390  ASSIGN * TO #2                    ! CLOSE "RANDOM"
400  !
410 Taskd:                             ! TASK D
420  ASSIGN #3 TO "NFC",R              ! CHECK TO SEE WHETHER "NFC" EXISTS.
430  IF R<>1 THEN Taske                ! IF "NFC" EXISTS, SKIP CREATION.
440  CREATE "NFC",4                    ! "NFC" IS 4 256-BYTE DEFINED RECORDS.
450  !
460 Taske:                             ! TASK E
470  ASSIGN #4 TO "AFC",R              ! CHECK TO SEE WHETHER "AFC" EXISTS.
480  IF R<>1 THEN Taskf                ! IF "AFC" EXISTS, THEN SKIP CREATION.
490  CREATE "AFC",16,64                ! "AFC" IS 16 64-BYTE DEFINED RECORDS.
500  !
510 Taskf:                             ! TASK F
520  ASSIGN #3 TO "NFC"                ! OPEN "NFC" FOR RECORD I/O.
530  FOR N=1 TO 14                     ! FOR EACH NFC TEAM.
540  PRINT #3;Serial$(N,1),Serial$(N,2),Random$(N,3) ! PRINT LOC, NAME, DIV.
550  NEXT N
560  PRINT #3;END                      ! PRINT END-OF-FILE MARK.
570  REDIM Nfc$(14,3)                  ! FOR NOW MAKE Nfc$ 14X3.
580  READ #3,1                         ! RESET FILE POINTER.
590  READ #3;Nfc$(*)                   ! TO VERIFY, READ "NFC" INTO ARRAY.
600  PRINT PAGE,"ORIGINAL ";Quote$;"NFC";Quote$;" FILE";LIN(1) ! TITLE.
610  PRINT USING "3(18A),/";Nfc$(*)    ! PRINT ARRAY.
620  !
```

```
630     Taskg:                          !  TASK G
640      ASSIGN #4 TO "AFC"             !  OPEN "AFC" FOR RECORD I/O.
650      FOR N=15 TO 28                 !  FOR REACH AFC TEAM.
660      PRINT #4,N-14;Random$(N,1),Random$(N,3)  !  PRINT LOCATION AND DIVISION.
670      NEXT N
680      PRINT #4,15;END                !  PRINT END-OF-FILE MARK.
690      REDIM Afc$(14,2)               !  FOR NOW MAKE Afc$ 14x2.
700      READ #4,1                      !  RESET FILE POINTER.
710      READ #4;Afc$(*)                !  TO VERIFY, READ "AFC" AND LIST IT.
720      PRINT LIN(4),"ORIGINAL";Quotes$;"AFC";Quotes$;"FILE";LIN(1)
730      PRINT USING "20A,10A,/";Afc$(*) !
740      !
750     Taskh:                          !  TASK H
760      READ #3,1                      !  RESET FILE POINTER TO FIRST BYTE IN FILE.
770      REDIM Nfc$(9,3)                !  MAKE Nfc$ 9X3 TO POSITION FILE POINTER.
780      READ #3;Nfc$(*)                !  POSITION FILE POINTER AT INSERTION POINT.
790      PRINT #3;" "," "," "           !  INSERT THREE BLANK STRINGS.
800      FOR N=10 TO 14                 !  REPRINT THE REMAINDER OF THE FILE.
810      PRINT #3;Serial$(N,1),Serial$(N,2),Random$(N,3)  !  PRINT LOC, NAME, DIV.
820      NEXT N
830      PRINT #3;END                   !  PRINT END-OF-FILE MARK.
840      REDIM Nfc$(15,3)               !  MAKE Nfc$ 15X3 TO READ FILE.
850      READ #3,1                      !  RESET FILE POINTER TO FIRST BYTE IN FILE.
860      READ #3;Nfc$(*)                !  READ "NFC" DATA.
870      PRINT PAGE;TAB(25);"UPDATED";Quotes$'"NFC";Quotes$;"FILE",LIN(1)
880      PRINT TAB(6):"EAST";TAB(30);"CENTRAL";TAB(54);"WEST";LIN(1)
890      FOR N=1 TO 5                   !  FOR EACH OF FIVE ROWS,
900      FOR M=N TO N+10 STEP 5         !  FOR EACH OF THREE DIVISION,
910      PRINT USING "#,24A";Nfc$(M,1)&" "&Nfc$(M,2) ! PRINT LOCATION AND NAME.
920      NEXT M
930      PRINT                          !  GO TO NEXT LINE.
940      NEXT N
950      !
960     Taski:                          !  TASK I
970      FOR N=1 TO 14                  !  FOR EACH TEAM IN THE "AFC" FILE,
980      READ #4,N;Loc$,Div$            !  READ LOCATION AND DIVISION.
990      PRINT #4,N;Loc$,Serial$(N+14,2),Div$ ! INSERT NAME BETWEEN LOC & DIV.
1000     NEXT N
1010     REDIM Afc$(14,3)               !  MAKE Afc$ 14X3.
1020     READ #4,1                      !  RESET FILE POINTER TO FIRST BYTE IN FILE.
1030     READ #4;Afc$(*)                !  SERIALLY READ CONTENTS OF "AFC".
1040     PRINT LIN(2);TAB(25);"UPDATED";Quotes$;"AFC";Quotes$;"FILE";LIN(1)
1050     PRINT TAB(6);"EAST";TAB(30);"CENTRAL";TAB(54);"WEST";LIN(1)
1060     FOR N=1 TO 5                   !  FOR EACH OF FIVE ROWS,
1070     FOR M=N TO N+10 STEP 5         !  FOR EACH OF THREE DIVISIONS,
1080     IF M<=14 THEN PRINT USING "#24A";Afc$(M,1)&" "&Afc$(M,2) ! PRINT DATA.
1090     NEXT M
1100     PRINT                          !  GO TO NEXT LINE.
1110     NEXT N
1120     REWIND                         !  REWIND TAPE CARTRIDGE.
1130     END
```

"RANDOM" FILE, RANDOMLY READ

DALLAS	NATIONAL	EAST
NEW YORK	NATIONAL	EAST
PHILADELPHIA	NATIONAL	EAST
ST. LOUIS	NATIONAL	EAST
WASHINGTON	NATIONAL	EAST
CHICAGO	NATIONAL	CENTRAL
DETROIT	NATIONAL	CENTRAL
GREEN BAY	NATIONAL	CENTRAL
MINNESOTA	NATIONAL	CENTRAL
ATLANTA	NATIONAL	WEST
LOS ANGELES	NATIONAL	WEST
NEW ORLEANS	NATIONAL	WEST
SAN FRANCISCO	NATIONAL	WEST
TAMPA BAY	NATIONAL	WEST
BALTIMORE	AMERICAN	EAST
BUFFALO	AMERICAN	EAST
MIAMI	AMERICAN	EAST
NEW ENGLAND	AMERICAN	EAST
NEW YORK	AMERICAN	EAST
CINCINNATI	AMERICAN	CENTRAL
CLEVELAND	AMERICAN	CENTRAL
HOUSTON	AMERICAN	CENTRAL
PITTSBURGH	AMERICAN	CENTRAL
SEATTLE	AMERICAN	CENTRAL
DENVER	AMERICAN	WEST
KANSAS CITY	AMERICAN	WEST
OAKLAND	AMERICAN	WEST
SAN DIEGO	AMERICAN	WEST

"RANDOM" FILE, SERIALLY READ

DALLAS	NATIONAL	EAST
NEW YORK	NATIONAL	EAST
PHILADELPHIA	NATIONAL	EAST
ST. LOUIS	NATIONAL	EAST
WASHINGTON	NATIONAL	EAST
CHICAGO	NATIONAL	CENTRAL
DETROIT	NATIONAL	CENTRAL
GREEN BAY	NATIONAL	CENTRAL
MINNESOTA	NATIONAL	CENTRAL
ATLANTA	NATIONAL	WEST
LOS ANGELES	NATIONAL	WEST
NEW ORLEANS	NATIONAL	WEST
SAN FRANCISCO	NATIONAL	WEST
TAMPA BAY	NATIONAL	WEST
BALTIMORE	AMERICAN	EAST
BUFFALO	AMERICAN	EAST
MIAMI	AMERICAN	EAST
NEW ENGLAND	AMERICAN	EAST
NEW YORK	AMERICAN	EAST
CINCINNATI	AMERICAN	CENTRAL
CLEVELAND	AMERICAN	CENTRAL
HOUSTON	AMERICAN	CENTRAL
PITTSBURGH	AMERICAN	CENTRAL
SEATTLE	AMERICAN	CENTRAL
DENVER	AMERICAN	WEST
KANSAS CITY	AMERICAN	WEST
OAKLAND	AMERICAN	WEST
SAN DIEGO	AMERICAN	WEST

ORIGINAL "NFC" FILE

DALLAS	COWBOYS	EAST
NEW YORK	GIANTS	EAST
PHILADELPHIA	EAGLES	EAST
ST. LOUIS	CARDINALS	EAST
WASHINGTON	REDSKINS	EAST
CHICAGO	BEARS	CENTRAL
DETROIT	LIONS	CENTRAL
GREEN BAY	PACKERS	CENTRAL
MINNESOTA	VIKINGS	CENTRAL
ATLANTA	FALCONS	WEST
LOS ANGELES	RAMS	WEST
NEW ORLEANS	SAINTS	WEST
SAN FRANCISCO	49ERS	WEST
TAMPA BAY	BUCCANEERS	WEST

ORIGINAL "AFC" FILE

BALTIMORE	EAST
BUFFALO	EAST
MIAMI	EAST
NEW ENGLAND	EAST
NEW YORK	EAST
CINCINNATI	CENTRAL
CLEVELAND	CENTRAL
HOUSTON	CENTRAL
PITTSBURGH	CENTRAL
SEATTLE	CENTRAL
DENVER	WEST
KANSAS CITY	WEST
OAKLAND	WEST
SAN DIEGO	WEST

UPDATED "NFC" FILE

EAST	CENTRAL	WEST
DALLAS COWBOYS	CHICAGO BEARS	ATLANTA FALCONS
NEW YORK GIANTS	DETROIT LIONS	LOS ANGELES RAMS
PHILADELPHIA EAGLES	GREEN BAY PACKERS	NEW ORLEANS SAINTS
ST. LOUIS CARDINALS	MINNESOTA VIKINGS	SAN FRANCISCO 49ERS
WASHINGTON REDSKINS		TAMPA BAY BUCCANEERS

UPDATED "AFC" FILE

EAST	CENTRAL	WEST
BALTIMORE COLTS	CINCINNATI BENGALS	DENVER BRONCOS
BUFFALO BILLS	CLEVELAND BROWNS	KANSAS CITY CHIEFS
MIAMI DOLPHINS	HOUSTON OILERS	OAKLAND RAIDERS
NEW ENGLAND PATRIOTS	PITTSBURGH STEELERS	SAN DIEGO CHARGERS
NEW YORK JETS	SEATTLE SEAHAWKS	

```
100  ! DATA BASICS EXERCISE 3    SOLUTION    27APR78
110  !
120  OPTION BASE 1                          ! LOWER BOUND OF ARRAYS IS 1.
130  DIM Numbers(64)                        ! 64 FULL-PRECISION NUMBERS.
140  RANDOMIZE                              ! CHOOSE SEED FOR RND FUNCTION.
150  ASSIGN #1 TO "TEST",Status             ! ASSIGN "TEST" FILE.
160  IF Status=1 THEN Create                ! IF 1 THEN "TEST" DOES NOT EXIST.
170  IF Status<>0 THEN Error                ! IF 0 THEN "TEST" EXISTS & IS OK.
180 Input: !
190  INPUT "USE OLD TEST DATA?  (Y OR N)",S$
200  S$=UPC$(S$[1,1])                       ! GUARANTEE UPPER CASE.
210  IF S$="Y" THEN Done                    ! IF YES THEN DONE.
220  IF S$="N" THEN Purge                   ! IF NO THEN PURGE AND RECREATE.
230  DISP "IMPROPER ENTRY.  TRY AGAIN.  ";
240  GOTO Input
250 Purge: !
260  PURGE "TEST"                           ! PURGE CURRENT DATA.
270 Create:!
280  CREATE "TEST",2                        ! RECREATE "TEST".
290  ASSIGN #1 TO "TEST"                    ! OPEN "TEST".
300  FOR N=1 TO 64                          !
310  Numbers(N)=100*RND                     ! GENERATE 64 RANDOM NUMBERS.
320  NEXT N
330  PRINT #1;Numbers(*)                    ! WRITE NUMBERS IN FILE.
340  PRINT LIN(1),"NEW TEST DATA",LIN(1)    ! PRINT TITLE.
350  PRINT USING "8(5D,4D)/";Numbers(*)     ! PRINT "TEST" DATA.
360 Done: !
370  PRINT LIN(3)                           ! SKIP 3 LINES.
380  REWIND                                 ! REWIND TAPE CARTRIDGE.
390  STOP
400 Error: !
410  DISP "FAILURE TO ASSIGN PROPERLY THE TEST FILE."
420  END
```

NEW TEST DATA

37.2190	1.1868	98.1348	31.3342	52.0846	3.1437	43.8321	62.7791
97.1952	41.1810	33.2051	25.4661	16.9737	74.8871	80.0903	55.9254
33.2419	56.4058	37.2812	53.4545	55.2689	81,1730	66.5136	37.9348
3.1717	67.4193	99.6491	4.8611	88.1641	45.9943	81.2417	24.2878
26.0072	72.0363	82.5240	2.6988	69.6726	94.6860	30.9232	6.4500
25.0854	96.8082	15.6661	75.2233	62.7959	11.3126	13.9362	20.3250
93.3438	2.1761	30.0975	12.0273	14.9683	88.3806	28.0795	14.8993
30.3202	99.2982	9.7928	35.7861	96.1190	36.0862	48.4631	57.4535

Continued--

```
USE OLD TEST DATA?  (Y OR N)
N

NEW TEST DATA

   52.19Ø1   91.8684   61.312Ø   63.397Ø   16.9Ø37   16.Ø471   95.59Ø7   91.7825
   89.Ø731   1Ø.4966   27.6Ø9Ø   19.2Ø78   53.7439   98.64Ø9   57.Ø36Ø   67.2559
   62.1817   94.8472   66.53Ø7   52.3429   2Ø.3752   35.5224   74.3217    4.5337
   12.8Ø91   72.47Ø2   47.4758   27.1519   34.7521   26.49Ø8   78.7829   56.4119
   42.3989   57.4433    9.8288   66.Ø594   55.9574   6Ø.2Ø77   34.6389   31.3242
   43.3989   85.1453    7.1747   33.9198   26.5366   17.3Ø58   54.17Ø2   57.1446
   58.6223    1.33Ø1   18.5851   3Ø.1Ø67   19.7511   1Ø.7Ø23     .6713   64.5828
   14.1224   76.8978   71.Ø686   68.6769   57.2341   33.8884     .1241    4.3731
```

```
100  ! IMPROVING SPEED EXERCISE 1    SOLUTION     5MAY78
110  !
120  ASSIGN #8 TO "SERIAL",R,"WRITE"  ! OPEN "SERIAL" FILE FOR RECORD I/O.
130  ASSIGN #9 TO "RANDOM",R,"WRITE   ! OPEN "RANDOM" FILE FOR RECORD I/O.
140  BUFFER #8                        ! SET UP FILE BUFFER FOR "SERIAL".
150  BUFFER #9                        ! SET UP FILE BUFFER FOR "RANDOM".
160  PRINT LIN(6);TAB(28);"NATIONAL FOOTBALL LEAGUE";LIN(2)
170  PRINT TAB(21);"TEAM";TAB(42);"CONFERENCE";TAB(56);"DIVISION";LIN(1)
180  FOR N=1 TO 29                    ! FOR EACH TEAM.
190  READ #8;Loc$,Name$               ! READ LOCATION AND NAME.
200  READ #9,N;Dummy$,Conf$,Div$      ! READ CONFERENCE AND DIVISION.
210  PRINT TAB(15);Loc$;" ";Name$;TAB(42);Conf$;TAB(57);Div$ ! PRINT.
220  NEXT N
230  ASSIGN #8 TO *                   ! CLOSE "SERIAL" FOR RECORD I/O.
240  ASSIGN #9 TO *                   ! CLOSE "RANDOM" FOR RECORD I/O.
250  END
```

NATIONAL FOOTBALL LEAGUE

TEAM	CONFERENCE	DIVISION
DALLAS COWBOYS	NATIONAL	EAST
NEW YORK GIANTS	NATIONAL	EAST
PHILADELPHIA EAGLES	NATIONAL	EAST
ST. LOUIS CARDINALS	NATIONAL	EAST
WASHINGTON REDSKINS	NATIONAL	EAST
CHICAGO BEARS	NATIONAL	CENTRAL
DETROIT LIONS	NATIONAL	CENTRAL
GREEN BAY PACKERS	NATIONAL	CENTRAL
MINNESOTA VIKINGS	NATIONAL	CENTRAL
ATLANTA FALCONS	NATIONAL	WEST
LOS ANGELES RAMS	NATIONAL	WEST
NEW ORLEANS SAINT	NATIONAL	WEST
SAN FRANCISCO 49ERS	NATIONAL	WEST
TAMPA BAY BUCCANEERS	NATIONAL	WEST
BALTIMORE COLTS	AMERICAN	EAST
BUFFALO BILLS	AMERICAN	EAST
MIAMI DOLPHINS	AMERICAN	EAST
NEW ENGLAND PATRIOTS	AMERICAN	EAST
NEW YORK JETS	AMERICAN	EAST
CINCINNATI BENGALS	AMERICAN	CENTRAL
CLEVELAND BROWNS	AMERICAN	CENTRAL
HOUSTON OILERS	AMERICAN	CENTRAL
PITTSBURGH STEELERS	AMERICAN	CENTRAL
SEATTLE SEAHAWKS	AMERICAN	CENTRAL
DENVER BRONCOS	AMERICAN	WEST
KANSAS CITY CHIEFS	AMERICAN	WEST
OAKLAND RAIDERS	AMERICAN	WEST
SAN DIEGO CHARGERS	AMERICAN	WEST

```
100  ! IMPROVING SPEED EXERCISE 2    SOLUTION    5MAY78
110  !
120  OPTION BASE 1                   ! LOWER BOUND OF ARRAYS IS 1.
130  DIM Team$(28,2)                 ! TEAM LOCATIONS AND NAMES.
140  DIM Conf$(28,2)                 ! TEAM LOCATIONS, CONFERENCES, DIVISIONS.
150  ASSIGN #8 TO "SERIAL",R,"WRITE" ! OPEN "SERIAL" FILE FOR RECORD I/O.
160  ASSIGN #9 TO "RANDOM",R,"WRITE" ! OPEN "RANDOM" FILE FOR RECORD I/O.
170  READ #8;Team$(*)                ! READ LOCATIONS AND NAMES.
180  READ #9;Conf$(*)                ! READ LOCATIONS, CONFERENCES, DIVISIONS.
190  PRINT LIN(6);TAB(28);"NATIONAL FOOTBALL LEAGUE";LIN(2)
200  PRINT TAB(21);"TEAM";TAB(41);"CONFERENCE";TAB(56);"DIVISION";LIN(1)
210  FOR N=1 TO 28                   ! PRINT DATA.
220  PRINT TAB(15);Team$(N,1);" ";Team$(N,2);TAB(42);Conf$(N,2);TAB(57);Conf$(N,
3)
230  NEXT N
240  ASSIGN #8 TO *                  ! CLOSE "SERIAL" FOR RECORD I/O.
250  ASSIGN #9 TO *                  ! CLOSE "RANDOM" FOR RECORD I/O.
260  END
```

NATIONAL FOOTBALL LEAGUE

TEAM	CONFERENCE	DIVISION
DALLAS COWBOYS	NATIONAL	EAST
NEW YORK GIANTS	NATIONAL	EAST
PHILADELPHIA EAGLES	NATIONAL	EAST
ST. LOUIS CARDINALS	NATIONAL	EAST
WASHINGTON REDSKINS	NATIONAL	EAST
CHICAGO BEARS	NATIONAL	CENTRAL
DETROIT LIONS	NATIONAL	CENTRAL
GREEN BAY PACKERS	NATIONAL	CENTRAL
MINNESOTA VIKINGS	NATIONAL	CENTRAL
ATLANTA FALCONS	NATIONAL	WEST
LOS ANGELES RAMS	NATIONAL	WEST
NEW ORLEANS SAINT	NATIONAL	WEST
SAN FRANCISCO 49ERS	NATIONAL	WEST
TAMPA BAY BUCCANEERS	NATIONAL	WEST
BALTIMORE COLTS	AMERICAN	EAST
BUFFALO BILLS	AMERICAN	EAST
MIAMI DOLPHINS	AMERICAN	EAST
NEW ENGLAND PATRIOTS	AMERICAN	EAST
NEW YORK JETS	AMERICAN	EAST
CINCINNATI BENGALS	AMERICAN	CENTRAL
CLEVELAND BROWNS	AMERICAN	CENTRAL
HOUSTON OILERS	AMERICAN	CENTRAL
PITTSBURGH STEELERS	AMERICAN	CENTRAL
SEATTLE SEAHAWKS	AMERICAN	CENTRAL
DENVER BRONCOS	AMERICAN	WEST
KANSAS CITY CHIEFS	AMERICAN	WEST
OAKLAND RAIDERS	AMERICAN	WEST
SAN DIEGO CHARGERS	AMERICAN	WEST

```
10   ! ADDITIONAL CAPABILITIES EXERCISE 1    SOLUTION    5MAY78
20   !
30   DIM Line$[30]                          ! ONE LINE OF POEM.
40   ASSIGN #10 TO "POEM"                   ! OPEN "POEM" FOR RECORD I/O.
50   ON END #10 GOTO Done                   ! SET UP EOF BRANCH.
60 Loop:  READ #10;Line$                    ! SERIALLY READ A LINE.
70   PRINT Line$                            ! PRINT THE LINE.
80   GOTO Loop                              ! CONTINUE.
90 Done:  END
```

```
IF IN THE COURSE OF TIME
YOUR PROGRAM RUNS JUST FINE
THEN NO HARSH BEEP
YOUR EARS WILL GREET
AND SWEET SUCCESS IS THINE.
```

```
100  ! ADDITIONAL CAPABILITIES EXERCISE 2    SOLUTION    5MAY78
110  !
120  INTEGER N
130  ASSIGN #1 TO "MESSAG"                  ! OPEN "MESSAGE" FILE
140  PRINT LIN(3)                           ! SKIP 3 LINES.
150 Loop: !
160  IF TYP(1)=5 THEN Integer               ! CHECK FOR INTEGER.
170  IF TYP(1)=2 THEN String                ! CHECK FOR STRING.
180  IF TYP(1)=3 THEN Done                  ! CHECK FOR END-OF-FILE.
190 Error:  PRINT "*****  ERROR  *****"     ! IF NONE OF THE ABOVE THEN ERROR.
200  PAUSE                                  ! HALT EXECUTION.
210 Integer:  READ #1;N                     ! READ INTEGER DATUM.
220  PRINT USING "#,D";N                    ! PRINT INTEGER WITH CRLF SUPPRESSION.
230  GOTO Loop                              ! GO TO NEXT  DATUM.
240 String:  READ #1;N$                     ! READ STRING DATUM.
250  PRINT N$;                              ! PRINT STRING WITH CRLF SUPPRESSION.
260  GOTO Loop                              ! GO TO NEXT DATUM.
270 Done: PRINT                             ! PRINT MESSAGE.
280  REWIND                                 ! REWIND TAPE CARTRIDGE.
290  END
```

```
1 CAN GO 2 FAR 4 INTERESTING D8A.
```

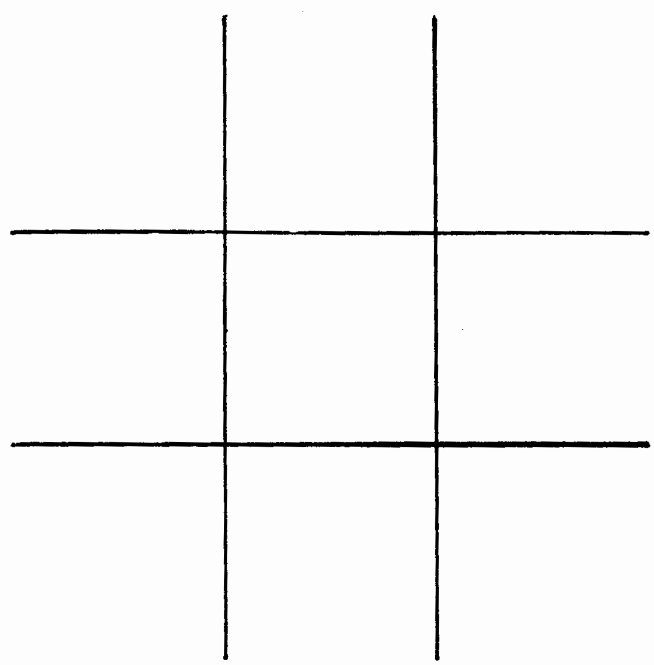
```
1Ø0  ! DATA BASICS EXERCISE 2    DATA CREATION    4MAY78
11Ø  !
12Ø  DATA DALLAS,COWBOYS,NATIONAL,EAST
13Ø  DATA NEW YORK,GIANTS,NATIONAL,EAST
14Ø  DATA PHILADELPHIA,EAGLES,NATIONAL,EAST
15Ø  DATA ST. LOUIS,CARDINALS,NATIONAL,EAST
16Ø  DATA WASHINGTON,REDSKINS,NATIONAL,EAST
17Ø  DATA CHICAGO,BEARS,NATIONAL,CENTRAL
18Ø  DATA DETROIT,LIONS,NATIONAL,CENTRAL
19Ø  DATA GREEN BAY,PACKERS,NATIONAL,CENTRAL
2ØØ  DATA MINNESOTA,VIKINGS,NATIONAL,CENTRAL
21Ø  DATA ATLANTA,FALCONS,NATIONAL,WEST
22Ø  DATA LOS ANGELES,RAMS,NATIONAL,WEST
23Ø  DATA NEW ORLEANS,SAINTS,NATIONAL,WEST
24Ø  DATA SAN FRANCISCO,49ERS,NATIONAL,WEST
25Ø  DATA TAMPA BAY,BUCCANEERS,NATIONAL,WEST
26Ø  DATA BALTIMORE,COLTS,AMERICAN,EAST
27Ø  DATA BUFFALO,BILLS,AMERICAN,EAST
28Ø  DATA MIAMI,DOLPHINS,AMERICAN,EAST
29Ø  DATA NEW ENGLAND,PATIROTS,AMERICAN,EAST
3ØØ  DATE NEW YORK,JETS,AMERICAN,EAST
31Ø  DATA CINCINNATI,BENGALS,AMERICAN,CENTRAL
32Ø  DATA CLEVELAND,BROWNS,AERMICAN,CENTRAL
33Ø  DATA HOUSTON,OILERS,AMERICAN,CENTRAL
34Ø  DATA PITTSBURGH,STEELERS,AMERICAN,CENTRAL
35Ø  DATA SEATTLE,SEAHAWKS,AMERICAN,CENTRAL
36Ø  DATA DENVER,BRONCOS,AMERICAN,WEST
37Ø  DATA KANSAS CITY,CHIEFS,AMERICAN,WEST
38Ø  DATA OAKLAND,RAIDERS,AMERICAN,WEST
39Ø  DATA SAN DIEGO,CHARGERS,AMERICAN,WEST
4ØØ  !
41Ø  CREATE "SERIAL",8           ! CREATE "SERIAL" FILE TO BE 8 256-BYTE RECORDS.
42Ø  CREATE "RANDOM",32,64       ! CREATE "RANDOM" FILE TO BE 32 64-BYTE RECORDS.
43Ø  ASSIGN #1 TO "SERIAL"       ! OPEN "SERIAL" FOR RECORD I/O.
44Ø  ASSIGN #2 TO "RANDOM"       ! OPEN "RANDOM" FOR RECORD I/O.
45Ø  BUFFER #1                   ! CREATE FILE BUFFER FOR "SERIAL".
46Ø  BUFFER #2                   ! CREATE FILE BUFFER FOR "RANDOM".
47Ø  FOR I=1 TO 28               ! FOR EACH OF TEAMS,
48Ø  READ Loc$,Name$,Conf$,Div$ ! READ LOCATION, NAME, CONFERENCE, DIVISION.
49Ø  PRINT #1;Loc$,Name$        ! SERIALLY PRINT LOCATION AND NAME.
5ØØ  PRINT #2,I;Loc$,Conf$,Div$ ! RANDOMLY PRINT LOCATION, CONFERENCE AND DIVISION.
51Ø  NEXT I
52Ø  PRINT #1;END                ! PRINT END-OF-FILE IN "SERIAL".
53Ø  PRINT #2,I;END              ! PRINT END-OF-FILE IN "RANDOM" AT NEXT RECORD.
54Ø  ASSIGN #1 TO *              ! CLOSE "SERIAL" TO DUMP BUFFERS.
55Ø  ASSIGN #2 TO *              ! CLOSE "RANDOM" TO DUMP BUFFERS.
56Ø  PROTECT "SERIAL","WRITE"    ! PROTECT "SERIAL".
57Ø  PROTECT "RANDOM","WRITE"    ! PROTECT "RANDOM".
58Ø  REWIND                      ! REWIND TAPE CARTRIDGE.
59Ø  END
```

```
100  ! IMPROVING SPEED EXERCISE 1    PROGRAM    5MAY78
110  !
120  ASSIGN #8 TO "SERIAL",R,"WRITE" ! OPEN "SERIAL" FILE FOR RECORD I/O.
130  ASSIGN #9 TO "RANDOM",R,"WRITE" ! OPEN "RANDOM" FILE FOR RECORD I/O.
140  PRINT LIN(6);TAB(28);"NATIONAL FOOTBALL LEAGUE";LIN(2)
150  PRINT TAB(21);"TEAM";TAB(41);"CONFERENCE";TAB(56);"DIVISION";LIN(1)
160  FOR N=1 TO 28                   ! FOR EACH TEAM,
170  READ #8;Loc$,Name$              ! READ LOCATION AND NAME.
180  READ #9,N;Dummy$,Conf$,Div$     ! READ CONFERENCE AND DIVISION.
190  PRINT TAB(15);Loc$;" ";Name$;TAB(42);Conf$;TAB(57);Div$ ! PRINT.
200  NEXT N
210  ASSIGN #8 TO *                  ! CLOSE "SERIAL" FOR RECORD I/O.
220  ASSIGN #9 TO *                  ! CLOSE "RANDOM" FOR RECORD I/O.
230  END
```

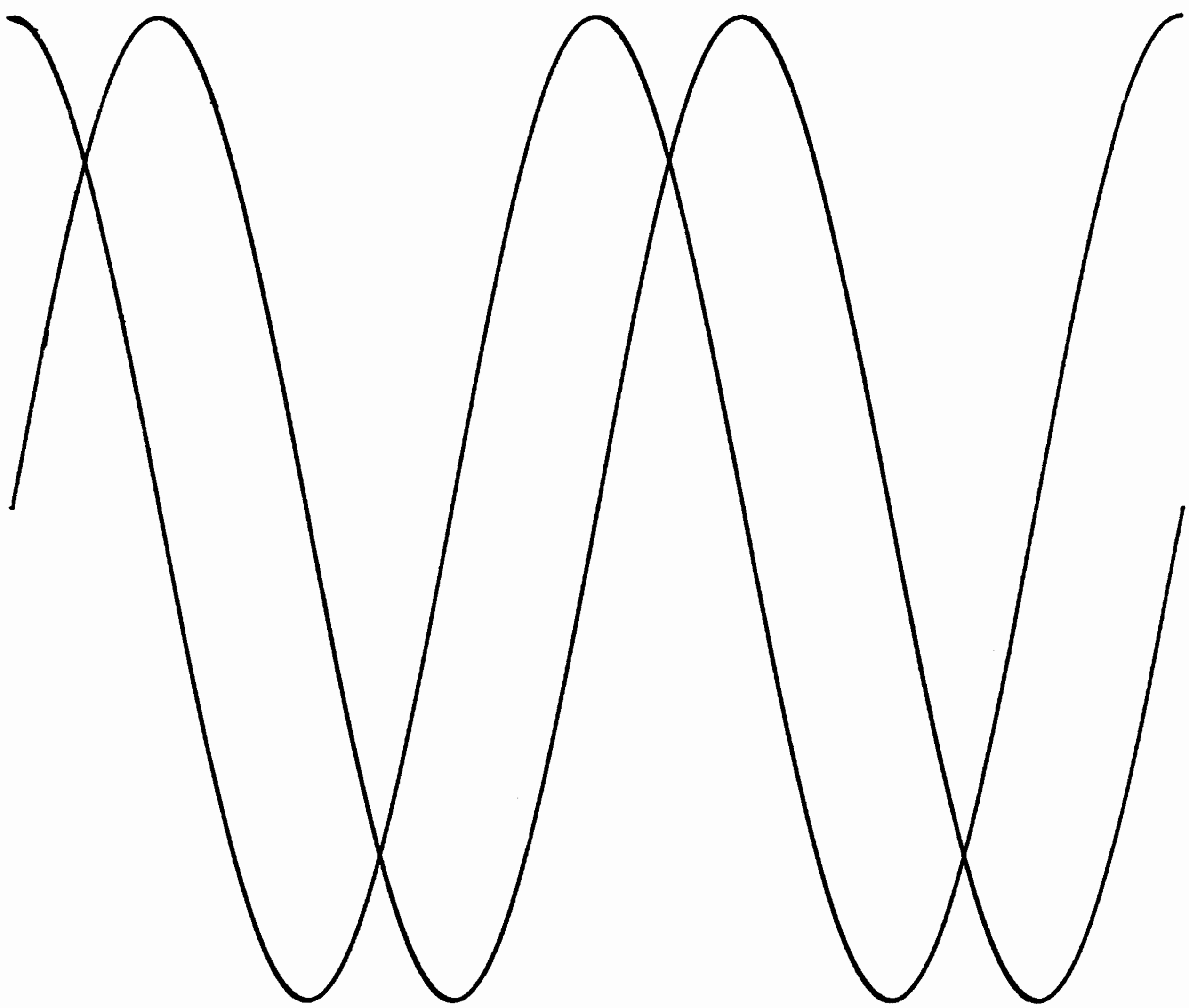
```
10   ! ADDITIONAL CAPABILITIES EXERCISE 1   DATA CREATION   5MAY78
20   !
21   DIM Line$[30]                          ! ONE LINE OF POEM.
30   DATA IF IN THE COURSE OF TIME          ! DATA
40   DATA YOUR PROGRAM RUNS JUST FINE
50   DATA THEN NO HARSH BEEP
60   DATA YOUR EARS WILL GREET
70   DATA AND SWEET SUCCESS IS THINE.
80   ASSIGN #10 TO "POEM",R                 ! SEE WHETHER "POEM" EXISTS.
90   IF R<>1 THEN Print                     ! IF SO, SKIP TO PRINT.
100  CREATE "POEM",1                        ! ONE 256-BYTE DEFINED RECORD.
110  ASSIGN #10 to "POEM"                   ! OPEN "POEM" FOR RECORD I/O.
120 Print:  !
130  FOR N=1 TO 5
140  READ Line$                             ! READ ONE LINE OF THE POEM.
150  PRINT #10;Line$                        ! SERIALLY PRINT THE LINE.
160  NEXT N
170  ASSIGN #10 TO *                        ! CLOSE "POEM".
180  END
```

```
100  ! ADDITIONAL CAPABILITIES EXERCISE 2   DATA CREATION    5MAY78
110  !
120  DATA Ø,1,1," CAN ",1,"GO ",Ø,2,1," FAR "  ! MESSAGE IN DATA STATEMENTS.
130  DATA Ø,4,1," INTERESTING ",1,"D",Ø,8,1,"A."
140  INTEGER N
150  ASSIGN #1 TO "MESSAG",R                 ! CHECK WHETHER "MESSAG" EXISTS.
160  IF R  1 THEN Data                       ! IF SO THEN SKIP CREATION.
170  CREATE "MESSAGE",1                      ! ONE 256-BYTE DEFINED RECORD.
180  ASSIGN #1 TO "MESSAG"                   ! OPEN "MESSAG" FILE.
190 Data: !
200  FOR I=1 TO 1Ø                           ! PUT 1Ø ITEMS IN THE FILE.
210  READ Type                               ! READ ITEM TYPE.
220  IF Type=1 THEN String                   ! Ø=INTEGER.  1=STRING.
230 Integer:!
240  READ N                                  ! READ INTEGER.
250  PRINT #1;N                              ! PRINT INTEGER IN FILE.
260  GOTO Nexti                              ! GO TO NEXT DATUM.
270 String: !
280  READ N$                                 ! READ STRING.
290  PRINT #1;N$                             ! PRINT STRING IN FILE.
300 Nexti: NEXT I                            ! GO TO NEXT DATUM.
310  REWIND                                  ! REWIND TAPE CARTRIDGE.
320  END
```

```
100  ! ABSOLUTE PLOTTING EXERCISE 1       SOLUTION    18MAY78
110  !
120  PLOTTER IS "GRAPHICS"              ! SELECT CRT FOR PLOTTER.  INITIALIZE.
130  GRAPHICS                           ! SELECT GRAPHICS MODE FOR CRT.
140  MOVE 20,60                         ! FOR EACH LINE, MOVE TO START POINT
150  DRAW 80,60                         ! AND THEN DRAW TO END POINT.
160  MOVE 80,40
170  DRAW 20,40
180  MOVE 40,20
190  DRAW 40,80
200  MOVE 60,80
210  DRAW 60,20
220  PENUP                              ! LIFT PEN.
230  END
```



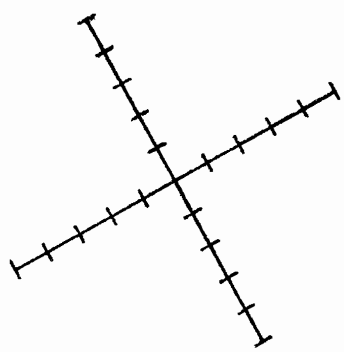
```
1ØØ  ! ABSOLUTE PLOTTING EXERCISE 2    SOLUTION    18MAY78
11Ø  !
12Ø  PLOTTER IS "GRAPHICS"              ! CHOOSE CRT FOR PLOTTER.  INITIALIZE.
13Ø  GRAPHICS                           ! CHOOSE CRT GRAPHICS MODE.
14Ø  DEG                                ! SET CURRENT ANGLE UNITS TO DEGREES.
15Ø  FOR Angle=Ø TO 72Ø                 ! LET Angle VARY FROM Ø TO 72Ø DEGREES.
16Ø  PLOT Angle/6,5Ø*(1+SIN(Angle))     ! PLOT POINT ON SINE WAVE.
17Ø  NEXT Angle
18Ø  PENUP                              ! LIFT PEN
19Ø  FOR Angle=Ø TO 27Ø                 ! LET Angle VARY FROM Ø TO 72Ø DEGREES.
2ØØ  PLOT Angle/6,5Ø*(1+COS(Angle))     ! PLOT POINT ON COSINE WAVE.
21Ø  NEXT Angle
22Ø  PENUP                              ! LIFT PEN
23Ø  END
```



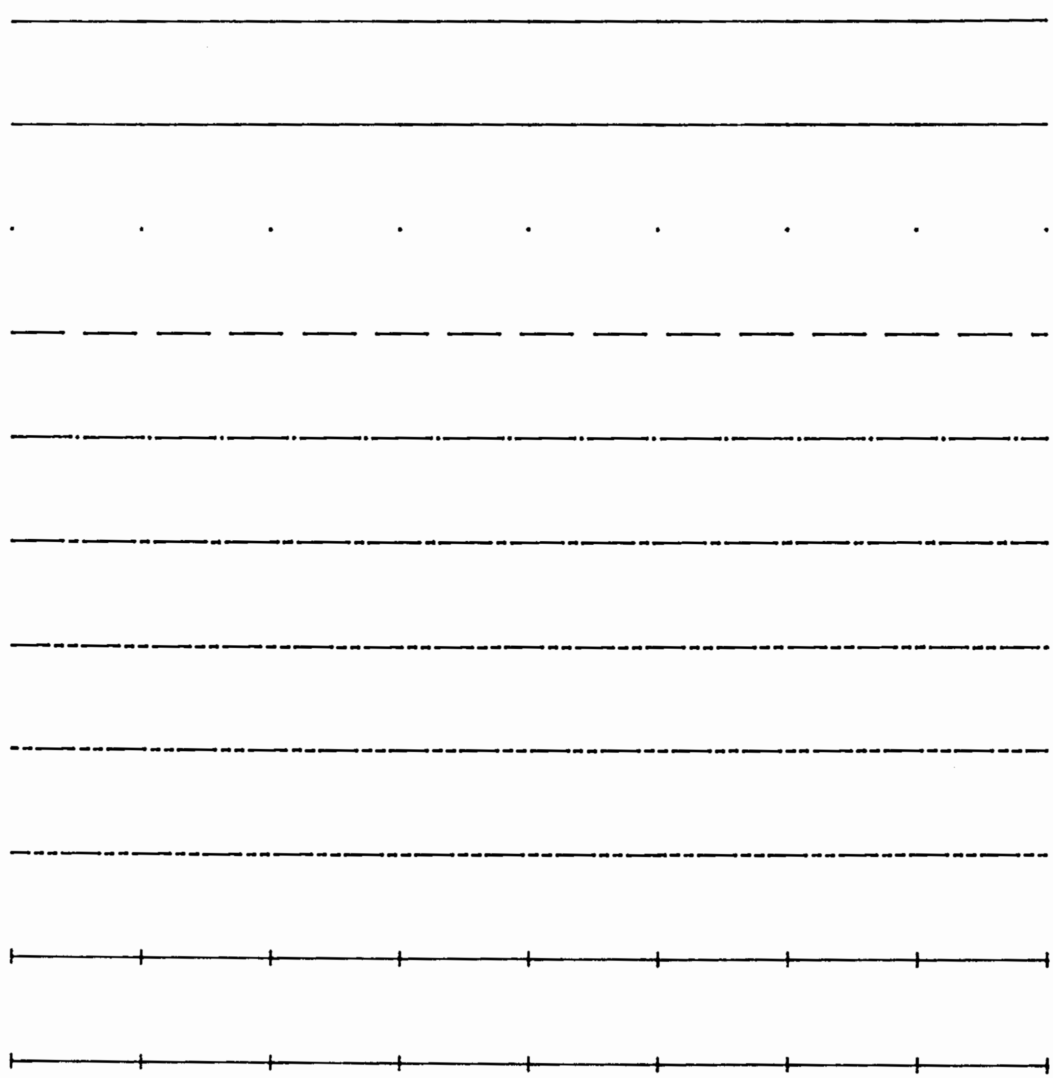
```
1ØØ   ! RELATIVE PLOTTING EXERCISES 2    SOLUTION   18MAY78
11Ø   !
12Ø   PLOTTER IS "GRAPHICS"          ! CHOOSE CRT AS PLOTTER.  INITIALIZE.
13Ø   GRAPHICS                       ! SELECT GRAPHICS MODE.
14Ø   CALL Tri(2Ø,4Ø,2Ø)             ! DRAW TRI. CORNER=(2Ø,4Ø).  SIDE=2Ø GDUS.
15Ø   CALL Tri(1Ø,35,35)             ! DRAW TRI. CORNER=(1Ø,35).  SIDE=35 GDUS.
16Ø   CALL Tri(7Ø,3Ø,4Ø)             ! DRAW TRI. CORNER=(7Ø,3Ø).  SIDE=4Ø GDUS.
17Ø   CALL Tri(6Ø,25,55)             ! DRAW TRI. CORNER=(6Ø,25).  SIDE=55 GDUS.
18Ø   END
19Ø   !
2ØØ   SUB Tri(A,B,L)                 ! SUBROUTINE TO DRAW RIGHT TRIANGLE.
21Ø   PENUP                          ! LIFT PEN.
22Ø   PLOT A,B                       ! PLOT TO LOWER LEFT CORNER.
23Ø   IPLOT L,Ø                      ! PLOT TO LOWER RIGHT CORNER.
24Ø   IPLOT Ø,L                      ! PLOT TO UPPER CORNER.
25Ø   IPLOT -L,-L                    ! PLOT TO LOWER LEFT CORNER.
26Ø   PENUP                          ! LIFT PEN.
27Ø   SUBEND                         ! RETURN.
```

```
100  ! RELATIVE PLOTTING EXERCISE 3    SOLUTION    18MAY78
110  !
120  PLOTTER IS "GRAPHICS"             ! CHOOSE CRT FOR PLOTTER.  INITIALIZE.
130  GRAPHICS                          ! SELECT GRAPHICS MODE FOR CRT.
140  DEG                               ! SET CURRENT ANGLE UNITS TO DEGREES.
150  INPUT "ANGLE (DEGREES)?",Angle    ! ENTER ROTATION ANGLE FROM KEYBOARD.
160  FOR D=Angle TO Angle+27Ø STEP 9Ø ! DRAW FOUR AXES SECTIONS FROM CENTER OUT.
170  PDIR D                            ! PLOT DIR.=ROT.ANG.+ ONE OF (Ø,9Ø,18Ø,27Ø)
180  MOVE 6Ø,5Ø                        ! MOVE TO CROSSPOINT OF AXES.
190  FOR I=1 TO 5                      ! FOR FIVE SEGMENTS,
200  GOSUB Segment                     ! DRAW LINE SEGMENT FOLLOWED BY TIC.
210  NEXT I                            ! NEXT SEGMENT.
220  NEXT D                            ! NEXT SECTION.
230  END
240 Segment:  IPLOT 4,Ø,-1             ! DRAW LINE SEGMENT.
250  RPLOT Ø,1                         ! DRAW UPPER PART OF TIC.
260  RPLOT 0,-1                        ! DRAW LOWER PART OF TIC.
270  RPLOT Ø,Ø                         ! MOVE TO CENTER OF TIC.
280  RETURN                            ! RETURN TO CALLING PROGRAM.
```

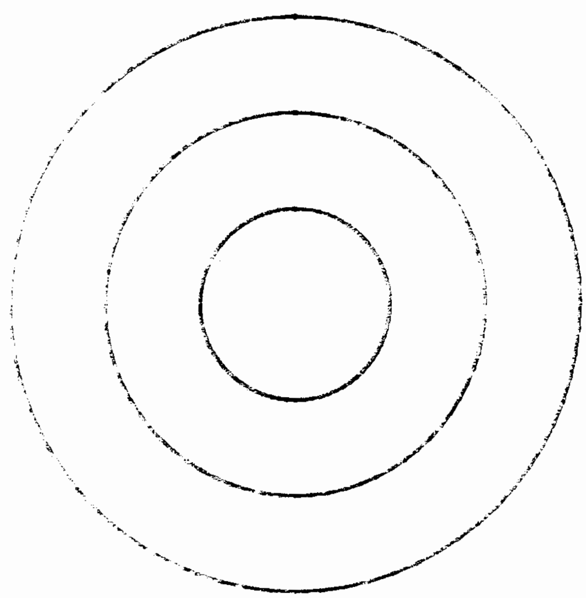
```
ANGLE (DEGREES)?
3Ø
```



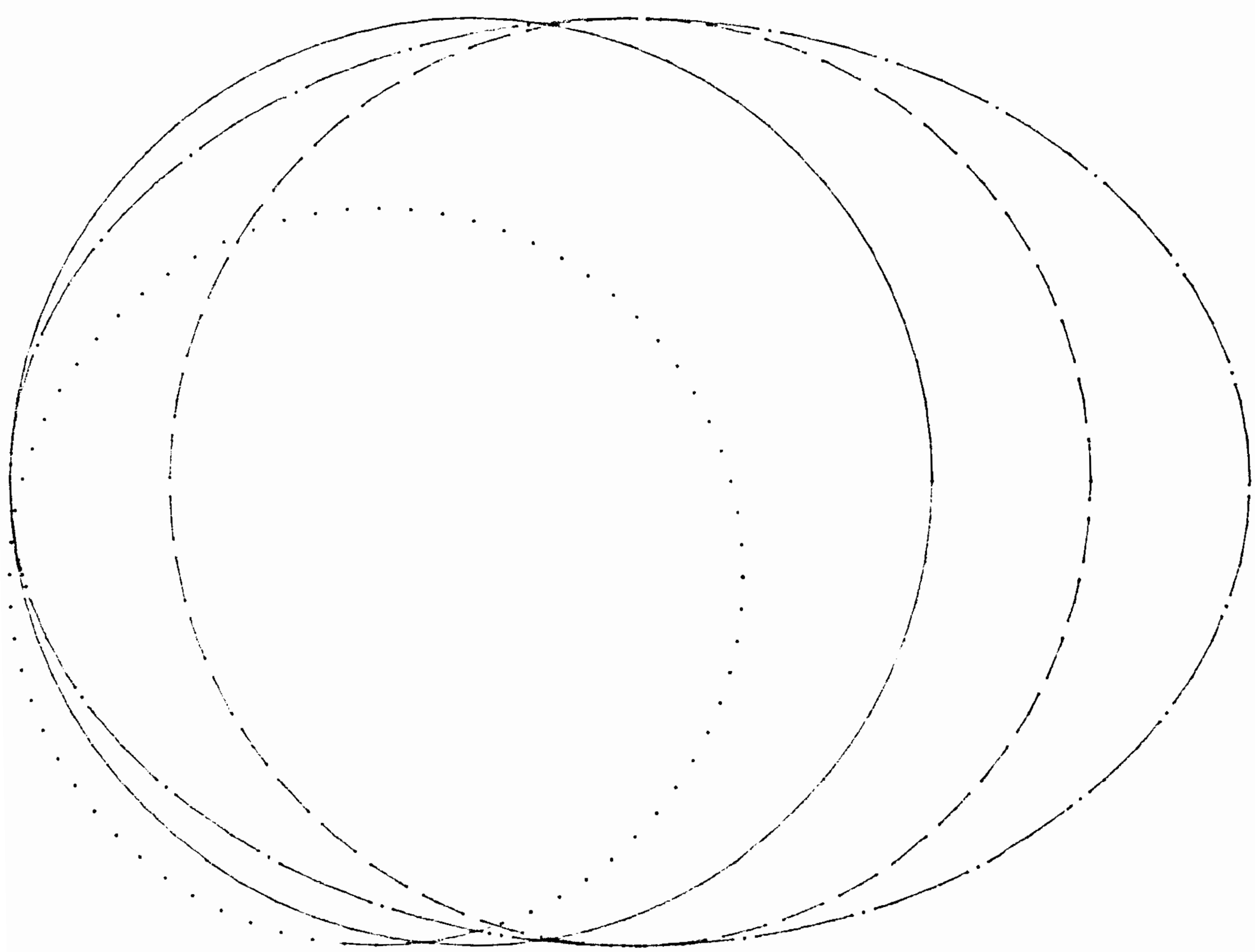
```
100  !  LINE DIFFERENTIATION EXERCISE 1    SOLUTION    18MAY78
110  !
120  PLOTTER IS "GRAPHICS"            ! CHOOSE CRT FOR PLOTTER, INITIALIZE.
130  GRAPHICS                         ! SELECT GRAPHICS MODE FOR CRT.
140  L=-1                             ! INITIALIZE LINE TYPE.
150  FOR Y=88 TO 8 STEP -8             ! DRAW 11 LINES.
160  L=L+1                            ! USE 11  DIFFERENT LINE TYPES.
170  LINE TYPE L                      ! SELECT LINE TYPE.
180  FOR X=10 TO 100 STEP 10          ! PLOT 11 POINTS IN EACH LINE TYPE.
190  PLOT X,Y                         ! DRAW TO EACH POINT.
200  NEXT X
210  PENUP                            ! LIFT PEN AT END OF EACH LINE.
220  NEXT Y
230  LINE TYPE 0
240  END
```



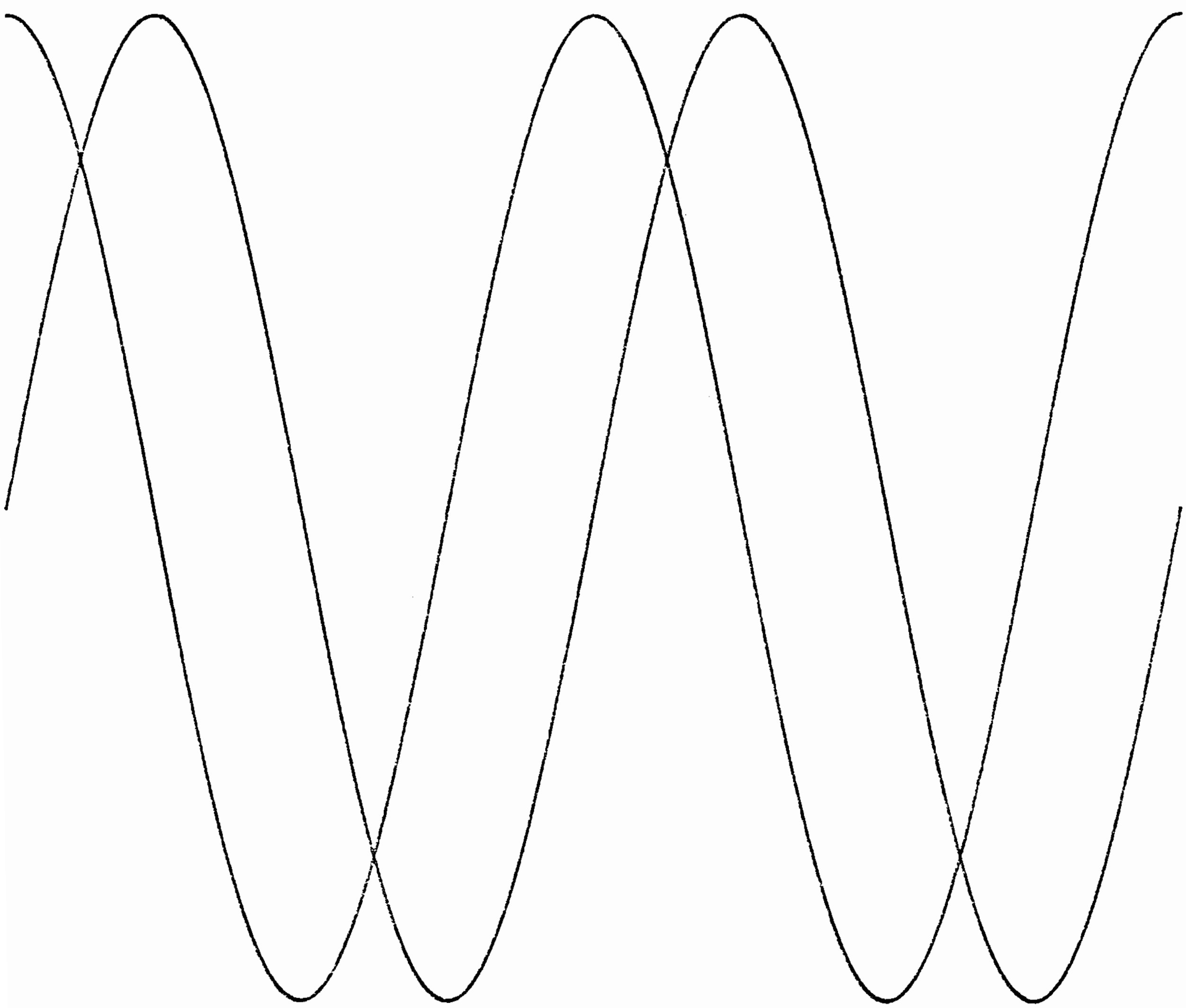
```
100  ! LINE DIFFERENTIATION EXERCISE 2    SOLUTION    18MAY78
110  !
120  PLOTTER IS "GRAPHICS"             ! CHOOSE CRT FOR PLOTTER.  INITIALIZE.
130  GRAPHICS                          ! SELECT GRAPHICS MODE.
140  FOR Radius=10 TO 30 STEP 10       ! DRAW 3 CIRCLES WITH RADII 10, 20, 30.
150  CALL Circle(60,50,Radius)
160  NEXT Radius
170  PEN -1                            ! SWITCH TO LINE ERASURE.
180  FOR Radius=10 TO 30 STEP 10       ! ERASE THE 3 CIRCLES.
190  CALL Circle(60,50,Radius)
200  NEXT Radius
210  PEN 1                             ! SWITCH TO LINE GENERATION.
220  END
250  SUB Circle(A,B,R)                 ! SUBROUTINE TO DRAW CIRCLE.
260  DEG                               ! SET CURRENT ANGLE UNITS TO DEGREES.
270  MOVE A+R,B                        ! MOVE TO ZERO DEGREE POSITION.
280  FOR D=0 TO 360 STEP 5             ! IN 5 DEGREE INCREMENTS,
290  DRAW H+R*COS(D),B+R*SIN(D)        ! DRAW STRAIGHT LINE SEGMENTS TO
300  NEXT D                            ! POINT ON THE CIRCLE.
310  PENUP                             ! LIFT PEN.
320  SUBEND                            ! RETURN.
```



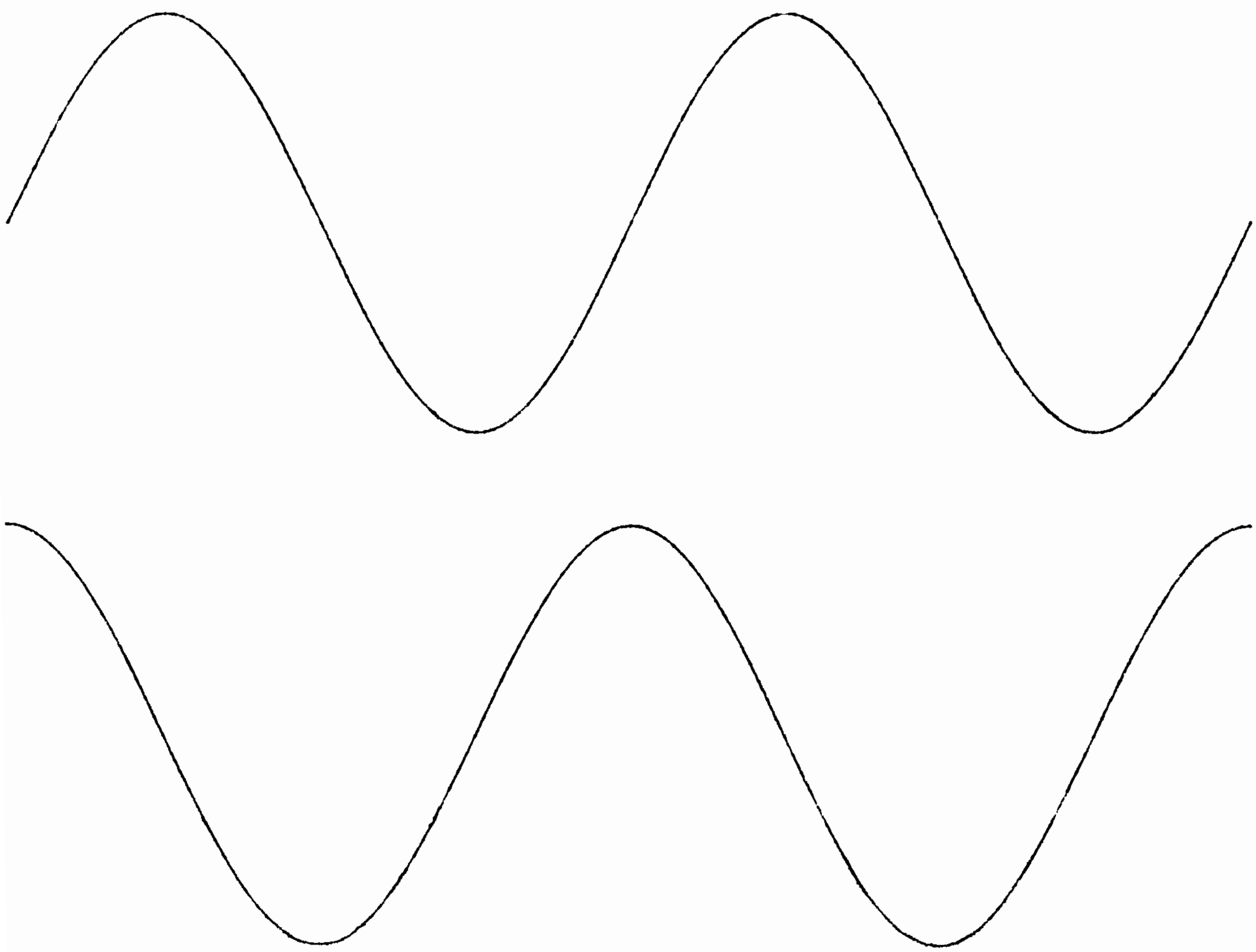
```
100  ! SCALING EXERCISE 1    SOLUTION    18MAY78
110  !
120  PLOTTER IS "GRAPHICS"            ! INITIALIZE.
130  GRAPHICS                         ! SET GRAPHICS MODE.
140  CALL Circle(50,50,50)            ! DRAW CIRCLE WITH SOLID LINE.
150  MSCALE 0,0                       ! SCALE IN MM.
160  LINE TYPE 2                      ! SELECT DOTTED LINE.
170  CALL Circle(50,50,50)            ! DRAW CIRCLE WITH DOTTED LINE.
180  SHOW 0,100,0,100                 ! SCALE WITH 1 X=1 Y.
190  LINE TYPE 3,80                   ! SELECT DASHED LINE.
200  CALL Circle(50,50,50)            ! DRAW CIRCLE WITH DASHED LINE.
210  SCALE 0,100,0,100                ! SCALE WITH 1 X#1 Y.
220  LINE TYPE 4                      ! SELECT LONG-SHORT LINE.
230  CALL Circle(50,50,50)            ! DRAW CIRCLE WITH LONG-SHORT LINE.
240  END
250  ! CIRCLE SUBROUTINE    STUDENT CARTRIDGE    18MAY78
260  !
270  SUB Circle(A,B,R)                ! SUBROUTINE TO DRAW CIRCLE.
280  DEG                              ! SET CURRENT ANGLE UNITS TO DEGREES.
290  MOVE A+R,B                       ! MOVE TO ZERO DEGREE POSITION.
300  FOR D=0 TO 360 STEP 5            ! IN 5 DEGREE INCREMENTS,
310  DRAW A+R*COS(D),B+R*SIN(D)       ! DRAW STRAIGHT LINE SEGMENTS TO
320  NEXT D                           ! POINT ON THE CIRCLE.
330  PENUP                            ! LIFT PEN.
340  SUBEND                           ! RETURN.
```



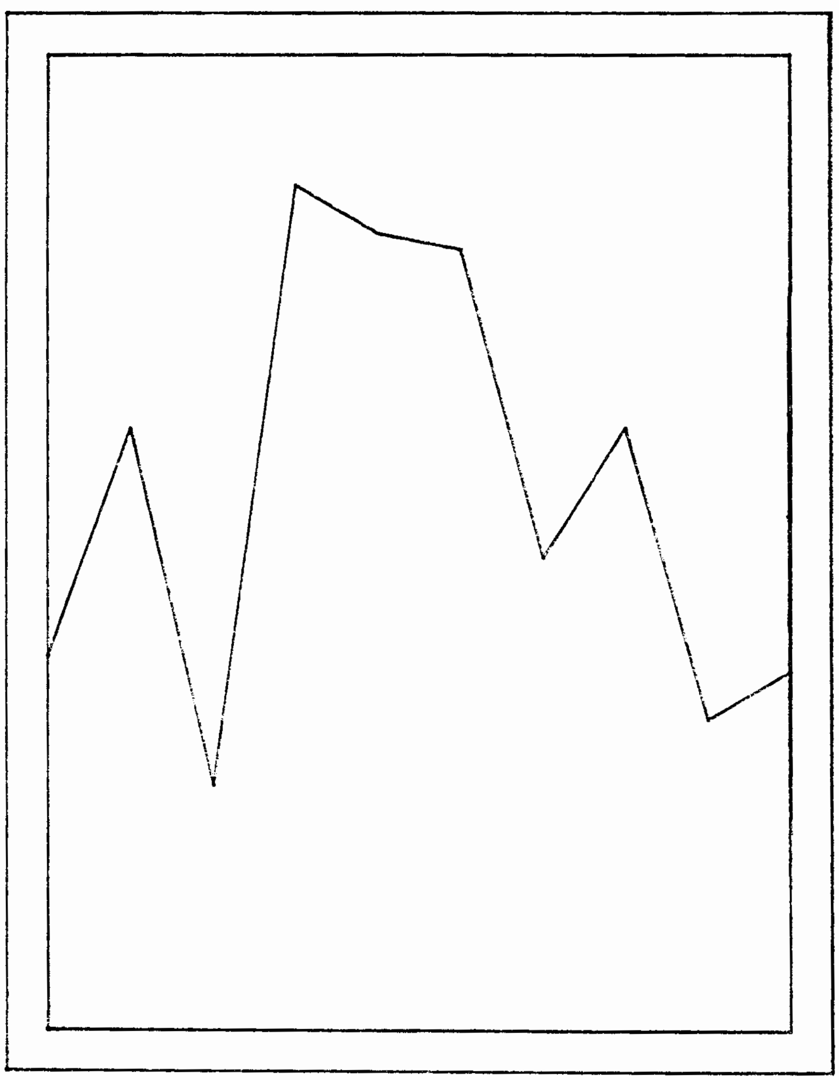
```
80   ! SCALING EXERCISE 2     SOLUTION     18MAY78
90   !
100  PLOTTER IS "GRAPHICS"            ! INITIALIZE.
110  GRAPHICS                         ! SET GRAPHICS MODE.
120  DEG                              ! SET CURRENT UNITS MODE TO DEGREES.
130  SCALE 0,720,-1,1                 ! SCALE IN PROBLEM UNITS.
140  FOR X=0 TO 720 STEP 5            ! DRAW TWO CYCLES OF A SINE WAVE
150  PLOT X,SIN(X)                    !   IN FIVE DEGREE INCREMENTS.
160  NEXT X
170  PEN UP                           ! LIFT PEN.
180  FOR X=0 TO 720 STEP 5            ! DRAW TWO CYCLES OF A COSINE WAVE
190  PLOT X,COS(X)                    ! IN FIVE DEGREE INCREMENTS.
200  NEXT X
210  PENUP                            ! LIFT PEN.
220  END
```



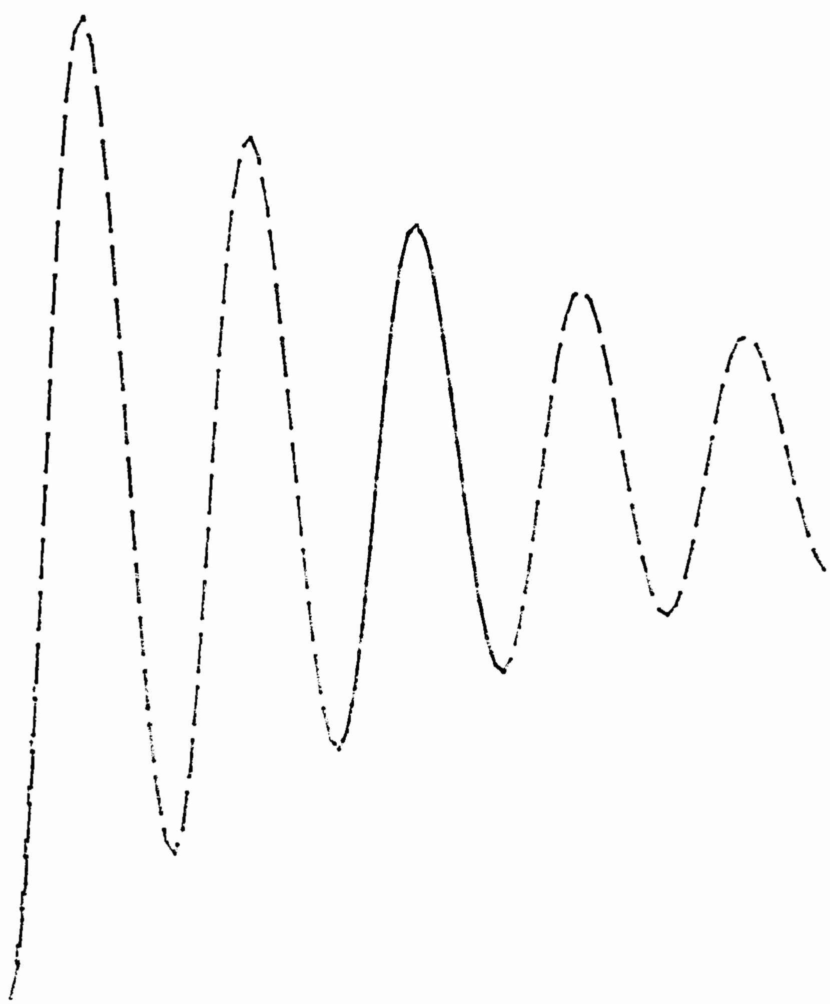
```
100  ! SCALING EXERCISE 3    SOLUTION    18MAY78
110  !
120  PLOTTER IS "GRAPHICS"           ! INITIALIZE.
130  GRAPHICS                        ! SET GRAPHICS MODE.
140  DEG                             ! SET CURRENT UNITS MODE TO DEGREES.
150  LOCATE Ø,1ØØ*RATIO,55,1ØØ       ! SELECT TOP PORTION OF SCREEN FOR SINE CURVE.
160  SCALE Ø,72Ø,-1,1                ! SCALE IN PROBLEM UNITS.
170  FOR X=Ø TO 72Ø STEP 5           ! DRAW TWO CYCLES OF A SINE WAVE
180  PLOT X,SIN(X)                   !   IN FIVE DEGREE INCREMENTS.
190  NEXT X
200  PENUP                           ! LIFT PEN
210  LOCATE Ø,1ØØ*RATIO,Ø,45         ! SELECT BOTTOM PORTION OF SCREEN FOR COS CURVE.
220  SCALE Ø,72Ø,-1,1                ! SCALE IN PROBLEM UNITS.
230  FOR X=Ø TO 72Ø STEP 5           ! DRAW TWO CYCLES OF A COSINE WAVE
240  PLOT X,COS(X)                   ! IN FIVE DEGREE INCREMENTS.
250  NEXT X
260  PENUP                           ! LIFT PEN.
270  END
```



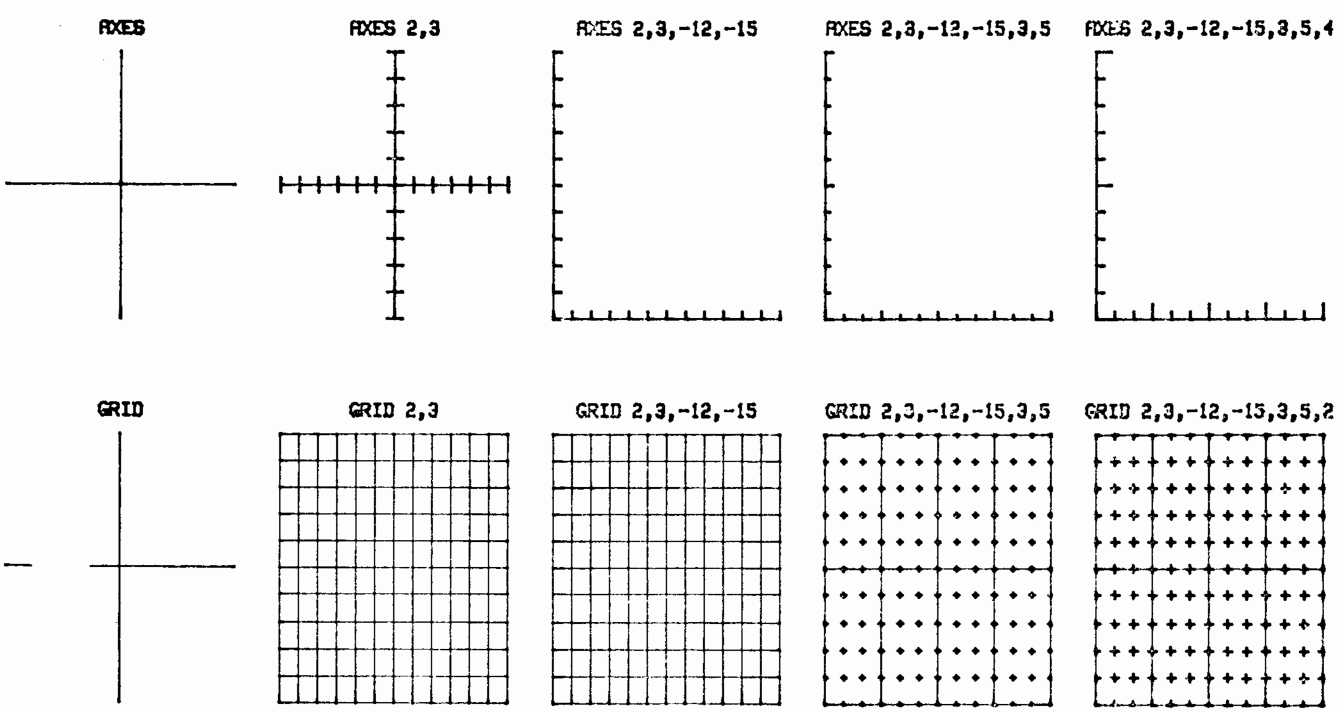
```
100 ! LIMITS EXERCISE 1    SOLUTION    18MAY78
110 !
120 PLOTTER IS "GRAPHICS"              ! INITIALIZE.
130 GRAPHICS                           ! SET GRAPHICS MODE.
140 LIMIT 0,110,0,140                  ! PROPORTION TO 220 X 280.
150 FRAME
160 LOCATE 5,95,5,100/RATIO-5          ! SET 5% MARGINS.
170 FRAME
180 SCALE 1969,1978,6,12               ! SCALE APPROPRIATELY.
190 FOR Year=1969 TO 1978              ! FOR YEARS 1968 THROUGH 1978.
200 READ Rainfall                      ! READ RAINFALL AND
210 PLOT Year,Rainfall                 ! PLOT YEAR VERSUS RAINFALL.
220 NEXT Year
230 PENUP
240 DATA 8,3,9,7,7,5,11,2              ! RAINFALL DATA.
250 DATA 10,9,10,8,8,9,9,7,7,9,8,2     ! RAINFALL DATA.
260 END
270 !
280 ! TO RUN THIS PROGRAM ON A HARD COPY PLOTTER WITH A SHEET OF
290 ! PREGRIDED PAPER, DELETE THE PARAMETERS FROM THE LIMIT AND
300 ! LOCATE STATEMENTS AND DIGITIZE THE LOWER LEFT AND UPPER RIGHT
310 ! CORNERS OF THE PAPER FOR LIMIT AND OF THE GRID FOR LOCATE.
```



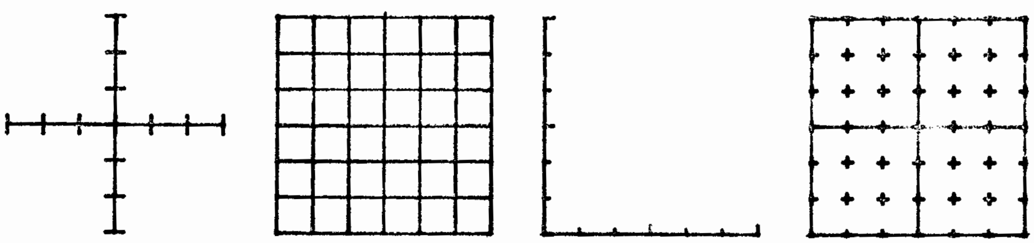
```
100  ! LIMITS EXERCISE 2    SOLUTION    18MAY78
110  !
120  PLOTTER IS "GRAPHICS"             ! INITIALIZE.
130  GRAPHICS                          ! SELECT GRAPHICS MODE.
140  RAD                               ! SET CURRENT ANGLE UNITS TO RADIANS.
150  SCALE Ø,PI,-1,1                   ! SCALE FOR FIVE CYCLES.
160  LINE TYPE 3                       ! SELECT DASHED LINE.
170  GOSUB Plot                        ! PLOT FIVE CYCLES.
180  CLIP .4*PI,.6*PI,-1,1             ! CLIP FOR THIRD CYCLE.
190  LINE TYPE 1                       ! SELECT SOLID LINE.
200  MOVE Ø,Ø                          ! MOVE TO ORIGIN.
210  GOSUB Plot                        ! PLOT THIRD CYCLE.
220  END                               ! END
230                                    !
240                                    ! PLOT SUBROUTINE.
250 Plot:   RAD                        ! SELECT SUBROUTINE.
260  FOR T=Ø TO PI STEP PI/1ØØ         ! FOR 5 CYCLES
270  PLOT T,EXP(-T/2)*COS(1Ø*T-PI)     ! PLOT FUNCTION.
280  NEXT T
290  PENUP                             ! LIFT PEN.
300  RETURN                            ! RETURN
```



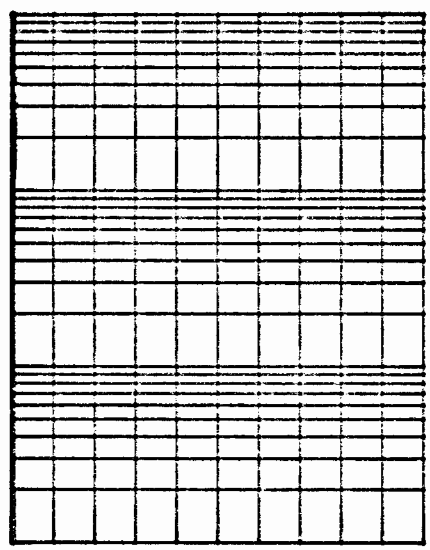
```
80   ! AXES EXERCISE 1    SOLUTION    18MAY78
90   !
100  PLOTTER IS "GRAPHICS"            ! INITIALIZE.
110  GRAPHICS                         ! SET GRAPHICS MODE.
120  LIMIT 0,180,25,125               ! REDUCE FOR FINAL OUTPUT.
130  DIM Title$[23]                   ! DIMENSION STRING FOR TITLE.
140  LORG 4                           ! SELECT LABEL ORIGIN 4.
150  CSIZE 3,.5                       ! SET CHARACTER SIZE.
160  FOR A=0 TO 4                     ! DRAW FIVE PLOTS PER ROW.
170  READ N(1),N(2),N(3),N(4),N(5),N(6),N(7) ! READ PARAMETERS
180  Title$="AXES "                   ! CONSTRUCT STATEMENT FOR TITLE.
190  FOR I=1 TO MIN(2*A,7)            ! APPEND PARAMETERS.
200  Title$=Title$&VAL$(N(I))&","! OMIT DEFAULT PARAMETERS.
210  NEXT I
220  LOCATE 36*A,36*A+30,60,95        ! SET CORNERS OF PLOT.
230  SCALE -12,12,-15,15              ! SCALE APPROPRIATELY.
240  MOVE 0,17                        ! MOVE TO TOP CENTER.
250  LABEL USING "K";Title$[1,LEN(Title$)-1] ! LABEL TITLE.
260  AXES N(1),N(2),N(3),N(4),N(5),N(6),N(7) ! DRAW AXES.
270  LOCATE 36*A,36*A+30,10,45        ! LOCATE CORNERS OF PLOT
280  SCALE -12,12,-15,15              ! SCALE WITH NEW CORNERS.
290  MOVE 0,17                        ! MOVE TO TOP CENTER.
300  Title$[1,5]="GRID "              ! REPLACE "AXES" BY "GRID" IN TITLE.
310  IF A=4 THEN Title$[LEN(Title$)-1;1]="2" ! FIX UP MINOR TIC LENGTH.
320  LABEL USING "K";Title$[1,LEN(Title$)-1] ! LABEL THE FILE.
330  GRID N(1),N(2),N(3),N(4),N(5),N(6),N(7)/2 ! DRAW THE GRID.
340  NEXT A                           ! CONTINUE.
350  DATA 0,0,0,0,1,1,2               ! USE ALL DEFAULTS.
360  DATA 2,3,0,0,1,1,2!              ! SET XSPC=2 AND YSPC=3.
370  DATA 2,3,-12,-15,1,1,2           ! SET (XINT,YINT) TO (-12,-15).
380  DATA 2,3,-12,-15,3,5,2           ! SET XCNT=3 AND YCNT=5.
390  DATA 2,3,-12,-15,3,5,4           ! SET LEN=4.
400  END
```



```
80   ! AXES EXERCISE 2    SOLUTION    18MAY78
90   !
100  PLOTTER IS "GRAPHICS"           ! INITIALIZE.
110  GRAPHICS                        ! SET GRAPHICS MODE.
120  LOCATE 0,24,70,94               ! LOCATE FAR LEFT PLOT.
130  SCALE -3,3,-3,3                 ! SCALE WITH ORIGIN IN CENTER.
140  AXES 1,1                        ! DRAW CROSSED AXES.
150  LOCATE 30,54,70,94              ! LOCATE CENTER LEFT PLOT.
160  SCALE -3,3,-3,3                 ! SCALE FOR NEW LOCATE POINTS.
170  GRID 1,1                        ! DRAW CROSS BAR GRID.
180  LOCATE 60,84,70,94              ! LOCATE CENTER RIGHT PLOT.
190  SCALE 0,6,0,6                   ! SCALE WITH ORIGIN AT LOWER LEFT.
200  AXES 1,1,0,0,3,3                ! DRAW BORDER AXES.
210  LOCATE 90,114,70,94             ! LOCATE FAR RIGHT PLOT.
220  SCALE 0,6,0,6                   ! RESCALE FOR NEW LOCATE POINTS.
230  GRID 1,1,0,0,3,3                ! DRAW CROSS TIC GRID.
240  END
```

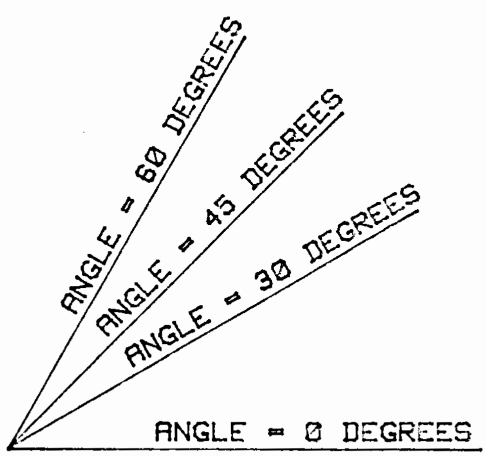


```
80   ! AXES EXERCISE 3    SOLUTION    18MAY78
90   !
100  PLOTTER IS "GRAPHICS"           ! INITIALIZE.
110  GRAPHICS                        ! SET GRAPHICS MODE.
120  LIMIT 65,120,70,140             ! SET HARD CLIP LIMITS.
130  SCALE 0,10,0,3                  ! SCALE FOR THREE CYCLES.
140  GRID 1,0                        ! DRAW LINEAR X AXIS.
150  FOR N=1 TO 9                    ! 1*10^CYCLE, ... , 9*10^CYCLE
160  GRID 0,1,0,LGT(N)               ! DRAW LOGARITHMIC Y AXIS.
170  NEXT N                          ! CONTINUE.
180  END
```

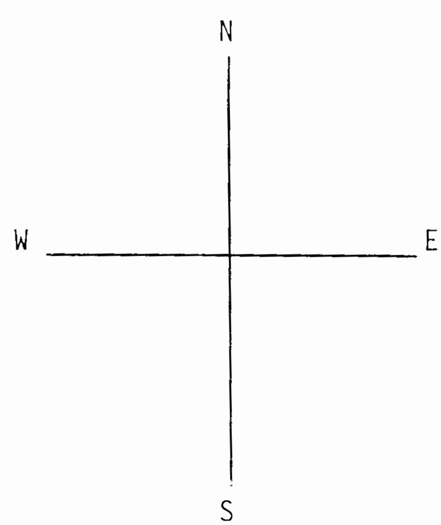


```
80   ! LABELING EXERCISE 1    SOLUTION    18MAY78
90   !
100  ! SETUP
110  PLOTTER IS "GRAPHICS"             ! SELECT CRT AND INITIALIZE.
120  GRAPHICS                          ! SELECT CRT GRAPHICS MODE.
130  !
140  ! MAP SCALE
150  LOCATE 30,110,10,90               ! LOCATE CORNERS OF MAP.
160  SHOW 0,4,0,4                      ! SCALE IN MILES.
170  AXES 1,1                          ! DRAW SCALE.
180  !
190  ! TITLE OF MAP
200  SETGU                             ! SELECT GRAPHIC DISPLAY UNITS.
210  MOVE 0,100                        ! MOVE TO UPPER LEFT CORNER.
220  CSIZE 6                           ! MAKE TITLE LARGE.
230  LORG 3                            ! SET LABEL ORIGIN TO UPPER LEFT CORNER.
240  LABEL USING "#,K";"TOWN OF "      ! DRAW TITLE.
250  WHERE X,Y                         ! READ PEN LOCATION.
260  POINTER X,Y,0                     ! MOVE CURSOR TO PEN LOCATION.
270  LETTER                            ! TYPE TOWN NAME.
280  !
290  ! Y AXIS LABELS
300  SETUU                             ! SELECT USER-DEFINED UNITS.
310  CSIZE 3.3,.6                      ! REDUCE CHARACTER SIZE.
320  LORG 8                            ! SET LABEL ORIGIN TO RIGHT CENTER.
330  FOR Y=0 TO 4
340  MOVE 0,Y                          ! MOVE TO CENTER OF Y-TIC.
350  LABEL USING "K,2X";Y              ! DRAW Y-TIC LABEL.
360  NEXT Y
370  !
380  ! Y AXIS TITLE
390  DEG                               ! SELECT DEGREESL
400  LDIR 90                           ! ROTATE TO 90 DEGREES.
410  LORG 4                            ! SET LABEL ORIGIN TO BOTTOM CENTER.
420  MOVE -.5,2                        ! MOVE TO BOTTOM CENTER OF Y-TITLE.
430  LABEL "MILES";                    ! DRAW Y-TITLE.
440  !
450  ! X AXIS LABELS
460  LDIR 0                            ! LABEL X-TICS AT 0 DEGREES.
470  LORG 6                            ! SET LABEL ORIGIN TO TOP CENTER.
480  FOR X=0 TO 4
490  MOVE X,-.2                        ! MOVE TO TOP CENTER OF X-TIC LABEL.
500  LABEL X;                          ! DRAW X-TIC LABEL.
510  NEXT X
520  !
530  ! X AXIS TITLE
540  MOVE 2,-.4                        ! MOVE TO TOP CENTER OF X-TITLE.
550  LABEL "MILES";                    ! DRAW X-TITLE.
560  END
```

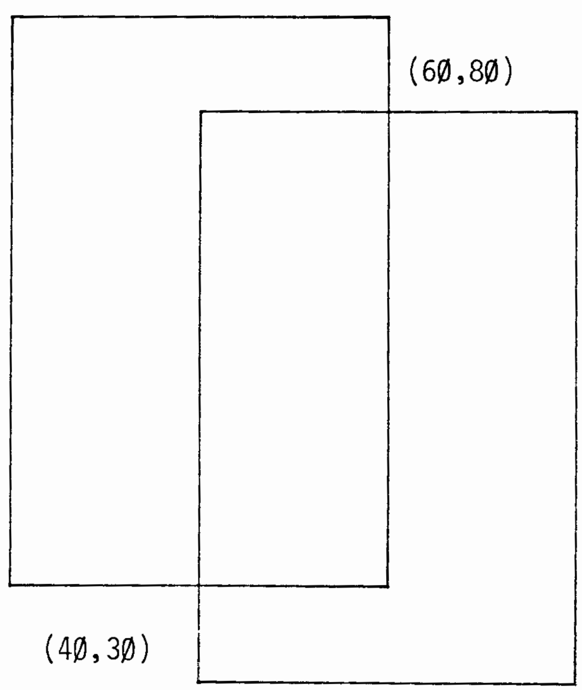
```
1ØØ  ! LABELING EXERCISE 2    SOLUTION    10APR78
11Ø  !
12Ø  PLOTTER IS "GRAPHICS"              ! SELECT CRT AND INITIALIZE.
13Ø  GRAPHICS                           ! SET CRT TO GRAPHICS MODE.
14Ø Loop:  INPUT "X? ,Y?",X,Y           ! TYPE COORDINATES OF POINT.
15Ø  MOVE X,Y                           ! MOVE TO THAT POINT.
16Ø  DRAW Ø,Ø                           ! DRAW TO (Ø,Ø).
17Ø  DEG                                ! SELECT DEGREES.
18Ø  Angle=9Ø                           ! DEFAULT ANGLE TO 9Ø DEGREES.
19Ø  IF X=Ø THEN GOTO 21Ø               ! TEST FOR 9Ø DEGREES.
2ØØ  Angle=PROUND(ATN(Y/X),Ø)           ! IF NOT 9Ø DEGREES, THEN COMPUTE ANGLE.
21Ø  PDIR X,Y                           ! SET PLOT DIRECTION.
22Ø  IPLOT Ø,1,-2                       ! OFFSET PEN FROM LINE.
23Ø  LDIR X,Y                           ! SET LABEL DIRECTION.
24Ø  LABEL USING "8X,K,K,K";"ANGLE = ",Angle," DEGREES" ! DRAW LABEL.
25Ø  WAIT 2ØØØ                          ! WAIT TWO SECONDS TO LOOK AT RESULTS.
26Ø  GOTO Loop                          ! CONTINUE.
27Ø  END
```



```
80    ! LABELING EXERCISE 3   SOLUTION   10APR78
90    !
100   PLOTTER IS "GRAPHICS"            ! INITIALIZE.
110   GRAPHICS                         ! SELECT GRAPHICS MODE.
120   DIM String$[1]                   ! SET UP STRING FOR "E","N","W","S".
130   DEG                              ! SELECT DEGREES.
140   FOR Angle=0 TO 270 STEP 90       ! FOR THE FOUR CARDINAL POINT,
150   MOVE 50,50                       ! MOVE TO THE CENTER,
160   PDIR Angle                       ! SET PLOT DIRECTION,
170   IPLOT 10,0,-1                    ! DRAW LINE,
180   IPLOT 2,0,-2                     ! LIFT PEN AND SPACE OVER,
190   READ String$,Origin              ! READ CARDINAL POINT AND LABEL ORIGIN,
200   LORG Origin                      ! SET LABEL ORIGIN,
210   LABEL String$;                   ! LABEL CARDINAL POINT,
220   NEXT Angle                       ! AND CONTINUE.
230   DATA "E",2,"N",4,"W",8,"S",6     ! CARDINAL POINTS AND LABEL ORIGINS.
240   END
250   !
260   ! FOR THOSE WHO LIKE TO BE CLEVER,
270   ! RATHER THAN READ THE LABEL ORIGIN FROM THE DATA STATEMENT,
280   ! GENERATE IT USING Origin=2^(1+Angle/90)MOD10.
```



```
100  ! DIGITIZING EXERCISE 2    SOLUTION    18MAY78
110  !
120  PLOTTER IS "GRAPHICS"                  ! INITIALIZE
130  GRAPHICS                               ! SET GRAPHICS MODE.
140  CALL Box(20,30,40,60)                  ! DRAW UPPER LEFT BOX.
150  CALL Box(40,20,40,60)                  ! DRAW LOWER RIGHT BOX.
160  GOSUB Digitize                         ! DIGITIZE AN INTERSECTION.
170  GOSUB Digitize                         ! DIGITIZE ANOTHER INTERSECTION.
180  END                                    ! DONE.
190  !
200 Digitize:  !                            ! DIGITIZE SUBROUTINE.
210  DIGITIZE X,Y                           ! DIGITIZE AND INTERSECTION.
220  MOVE X,Y                               ! MOVE PEN TO INTERSECTION.
230  FIXED 0                                ! SELECT INTEGER FORMAT.
240  X$="("&VAL$(X)&","                     ! GET X VALUE IN STRING AND TRIM.
250  Y$=VAL$(Y)&")"                         ! GET Y VALUE IN STRING AND TRIM.
260  INPUT "LABEL ORIGIN?",Origin           ! INPUT LABEL ORIGIN FROM KEYBOARD.
270  LORG Origin                            ! SET LABEL ORIGIN.
280  IF Origin>=7 THEN IPLOT -1,-1          ! OFFSET LABEL.
290  IF Origin<=3 THEN IPLOT 1,1            ! ADD SPACING.
300  LABEL X$;Y$;                           ! LABEL DIGITIZED POINT.
310  RETURN                                 ! DONE
320  !
330  SUB Box(X,Y,Dx,Dy)                     ! SUBROUTINE TO DRAW BOX.
340  PENUP                                  ! LIFT PEN.
350  PLOT X,Y                               ! MOVE TO LOWER LEFT CORNER.
360  IPLOT Dx,0                             ! DRAW BOTTOM SIDE.
370  IPLOT 0,Dy                             ! DRAW RIGHT SIDE.
380  IPLOT -Dx,0                            ! DRAW TOP SIDE.
390  IPLOT 0,-Dy                            ! DRAW LEFT SIDE.
400  PENUP                                  ! LIFT PEN.
410  SUBEND                                 ! RETURN.
```



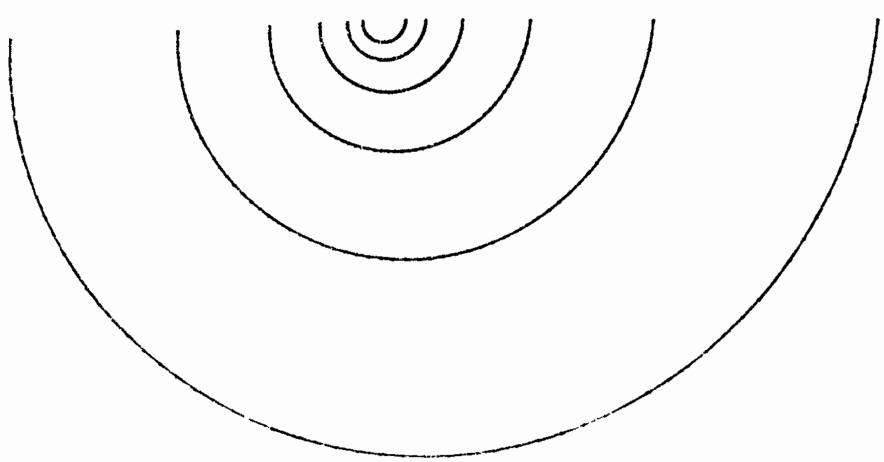
```
8Ø   ! CRT ONLY EXERCISE 1    SOLUTION    18MAY78
9Ø   !
1ØØ  PLOTTER IS "GRAPHICS"             ! SET PLOTTER TO CRT IN GRAPHICS MODE.
11Ø  GRAPHICS                          ! SELECT CRT GRAPHICS MODE.
12Ø  MOVE Ø,9Ø                         ! POSITION PEN FOR PLOTTED TABLE.
13Ø  PRINT PAGE                        ! CLEAR PRINT AREA ON CRT IN ALPHA MODE.
14Ø  FOR X=1 TO 1Ø                     ! X VARIES FROM 1 TO 1Ø.
15Ø  PRINTER IS 16                     ! SET PRINTER TO CRT IN ALPHA MODE.
16Ø  PRINT USING "DD,4X,D,DDD";X,X^(1/2) ! PRINT LINE.
17Ø  PRINTER IS Ø                      ! SET PRINTER TO INTERNAL PRINTER.
18Ø  PRINT USING "DD,4X,D,DDD";X,X^(1/2) ! PRINT LINE.
19Ø  LABEL USING "DD,4X,D,DDD";X,X^(1/2) ! LABEL LINE.
2ØØ  NEXT X                            ! CONTINUE
21Ø  PRINTER IS 16                     ! SET PRINTER TO CRT IN ALPHA MODE.
22Ø  EXIT GRAPHICS                     ! SELECT CRT IN ALPHA MODE.
23Ø  DUMP GRAPHICS                     ! DUMP LABELED OUTPUT.
24Ø  END                               ! DONE.
25Ø  !
26Ø  ! THE CHARACTERS GENERATED FOR PRINTING DIFFER IN SIZE AND SHAPE
27Ø  ! FROM THOSE GENERATED FOR PLOTTING USING DEFAULT CHARACTER WEIGHT
28Ø  ! AND ASPECT RATIO.
```

```
 1      1.ØØØØ
 2      1.4142
 3      1.7321
 4      2.ØØØØ
 5      2.2361
 6      2.4495
 7      2.6458
 8      2.8284
 9      3.ØØØØ
1Ø      3.1623
```

```
 1      1.ØØØØ
 2      1.4142
 3      1.7321
 4      2.ØØØØ
 5      2.2361
 6      2.4495
 7      2.6458
 8      2.8284
 9      3.ØØØØ
1Ø      3.1623
```

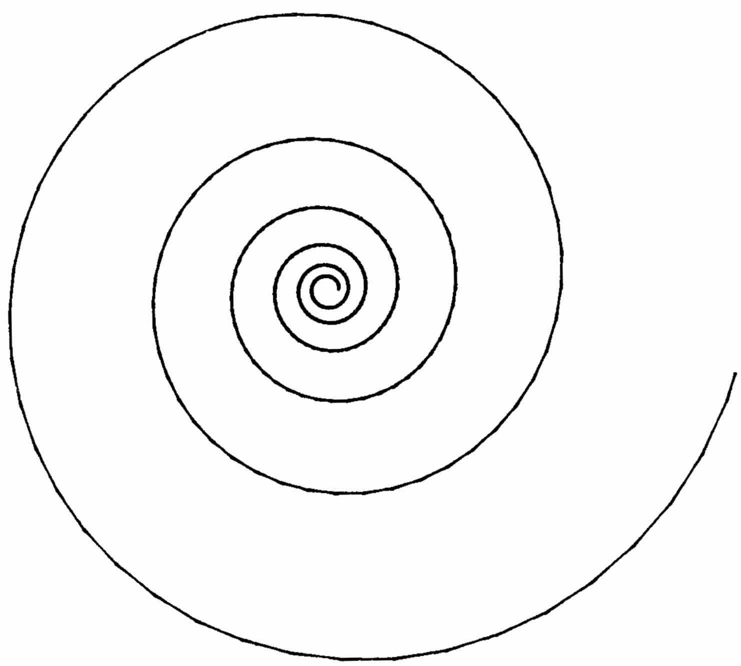
```
8Ø   ! CRT ONLY EXERCISE 2    SOLUTION    18MAY78
9Ø   !
95   INTEGER Memo(Ø:16380)              ! MEMO FORM.
1ØØ  PLOTTER IS "GRAPHICS"              ! INITIALIZE.
11Ø  GRAPHICS                           ! SET GRAPHICS MODE.
12Ø  GOSUB Form                         ! DRAW MEMO FORM.
124  GSTORE Memo(*)                     ! STORE MEMO FORM.
128 Loop: GLOAD Memo(*)                 ! LOAD MEMO FORM.
13Ø  INPUT "NAME?",Name$                ! INPUT NAME TO FOLLOW "TO:"
14Ø  MOVE 1,8                           ! MOVE TO START OF "TO:"
15Ø  LABEL "      "&Name$               ! FILL IN NAME.
16Ø  WAIT 1ØØØ                          ! WAIT ONE SECOND.
17Ø  GOTO Loop                          ! REPEAT.
18Ø                                     !
19Ø                                     ! SUBROUTINE TO DRAW MEMO FORM.
2ØØ Form: SHOW Ø,8,5,Ø,11               ! SCALE CONVENIENTLY.
21Ø  CLIP Ø,8,5,Ø,11                    ! SET SOFT CLIP LIMITS.
22Ø  FRAME                              ! DRAW BOX AROUND FORM.
23Ø  LORG 4                             ! SET LABEL ORIGIN TO BOTTOM CENTER.
24Ø  CSIZE 1Ø                           ! SET CHAR WEIGHT TO 1Ø GDUS FOR TITLE.
25Ø  MOVE 4,25,1Ø                       ! MOVE PEN NEAR TOP CENTER OF FORM.
26Ø  LABEL "MEMO";                      ! LABEL TITLE.
27Ø  LORG 1                             ! SET LABEL ORIGIN TO BOTTOM LEFT.
28Ø  CSIZE 5                            ! SET CHAR HEIGHT TO 5 GDUS FOR TEXT.
29Ø  MOVE 1,8                           ! MOVE TO LEFT MARGIN FOR "TO:".
3ØØ  LABEL "TO:"                        ! LABEL "TO:".
31Ø  MOVE 1,6                           ! MOVE TO LEFT MARGIN FOR MESSAGE.
32Ø  LABEL "TOMORROW'S MEETING"         ! LABEL MESSAGE.
33Ø  LABEL "WILL BE HELD FROM"
34Ø  LABEL "8:3Ø A.M. TO 5:00"
35Ø  LABEL "P.M. IN THE SOUTH"
36Ø  LABEL "CONFERENCE ROOM."
37Ø  RETURN                             ! MEMO FORM IS DRAWN.
```

```
80   ! MULTIPLE PLOTTERS EXERCISE 1    SOLUTION    18MAY78
90   !
100  PLOTTER IS "GRAPHICS"                    ! INITIALIZE.
110  GRAPHICS                                 ! SELECT GRAPHICS MODE.
120  SHOW -35,35,-40,30                       ! SCALE.
130  DEG                                      ! SELECT DEGREES
140  FOR Cycle=1 TO 6                         ! SET CYCLE NUMBER.
150  A=360*(Cycle-1)                          ! COMPUTE INITIAL ANGLE.
160  B=360*Cycle                              ! COMPUTE FINAL ANGLE.
165  PLOTTER 13 IS OFF                        ! TURN OFF FOR UPPER PORTION.
170  FOR Angle=A TO B STEP 18                 ! DRAW ONE CYCLE.
180  Radius=EXP(Angle/600)                    ! USE EXPONENTIAL RADIUS.
190  X=Radius*COS(Angle)                      ! COMPUTE X COORDINATE.
200  Y=Radius*SIN(Angle)                      ! COMPUTE Y COORDINATE.
210  PLOT X,Y                                 ! PLOT SPIRAL.
215  IF Angle=A+180 THEN PLOTTER 13 IS ON     ! TURN ON FOR LOWER PORTION.
220  NEXT Angle                               ! CONTINUE CURRENT CYCLE.
230  NEXT Cycle                               ! CONTINUE WITH NEXT CYCLE.
240  PENUP                                    ! LIFT PEN WHEN DONE.
250  END
```

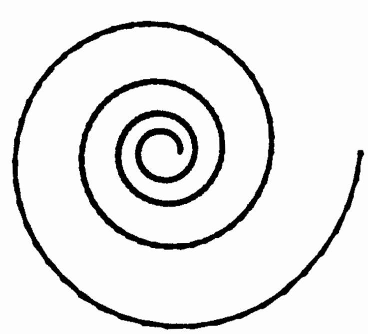


```
80   ! MULTIPLE PLOTTERS EXERCISE 2    SOLUTION    18MAY78
90   !
100  PLOTTER IS "GRAPHICS"                        ! INITIALIZE.
110  GRAPHICS                                     ! SELECT GRAPHICS MODE.
120  SHOW -35,35,-40,30                           ! SCALE.
130  DEG                                          ! SELECT DEGREES.
135 Loop:   GCLEAR                                ! CLEAR SCREEN.
136  INPUT "NUMBER OF CYCLES?",N                  ! INPUT NUMBER OF CYCLES.
140  FOR Cycle=1 TO N                             ! SET CYCLE NUMBER.
150  A=360*(Cycle-1)                              ! COMPUTE INITIAL ANGLE.
160  B=360*Cycle                                  ! COMPUTE FINAL ANGLE.
170  FOR Angle=A TO B STEP 18                     ! DRAW ONE CYCLE.
180  Radius=EXP(Angle/600)                        ! USE EXPONENTIAL RADIUS.
190  X=Radius*COS(Angle)                          ! COMPUTE X COORDINATE.
200  Y=Radius*SIN(Angle)                          ! COMPUTE Y COORDINATE.
210  PLOT X,Y                                     ! PLOT SPIRAL.
220  NEXT Angle                                   ! CONTINUE CURRENT CYCLE.
230  NEXT Cycle                                   ! CONTINUE WITH NEXT CYCLE.
240  PENUP                                        ! LIFT PEN WHEN DONE.
245  GOTO Loop                                    ! REPEAT.
250  END
```

NUMBER OF CYCLES:
6



NUMBER OF CYCLES?
4



Date:____________________

Help us in our continuing effort to improve this class. Please complete the following questionnaire.

I. For the following sections of the course, circle the one answer which best describes your feelings.

	Poor		OK		Outstanding
Basics					
Style of Materials	0	1	2	3	4
Content of Materials	0	1	2	3	4
Instructor Presentation	0	1	2	3	4
Exercises	0	1	2	3	4
Comments:					
Mass Storage					
Style of Materials	0	1	2	3	4
Content of Materials	0	1	2	3	4
Instructor Presentation	0	1	2	3	4
Exercises	0	1	2	3	4
Comments:					
I/O					
Style of Materials	0	1	2	3	4
Content of Materials	0	1	2	3	4
Instructor Presentation	0	1	2	3	4
Exercises	0	1	2	3	4
Comments:					
Graphics					
Style of Materials	0	1	2	3	4
Content of Materials	0	1	2	3	4
Instructor Presentation	0	1	2	3	4
Exercises	0	1	2	3	4
Comments:					
Facilities	0	1	2	3	4
Comments:					
Overall Evaluation	0	1	2	3	4
Comments:					

II. What are your feelings about the balance of time between the lecture and the lab?

	More Lab		OK		More Lecture
Basics	0	1	2	3	4
Mass Storage	0	1	2	3	4
I/O	0	1	2	3	4
Graphics	0	1	2	3	4

III. Suggestion Box. Please comment on any aspect of your experience this week. (Use back of page)

1) What were your expectations in taking this class?

2) Were your expectations met? (If not, where did we fall short?)

3) On the basis of this class, would you like any additional information on:

the 9845?

any other HP Product?

MAILING ADDRESS:

NAME PHONE NO.

COMPANY

ADDRESS

4) Fantasy Box. Dream a little - What capabilities would you like to see in future desk top computers?

5) In order for us to guide the next class, we are interested in how you enjoyed your stay in this area. What local activities, restaurants, etc., would you recommend that the next class take advantage of? Are there any activities, restaurants, etc. you would recommend they not try?