Graphics.

This document describes simple methods in terms of operator interface and programs needed to produce basic graphics.

A wide variety of graphics can be produced using the standard characters in various interface modes.

* Keyboard - interactivity from the keyboard;
* BASIC - use of PRINT, TAB, ARRAYS, Escape Sequences, etc.;
* Editor - use of an editor to define and save templates, tabular and other graphics.

This document will present a number of ways to produce graphics from a previously defined character set. The methods employ clever ways of controlling the position in which characters display on the screen and as a result print on paper. The approach describing these methods will be by way of presenting examples which are created under the three environments described above.

2. VICTOR 9000 VIDEO DISPLAY SYSTEM

The following deals with those special hardware design aspects of the VICTOR 9000's CRT display subsystem and how they can be used to produce character graphics.

To begin with, the terms Dot RAM and Screen RAM will be described. Dot RAM is that portion of the main dynamic memory which is dedicated to the patterns of dots (CRT beam "on" signals) required to generate a particular symbol on the screen. The hardware is designed so that when the CRT beam is pointing at a particular location on the screen that bit of data from Dot RAM that is associated with the portion of the symbol being displayed at that locaation on the screen controls whether the beam is on, the screen is made to glow at that point, or off, the screen goes dark. The set of graphics to be displayed is selected as part of the system generation process. When that system is loaded in response to a "cold boot" request the dot patterns are loaded into main memory as part of the "cold boot" process.

Screen RAM is a memory sub-system completely seperate from main memory. The CRT screen is divided into 25 lines of characters, each line able to display 80 chacters. This is a total of 25 X 80 or 2000 seperate characters. The character to be displayed on the screen is determined by the data stored in two consequtive bytes of screen RAM associated with the character position. What is stored at this location is the Dot RAM number of the character to be displayed.

3. CHARACTER SETS

The initial release of the VICTOR 9000 System Generation process allows the user to choose between two character sets. One is described as the VTT52 Type Display and the other as the Victor International. The VT52 ASCII set has 128 characters and the International set has 256. The System Generation process allows the choice of one or the other as the set to be used. These two character sets are shown below in Figures 7 and 8.

This is not to say that character sets other than VT52 ASCII and the Victor International 01 set are not available with the VICTOR 9000. There is a variety of other character sets of various standards (IBM, etc.) and languages that may be used providing different capabilities and effects using character graphics. Additionally a user may alter an already existing character set or define a completely unique set (using a special Victor 9000 utility) in order to satisfy a special set of requirements.

The more familiar graphics of a character set as displayed on the CRT are the alphabetics, numerics, and special characters accessible directly from the keyboard. There are, however, additional characters accessible from the keyboard that are avaialable for use by programs. These characters are keyboard accessible when the system is in the Graphics Mode. In order to put the system into and out of the Graphics Mode two escape sequences are provided. They are:

1.)	Escape F	Graphics Mode On
2.)	Escape G	Graphics Mode Off

There is a third group of characters not accessible by the keyboard with a single key stroke. This group, as well as the first two described above, is accessible by use of a third escape sequence:

```
Escape 8 x  where the hex value of x
            is interpreted literally.
```

Short BASIC programs can be used to demonstrate how to access and display all the characters in a set. The character set used in the following discussion is the Victor International. If the VT52 ASCII type is used, the results will not necessarily be the same.

```
10 REM NON-GRAPHICS MODE DEMONTRATION (NONGRAF.BAS)
20 REM         THIS ROUTINE ACCEPTS A CHARACTER NUMBER
30 REM         FROM THE KEYBOARD AND DISPLAYS THE ASSOCIATED
40 REM         CHARACTER IN THE NORMAL (NON-GRAPHICS) MODE.
50 DEFINT A-Z
60 PRINT CHR$(27);"E";
```

```
70 INPUT "ENTER CHARACTER NUMBER. ",A
90 PRINT CHR$(A)
110 GOTO 70
120 END
```

If NONGRAF above is run and the values 0 to 32 are entered no character will be displayed. This is due to the fact that the program operates in the normal (non-graphic) mode and those characters numbbered from 0 to 32 are accessable only in the graphics mode. If number 33 is entered the ! is displayed. Different characters will be displayed for input values from 33 to 253. For those values from 97 to 128 the lower case alphabetics are displayed along with a few special character from 123 to 128.

```
10 GRAPHICS MODE DEMONSTRATION (GRAFMODE.BAS)
20 REM          THIS ROUTINE ACCEPTS A CHARACTER NUMBER
30 REM          FROM THE KEYBOARD AND DISPLAYS THE ASSOCIATED
40 REM          GRAPHICS MODE CHARACTER.
50 DEFINT A-Z
60 PRINT CHR$(27);"E";
70 INPUT "ENTER CHARACTER NUMBER. ",A
80 PRINT CHR$(27);"F";
90 PRINT CHR$(A)
100 PRINT CHR$(27);"G";
110 GOTO 70
120 END
```

This second program differs from the first in that the graphics mode is entered at line #80 and exited at line #100. If the character numbers 0 to 32 are entered into this program, that group of characters not accessible by NONGRAF is still not accessible. But they are accessible by entering the numbers 97 to 128. However, those lower case alphabetics and special characters which appear for these values in the normal mode are no longer accessible. All other values should produce the same characters as for NONGRAF.

```
10 REM LITERAL MODE DEMONSTRATION (LTRLMODE.BAS)
20 REM          THIS ROUTINE ACCEPTS A CHARACTER NUMBER
30 REM          FROM THE KEYBOARD AND DISPLAYS THE ASSOCIATED
40 REM          CHARACTER AFTER INTERPRETING THE INPUT
50 REM          NUMBER LITERALLY.
60 DEFINT A-Z
70 PRINT CHR$(27);"E";
80 INPUT "ENTER CHARACTER NUMBER. ",A
90 PRINT CHR$(27);"8";CHR$(A)
100 GOTO 80
110 END
```

This program is identical to NONGRAF, except for the second PRINT statement. This statement instructs BASIC to interpret the number entered literally. This program will display all characters of the set from 0 to 256.

```
10 REM DISPLAY CHARACTER SET (DSPLAYCS.BAS)
20 REM THIS ROUTINE USES ABSOLUTE CURSOR POSITIONING ANDD
30 REM PRINTING OF CHARACTERS IN THE 'LITERAL' MODE IN
40 REM ORDER TO DISPLAY ALL THE AVAILABLE CHARACTERS.
50 PRINT CHR$(27);"E"
60 WIDTH 255
70 PRINT "      0  1  2  3  4  5  6  7  8  9  10";
80 PRINT " 11 12 13 14 15 16 17 18 19"
90 PRINT
100 FOR B=0 TO 240 STEP 20
110 PRINT B;
120 C=38
130 FOR I=0 TO 19
140 IF I+B=256 THEN STOP
150 PRINT CHR$(27);"Y";CHR$(99);CHR$(C);
160 PRINT CHR$(27);"8";CHR$(I+B);"  ";
170 C=C+3
180 NEXT I
190 PRINT
200 NEXT B
200 END
```

This program will display in a tabular form all the characters available in the set alone with their number as shown below in Figures 7 and 8.

4. USE OF ESCAPE SEQUENCES

Although escape sequences are not strictly part of the subject of Character Graphics, they provide a means of simply accomplishing CRT display effects. Escape sequences are grouped into the following:

* Cursor Control Functions
* Screen, line and character editing
* Configuration functions
* Operation mode functions
* Special functions

The following table lists the escape sequences incorporated into the system software. The leading "*"'s are not part of the escape sequence but indicate aa VT52 compatible sequence. Also brackets enclose parameters and are not part of the sequence.

Cursor Functions

*Esc H	1B,48	Sets the cursor at home position.
*Esc C	1B,43	Moves the cursor forward one character position.
*Esc D	1B,44	Moves the cursor backward one

		character position.
*Esc B	1B,42	Moves the cursor down one line without changing columns.
*Esc A	1B,41	Moves the cursor up one line.
*Esc I	1B,49	Moves the cursor to the same horizontal position on the preceding line.
Esc n	1B,6E	Reports the cursor position.
Esc j	1B,6A	Saves the cursor position.
Esc k	1B,6B	Returns the cursor to the previously saved cursor position.
*Esc Y[l][c]	1B,59	Moves the cursor via direct cursor addressing, where "l" represents the hexadecimal line number and "c" represents the hexadecimal column number. The first line and the left column are both 20 (hex) (the smallest value of the printing characters) and increase from there. Since the lines are numbered from 1 to 19 (hex) (from top to bottom) and the columns from 1 to 50 (hex) (from left to right), you must add the proper line and column numbers to 1F.

Editing Functions

Esc E	1B,45	Erases the entire screen.
Esc b	1B,62	Erases from the start of the screen up to and including the cursor position.
*Esc J	1B,4A	Erases from the cursor position to the end of the page.
Esc l	1B,6C	Erases entire line.
Esc o	1B,6F	Erases the beginning of the line up to and including the cursor position.
*Esc K	1B,4B	Erases from the cursor position to the end of the line.
Esc L	1B,4C	Inserts a blank line; the current line and all following lines are moved down one line. The cursor is moved to the beginning of the blank line.
Esc M	1B,4D	Deletes the current line, placing the cursor at the beginning of the line, and moves all following lines up one line. A blank line is inserted at line 24.
Esc N	1B,4E	Deletes cursor position character and shifts the rest of the line one character position to the left.
Esc @	1B,40	Enters the insert character mode, allowing insert into text on the screen. As each new character is inserted, the character at the end of the line is lost.

Esc O	1B,4F	Exits from the insert character mode.

Configuration Functions

Esc x[Ps]	1B,78	Sets mode(s) as follows:	
		Ps	Mode
		1	Enable 25th line
		4	Block cursor
		5	Cursor off
		6	Keypad shifted
		8	Auto line feed on receipt of a carriage return
		9	Auto carriage return on receipt of a line feed
		A	Increase audio volume
		B	Increase CRT brightness
		C	Increase CRT contrast
Esc y[Ps]	1B,79	Resets mode(s) as follows:	
		Ps	Mode
		1	Disable 25th line
		4	Underscore cursor
		5	Cursor on
		6	Keypad unshifted
		8	No auto line feed
		9	No auto carriage return
		A	Decrease audio volume
		B	Decrease CRT brightness
		C	Decrease CRT contrast

Operation Mode Functions

Esc P	1B,70	Enters the reverse video mode.
Esc q	1B,71	Exits the reverse video mode.

Special Functions

Esc }	1B,7D	Disables the keyboard.
Esc {	1B,7B	Enables the keyboard.
Esc v	1B,76	Enables wrap around at the end of the line.
Esc w	1B,77	Discards at the end of the line.
*Esc Z	1B,7A	Resets terminal to power-on configuration.
Esc]	1B,5D	Transmits the 25th line.
Esc #	1B,23	Transmits the page.
Esc (	1B,28	Sets high intensity.
Esc)	1B,29	Sets low intensity.

Esc +	1B,2B	Clears the foreground.
Esc 0	1B,30	Sets the underline mode.
Esc 1	1B,31	Resets the underline mode.
Esc 2	1B,32	Enables cursor blink.
Esc 3	1B,33	Disables cursor blink.
Esc 8	1B,38	Sets the test (literally) mode.
Esc 9	1B,67	Resets the test (literally) mode.
Esc 4[m][lk][kv]	1B,34	Resets the key value. Used to reconfigure the ASCII code generated by the key where – "m" represents the characters 1, 2 or 3 – 1 stands for unshifted mode – 2 stands for shifted mode – 3 stands for alternate mode "lk" represents the logical key number (00-67 hexadecimal) "kv" represents the hexadecimal ASCII key code of the new key value
Esc i	1B,69	Returns configuration information.

The following BASIC program provides a means for demonstrating most of the escape sequences listed above. The program will prompt for an escape sequence. When the sequence is entered the program will move the cursor to the middle of the "3" line, display an "*" to mark its starting location, and then execute the sequence. If, for example, a D is entered, the program will put an * in the middle of line #3, and then move the cursor to the left one character position.

```
10 REM ESCAPE SEQUENCE DEMONSTRATION (ESCAPSEQ.BAS)
20 REM THIS ROUTINE ACCEPTS A GIVEN ESCAPE SEQUENCE
30 REM FROM THE KEYBOARD AND THEN EXECUTES IT OVER
40 REM FIVE LINES OF THE NUMBERS 1 TO 5 AS AN AID
50 REM IN VISUALIZING THE SEQUENCE.
60 PRINT CHR$(27);"E";:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
70 PRINT "1111111111111111111111111111111111111111";
80 PRINT "1111111111111111111111111111111111111111"
90 PRINT "2222222222222222222222222222222222222222";
100 PRINT "2222222222222222222222222222222222222222"
110 PRINT "3333333333333333333333333333333333333333";
120 PRINT "3333333333333333333333333333333333333333"
130 PRINT "4444444444444444444444444444444444444444";
140 PRINT "4444444444444444444444444444444444444444"
150 PRINT "5555555555555555555555555555555555555555";
160 PRINT "5555555555555555555555555555555555555555"
170 INPUT "ENTER ESCAPE FUNCTION: ",EF$
```

```
180 EFN= ASC(EF$)
190 PRINT CHR$(27);"Y";CHR$(41);CHR$(71);' MOVE CURSOR
200 PRINT "*";'                               DISPLAY "*"
210 PRINT CHR$(8);
220 PRINT CHR$(27);EF$;'                      EXECUTE SEQUENCE
230 CONTCHR$=INKEY$
240 IF CONTCHR$="" THEN GOTO 230'             WAIT FOR KEY ENTRY
250 GOTO 60
260 END
```

5. CHARACTER GRAPHICS UNDER BASIC

General:

Before continuing with schemes and examples for producing character graphics, the reader should be familiar withh the background material discussed above in this report:

- The VICTOR 9000 Video Display System
- Character Sets
- Use of Escape Sequences

The remainder of this section will show examples of character graphics using the BASIC language.

The TAB Statement

The following simple BASIC program:

```
10 FOR X=1 TO 25
20 PRINT TAB(X);"*"
30 NEXT X
```

will produce a line which begins at column zero and moves downward at approximately 60 degrees as shown in Figure 1.

In other words, TAB(X) means move right to the Xth position before printing the asterisk. (Don't forget - column positions are numbered 0,1,2,3,...).

Now, if we change line 20 to use a more complicated TAB expression such as:

```
20 PRINT TAB(X*X/10);"*"
```

a "curved" line is produced. This is because increasing X from 1 to 25 will increase X*X/10 from 0.1 to 62.5. Since TAB uses the integer part of its argument, the asterisks will print in positions determined by the numbers in the third column of the following table:

```
      X            X*X/10          TAB(X*X/10)

      1              .1                 0
      2              .4                 0
      3              .9                 0
      4             1.6                 1
      5             2.5                 2
      6             3.6                 3
      7             4.9                 4
      8             6.4                 6
      9             8.1                 8
     10            10.0                10
```

This approximates a "quadratic" curve as shown in Figure 2.

TAB can have any legal BASIC expression as its argument, including expressions that use BASIC functions. The following is an example where line 20 prints the symbol "1" in the position determined by TAB(X+3), while line 30 prints the symbol "2" in the position determined by TAB(ABS(3*X-36)+3). The effect is something like graphing the path of two billiard balls. Notice that we are printing the "1" and "2" on alternate lines (see Figure 3). (This was done to simplify the program.)

```
                                             (Using ABS makes the
    10 FOR X=0 TO 25                         (argument of TAB go
    20 PRINT "A:";TAB(X+3);"1"               (from +39 to +3 when
    30 PRINT "B:";TAB(ABS(3*X-36)+3);"2"     (X goes from 0 to +12
    40 NEXT X                                (and from +6 to +42
    50 END                                   (when X goes from +13
                                             (to +25
```

Another way to use TAB(X) is to read values of X from data statements. This allows us to print computer graphs that show pictorially what the data "looks" like. For example, the plot from the weekly weigh-ins of someone on a reducing diet is shown in Figure 4.

```
    10 PRINT "GRAPH OF WEEKLY WEIGHTS"
    20 PRINT "    ";
    30 FOR K=100 TO 200 STEP 10:PRINT K;:NEXT K:PRINT)Lines
    40 PRINT "    ";                                )10-50
    50 FOR K= 0 TO 10: PRINT "  +  ";:NEXT K: PRINT  )Print the
                                                    )"Heading"
                                                    )Lines.
    55 LET S = 0
    60 FOR X=1 TO 30
    70 READ W                                    (This line
    80 IF W<0 THEN 150                           (graph. The
    90 PRINT X;TAB(4);"I";TAB(W-100)/2+6);"*" -  (formulas used
   100 NEXT X                                    (are explained
                                                 (further on.
110 DATA 155, 149,144, 141, 138, 135, 134.5, 132, 133, 133.7
120 DATA 134, 135, 136, 136, 137, 139, 140.2, 142, 144, 147
130 DATA 150, 143, 135, 130, 126, 123, 121, 120, 119, 119
140 DATA -1
150 PRINT "AVERAGE WEIGHT ="; S/30
160 END
```

The BASIC functions SIN, COS, LOG, EXP, TAN, ATN, SGN evaluate the mathematic functions "Sine", "Cosine", "Logarithmic", "Exponential", "Tangent", "Arctangent" and "Sign" respectively for a given argument. These functions can be used to produce displays that show graphically what the functions look like and that are also attractive as design elements.

Figure 5 is an example showing what the SIN function looks like when graphed using the following program:

```
10 FOR F=-1 TO 1.1 STEP .4
20 PRINT TAB(9+30*(F+1);INT(F*100)/100;
30 NEXT F
40 PRINT
50 FOR A = 0 TO 6.3 STEP .1
60 PRINT A;TAB(10+30*(SIN(A)+1));"*"
70 NEXT A
80 END
```

The first loop in lines 10-30 puts numbers across the top of the page to show what values of the SIN function are being graphed. (The numbers were selected as shown because we know from trigonometry that the SIN function has values that range from -1 to +1).

The second loop in lines 50-70 prints A (the argument), and then prints an asterisk in a position determined by the value of SIN(A). We used SIN(A)+1 in our TAB so that the values -1 to +1 would be changed to the range 0 to 2 (you can't TAB negative values). We multiplied by 30 to spread the picture out from columns 0 to 60, and then added 10 to shift all values 10 columns to the right (to leave room for print A). So the final graph goes from 10 to 70. On a terminal with a smaller number of columns, the multiplier 30 should be reduced to about 15.

Of course, we can print other things besides a single asterisk "*". See Figure 6. Figure 6 was generated by combining functions and putting multipliers in front of the arguments giving "intermodulation" affects.

The combined function

```
Y=COS(2*A)+SIN(A)
```

was used. The two signals are "90 degrees out of phase", and the first one has "twice the frequency" of the second.

By running the following programs, the graphs pictured in Figures 6 and 6A were produced.

```
      10 FOR A=0 TO 9.5 STEP .2
      20 LET Y=COST(2*A)+SIN(A)
      30 PRINT TAB(15*Y+30);"HARVEY KILOBIT"
      40 NEXT A
      50 END
```

or

```
      10 FOR A=0 TO 9.5 STEP .2
      20 LET Y=COST(2*A)+SIN(A)
      30 PRINT TAAB(15*Y+30);"*"
      40 NEXT A
      50 END
```

Additional Examples

This section lists three BASIC programs which produce examples that are called CALENDAR, BARGRAPH, AND 3D. The results are pictured in Figures 9, 10, and 11.

The following program will display a calender for the given month of the year 1982. It will first ask for the month desired and wait for a number in the range of 1 to 12 to be entered. After the entry it will display the month's calendar and then wait for a key to be depressed when it will return to the BASIC interpreter.

```
10 REM CALENDAR DEMONSTRATION (CALENDAR.BAS)
20 WIDTH 255
30 DEFINT A-Z

40 YR$="1982":INPUT "Enter month (1=January)";MN
50 DATA JANUARY,5,31,FEBRUARY,1,28,MARCH,1,31
60 DATA APRIL,4,30,MAY,6,31,JUNE,2,30
70 DATA JULY,4,31,AUGUST,0,31,SEPTEMBER,3,30
80 DATA OCTOBER,5,31,NOVEMBER,1,30,DECEMBER,3,31
90 DATA DECEMBER,3,31
100 FOR I=1 TO MN:READ MON$,FD,NDAYS:NEXT I
110 PRINT CHR$(27);"z";

120 CN=FD*10-5:LN=9
130 FOR I=1 TO NDAYS
140 CN=CN+10
150 IF CN<72 THEN GOTO 180
160 LN=LN+3:CN=5
170 IF LN>21 THEN LN=22:CN=CN+7:
```

```
                    SL$=CHR$(8)+CHR$(8)+CHR$(8)+CHR$(8)+CHR$(8)+
                    CHR$(8)+"/"+CHR$(27)+"A/"
180 GOSUB 820
190 PRINT I;SLL$
200 NEXT I
210 LN=2:CN=5:GOSUB 820
220 STRT=1:FINISH=70:CHRNMB=205:GOSUB 860' TOP BORDER
230 PRINT CHR$(27);"8";CHR$(187);
240 PRINT CHR$(27);"D";CHR$(27);"B";
250 STRT=1:FINISH=20:CHRNMB=186:GOSUB 910'     RIGHT BORDER

260 REM:LN=24:CN=5:GOSUB 720
270 LN=23:CN=5:GOSUB 820
280 STRT=1:FINISH=70:CHRNMB=205:GOSUB 860' BOTTOM BORDER
290 PRINT CHR$(27);"8";CHR$(188);
300 LN=2:CN=5:GOSUB 820
310 PRINT CHR$(27);"8";CHR$(201);
320 PRINT CHR$(27);"B";CHR$(27);"D";
330 STRT=1:FINISH=20:CHRNMB=186:GOSUB 910'     LEFT BORDER
340 PRINT CHR$(27);"8";CHR$(200)

350 LLN=3
360 FOR H=1 TO 2
370 LN=LN+2:CN=5:GOSUB 820
 380 PRINT CHR$(27);"8";CHR$(204);
390 STRT=1:FINISH=69:CHRNMB=205:GOSUB 860'     HORZ LINES 1 & 2
400 PRINT CHR$(27);"8";CHR$(185);
410 NEXT H

420 LN=6:CN=7:GOSUB 820
430 PRINT "SUNDAY    MONDAY    TUESDAY  WEDNESDAY ";
440 PRINT "THURSDAY   FRIDAY   SATURDAY";

450 LN=7
460 FOR LC=1 TO 4
470 CN=5:LN=LN+3:GOSUB 820
480 PRINT CHR$(27);"8";CHR$(199);
490 STRT=1:FINISH=69:CHRNMB=196:GOSUB 860'     HORZ LINES 3 TO 6
500 PRINT CHR$(27);"8";CHR$(182)
510 NEXT LC

520 CN=5
530 FOR CC=1 TO 6'                             VERT LINES 1 TO 6
540 CN=CN+10:LN=5:GOSUB 820
550 PRINT CHR$(27);"8";CHR$(209);
560 GOSUB 970
570 PRINT CHR$(27);"8";CHR$(179);
580 GOSUB 970
590 PRINT CHR$(27);"8";CHR$(216);
600 FOR II=1 TO 5
610 GOSUB 970
620 PRINT CHR$(27);"8";CHR$(179);
```

```
630 GOSUB 970
640 PRINT CHR$(27);"8";CHR$(179);
650 GOSUB 970
660 PRINT CHR$(27);"8";CHR$(197);
670 NEXT II
680 PRINT CHR$(27);"D";
690 PRINT CHR$(27);"8";CHR$(207);
700 NEXT CC

710 CN=35:LN=2:GOSUB 820
720 PRINT "VICTOR 9000";
730 CN=29:LN=3:GOSUB 820
740 PRINT "CALENDAR DEMONSTRATION";
750 CN=35:LN=4:GOSUB 820
760 PRINT MON$;" ";YR$

770 PRINT CHR$(27);"x";"5"'                              CURSOR OFF
780 CONTTKEY$=INKEY$
790 IF CONTKEY$="" THEN GOTO 780
800 PRINT CHR$(27);"y";"5"'                              CURSOR ON
810 STOP

820 REM POSITION CURSOR ROUTINE.
830 LA=LN+32:CA=CN+32
840 PRINT CHR$(27);"Y";CHR$(LA);CHR$(CA);
850 RETURN

860 REM WRITE LINE OF GIVEN CHARACTER TO CRT.
870 FOR I=STRT TO FINISH
880 PRINT CHR$(27);"8";CHR$(CHRNMB);
890 NEXT I
900 RETURN

910 REM WRITE COLUMN OF GIVEN CHARACTER TO CRT.
920 FOR I=STRT TO FINISH
930 PRINT CHR$(27);"8";CHR$(CHRNMB);
940 PRINT CHR$(27);"D";CHR$(27);"B";
950 NEXT I
960 RETURN

970 REM MOVE CURSOR DOWN ONE LINE & LEFT ONE CHARACTER.
980 PRINT CHR$(27);"B";CHR$(27);"D";
990 RETURN
```

The following program will display a bar chart showing the % sales of a corporation for the years 1975 to 1981 for each product category (see Figure 8).

```
10 REM BAR CHART DEMONSTRATION (BARCHART.BAS)
```

```
20 WIDTH 255
30 DEFINT A-Z
40 PRINT CHR$(27);"E";
50 PRINT "                 PROFITABLE PRODUCTS INC. -";
60 PRINT " % SALES BY PRODUCT CATEGORY"
70 DIM CIA(4):CIA(1)=176:CIA(2)=177:CIA(3)=178:CIA(4)=219

80 PC=110'                                           FORM VERT SCALE
90 LN=0:CN=0:GOSUB 610
100 FOR I=1 TO 11
110 PC=PC-10
120 PRINT "    ";
130 PRINT CHR$(27);"8";CHR$(179)
140 PRINT USING "###";PC;
150 PRINT CHR$(27);"0";
160 PRINT CHR$(27);"8";CHR$(179);
170 PRINT CHR$(27);"1"
180 NEXT I

190 LN=21:CN=4:GOSUB 610                             FORM HORIZONTAL
SCALE
200 PRINT CHR$(27);"0";
210 PRINT "                                                    "
220 PRINT CHR$(27);"1";
230 PRINT "       1975   1976   1977   1978   1979   1980   1981";

240 RESTORE
250 CN=0
260 FOR Y=1 TO 7
270 CN=CN+6:LN=2:GOSUB 610
280 FOR IC=1 TO 4
290 READ LMT
300 CHARID=CIA(IC)
310 FOR IA=1 TO LMT*2
320 PRINT CHR$(27);"8";CHR$(CHARID);
330 PRINT CHR$(27);"8";CHR$(CHARID);
340 PRINT CHR$(27);"8";CHR$(CHARID);
350 PRINT CHR$(27);"8";CHR$(CHARID);
360 PRINT CHR$(8);CHR$(8);CHR$(8);CHR$(8);CHR$(27);"B";
370 NEXT IA
380 NEXT IC
390 NEXT Y

400 LN=0:CHARID=176:GOSUB 650'                        DISPLAY LEGEND
410 PRINT "COMPONENTS";
420 CHARID=177:GOSUB 650
430 PRINT "SUB-ASSEMBLIES";
440 CHARID=178:GOSUB 650
450 PRINT "HARDWARE SYSTEMS"
460 CHARID=219:GOSUB 650
470 PRINT "SOFTWARE";
```

```
480 PRINT CHR$(27);"x";"5";'                       CURSOR OFF
490 CONTKEY$=INKEY$
500 IF CONTKEY$="" THEN GOTO 490
510 PRINT CHR$(27);"y";"5"'                        CURSOR ON
520 STOP

530 REM THE DATA IS FOR THE YEARS 1975 TO 1981.
540 DATA 7,2,1,0
550 DATA 6,2,2,0
560 DATA 5,2,3,0
570 DATA 6,1,3,0
580 DATA 5,1,3,1
590 DATA 4,1,4,1
600 DATA 2,1,4,3

610 REM POSITION CURSOR ROUTINE.
620 LA=LN+32:CA=CN+32
630 PRINT CHR$(27);"Y";CHR$(LA);CHR$(CA);
640 RETURN

650 REM GENERATE LEGEND ROUTINE.
660 LN=LN+4:CN=55:GOSUB 610
670 FOR J=1 TO 4
680 PRINT CHR$(27);"8";CHR$(CHARID);
690 NEXT J
700 LN=LN+1:CN=55:GOSUB 610
710 RETURN

720 END
```

The following program will display the sales of a corporation for the years a 1975 to 1978 by product category. The display will illustrate these sales using three dimensional blocks constructed only from standard graphic characters.

```
10 REM 3-DIMENTION BAR GRAPH DEMONSTRATION (3D.BAS)
20 WIDTH 255
30 DEFINT A-Z
40 PRINT CHR$(27);"E";
50 PRINT "       PROFITABLE PRODUCTS INC. - ";
60 PRINT "SALES BY PRODUCT CATEGORY IN MILLIONS OF $"
70 DIM CIA(4):CIA(1)=176:CIA(2)=177:CIA(3)=178:CIA(4)=219

80 PC=110'                                         FORM VERT SCALE
90 LN=0:CN=0:GOSUB 690
100 FOR I=1 TO 11
110 PC=PC-10
120 PRINT "    ";
130 PRINT CHR$(27);"8";CHR$(179)
140 PRINT USING "###";PC;
```

```
150 PRINT CHR$(27);"0";
160 PRINT CHR$(27);"8";CHR$(179);
170 PRINT CHR$(27);"1"
180 NEXT I

190 LN=21:CN=4:GOSUB 6990                           FORM HORIZONTAL
SCALE
200 PRINT CHR$(27);"0";
210 PRINT "                                                          "
220 PRINT CHR$(27);"1";
230 PRINT "          1975         1976         1977         1978";
240 RESTORE

250 FOR Y=1 TO 4
260 CN=66-(Y*12):LN=21:GOSUB 690
270 FOR IC=1 TO 4
280 CN=CN-6:LN=21:GOSUB 690
290 READ LMT
300 CHARID=CIA(IC)

310 FOR IA=1 TO LMT'                      FORM FACE OF BLOCK
320 PRINT CHR$(27);"8";CHR$(CHARID);
330 PRINT CHR$(27);"8";CHR$(CHARID);
340 PRINT CHR$(27);"8";CHR$(CHARID);
350 PRINT CHR$(27);"8";CHR$(CHARID);
360 PRINT CHR$(8);CHR$(8);CHR$(8);CHR$(8);CHR$(27);"A";
370 NEXT IA

380 IF LMT=0 THEN GOTO 490
390 PRINT "/";'                                   FORM TOP OF BLOCK
400 FOR IA=1 TO 3
410 PRINT CHR$(27);"8";CHR$(196);
420 NEXT IA
430 PRINT "/";
440 CN=CN+4:LN=22-LMT:GOSUB 690'                  FORM SIDE OF BLOCK
450 FOR IA=1 TO LMT
460 PRINT CHR$(27);"8";CHR$(179);
470 PRINT CHR$(27);CHR$(8);CHR$(27);"B";
480 NEXT IA

490 NEXT IC
500 NEXT Y
510 LN=0:CHARID=176:GOSUB 730'                    DISPLAY LEGEND
520 PRINT "COMPONENTS";
530 CHARID=177:GOSUB 730
540 PRINT "SUB-ASSEMBLIES";
550 CHARID=178:GOSUB 730
560 PRINT "HARDWARE SYSTEMS"
570 CHARID=219:GOSUB 730
580 PRINT "SOFTWARE";

590 PRINT CHR$(27);"x";"5";'                      CURSOR OFF
```

```
600 CONTKEY$=INKEY$
610 IF CONTKEY$="" THEN GOTO 600
620 PRINT CHR$(27);"y";"5";                        CURSOR ON
630 STOP

640 REM THE DATA IS FOR THE YEARS 1975 TO 1978.
650 DATA 18,15,11,7
660 DATA 9,7,5,2
670 DATA 8,5,2,0
680 DATA 7,5,2,0

690 REM POSITION CURSOR ROUTINE.
700 LA=LN+32:CA=CN+32
710 PRINT CHR$(27);"Y";CHR$(LA);CHR$(CA);
720 RETURN

730 REM GENERATE LEGEND ROUTINE.
740 LN=LN+4:CN=55:GOSUB 690
750 FOR J=1 TO 4
760 PRINT CHR$(27);"8";CHR$(CHARID);
770 NEXT J
780 LN=LN+1:CN=55:GOSUB 690
790 RETURN

800 END
```

```
*
 *
  *
   *
    *
     *
      *
       *
        *
         *
          *
           *
            *
             *
              *
               *
                *
                 *
                  *
                   *
                    *
                     *
                      *
                       *
                        *
```

FIGURE 1

STRAIGHT LINE

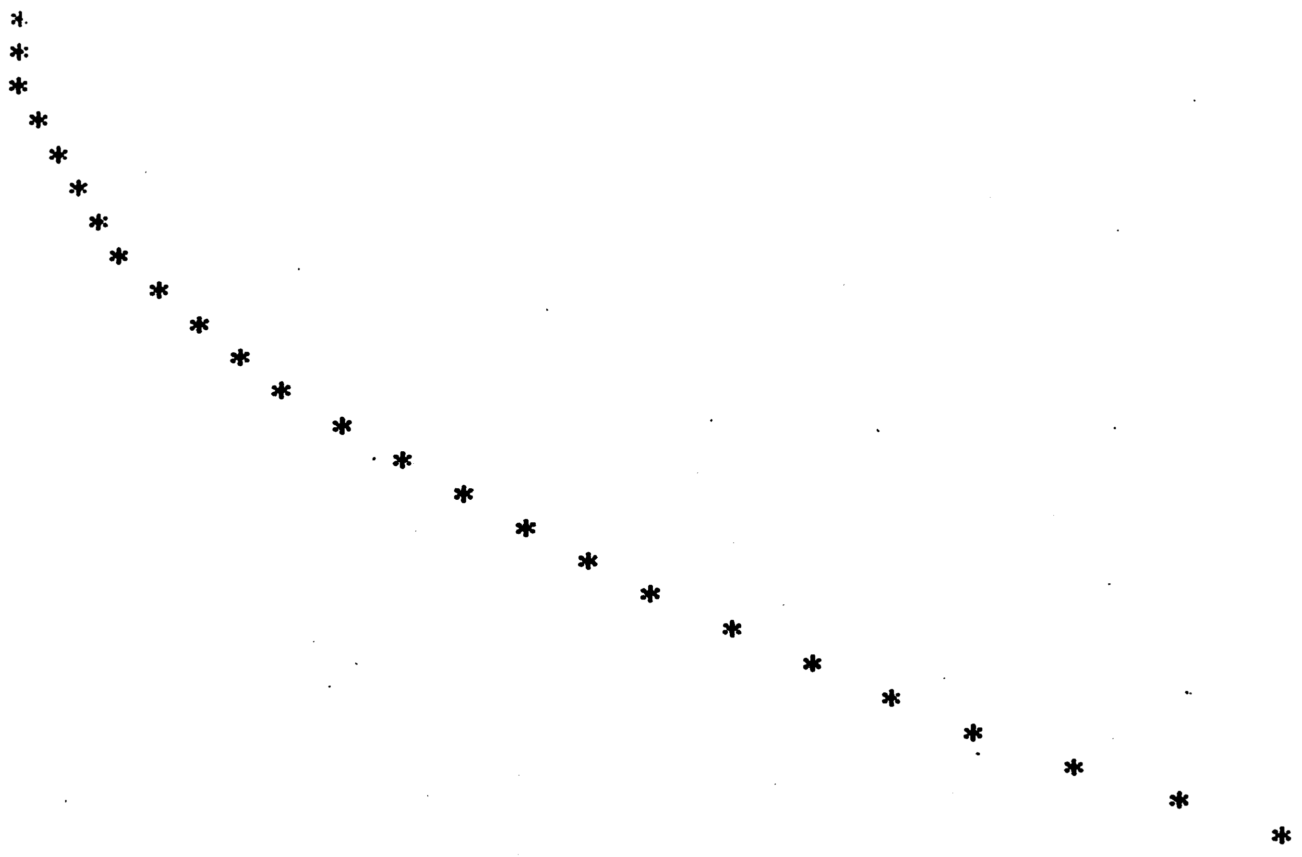
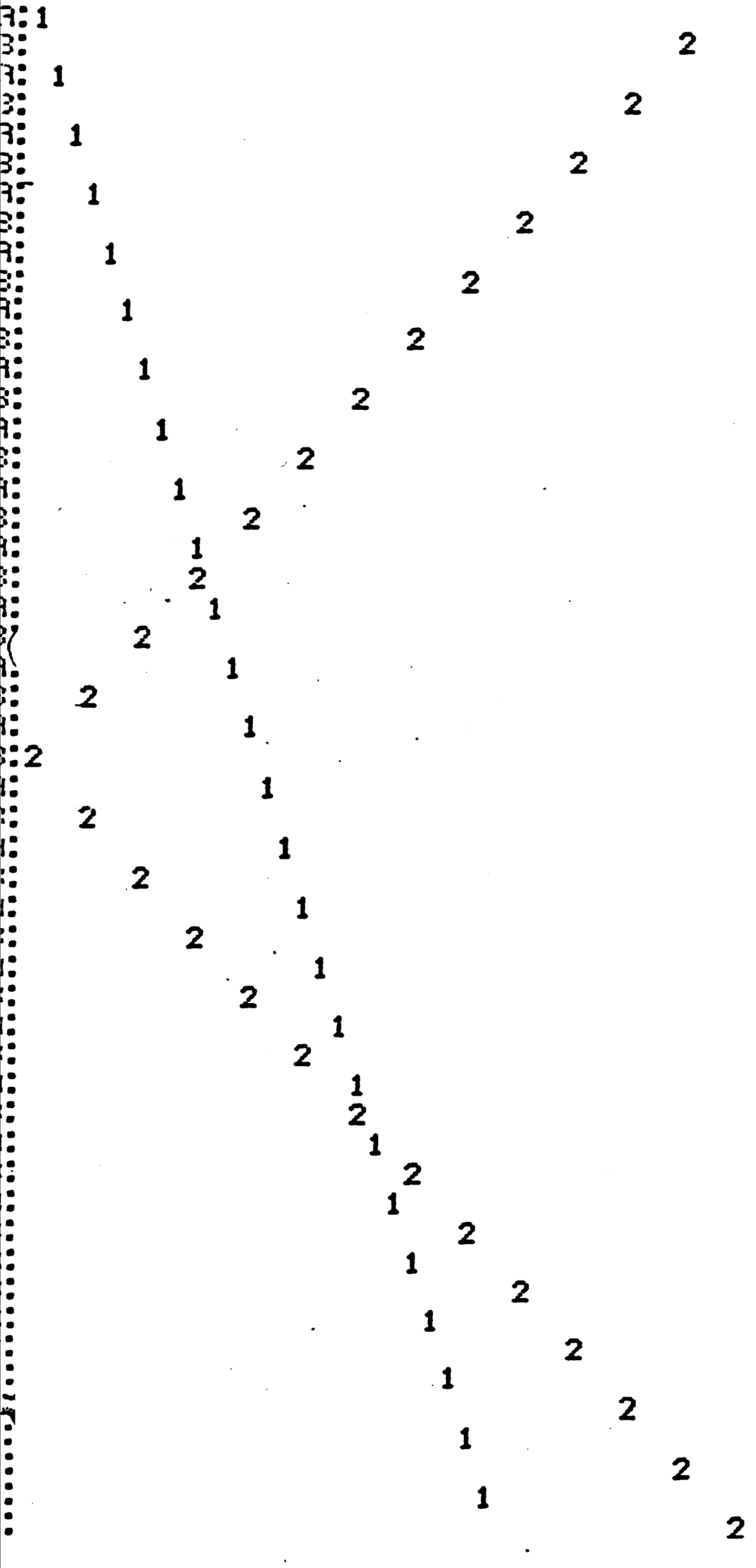


FIGURE 2

QUADRATIC CURVE

FIGURE 3

TAB LINES



```
GRAPH OF WEEKLY WEIGHTS

    100  110  120  130  140  150  160  170  180  190  200
     +    +    +    +    +    +    +    +    +    +    +
1 (                             *
2                            *
3 .                       *
4 I                      *
5 I                    *
6 I                   *
7 I                  *
8 I                 *
9 I                  *
10
 I                   *
11
 I                   *
12
 I                    *
13
 I                    *
14
 I                    *
15
 I                     *
16
 I                      *
17
 I                      *
18
 .                       *
19
 I                        *
20
 I                          *
21
 I                           *
22
 I                        *
23
 I                   *
24
 I               *
25
 I             *
26
 I            *
27
 I           *
28
 I          *
29
 I          *
30
 I          *
  GE WEIGHT = 135.7133
```

<u>FIGURE 4</u>

WEIGHT GRAPH

```
       -1          -.61        -.21         .19         .6          1  8
                                       *
.2                                           *
.4                                                 *
.6                                                      *
.8                                                           *
1                                                               *
1.2                                                                *
1.4                                                                  *
1.6                                                                  *
1.8                                                                  *
2                                                                 *
2.2                                                            *
2.4                                                        *
2.600001                                              *
2.800001                                         *
3.000001                                   *
3.200001                             *
3.400001                       *
3.600001                  *
3.800001             *
4.000001        *
4.2          *
4.4       *
4.6      *
4.8      *
5         *
5.2         *
5.399999        *
5.599999            *
5.799999                 *
5.999999                       *
6.199999                             *
```

FIGURE 5

SIN GRAPH

```
                                            HARVEY KILOBIT
                                              HARVEY KILOBIT
                                             HARVEY KILOBIT
                                           HARVEY KILOBIT
                                       HARVEY KILOBIT
                                   HARVEY KILOBIT
                                HARVEY KILOBIT
                              HARVEY KILOBIT
                             HARVEY KILOBIT
                              HARVEY KILOBIT
                                 HARVEY KILOBIT
                                     HARVEY KILOBIT
                                        HARVEY KILOBIT
                                            HARVEY KILOBIT
                                              HARVEY KILOBIT
                                              HARVEY KILOBIT
                                           HARVEY KILOBIT
                                      HARVEY KILOBIT
                               HARVEY KILOBIT
                        HARVEY KILOBIT
               HARVEY KILOBIT
        HARVEY KILOBIT
     RVEY KILOBIT
HARVEY KILOBIT
HARVEY KILOBIT
  HARVEY KILOBIT
       HARVEY KILOBIT
              HARVEY KILOBIT
                       HARVEY KILOBIT
                               HARVEY KILOBIT
                                     HARVEY KILOBIT
                                           HARVEY KILOBIT
                                             HARVEY KILOBIT
                                              HARVEY KILOBIT
                                            HARVEY KILOBIT
                                         HARVEY KILOBIT
                                     HARVEY KILOBIT
                                 HARVEY KILOBIT
                              HARVEY KILOBIT
                             HARVEY KILOBIT
                             HARVEY KILOBIT
                                HARVEY KILOBIT
                                   HARVEY KILOBIT
                                       HARVEY KILOBIT
                                           HARVEY KILOBIT
                                             HARVEY KILOBIT
                                              HARVEY KILOBIT
                                            HARVEY KILOBIT
```

FIGURE 6

COMBINED TRIG FUNCTIONS

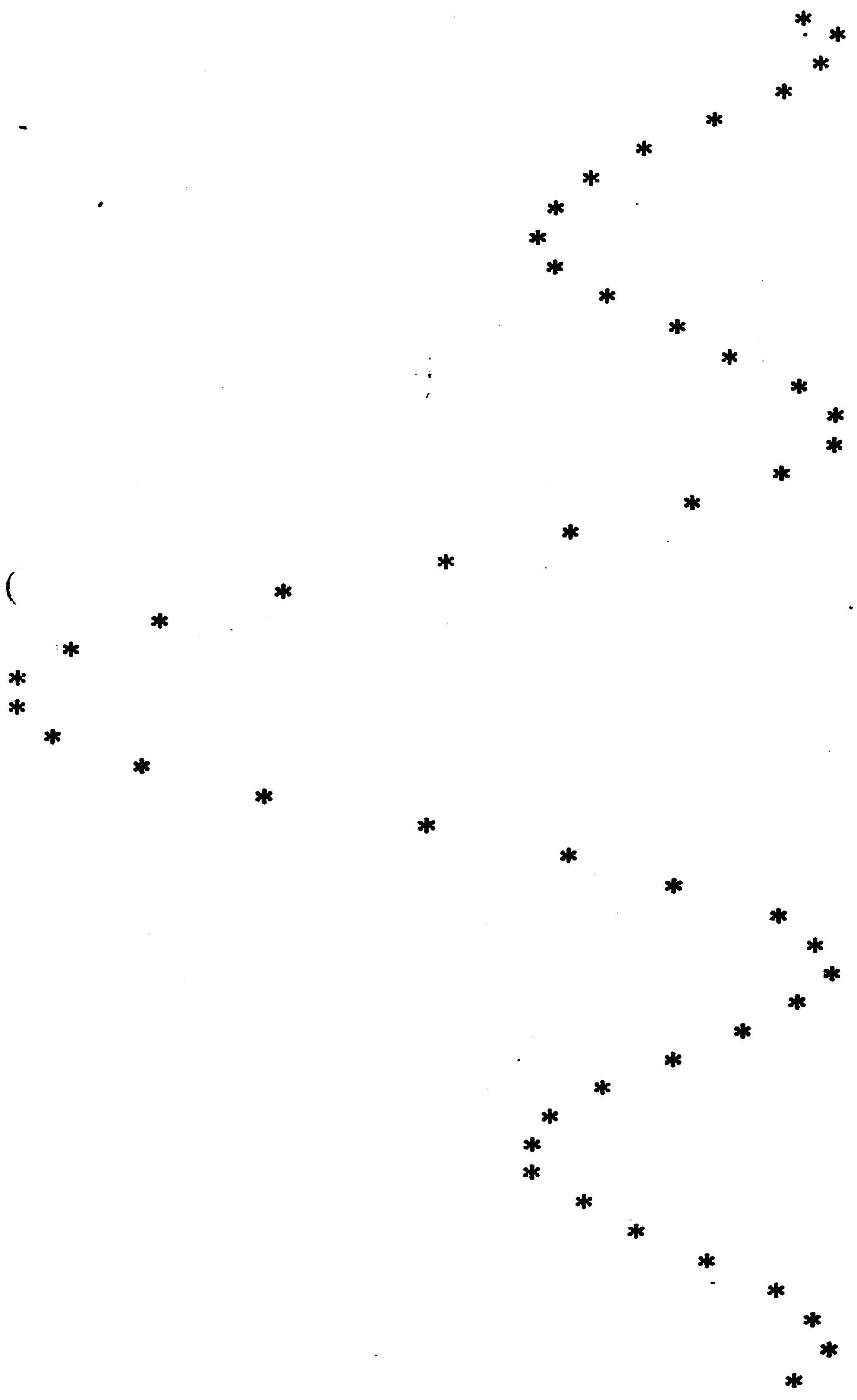


FIGURE 6A

COMBINED TRIG FUNCTIONS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0				▓	1/	3/	5/	7/	°		→	…	÷	↓	¯	¯	¯	¯	¯	¯
20	¯	¯	^{0}	^{1}	^{2}	^{3}	^{4}	^{5}	^{6}	^{7}	^{8}	^{9}		!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	¦	}	~	¶				▓	1/	3/	5/	7/	°	±	→	…
140	÷	↓	¯	¯	¯	¯	¯	¯	¯	¯	^{0}	^{1}	^{2}	^{3}	^{4}	^{5}	^{6}	^{7}	^{8}	^{9}
160		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3
180	4	5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E	F	G
200	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[
220	\	]	^	_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
240	p	q	r	s	t	u	v	w	x	y	z	{	¦	}						

```
PRINT SCREEN ? (Y/N) y
```

FIGURE 7

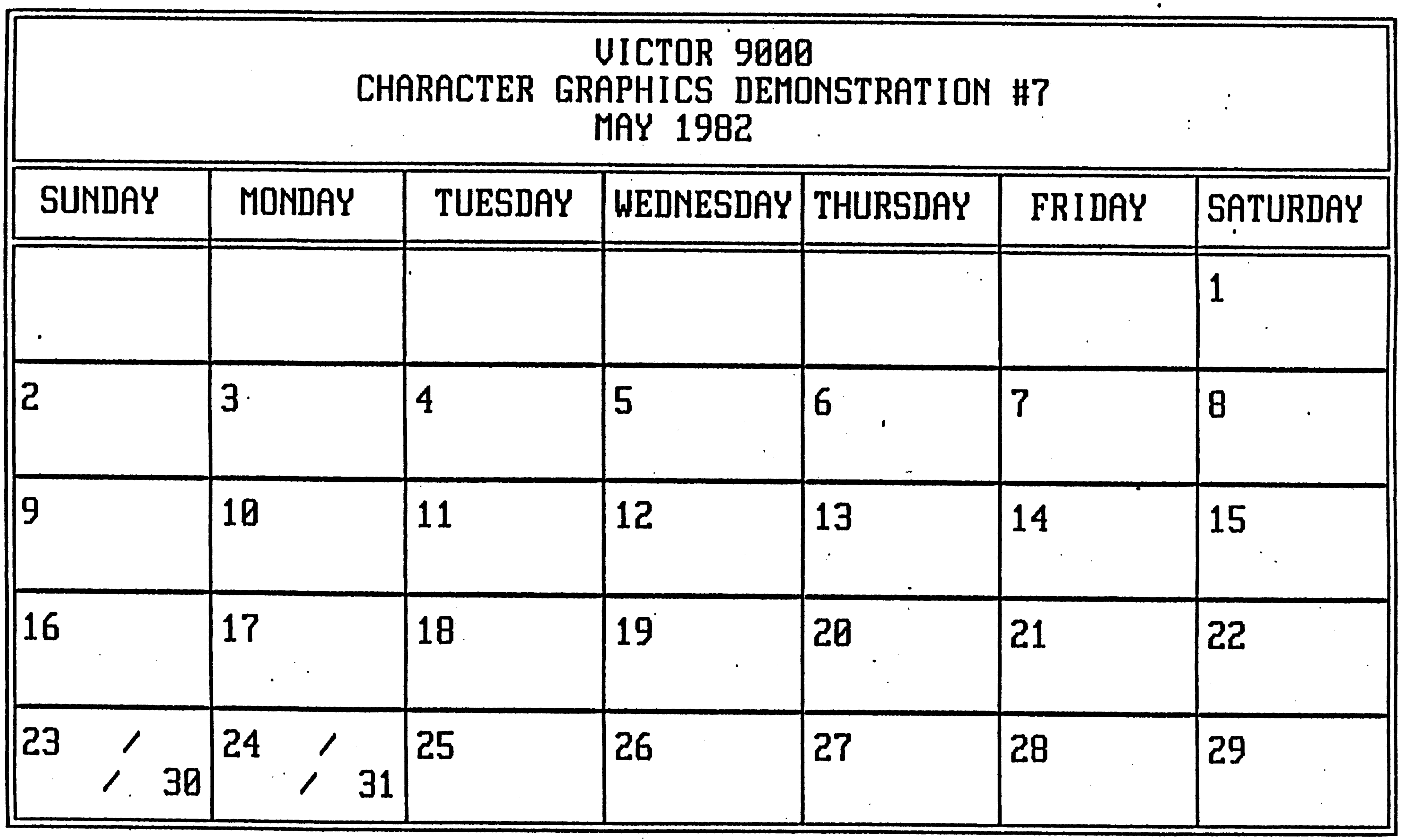
VT52T.CHR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		☺	☻	♥	♦	♣	♠	•	◘		◙	♂	♀	♪	♫	☼	►	◄	↕	‼
20	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼		!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	⌂	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï
140	î	ì	Ä	Å	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
160	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»	░	▒	▓	│
180	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐	└	┴	┬	├	─	┼	╞	╟
200	╚	╔	╩	╦	╠	═	╬	╧	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█
220	▄	▌	▐	▀	α	β	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
240	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²						

```
PRINT SCREEN ? (Y/N) y
```

FIGURE 8

VINTL01.CHR



VICTOR 9000
CHARACTER GRAPHICS DEMONSTRATION #7
MAY 1982

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23 / / 30	24 / / 31	25	26	27	28	29

FIGURE 9

CALENDAR

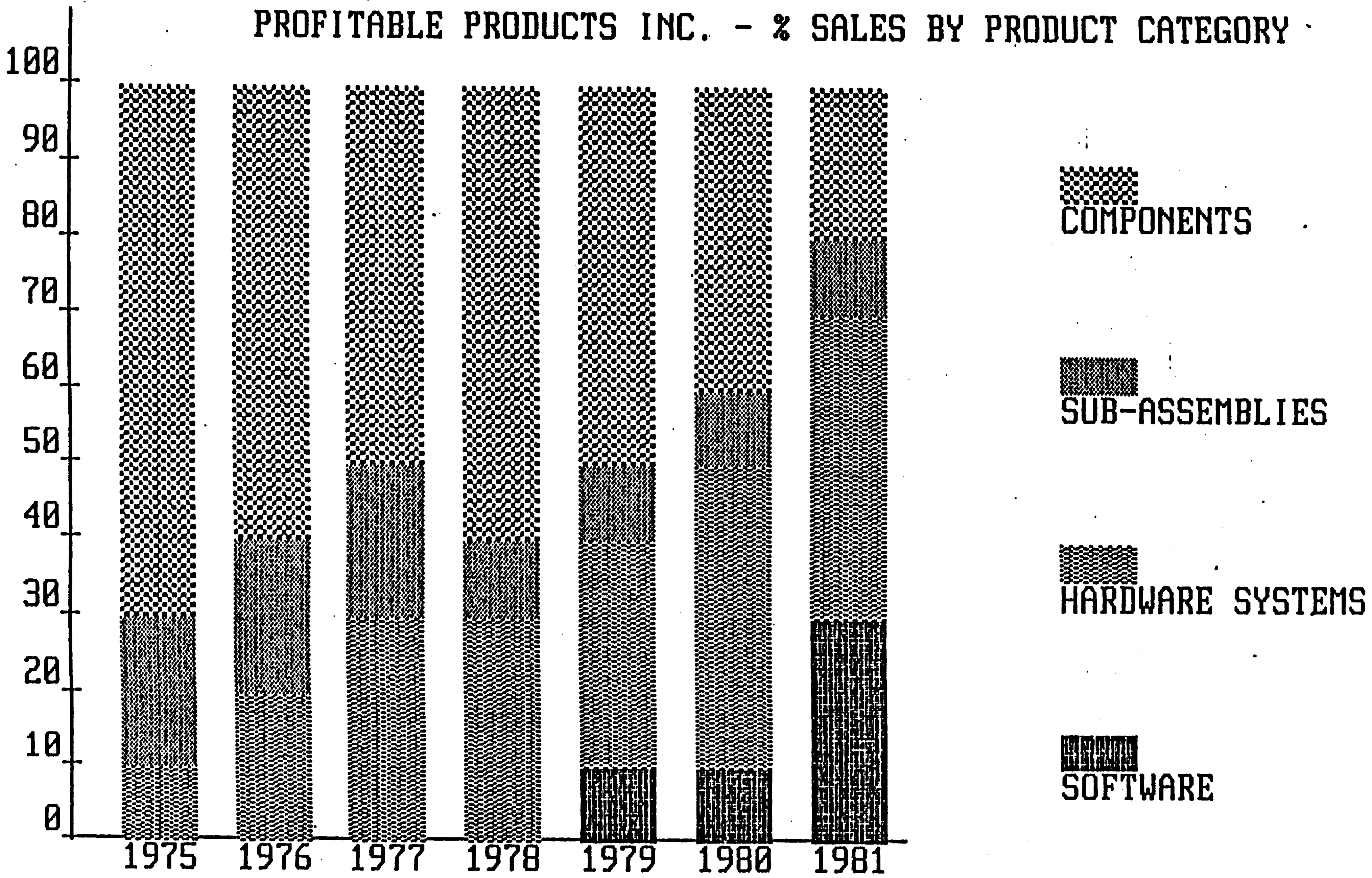
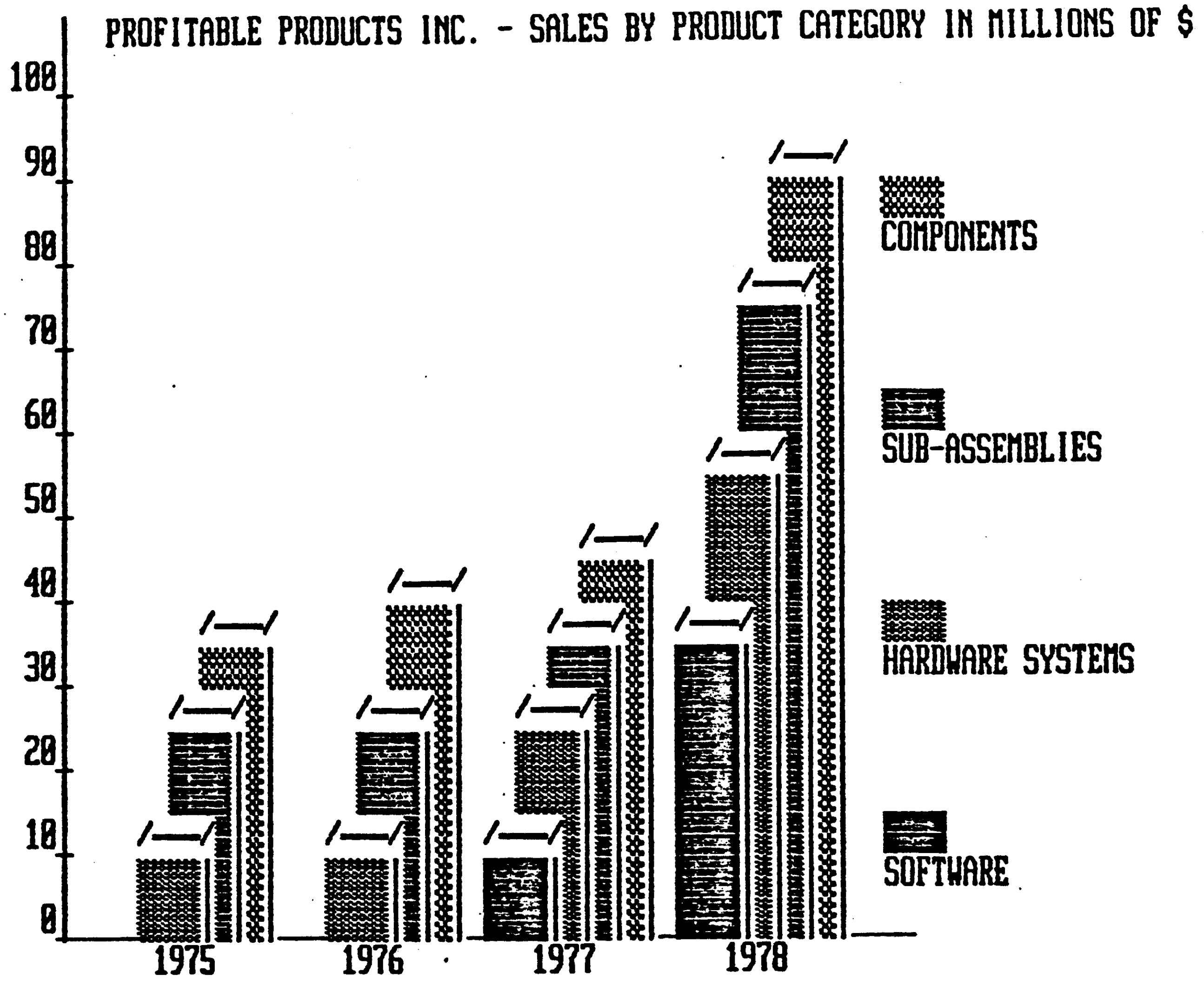


FIGURE 10

% SALES BY PRODUCT CATEGORY


PROFITABLE PRODUCTS INC. - SALES BY PRODUCT CATEGORY IN MILLIONS OF $
100
90
80
70
60
50
40
30
20
10
0
1975
1976
1977
1978
COMPONENTS
SUB-ASSEMBLIES
HARDWARE SYSTEMS
SOFTWARE

<u>FIGURE 11</u>

<u>3-D</u>

VICTOR 9000 CHARACTER GRAPHICS

ABSTRACT

This document provides an introduction to the character graphics capabilities of the Victor 9000. Terms are defined, the Victor 9000 Display System is discussed, available character sets are pictured, and short BASIC programs are included to aid in displaying individual characters from the set as well as the complete set. Additional short BASIC programs are included to demonstrate the usefulness of the TAB function in displaying graphics and to generate additional illustrative displays using the features discribed above.

TABLE OF CONTENTS

1. INTRODUCTION

Character Graphics is defined as the process of constructing fundamental graphic forms by using a previously defined character set. The VICTOR 9000 provides an extended capability through special characters within selected character sets (eg, Victor International) or through the character creation utility.

This document does not address the method and use of the High Resolution bit mapped graphics capability of the VICTOR 9000. This is covered by other documentation on the subject of HI RES