CP-30 CHARACTER PRINTER

MARCH, 1979

INFOTEK SYSTEMS CORPORATION
1400 N. BAXTER STREET
ANAHEIM, CA 92806

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

TABLE OF CONTENTS

INFOTEK CP-30 CHARACTER PRINTER
FOR THE HEWLETT-PACKARD 9830A/B COMPUTER
INSTRUCTION MANUAL

The INFOTEK CP-30 character printer is a fast, highly versatile printer which is ideally suited for word-processing operations and for making graphs or plots of any kind for which typewriter characters are acceptable. The quality of the output is equal to that of the finest typewriters and it is possible to make excellent carbon copies. The printing mechanism is the Diablo Systems, Inc. HYTYPE II and print wheels and ribbons for the equipment are usually readily available locally or can be obtained from INFOTEK.

The great speed of the printer is achieved through the use of a "daisy wheel" which is a very light plastic disk with long fingers carrying at their ends the character set. Printing occurs when an electromagnetically actuated "hammer" strikes the back of the finger carrying the desired character, driving it against the ribbon and transferring the impression to the paper. The moment of inertia of the disk is very small and this facilitates rotation of the disk to position the proper character in front of the hammer.

This manual is divided into several sections. First there is a simple introduction into the use of the CP-30 in conjunction with the HP 9830A/B which is designed to familiarize you with the characteristics of the CP-30 and something of its use for word-processing operations. Part II will explain operation as a plotter. Part III will describe how you can use the INFOTEK Option 214 14K buffer as an extended memory for the HP 9830A/B calculator.

You should recognize that programming the CP-30 is much easier if you have the INFOTEK Typewriter ROM, but all of the same operations can be carried on without it, albeit at the cost of more difficult debugging, and use of additional programming memory.

PART I

BECOMING FAMILIAR WITH THE CP-30

suming that you have connected up the CP-30 to your HP 9830A/B calcu-
tor in accord with the instructions in Appendix A, turn on the 9830
d press the POWER and ON LINE buttons on the CP-30. The red and green
ghts should come on and the CP-30 go into its power-up routine which
volves a self-testing procedure, moving of the print head to the left
rgin, and an upward movement of the ribbon into the printing position.
ter power-up the CP-30 assumes a number of default print parameters
ich will be described later in detail.

ur first task will be to learn how to communicate with the CP-30 which
es not understand the HP BASIC but only a modified set of ASCII codes.
ese codes are summarized in Table 1 and you will see that they corr-
oond to the decimal numbers 1 to 126. Of these, 32 to 126 code for
ious printed characters, while 1 to 31 are codes for non-printing
mands. There are also a number of multiple codes, each preceded by 27
h as 27,69 which is the equivalent of RESET.

can not send the simple number 65 to the CP-30 and have it print
. Instead, you have several options in getting the ASCII code 65
the CP-30 to achieve the printed "A". First is the simple and
iliar WRITE(6,*)"A" and EXECUTE which is converted to ASCII 65 by
9830 and sent to the CP-30. (Here and throughout we assume your
30 has been assigned SELECT CODE 6). Second and probably less
iliar is WRITE(6,100)65; and EXECUTE, where 100 is a format statement
h as 100 FORMAT 80B. The use of the B format is explained in some

Table 1. ASCII Decimal Codes and CP-30 Responses

Decimal code	CP-30	Decimal code	CP-30	Decimal code	CP-30
0		32	space	64	@
1		33	!	65	A
2		34	"	66	B
3		35	#	67	C
4		36	$	68	D
5		37	%	69	E
6		38	&	70	F
7	bell	39	'	71	G
8	back space	40	(	72	H
9	horizontal tab	41	)	73	I
10	line feed	42	*	74	J
11	vertical tab	43	+	75	K
12	form feed	44	,	76	L
13	carriage return	45	-	77	M
14	shift out	46	.	78	N
15	shift in	47	/	79	O
16		48	0	80	P
17		49	1	81	Q
18		50	2	82	R
19		51	3	83	S
20		52	4	84	T
21		53	5	85	U
22		54	6	86	V
23		55	7	87	W
24		56	8	88	X
25		57	9	89	Y
26		58	:	90	Z
27		59	;	91	[
28		60	<	92	\
29		61	=	93	]
30		62	>	94	^
31		63	?	95	_

able 1 (cont). ASCII Decimal Codes and CP-30 Responses

ecimal code	CP-30	Decimal code	CP-30
96	`	27,10	reverse line feed
97	a	27,25	pointer buffer control
98	b	27,32	again
99	c	27,37	buffer pointer value
100	d	27,38	FIFO buffer control
101	e	27,46	character-fill setup
102	f	27,49	set horizontal tab
103	g	27,50	clear horizontal tab
104	h	27,51	clear all horizontal tabs
105	i	27,52	horizontal tab left
106	j	27,53	set vertical tab
107	k	27,54	clear vertical tab
108	l	27,55	clear all vertical tabs
109	m	27,56	vertical tab up
110	n	27,65	absolute plot
111	o	27,67	character replacement
112	p	27,68	peek
113	q	27,69	reset
114	r	27,70	form length
115	s	27,72	horizontal spacing
116	t	27,76	text length
117	u	27,77	set left margin
118	v	27,79	plot origin
119	w	27,82	relative plot
120	x	27,83	variable spacing
121	y	27,84	top of form
122	z	27,86	vertical spacing
123	{	27,87	text width
124	¦	27,97	absolute plot with fill
125	}	27,114	relative plot with fill
126	~	27,122	self test
		27,256	stop, reset pointer

detail in the HP 9830 A/B instruction manual and particularly in Appendix F. Your final option for sending ASCII 65 to the CP-30 is possible if you have the HP 11272B Extended I/O ROM by use of WRITE(6,*)WBYTE65 and EXECUTE. This mode has no advantage over use of Format B except in being slightly faster in execution and not requiring that a particular format statement (as 100 FORMAT 80B) be previously placed in the calculator memory. To summarize, execution of each of the following will print the letter "A".

```
WRITE(6,*)"A"

WRITE(6,100)65
(where 100 FORMAT 80B, or a similar Format B statement has
already been entered in the 9830 memory)

WRITE(6,*)WBYTE65
```

If you try these out you will find that after print of the letter "A", each option results in a carriage return and line feed. These subsequent operations can be suppressed if the appropriate WRITE statement is followed by a semicolon, as in the following examples:

```
WRITE(6,200)"A";
(where 200 FORMAT F1.0, or a similar format specification for
printing a number is in the 9830 memory and is unfulfilled
by execution of the WRITE statement. Note that WRITE(6,*)WBYT
256; is a fulfilled statement and the semicolon with it
does not suppress the line feed and character return.)

WRITE(6,100)65;
(where 100 FORMAT 80B has been entered previously)
```

With last of these options you can print out all of the characters on the print wheel of your CP-30 on a single line with a program

uch as the following:

```
10 FOR J=32 TO 126
20 WRITE(6,100)J;
30 NEXT J
40 WRITE(6,100)10,13;
100 FORMAT 80B
110 END
```

ne statement 40 WRITE(6,100)10,13; produces a carriage return and line eed and is included so that the print head does not remain at the far ight of the paper.

t is usually easier to print words or phrases with statements as RITE(6,*)"Now is the time..." or WRITE(6,*)A$ where A$ is a string ontaining the desired characters to be printed rather than using Format or WBYTE. However, there are a number of characters that the CP-30 can rint which are not on the 9830 keyboard as well as the non-printing nstructions to the CP-30, all of which must be transmitted by the SCII codes using Format B or WBYTE statements. This is true also of which can not be printed directly, but only by ASCII 34. Numerical nformation gives no problems because it can be printed by the usual RITE(6,*)N or WRITE(6,200)N statements, as dictated by whatever format ormat you require.

n order to permit you to use your CP-30 effectively for simple perations, It is necessary that you become acquainted with some of the efault (powerup) parameters and also with the concept of "print rea" as it applies to this printer. You will notice on powerup that ne print head moves from whatever position it was in to its left-hand echanical limit and zero on the character-spacing scale. If you set the

paper guide in back of the platen (print roller) to the same zero and insert your paper to line up with this zero then you will have the who area of the sheet with which to work.

On powerup or after pressing the RESET button on the CP-30, the print area is defined as a rectangle with its upper left-hand corner being t exact location of the print head. This location is called the reference point and because the CP-30 remembers this location throughout subsequent operations, and because the print head can be returned to it as long as you stay within the print area, it is important that you understand how it is defined and how its location ca be changed. The default width of the print area is 13.2 inches and the default length is one line less than 11 inches. The default value for the so-called "Form Length" is 11 inches. The form-length parameter is important when you use Z-fold paper or forms in strips and wish to go to a precisely determined location on a new page or form when the current page or form is finished. The way this works is that when the print head goes below the lower bound of the print area, a "Form Feed" is executed and the paper moves to the reference point on the next sheet. If you use single sheets, the sheets are simply ejected - i this is not desired then you can make the form-length parameter larger than the length of your single sheets, but as you will see a little later, this has to be done with certain precautions.

The default character spacing for the CP-30 is 10 characters per inch, and the default line spacing is six lines per inch. These parameters with the standard Diablo print wheel (Pica 10) supplied by INFOTEK with your CP-30 give an attractive printed page. Later, you will learn how to change these default spacings. Because you will want to set your own limits on the print area and be able to specify your margins, these will

e our next topics.

n defining a print area, you need first to establish the reference oint. To clear your CP-30's memory and get ready for a fresh start, ress RESET or turn the CP-30 POWER off and then on. (Another reset rocedure will discussed later.) Assuming you do not have the HP 1272B Extended I/O ROM, it is best to set up a B format in the 9830 emory by executing: 100 FORMAT 80B. Next decide how wide you want your argin to be in tenths of inches. This number will correspond to the umber of spaces (at ten spaces per inch) you use in the sequence RITE(6,100)" ",27,77; and EXECUTE. The semicolon is ecessary to suppress carriage return and line feed, and the 27,77 code ets the "Left Margin", at the location of the print head after the pacing is complete. For a one-inch margin, use ten spaces as corres- onds to ten characters. These steps define the X bound of the refer- nce point, Left Margin and left edge of the print area - all of which re the same distance from the left-hand limit of travel of the print ead.

ou can also set Left Margin by execution of the two-step sequence RITE(6,100)" "; followed by WRITE(6,100)27,77; but you will ind that the print head will not move over the specified steps in the irst step. This is not a malfunction - the CP-30 is programmed not to espond to commands asking it to print spaces until it has been told to o a specific task after the spacing commands are completed. Then, nstead of stepping off the spaces one by one, it jumps to the end of he asked-for total space and does its next job. This mode of dealing ith spaces is an important time saver in long printing operations with any spaces in the text.

ow turn the paper in the platen so that, when you are ready, printing ill start at the desired position down from the top of the page. The

print head is now at your reference point and you can set its upper bound with WRITE(6,100)27,84; and EXECUTE. Here 27,84 codes for establishing the upper (Y bound) of the reference point, or as it is often called "Top of Form". Printing of a statement such as WRITE(6,*)"The quick brown fox.." now will be followed by return of the print head to the left margin.

The previous RESET put the default right-hand margin specification of 13.2 inches in the memory of the CP-30 and this distance is now the "Text Width" of the print area. However, because you have just set a ten-character left margin, it has to be a hypothetical width because t right-hand edge of the print area is beyond the mechanical right-hand limit of movement of the print head. If you now send the CP-30 a strin of more than 122 characters to print, and the margin is 10 characters wide, the print head will reach, and have to move against, its mechanical limit. This results in an automatic RESET, resumption of the default print characters, and loss of all of the special print parameters that you have fed in. Consequently, it is desirable to set right-hand margin limit which at least prevents that eventuality. With left margin of 1 inch, this would be a maximum of 12.2 inches. The widt of the print area or "Text Width" is set by a complicated coding system.

You will be able to save much time in working through what follows if you have the following expressions on special-function keys.

```
INT(X*120/64-INT((X*120/64)/256)*256);
INT(X*120-INT((X*120)/256)*256););
INT(Y*2*48/64-INT((Y*2*48/64)/256)*256);
INT(Y*2*48-INT((Y*2*48)/256)*256);
```

r if you have the FAST BASIC I ROM:

```
MOD(INT(X*120/64),256),MOD(INT(X*120),256);
MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
```

and Y in these expressions correspond to directions to the CP-30 re-
arding distances in inches. The reason the expressions in X and Y
iffer is because the horizontal (X) movements are in 1/120-inch steps
thus X*120) while the vertical movements are in 1/48-inch steps (thus
*48). We will henceforth use the coding compatible with the FAST BASIC
ROM because it is shorter. If you do not have this ROM, the longer
xpressions will work equally well.

et us suppose that you decide on a Text Width of 11.5 inches. Execution
f X=11.5 followed by execution of WRITE(6,100)27,87,MOD(INT(X*120/64),
56),MOD(INT(X*120),256); will set this Text Width. In general, for a
ext Width of N inches, the sequence is X=N and EXECUTE followed by
RITE(6,100)27,87,MOD(INT(X*120/64),256),MOD(INT(X*120),256);.

ou could well ask at this point, "Why not set the Text Width to the
eal print-area width that I want?" You can, but the results under the
ormal mode of operation may not be what you want. The reason is that
hen the CP-30 reaches the left edge of the print area it automatically
xecutes a carriage return and a line feed – so if you are printing a
ong phrase and the print head reaches the right margin, the printing
ill stop (in the middle of a word you might not want to see divided)
nd then continue on a new line. For most text material it is usually
etter to regulate the line lengths, line feeds, and word division
nder program control.

y locating the reference point and setting the right margin, you have
efined the print area except for its length ("Text Length"). This
an be done for N inches through execution of Y=N followed by:

WRITE(6,100)27,76,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);

The rationale here is that the platen rolls the paper in 1/48 inch steps. As a result if you want to have the text length be 9 inches, th would be 432 1/48 inch increments.

Some ways of moving the platen cause the print head to reach but not exceed the lower bound of the print area. However, if the CP-30 is typing text and would exceed the lower limit as the result of a line feed, then a Form Feed occurs and printing continues on the next sheet if you are using Z-fold paper or on the bare platen if single sheets.

The Form Length parameter and its default value of 11 inches was mentioned earlier. The value of 11 inches was chosen because of its useful ness with Z-fold paper. Each time the print head crosses the lower limi of the print area, the platen will turn just the right amount to bring a fresh page into position. You can cause a "Form Feed" to occur at any time in either the calculator or program modes by WRITE(6,100)12; and EXECUTE. When using single sheets this is a rapid way of removing the sheets from the printer.

You can set a different Form Length by Y=N and WRITE(6,100)27,70, MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256); and EXECUTE, where N is the Form Length in inches and can be considered to be the result of N*4 turns of the platen in 1/48 inch increments.

Unwanted or unexpected form feeds can be a nuisance in the following kind of situation. Suppose that you have the CP-30 print half of a page and find that it has a serious enough error to warrant stopping the process, making the desired corrections, and putting in fresh paper for a new run. As far as the CP-30 is concerned, it only knows that you stopped it with half a page yet to go. As a result when the CP-30

reaches the middle of your new page, it thinks it has come to the end of your old page and executes a Form Feed, leaving you with a blank half page. The solution to this problem is before you put in your fresh paper, WRITE(6,100)12; and EXECUTE (ASCII12 codes for Form Feed). An alternative is to execute a new Top of Form (Code 27,84). These operations do not change the other print parameters. One should not press RESET on the CP-30 to attempt to avoid an unwanted Form Feed, because this will cause loss of the margin setting and reset all of the print parameters to their default values. Execution of WRITE(6,100)256; (to be explained a little later) clears the buffer but does not set a new Top of Form.

Another way to get an unexpected Form Feed is to set your Text Length greater than your Form Length. When you do this and the print head crosses the lower limit of the print area, the platen turns the paper down until the FORM Length specification is reached and then types on top of what is already there or possibly on the bare platen.

A tremendous advantage of the CP-30 over an ordinary typewriter is the ease with which the line and letter spacings can be changed. This is of particular value when it is necessary to fill in forms that have odd spacings or when it is desired to squeeze a few more lines of a letter on one page rather than have the extra trouble of preparing a continuation page. The "Horizontal Spacing" between the characters is changed by the statement WRITE(6,100)27,72,INT(120/X/64),INT(120/X) where X is the desired number of characters per inch. The "Vertical Spacing" is set by WRITE(6,100)27,86,INT(2*48/Y/64),INT(2*48/Y); where Y is the number of lines you want per inch. At more than 13 characters per inch or 8 lines per inch and using the standard Pica 10 print

print wheel, you will find that overlap of the typed characters bec
important.

The power of the spacing and form commands can best be illustrated
making a listing of one of your programs on Z-fold paper. Assuming
the page size is 8.5x11 inches and there is a perforated tear-off m
gin 21/32 inch wide, you should make several trial listings using t
following sequence. Put your paper in the printer and press the RES
button on the CP-30. Follow this with:

Y=11 and WRITE (6,100)" ",27,77,27,84; and WRITE(6,100)27,
70,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);

You should be able to recognize in this sequence the concatenation
Left Margin, Top of Form, and Form Length (set at 11 inches). If we
the Text Length as 9 inches and the Text Width as 8 inches, the pro
statements are:

Y=9 and WRITE(6,100)27,76,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),25
then
X=8 and WRITE(6,100)27,87,MOD(INT(X*120/64),256),MOD(INT(X*120),256)

We need now to specify the Horizontal Spacing (H) and the Vertical S
ing (V) by X=H and Y=V followed by:

WRITE(6,100)27,72,INT(120/X/64),INT(120/X);
and
WRITE(6,100)27,86,INT(2*48/Y/64),INT(2*48/Y);

Good trial parameters for H and V would be respectively 13,7 (very tight), 10,6 (standard), and 8,5 (quite loose). You should now load a program and try LIST#6 and EXECUTE with these horizontal and vertical spacing parameters.

If you have the INFOTEK OPT 140 Buffer for your CP-30, you may want to interrupt the listing at some point and you will find to your surprise that, unlike other printers you might use with the 9830, the printing of the listing does not stop when you press the STOP key. The reason is that the buffer on the CP-30 already contains part or all of the list and is continuing to execute even though the 9830 is no longer feeding it fresh material. The way to suspend CP-30 operation in this situation is to take the printer off-line by pressing the ON LINE button on the CP-30. This, however, does not clear the buffer and, if you return the printer to ON LINE, it will resume operation where it left off. RESET will clear the buffer, but will also cause the CP-30 to assume the default parameters and lose the margin and form settings. Unless you wish to continue with the printing of the material in the buffer, the commands WRITE(6,100)256; or WRITE(6,*)WBYTE256; and EXECUTE should be used to clear the buffer without resetting the print parameters. However, as mentioned earlier, WRITE(6,100)256; or its equivalent WRITE(6,*)WBYTE256; does not set a new Top of Form.

Entering new parameters by this procedure is tedious at best and you may wish to include a set of steps like the following at the beginning of each of your programs that uses the CP-30 or put them on a special function key.

```
10 DIM A$[80]
20 BEEP
30 DISP "TURN ON CP-30 PWR AND 'ON LINE'"
40 WAIT 30000
50 DISP "READY, YES=' '";
60 INPUT A$
70 IF A$#" " THEN 20
80 REM RESETS CP-30, IN CASE POWER WAS ALREADY ON.
90 WRITE (6,100)27,69;
100 FORMAT 80B
110 REM SETS TOP OF FORM
120 WRITE (6,100)27,84;
130 DISP "CHARACTERS/IN? ' ' FOR STND (10)";
140 INPUT A$
150 IF A$=" " THEN 190
160 X=VAL(A$)
170 REM SETS CHARACTER SPACING
180 WRITE (6,100)27,72,INT(120/X/64),INT(120/X);
190 DISP "LINES/IN? ' ' FOR STND (6)";
200 INPUT A$
210 IF A$=" " THEN 250
220 Y=VAL(A$)
230 REM SETS LINE SPACING
240 WRITE (6,100)27,86,INT(2*48/Y/64),INT(2*48/Y);
250 DISP "LEFT MARGIN, # CHARACTERS WIDE";
260 INPUT N
270 REM MOVES HEAD TO MARGIN POSITION
280 FOR J=1 TO N
290 WRITE (6,100)32;
300 NEXT J
310 REM SETS LEFT MARGIN
320 WRITE (6,100)27,77;
330 DISP "TEXT WIDTH IN INCHES";
340 INPUT X
350 REM SETS TEXT WIDTH
360 WRITE (6,100)27,87,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
370 DISP "TEXT LENGTH IN INCHES";
380 INPUT Y
390 REM SETS TEXT LENGTH
400 WRITE (6,100)27,67,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
410 DISP "FORM LENGTH IN INCHES";
420 INPUT Y
430 REM SETS FORM LENGTH
440 WRITE (6,100)27,70,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
450 END
```

e CP-30 has both horizontal and vertical tabs. These tabs can be set, eared, or used to position the print head either in the calculator or ogram mode. We will assume now that you will want to have your CP-30 erate like a typewriter in the sense that the tab positions are set d used in character units. First, you should remove any tabs present the CP-30's memory (also done routinely by RESET or CP-30 POWER off, en on - both, of course, at the cost of reverting to the default rgin and form parameters). "Clear All Horizontal Tabs" is achieved WRITE(6,100)27,51; while "Clear All Vertical Tabs" results from ITE(6,100)27,55; and EXECUTE.

ndividual tabs are set and cleared at the current position of the print ead. Therefore, to set two horizontal tabs at 5 and 15 spaces from the eft margin, you should first be sure the print head starts from that argin by causing a carriage return through WRITE(6,100)13; and EXECUTE. ne tab can be set at 5 and 15 spaces by execution of:

```
RITE(6,100)"     ",27,50,"          ",27,50,13;
```

n this sequence, the print head moves five spaces and then code 27,49 erforms "Set Horizontal Tab", without a carriage return. The print ead now moves the next ten spaces and code 27,49 sets the second tab. ode 13 returns the print head to the left margin. In printing perations, the print head moves right to successive tabs on Horizontal Tab Right" which can be brought about by WRITE(6,100)9; nd EXECUTE. "Horizontal Tab Left" uses WRITE(6,100)27,52;.
o clear an individual horizontal tab, the print head has to be at the ab location and this is most easily done by starting at the right

margin and using the necessary WRITE(6,100)9; statements to arrive at the tab. This should be followed by WRITE(6,100)27,50; ("Clear Horizontal Tab").

Vertical tabs can be set in the same manner as horizontal tabs, except that now you have to put in the requisite number of line feeds to get the platen, instead of the print head, at the proper position to set the tabs. There is no simple way to return the platen to the upper bound of the print area which is comparable to ASCII code 13 which takes the head to the left margin. One way is to use enough "Reverse Line Feeds" in the form of WRITE(6,100)27,10; statements to go to the upper bound or else you could employ the more complex following statement (the rationale of which will be explained later) where Y has to be set greater or equal to the Text Length:

```
WRITE(6,100)27,65,0,0,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
```

When you have the paper back to the upper bound of the print area, vertical tabs can be set, used, and cleared by execution of the following statements. For the purpose of providing a definite example, we will assume that you want to set vertical tabs at lines 4, 10, and 13. For this you will use "Set Vertical Tab", Code 27,53;

For our specific example:

```
WRITE(6,100)10,10,10,27,53,10,10,10,10,10,10,27,53,10,10,10,27,53;
```

This sequence would be followed by Y=20 and then

```
WRITE(6,100)27,65,0,0,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
```

if you want to return to the top of the print area (assumes Text Length

less than 20 inches).

ertical Tab Up", Code 27,58;
lls the platen up to the next tab or to the upper bound of the print
ea, whichever comes first.

ertical Tab Down", Code 11;
lls the platen down to the next tab or to the lower bound of the print
ea, whichever comes first.

lear All Vertical Tabs", Code 27,55;

lear Vertical Tab", Code 27,54;
ears a vertical tab, if there is one, at the current position of the
aten. Probably the easiest way to clear a particular tab is to go to
e bottom edge of the print area with WRITE (6,100)27,65,0,0,0,0; (the
micolon is vital here to prevent a Form Feed at the end of movement of
e platen) and EXECUTE. Then you can step back through the vertical
bs with Vertical Tab Up (Code 27,56) until the tab desired to be
moved is reached, followed by WRITE(6,100)27,54; and EXECUTE.

e tedium of setting tabs by the above procedure can be greatly
lieved by carrying on the process under program control. The following
ogram is illustrative of how this can be done, and you may find it
eful to combine this program with the one given earlier for setting
e form and print parameters.

```
10 DIM A$[80]
20 REM THIS PROGRAM ASSUMES THAT THE CP-30 IS POWERED AND ON LINE - ALSO
30 REM THAT THE FORM AND PRINT PARAMETERS ARE ALREADY SET. IT IS POSSIBLE
40 REM TO SET THE TABS IN EITHER INCHES OR UNITS OF CHARACTERS AND LINE F
50 DISP "TAB UNITS? IN.(0),CRTRS&LINES(1)";
60 INPUT N
70 IF N#0 THEN 110
80 CFLAG 0
90 GOTO 120
100 FORMAT 80B
110 SFLAG 0
120 REM RETURNS PLATEN AND PRINT HEAD TO REFERENCE POSITION ON ASSUMPTION
130 REM TEXT LENGTH IS LESS THAN 20 INCHES AND FAST BASIC 1 ROM IN 9830A/
140 Y=20
150 WRITE (6,100)13,27,65,0,0,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256
160 REM CLEARS ALL EXISTING TABS.
170 WRITE (6,100)27,51,27,55;
180 DISP "HOW MANY HORIZONTAL TABS";
190 INPUT N
200 IF FLAG0 THEN 330
210 FOR J=1 TO N
220 IF J>1 THEN 260
230 DISP "INCHES TO FIRST TAB";
240 GOTO 270
250 REM DISTANCES ARE FROM ONE TAB TO THE NEXT.
260 DISP "INCHES TO NEXT TAB, #";J;
270 INPUT X
280 REM MOVES PRINT HEAD TO TAB POSITION AND SETS TAB.
290 WRITE (6,100)27,82,MOD(INT(X*120/64),256),MOD(INT(X*120),256),0,0,27
300 NEXT J
310 WRITE (6,100)13;
320 GOTO 460
330 FOR J=1 TO N
340 IF J>1 THEN 370
350 DISP "CHRCTRS TO FIRST TAB";
360 GOTO 380
370 DISP "CHRCTRS TO NEXT TAB, #";J;
380 INPUT L
390 FOR K=1 TO L
400 WRITE (6,100)32;
```

```
0 NEXT K
0 REM SETS HORIZONTAL TAB.
0 WRITE (6,100)27,49;
0 NEXT J
0 WRITE (6,100)13;
0 DISP "HOW MANY VERTICAL TABS";
0 INPUT N
0 IF FLAG0 THEN 620
0 FOR J=1 TO N
0 IF J>1 THEN 530
0 DISP "INCHES TO FIRST TAB";
0 GOTO 540
0 DISP "INCHES TO NEXT TAB, #";N;
0 INPUT Y
0 Y=-Y
0 REM TURNS PLATEN TO TAB POSITION AND SETS TAB.
0 WRITE (6,100)27,82,0,0,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),27,5
0 NEXT J
0 Y=20
0 WRITE (6,100)13,27,65,0,0,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
0 GOTO 770
0 FOR J=1 TO N
0 IF J>1 THEN 660
0 DISP "LINES DOWN TO FIRST TAB";
0 GOTO 670
0 DISP "LINES DOWN TO NEXT TAB, #";N;
0 INPUT L
0 FOR K=1 TO L
0 WRITE (6,100)10;
0 NEXT K
0 REM SETS VERTICAL TAB
0 WRITE (6,100)27,53;
0 NEXT J
0 REM RETURNS PRINT HEAD AND PLATEN TO REFERENCE POINT.
0 Y=20
0 WRITE (6,100)13,27,65,0,0,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
0 DISP "TABS COMPLETE, OK=' '";
0 INPUT A$
0 IF A$#" " THEN 50
0 END
```

In some circumstances you may wish to move the print head to a partic. ular location in the print area to insert words, phrases, or special symbols. If you know, or can calculate, the X,Y coordinates of the poi where you want the print head to relocate with respect to an origin located at the lower left-hand corner of the print area, then executio of the following "Plot Absolute" (Code 27,65) statements will do the job. However, first you have to enter the desired X and Y values for t expressions in inches:

WRITE(6,100)27,65,MOD(INT(X*120)/64,256),MOD(INT(X*120),256); followed by WRITE(6,100)MOD(INT(Y*2*48)/64,256),MOD(INT(Y*2*48),256);.

If you have previously set a "Plot Origin" (Code 27,79) in a plotting mode as is described in Part II of this manual, it may be first necessary to restore the origin to the lower left corner of the print area before trying to move the print head to a particular X,Y location. This can be done with WRITE(6,100)27,79,0,0,0,0; and EXECUTE.

Before you power up the CP-30, you turn the platen (paper roller) by hand, you will find it turns quite easily, without the usual clicking detent action that characterizes an ordinary typewriter. After powerup however, you will encounter a rather stiff detent action and each "click" corresponds to a paper advance of 0.083 inch. Two clicks therefore amount to the default line spacing of six lines per inch. In circumstance where you wish to insert characters or words in particula locations on an already typed page and if the line spacing is six line per inch, you can turn the platen by hand from the reference position line up the characters to be added with the already printed lines. Thi

ocedure does not work if the line spacing is different from six per
ch or if the print head or platen have been moved in such a way as to
ke the desired location not an integral number of clicks of the platen
tent away from the reference position.

ere is no easy way to place accurately a particular character in any
sired X,Y location on a page if X and Y are not well enough known so
to permit use of the X,Y values and the following Code 27,65
pressions:

ITE(6,100)27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256); followed
ITE(6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);

nother way is to bring the print head into the desired position
sually through successive Code 27,65 operations with trial X,Y values
or the incremental "Plot Relative" with Code 27,82, see below) and
ake the horizontal alignment by sighting along the red line on top of
he print head. Vertical alignment is more difficult, but fairly good
esults are possible if you push in on the hand wheel at platen so as to
llow it to turn independently of the detent and then line up the top
dge of the ribbon with the desired position of the bottom edge of the
haracter(s) to be printed. This alignment will make the printed
osition too low on the paper, so follow it with a "Plot Relative"
Code 27,82) upward move of the paper:

RITE(6,100)27,82,0,0,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);

here Y (in inches) will depend on the mechanical adjustments of your

where Y (in inches) will depend on the mechanical adjustments of your CP-30. Usually Y = 0.07 is about right. WRITE(6,100)"(your text)"; and EXECUTE will then allow you to make your insertion. Some experimentation to find the best value of Y is desirable but the value should remain unchanged unless the ribbon height is readjusted.

Movement of the print head to some particular point with an unknown X,Y position is virtually impossible to do conveniently except under program control. The following program provides one way that this might be done. If you would only want to type in one place on a sheet, you can turn the platen manually down to the desired vertical level and then use the program which follows, with Y=0, to obtain the proper horizontal location of the print head. The program is more general in that both the vertical and the horizontal alignments are done under program control and then the print head and platen can be returned to the reference point.

```
10 DIM A$[80]
20 REM THIS PROGRAM ASSUMES THAT THE CP-30 IS POWERED AND ON LINE AS
30 REM WELL AS HAVING THE FORM AND PRINT PARAMETERS SET.
40 REM THE PLATEN AND PRINT HEAD ARE MOVED TO THE "PLOT ORIGIN" (LOWER
50 REM LEFT-HAND CORNER OF PRINT AREA) TO BE ENSURE THAT THE DESIRED FI
60 REM LOCATION IS IN VIEW. THE VERTICAL LOCATION IS BEST TAKEN AS THE
70 REM TOP EDGE OF THE RIBBON AND A SMALL UPWARD MOTION USED TO POSITIO
80 REM THE PRINT WHEEL.
90 REM RESETS THE ORIGIN AND MOVES TO THE ORIGIN.
100 FORMAT 80B
110 WRITE (6,100)27,79,0,0,0,0,27,65,0,0,0,0;
120 DISP "INITIAL GUESS OF X,Y (INCHES)";
130 INPUT X,Y
140 REM MOVES TO X,Y.
150 WRITE (6,100)27,82,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
160 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
170 REM -X TO THE LEFT, +X TO THE RIGHT, -Y DOWN, +Y UP
180 DISP "NEW X,Y INCREMENTS, (0,0=END)";
190 INPUT X,Y
200 IF X OR Y THEN 150
210 REM MOVES THE X,Y LOCATION BELOW THE RIBBON EDGE TO PERMIT TYPING
220 REM AT THE DESIRED X,Y LOCATION, REMEMBER THE PLATEN ONLY MOVES IN
230 REM IN 1/48 INCH STEPS.
240 DISP "MOVE BELOW RIBBON EDGE, ' '=YES";
250 INPUT A$
```

```
0 IF A$#" " THEN 360
0 DISP "Y=0.07 INCH, ' '=YES; OR Y=";
0 INPUT A$
0 Y=0.07
0 IF A$=" " THEN 360
0 A$[2]=A$[1]
0 A$[1,1]="0"
0 Y=VAL(A$)
0 IF  NOT Y THEN 360
0 WRITE (6,100)27,82,0,0,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
0 DISP "RETURN TO REF. PT., ' '=YES";
0 INPUT A$
0 IF A$#" " THEN 410
0 Y=20
0 WRITE (6,100)27,65,0,0,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
0 END
```

some circumstances, you may wish to move the print head in aracter units rather than in inches. In the program or calculator de, the head will move to the right on execution of statements such as ITE(6,100)" ";. The move will occur immediately if there a command to perform some operation at the end of the move, in the sence of such a command, the spaces are stored in the CP-30 buffer and e not executed until, and if, it is received). However, moving the int head to the left in character units is less straightforward and an best be done by the "Backspace" statement (Code 8). Thus, RITE(6,100)8,8,8,8,8; and EXECUTE moves the head back five spaces. It s important to know that Backspace can also be used to move the print ead into the region beyond the left margin of the print area. It is he only command other than a complete RESET which allows one to go to he left of this margin. A limitation on Backspace is that if the print ead is 5 spaces from the left edge of the print area and that edge of he print area is 10 spaces from the left-hand mechanical limit of ovement of the print head, and you then execute Backspace fifteen imes (or any number >15), the print head is backspaced against its echanical limit, the CP-30 goes into its RESET routine, and reverts to he default print parameters.

The beeping tone of the CP-30 can be activated in either the calculator or the program mode by "Bell" (Code 7) using WRITE(6,100)7,7,7;. The Bell can be used as an alarm to indicate completion of particular program operations and is louder and more distinctive than the 9830 beep. If Bell is used independently of printing operations, you must be sure that the CP-30 is powered and ON LINE.

The CP-30, like all daisy-wheel printers, has the disadvantage relative to an ordinary typewriter of not permitting you to see the character that has just been struck or the precise location of where the next character will go. The second difficulty has been addressed in some detail earlier in this manual. The first can be alleviated by the following device called "Peek" (Code 27,68). Peek is inoperative as long the CP-30 buffer contains print information. When the buffer becomes empty, Peek moves the print head to the right following a delay in milliseconds that you specify in the peek statement. For a delay of N milliseconds, you need WRITE(6,100)27,68,MOD(INT(N/64),256), MOD(INT(N),256); where the maximum value of N is 2000 milliseconds. Peek is cancelled by executing the Peek statement with N as a negative number. The delay parameter is designed so as not to reduce the printing speed of the CP-30 when there is material in the buffer to be printed, but give you a chance to view what has been printed when input stops. This is illustrated by the following program which permits you to print words or lines of text using the Peek feature. In this program, a line feed and carriage return will occur whenever a ^ is encountered. This character was chosen to indicate the end of lines because it is not often used in ordinary text material. A form feed occurs whenever ^^ is encountered.

```
DIM A$[80]
REM THIS PROGRAM ASSUMES THAT THE CP-30 IS POWERED AND ON LINE
DISP "DESIRED PEEK DELAY IN MILLISECS";
INPUT N
IF N>2000 THEN 30
REM SETS PEEK VALUE, NEGATIVE NUMBERS CANCEL.
WRITE (6,100)27,68,MOD(INT(N/64),256),MOD(INT(N),256)
DISP "TEXT";
INPUT A$
) FORMAT 80B
) P=POS(A$,"^")
) IF  NOT P THEN 190
) IF P>1 THEN 130
) WRITE (6,100)10,13;
) GOTO 80
) Q=POS(A$,"^^")
) WRITE (6,100)A$[1,P-1]
) IF P=Q THEN 210
) IF P=LEN(A$) THEN 80
) A$=A$[P+1]
) GOTO 110
) WRITE (6,100)A$;
) GOTO 80
) WRITE (6,100)12;
) GOTO 80
) END
```

many word-processing operations, it is desirable to produce "just-
ed" print which means that both the left-hand and the right-hand
rgins are even from the top to the bottom of the page. There are var-
us ways of achieving justification. The most common way used in print-
g books is to change the spacings between the words to make the lines
me out right. An alternative method is to change the spaces between
l of the characters. Either way can be used with the CP-30. To change
e spacing between the characters, you only need to have the contents
the line to be printed contained in a string, have the calculator
termine the length of the string, and adjust the line spacing for each
ne by adjusting the Set Horizontal Spacing parameter (Code 27,72) as
scribed earlier. Lines can be justified by changing the spaces between

the words in several ways and, of these, we will discuss here only that based on the CP-30's ability to provide "Variable Spacing" (Code 27,83).

Variable Spacing, WRITE(6,100)27,83 and EXECUTE, establishes an operating mode for the CP-30 in which each character that is sent to the printer has to be followed by a parameter which specifies the number of 1/120's of an inch that the character occupies. While in the Variable Spacing mode, bidirectional print by the CP-30 is disabled. To cancel Variable Spacing, a Reset (through Code 27,69 or the button on the CP-30) is required.

To justify lines by providing the desired changes in spacing between th words using Variable Spacing, we have to determine the length of the string containing our line, determine the number of spaces in the line and then apportion the needed increment to each space. Finally, the characters have to be sent one by one to the CP-30, each followed by th appropriate spacing parameter, which can have a maximum value of 255 (1/120's of an inch). One problem with this method of justification is that the extra width to be added to the spaces is an integral multiple of 1/120 inch and a roundoff error can be introduced which will prevent precise alignment of the right-hand margins.

To avoid this problem, the following program adds the accumulated roundoff error to the last space in the line. This in turn may lead to considerable uneveness in the last spaces, so when the roundoff error is 50% or more of the width between the words it is then divided equally between the last two spaces.

```
) DIM A$[80]
) REM THIS PROGRAM ASSUMES THAT THE CP-30 IS POWERED AND ON LINE WITH
) REM PRINT AND FORM PARAMETERS ALREADY SET.
) DISP "LENGTH OF LINE DESIRED, INCHES";
) INPUT L
) DISP "MAXIMUM # OF CHRCTRS IN LINE";
) INPUT K
) DISP "DESIRED CHRCTRS/INCH IN WORDS";
) INPUT M
00 FORMAT 80B
10 S=120/M
20 L=INT(L*10)
30 REM SETS CP-30 IM VARIABLE SPACING MODE
40 WRITE (6,100)27,83;
50 DISP "ENTER LINE; '^' AT END=NO JSTFY";
60 INPUT A$
70 A=LEN(A$)
80 IF A <= K THEN 230
90 BEEP
00 DISP "LINE LONGER THAN MAXIMUM"
10 WAIT 30000
20 GOTO 150
30 P=POS(A$,"^")
40 IF  NOT P THEN 320
50 FOR J=1 TO P-1
60 WRITE (6,100)A$[J,J],S;
70 NEXT J
80 REM NOTE HOW EVEN ASCII CODE MUST BE FOLLOWED BY A SPACING
90 REM PARAMETER TO BE INTERPRETED CORRECTLY.
00 WRITE (6,100)10,0,13,0;
10 GOTO 150
20 Q=0
30 REM DETERMINES # OF SPACES
40 FOR J=1 TO A
50 IF A$[J,J]#" " THEN 370
60 Q=Q+1
70 NEXT J
80 REM DETERMINES NUMBER OF 1/120 INCH INCREMENTS TO ADD TO EACH SPACE
90 X=120/M/Q*(L-A)+S
00 REM CALCULATES ROUNDOFF ERROR IN X
10 R=INT(Q*X-Q*INTX)
20 X=INTX
30 IF R<0.5*X THEN 460
40 SFLAG 0
50 GOTO 470
60 CFLAG 0
70 IF X<255 THEN 520
80 BEEP
90 DISP "LINE TOO SHORT TO JUSTIFY"
500 WAIT 30000
```

```
510 GOTO 150
520 N=1
530 FOR J=1 TO A
540 IF A$[J,J]=" " THEN 570
550 WRITE (6,100)A$[J,J],S;
560 GOTO 690
570 IF FLAG0 THEN 590
580 GOTO 620
590 IF N#Q-1 THEN 630
600 X=X+0.5*R
610 GOTO 630
620 IF N=Q THEN 660
630 WRITE (6,100)32,X;
640 N=N+1
650 GOTO 690
660 REM ADDS ROUNDOFF ERROR TO LAST SPACE
670 X=X+R
680 GOTO 630
690 NEXT J
700 GOTO 300
710 REM BE SURE TO RESET THE CP-30 WHEN YOU ARE FINISHED USING VARIABL
720 REM SPACING. IT WILL DO STRANGE THINGS WHEN EACH CHARACTER IS NOT
730 REM FOLLOWED BY A SPACING PARAMETER UNTIL RESET.
740 END
```

Some examples of justified lines printed with this program follow:

```
INFOTEK's CP-30 with Variable Spacing can easily produce justified lines
INFOTEK's CP-30 with Variable Spacing can easily produce justified
INFOTEK's CP-30 with Variable Spacing can easily produce just-
INFOTEK's CP-30 with Variable Spacing can easily produce
INFOTEK's CP-30 with Variable Spacing can easily pro-
INFOTEK's CP-30 with Variable Spacing can easily
INFOTEK's CP-30 with Variable Spacing can
INFOTEK's CP-30 with Variable Spacing
INFOTEK's CP-30 with Variable Spac-
```

ote how in the foregoing program even non-printing codes are followed y spacing parameters (as for example in Statement 300). It is very mportant that Variable Spacing be cancelled by a RESET before any rint operations from the keyboard or in the program mode that does not se Variable Spacing be undertaken. If this is not done, quite amazing esults may be observed, including unprogrammed line feeds, strange arriage motions, and so on.

An interesting and very powerful capability of the CP-30 is "Character Replacement" (Code 27,67). Use of this is most easily illustrated by a specific example. Suppose you are generating text that would normally have many quotation marks (ASCII Code 34) in it, such as in a report of conversation. Quotation marks can not be entered directly from the keyboard as input to a string. However, one could type the text using @ (ASCII Code 64) wherever a quotation mark would be desired to appear in the final text. Thus:

Tom said, @I stopped to talk to Huckleberry Finn!@

With Character Replacement, all of the @'s would be replaced by " as the result of a prior instruction to the CP-30 that henceforth @ is to be printed as ". For this the prior instruction would be WRITE(6,100)27,67,64,1,34; or WRITE(6,100)27,67,"@",1,34; but not WRITE(6,100)27,67,1,""" which would create rather obvious confusion. The general statement is WRITE(6,100)27,67, the character to be replaced (either as the ASCII code number or in quotes), an integer representing the number of characters in the replacement list, and finally the replacement list. For our example, the integer is 1, and the replacement list is 34, the ASCII code for ". The fact that one can replace a given character, by more than just another character, in fact,

with a string of characters, or even by a sequence of ASCII codes suggests many possibilities, because nonprinting commands can be part or all of the replacement list. The exception is 27, which is not allowed to be the replacement character, although it can be part of th replacement list. To remove a Character Replacement condition, enter WRITE(6,100)27,67, character to be replaced,0; (i.e. a zero list).

An example of how to use Character Replacement to achieve other than print would be in storing text that might otherwise be stored as indi vidual rows in an integer array, such as AI(60,40) through having the lines generated as A$ and transferred to a row in the array. This is an efficient use of the storage space because many lines would have less than the full 80 possible characters. More efficient storage is possible by terminating each line as it is generated with a symbol s as ^ and combining successive lines so as to store the text essent- ially as a continuous string. Execution of WRITE(6,100)27,67,"^",2, 10,13; followed by successive transfers of the array to A$ and thenc to the CP-30 by WRITE(6,100)A$; would lead to a carriage return and feed each time ^ is encountered in the printing of any given A$. An- other character such as @ could be used to code for end of paragrap because it is possible to have more than one Character Replacement effect at once.

There are many non-printing codes that are not used by the CP-30, s Table 1, and these can be used to trigger even quite complex operat A non-printing code can be introduced into a string A$ at location OUTPUT (A$(J,J),100)N; where 100 refers to a Format B statement, N the ASCII code number, and the HP 11272B Extended I/O ROM is requi The availability of these unused codes permits greater versatility would be possible for the limited number of printing characters th might be use for replacement codes.

ne very profitable use of Character Replacement would be in the printng of texts containing subscripts and superscripts as are common with athematical expressions, degree signs, chemical formulas, literature eferences, and footnotes. This can be done by using some symbol or code uch as ^ to denote the start of a superscript and another such as @ o indicate the end of the superscript. In this mode, a line typed as:
The weight of C@2^H@5^OH is A@2^^2@ at 0^o@" is typed:

The weight of C_2H_5OH is A_2^2 at 0^o.

he character replacement to make the platen turn up or down (the order epends on whether ^ or @ comes first) uses Plot Relative which was iscussed earlier in connection with moving the print head to a particlar location in your print area. The statements, if you will allow that .0625 inch as used above is a reasonable platen movement, are:

```
RITE(6,100)27,67,"@",6,27,82,0,0,255,250;
nd
RITE(6,100)27,67,"^",6,27,82,0,0,0,6;
```

or @ and ^, respectively.

The general statements for a turn down or turn up value of Y are more complicated. For a turn down the expressions are:

```
WRITE(6,100)27,67,"@",6,27,82,0,0,MOD(INT(-Y*2*48/64),256);
and
WRITE(6,100)MOD(INT(-Y*2*48),256);
```

For turn up:

```
WRITE(6,100)27,67,"^",6,27,82,0,0,MOD(INT(Y*2*48/64),256);
and
WRITE(6,100)MOD(INT(Y*2*48),256);
```

The principal limitation of Character Replacement has to do with the size and structure of the buffer in which the character replacement instructions are stored. There are 300 total available spaces in the buffer and each Character Replacement occupies the integer number of spaces that represent the length of the replacement list plus two. Thus the "turn down" statement used in the foregoing program would take 8 spaces in the buffer. When a program makes many changes in Character Replacement while it is running, there is a serious possibility of overflow. You should know that cancelling a Character Replacement does not remove it from the buffer unless it is the very last one sent in. Furthermore, a change in a replacement specification which is not the last one in, simply adds one new replacement instruction to the set already resident in the buffer. When the buffer is full, the CP-30 will beep if more replacement commands are received. All of the current Character Replacements are cleared from the buffer by RESET.

The statement "Again" (Code 27,32) tells the CP-30 to run off the contents of its print buffer on the basis of 'first-in, first-out' without need for further processing from the 9830. This can be very useful if the contents of the buffer have been obtained by lengthy 9830 operations. However, if you can foresee the need for this, it is a good idea to WRITE(6,100)256; to clear the buffer before beginning, because Again will lead to a printout of any preliminary material that the buffer has accumulated as well as what you want reprinted. If the material exceeds the 14K capacity of the buffer, the overflow will be lost and not reprinted.

"Self Test" (Code 27,122) allows you to check the operation of the printing electronics and produces all of the print wheel characters.

Reset" (Code 27,69) is the program equivalent of the RESET button
n the CP-30. It is a useful step to include at the start of any program
here the print and form parameters are to be changed to ensure that
othing undesired is held over from the previous program. The results of
eset are much the same as turning the CP-30 power off and then back on
gain except that power off loses the contents of the buffer. The list
f default parameters so achieved follows:

1. Print head at position zero, the extreme-left mechanical limit of its travel.
2. All tabs are cleared.
3. Horizontal Spacing is set to 10 characters per inch and Variable Spacing cancelled.
4. Vertical Spacing set to 6 lines per inch.
5. Peek operation cancelled.
6. Top of Form set to the current position of the platen.
7. Left Margin at left mechanical limit.
8. Text Width set to 13.2 inches.
9. Form Length set to 11 inches.
10. Text Length set to be one line less than Form Length (11 inches).
11. All Character Replacements cancelled.
12. Origin for X,Y plotting set to lower left-hand corner of print area.
13. Character Fill parameters (to be described in Part II) set to: decimal point, spacing of 3/48 inch and fill-character offset of +5/120 inch.
14. Buffer pointer (to be described in Part III) reset so as to make the buffer appear to be cleared.

PART II

USE OF THE CP-30 AS A PLOTTER

Setting up the CP-30 to make plots with the HP9830A/B is especially convenient with the INFOTEK Typewriter ROM. Nonetheless, useful plots can be made without it in a straightforward way, although as you will have seen in Part I, the commands for moving the print head to a given location within the print area are relatively complicated. For this reason, you will, as before, find it helpful while doing programming for a plot to have the following expressions on special-function keys:

```
INT(X*120/64-INT((X*120/64)/256)*256),
INT(X*120-INT((X*120)/256)*256),
and
INT(Y*2*48/64-INT((Y*2*48/64)/256)*256),
INT(Y*2*48-INT((Y*2*48)/256)*256),
```

or if you have the FAST BASIC I ROM:

```
MOD(INT(X*120/64),256),MOD(INT(X*120),256),
and
MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),
```

In these expressions, X and Y represent movements in inches. In what follows, we will assume you have the FAST BASIC I ROM but if you do not the first group of expressions will work perfectly well.

For a concrete introduction to the use of the CP-30 as a plotter, you will find it helpful to work through the following steps to produce a plot of the spiral defined by X=K/12+K*cos(k) and Y=K/12+K*sin(k) over the range K=0 to K=1080 where K is in degrees.

You must decide on the total area to be occupied by the plot includ-
7 by 9 inches and, within this area, you will have a 1 inch margin
around for labels so that the actual plot area will take up 6 by 8
hes with the origin in the center of this area. Now RESET and put
r paper in the CP-30 with the upper edge about 1 inch above the print
d and the print head at the left margin. Because the default spacing
10 characters per inch, you can set the left edge of the print area
WRITE(6,100)"   ",27,77,27,84; (see page 8) where
          " covers ten spaces. The select code of your CP-30 is
umed here to be 6 as in Part I and you should already have entered
FORMAT 80B.

e right margin (which calls for a Text Width of 7 inches) and a Text
ngth of 9 inches can be set by use of Codes 27,87 and 27,76 as des-
ibed on pages 10-11.

The default parameters put the origin of the plot at the lower left
rner of the 7 by 9 inch print area. To relocate the origin (Code 27,
) at the center of the print and plot area, execute X=3.5 and Y=4.5
llowed by:

ITE(6,100)27,79,MOD(INT(X*120/64),256),MOD(INT(X*120),256); and
RITE(6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);

emember that positive X values move the print head to the right and
ositive Y values roll the platen down. Negative values do the
everse.

he program steps to here are:

```
0 REM RESET
0 WRITE(6,100)27,69;
0 REM SET LEFT MARGIN AT 1 INCH (TEN SPACES) AND TOP OF FORM
0 WRITE(6,100)"          ",27,77,27,84;
```

```
50 REM SET TEXT WIDTH OF 7 INCHES
60 X=7
70 WRITE(6,100)27,87,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
80 REM SET TEXT LENGTH OF 9 INCHES
90 Y=9
100 FORMAT 80B
110 WRITE(6,100)27,76,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
120 REM SET PLOT ORIGIN TO CENTER OF PRINT AREA
140 Y=4.5
150 WRITE(6,100)27,79,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
160 WRITE(6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
```

3) The next task is to set up the equations for the plot and plan to provide for conversion of the calculated units (user units) to the X and Y units (plotter units) actually used for movement of the print head and platen.

If the calculated values for the function are taken to be X,Y are evaluated at 10-degree intervals, the additional program steps are:

```
170 DEG
180 FOR K=0 TO 1080 STEP 10
190 X=K/12+K*COS(K)
200 Y=K/12+K*SIN(K)
300 NEXT K
```

The X,Y values calculated in these program steps are not ready to plot because they have to be converted from user units to plotter units.

The planned range of the X axis is −3 to +3 inches and of the Y axis from −4 to +4 inches. It is convenient to make each inch in either the X or Y direction equal to 400 user units. This means that the plot will have its maximum value along the X axis equal to 1170/400 or 2.925 inches.

Next you have to decide what character to print at the X,Y location. If

between the current X,Y position of the print head and the new X,Y position. To set up Character Fill (Code 27,46), we need three parameters. The first one P1 is the ASCII code for the fill character. P1 has the default value of 46, the decimal point. The second parameter P2 has two definitions. First, when the line connecting the successive X,Y points is inclined 45 degrees or less with respect to the X axis, then P2 represents the number of characters of fill per inch. Second, when the line is inclined at greater than 45 degrees, P2 represents the number of characters per 1.25 inch. The default value of P2 is 40 characters per inch. (The reason for the double definition of P2 is that horizontal movements are carried on in 1/120-inch steps while vertical movements are in 1/48-inch steps.

The third parameter P3 is a vertical-offset parameter which represents the number of inches that a character is raised (positive value of P3) or lowered (negative value of P3) before it is printed. The default value is +5/48 inch (0.103 inch). The purpose of P3 is to permit a character, such as the decimal point (.) which is low in the print line, or the apostrophe (') which is high in the print line, to be centered along the fill line between the current and new values of X and Y.

The complete Character Fill statement is:

```
WRITE(6,100)27,46,P1,INT(120/P2/64),INT(120/P2),INT(P3*2*48);
```

If it should turn out that the difference in spacing of the fill characters with changes of the slope of the fill line causes an unsatisfactory appearance of the plot, the character-fill setup can be changed as the slope changes to maintain as even spacings as the 1/48-inch increment limitation on the vertical movement of the platen will allow.

Suppose that P2 is 10 characters per inch when the X,Y line is <=45 degrees with respect to the X axis. It then becomes 10/1.25 = 8 characters per inch when the angle is greater than 45 degrees. An illustra subroutine for our earlier program (if it were to use fill characters) that will operate to even up the fill spacings, where the decimal point (ASCII Code 46) is the fill character, P3 is 1.5/48=0.032 inch, X,Y are the new coordinates and A,B are the current coordinate:

```
1200 IF ABS((Y-B)/(X-A))>1 THEN 1230
1210 P2=10
1220 GOTO 1240
1230 P2=12.5
1240 P3=0.032
1250 REM CHARACTER FILL SETUP WITH P1=46, P2 VARIABLE, AND P3=0.032
1260 WRITE(6,100)27,46,46,INT(120/P2/64),INT(120/P2),INT(P3*2*48);
1270 A=X
1280 B=Y
1290 RETURN
1300 END
```

To return to the X-axis problem, you might use either closely spaced hyphens (-) or P3-raised underlinings (_) as the fill characters to make a solid line between the plusses (+) which make excellent tic mar for the 200-degree intervals. With hyphens, 20 characters per inch wil give a good solid axis line. The X-axis can then be generated as follows, where Code 27,114 is for "Plot Absolute with Character Fill", hyphens (Code 45) are the fill characters, P2=20 and P3=0.

```
310 REM RESTORES SPACING AFTER PRINTING OF CHARACTER
320 WRITE (6,100)27,72,INT(120/10/64),INT(120/10);
330 REM SET CHARACTER FILL, P1=45, P2=20, 93=0
340 P2=20
350 P3=0
360 WRITE (6,100)27,46,45,INT(120/P2/64),INT(120/P2),INT(P3*2*48);
```

```
REM MOVE TO LEFT END OF X AXIS, PRINT + (CODE 43)
REM SUPPRESSES SPACES AFTER PRINTING CHARACTER
WRITE (6,100)27,72,0,0;
X=-3
Y=0
WRITE (6,100)27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),43;
REM PRINT + AS TIC EVERY 1/2 INCH WITH HYPHENS AS CHARACTER FILL
X=0.5
FOR L=1 TO 12
REM PLOT RELATIVE WITH CHARACTER FILL (CODE 27,114)
WRITE (6,100)27,114,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),43;
NEXT L
```

the X-axis labels, we could use several different sequences, but easiest when printing 10 characters per inch is probably:

```
REM RESTORES SPACING AFTER PRINTING OF CHARACTER
WRITE (6,100)27,72,INT(120/10/64),INT(120/10);
REM CARRIAGE RETURN, PLOT RELATIVE TO TURN PAPER UP 0.14 INCH
X=0
Y=-0.14
WRITE (6,100)13,27,82,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
REM PRINT X-AXIS LABELS
WRITE (6,100)"  -1200          -600              0          ";
WRITE (6,100)"  600            1200"
```

Y axis is slightly more difficult because the only character on the nt wheel which will give a smooth vertical line is ¦ (Code 124) you will have to have the spacing close enough to overlap the gap in this character. Thus, P1=124, P2=20, and P3=0. The Y-axis line n then be made up as follows:

```
REM SET CHARACTER FILL, P1=124, P2=20, P3=0
P2=20
P3=0
WRITE (6,100)27,46,124,INT(120/P2/64),INT(120/P2),INT(P3*2*48);
```

```
50 REM MOVE TO TOP END OF Y AXIS, PRINT + (CODE 43)
660 REM SUPPRESSES SPACES AFTER PRINTING CHARACTER
670 WRITE (6,100)27,72,0,0;
680 X=0
690 Y=3
700 WRITE (6,100)27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
710 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),43;
720 REM PRINT + EVERY 1/2 INCH WITH CODE 124 AS CHARACTER FILL
730 X=0
740 Y=-0.5
750 FOR L=1 TO 12
760 REM PLOT RELATIVE WITH CHARACTER FILL
770 WRITE (6,100)27,114,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
780 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),43;
790 NEXT L
```

The Y-axis labels are easily printed with the aid of horizontal and vertical tabs (pages 16-20) and this way of doing it will provide you with another illustration of how these tabs are set and used.

```
800 REM RESTORES SPACING AFTER PRINTING OF CHARACTER
810 WRITE (6,100)27,72,INT(120/10/64),INT(120/10);
820 REM CLEAR ALL HORIZONTAL AND VERTICAL TABS
830 WRITE (6,100)27,51,27,55;
840 REM SET HORIZONTAL TAB FIVE SPACES TO RIGHT OF Y AXIS
850 WRITE (6,100)8,8,8,8,8,27,49;
860 REM GO TO LOWER END OF Y AXIS
870 X=0.5
880 Y=-3.5
890 WRITE (6,100)27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
900 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
910 REM SET VERTICAL TABS 1.5 INCH APART
920 FOR Y=-3 TO 3 STEP 1.5
930 REM SET VERTICAL TAB (CODE 27,53) AND PLOT ABSOLUTE (CODE 27,65)
940 WRITE (6,100)27,53,27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
950 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
960 NEXT Y
970 FOR L=1200 TO -1200 STEP -600
980 IF L=0 THEN 1030
990 REM HORIZONTAL TAB LEFT
1000 WRITE (6,100)27,52;
1010 WRITE (6,1020)L;
1020 FORMAT F5.0
1030 REM VERTICAL TAB DOWN
1040 WRITE (6,100)11;
1050 NEXT L
1060 REM FORM FEED
1070 WRITE (6,100)12;
1080 END
```

The complete spiral plot made without character fill between the plotted points is shown in Figure 2.

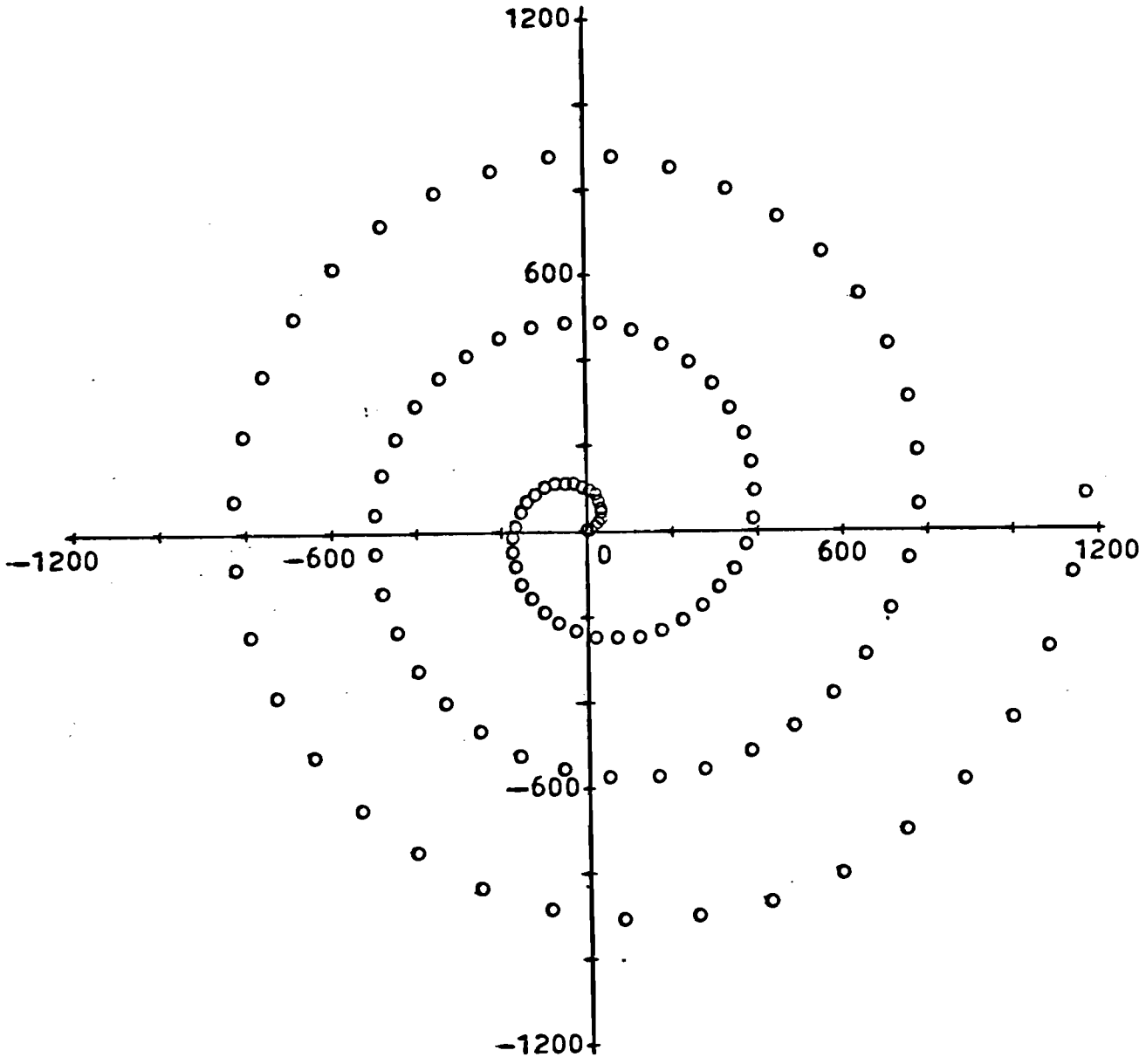


Figure 2. Plot of X=K/12+K*cos(K), Y=K/12+K*sin(K) after addition of axes, labels, and tics.

The corresponding plot with decimal points as character fill is shown in Figure 3. This plot utilizes the subroutine described on page 41 fo

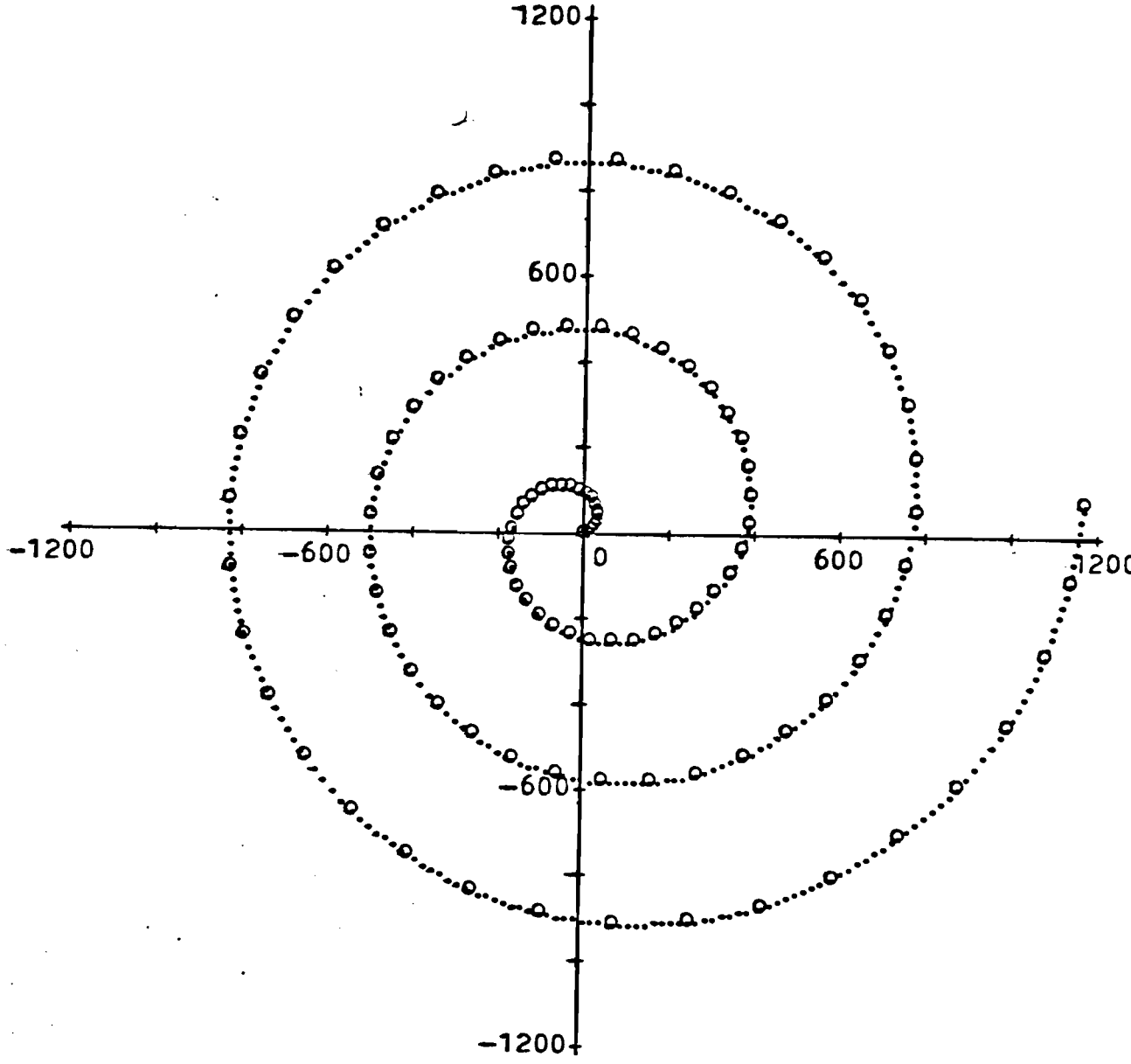


Figure 3. Spiral Plot of Figure 2 with decimals as character fill.

ening out the spacing between the fill characters and the detailed ogram for it is given in Appendix I. You may wish to see how much difrence the GOSUB makes in the appearance of the plot.

ere are too many ways in which graphs and plots can be made to provide u with very general rules or instructions for their form, labeling, d so on, beyond the kind of illustrative example just discussed. Hower, The need for conversion of data in user units to plotter units in ches is very general and a short routine such as the following can sily be made a subroutine on any plot program. This GOSUB handles the igin problem in a different way than the spiral-plot program in that e origin is always taken to be at the lower left corner of the int area. Suitable offset and scaling parameters are used to proply locate the plot in the print area.


```
000 DISP "OVERALL LENGTH OF X AXIS (INCHES)"
010 INPUT XO
020 REM INCHES BETWEEN LEFT EDGE OF PRINT AREA AND LEFT END OF X AXIS
030 DISP "INCHES MARGIN TO LEFT END OF X";
040 INPUT X1
050 DISP "OVERALL LENGTH OF Y AXIS (INCHES)";
060 INPUT YO
070 REM INCHES BETWEEN LOWER EDGE OF PRINT AREA AND LOWER END OF Y AXIS
080 DISP "INCHES MARGIN TO LOWER END OF Y";
090 INPUT Y1
100 REM ENTER VALUES OF X AND Y FOR EXTREMES OF X AND Y AXES
110 DISP "MAX VALS OF X & Y (USER UNITS)";
120 INPUT X2,Y2
130 DISP "MIN VALS OF X & Y (USER UNITS)";
140 INPUT X3,Y3
150 REM CALCN OF INCHES PER USER UNIT (XO,YO)
160 X4=XO/(X2-X3)
170 Y4=YO/(Y2-Y3)
180 RETURN
190 END
```

Conversion of user units to plotter units, Plot Absolute with print of "o" at the new X,Y point, and commands to prevent the print head from straying out of the bounds of the plot area can be achieved by the following additional GOSUB, where X and Y are initially in user units.

```
1200 X=(X-X3)*X4+X1
1210 Y=(Y-Y3)*Y4+Y1
1220 IF X<X1 or X>X1+X0 THEN 1360
1230 IF Y<Y1 OR Y>Y1+Y0 THEN 1370
1240 REM PLOT ABSOLUTE WITH "o" (CODE 111) TO BE PRINTED AT X,Y
1250 WRITE(6,100)27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
1260 WRITE(6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),111;
1270 RETURN
1280 END
```

PART III

USE OF THE CP-30'S BUFFER FOR STORAGE AND DATA PROCESSING

n combination with the HP-11272B Extended I/O ROM, the CP-30 buffer can e separated from its print function and used for read-write data stor-ge and even data processing, at least to the extent such processing can e done by Character Replacement (CODE 27,67, see pages 30-33). The use-ulness of the buffer for this purpose will depend on the Buffer Option ou selected for your CP-30. The regular CP-30 with a 256-character buf-er holds about 3 lines of 72 characters each, the 2048-character buffer bout 28 lines (close to a half page), and the 14,336-buffer almost 200 ines (almost three pages). The buffer is most simply used on a "first-in" "first-out" basis (FIFO) by execution of WRITE(6,100) 27,38;. This command disengages the print function and can only be overruled and operation restored as a printer by WRITE(6,100)256; or RESET.

To illustrate how the buffer may be used in a simple practical oper-ation, consider that we have generated 30 lines of a text with 72 char-acters per line and transferred the lines one-by-one into the first 30 rows of an integer array A(60,36). Now suppose that we wish to fill up the rest of the array with the first 30 lines from another integer array with the same dimensions stored in a data file. A program like the following can be used for this purpose, wherein the first array is transferred to the buffer memory, the file array is loaded, the first 30 of its lines transferred to the buffer and the combined text recovered. Any desired Character Replacement can be carried out at the same time and provision for any number of such replacements of a single character by a single character is made in this program.

```
10 DIM AI[60,36],A$[72]
20 REM ASCII CODES FOR CHARACTERS TO BE REPLACED BY A SINGLE CHARACTER;
30 REM 0,0 FOR THE END OF THE LIST
40 DISP "ASCII CODES-FROM? TO";
50 INPUT P,Q
60 IF  NOT P AND  NOT Q THEN 120
70 REM CHARACTER REPLACEMENT
80 WRITE (6,100)27,67,P,1,Q;
90 GOTO 40
100 FORMAT 80B
110 REM DESIGNATION OF MATERIAL TO STORE IN BUFFER
120 DISP "LINES TO GO IN BFFR; N1,N2 INCL";
130 INPUT N1,N2
140 REM CLEAR BUFFER THEN DISENGAGE PRINT FUNCTION
150 WRITE (6,100)256,27,38;
160 FOR J=N1 TO N2
170 TRANSFER A[J,1] TO A$
180 REM TRANSFER A$ TO BUFFER
190 WRITE (6,100)A$
200 NEXT J
210 DISP "FILE# OF TEXT TO ADD TO N1,N2";
220 INPUT N
230 REM ASSUME TEXT IS IN CASSETTE FILE
240 LOAD  DATA N,A
250 FOR J=N2-N1+2 TO 60
260 TRANSFER A[J,1] TO A$
270 WRITE (6,100)A$
280 NEXT J
290 REM RECOVER TEXT FROM BUFFER
300 FOR J=1 TO 60
310 REM FREE-FIELD FORMAT FOR RECEIVING LINES FROM BUFFER
320 ENTER (6,*)A$
330 TRANSFER A$ TO A[J,1]
340 NEXT J
350 REM RESTORE CP-30 TO PRINTING OPERATION
360 WRITE (6,100)256;
370 REM WRITE TEXT
380 FOR J=1 TO 60
390 TRANSFER A[J,1] TO A$
400 WRITE (6,100)A$
410 NEXT J
420 END
```

You will see from steps 150-200 of the foregoing program that the lines of text represented by the first 30 rows of the array are transferred to the buffer (after the print function is disengaged) by simple WRITE statements. If A$ is 72 characters or less, WRITE(6,*) will suffice, but if more, then you have to use WRITE(6,100)A$ to avoid truncating the text. The strings which are transferred to the CP-30 can be recovered one by one on the FIFO basis by the ENTER(6,*)A$ command (see steps 290-340). This can be done in a straightforward way because, as each string goes into the CP-30 buffer, it is followed by an "end of the line" character and, in the readout by ENTER(6,*)A$, one string at a time is accepted. If you execute ENTER(6,*)A$ and the buffer is clear, the 9830 will "hang up". You can regain control by STOP. You also need to know that in the FIFO mode (Code 27,38), the CP-30 does not respond to any of the normal commands until it receives a WRITE(6,100) 256; RESET, or is turned off and then on again.

A more sophisticated use of the CP-30 buffer is to prepare text in a form which can be edited with the ordinary EDIT keys, a luxury which is not usually possible for text that has been generated as a string and then transferred to an array. The operation here is somewhat similar to that possible with the TEXT command of the HP Data Communications 3 ROM. The procedure is to generate text in the same manner as if you were writing a program but in the form of "10 REM Now is the time for all good---". The REM is most easily added after the line number by entering *REM (with a trailing space) on a special function key. If you use line numbers under 999, you will be able to generate REM lines up to 72 characters long. The lines can be edited by FETCH and use of the EDIT keys. New lines are easily added, the lines renumbered, lines transferred, and so on, just as is possible when writing a program. PRT ALL is helpful to follow your progress and the text can be run off at any time, of course preceded by the line numbers and REM, through the usual LIST command.

An example of a short text prepared as REM program text follows:

```
10 REM This is an example of text prepared by the procedure used for
20 REM writing programs with the HP9830. The lines can be edited as th
30 REM are generated with the EDIT keys or can be recalled by the FETC
40 REM command and then edited in just the same way. Lowercase is
50 REM possible with either the FAST BASIC I ROM or with the HP Advanc
60 REM Programming II ROM. Lines can easily be inserted, deleted, or
70 REM renumbered, just as in regular programming.
80 REM
90 REM The REM which has to follow the line number is most easily intr
100 REM duced from a special-function key. If you forget to put it in,
110 REM you will get ERROR 15, but the line will not be lost. All you
120 REM you have do is to is press RECALL and insert REM at the start
130 REM the line of text. After REM in a line there is no problem abou
140 REM using " as part of the text, although more elaborate procedure
150 REM are necessary to introduce non-keyboard characters which are
160 REM available on the CP-30 print wheel. Because the text will be
170 REM processed by the CP-30 buffer before it is printed, one way to
180 REM to do this is to substitute non-keyboard characters for keyboa
190 REM characters by execution of Character Replacement statements.
```

Because text made as above is not in the computer in ASCII code it has to be translated. This is done automatically by a LIST command. So if you execute WRITE(6,100)27,38; followed by LIST#6, the text will be stored in ASCII code in the CP-30 buffer. Then when you have it read out, it can be edited line by line to remove the number and REM, and line transferred to a suitable array. Sample program steps to do this are given below:

```
10 DIM A$[80],AI[60,40]
20 DISP "NUMBER OF LINES TO RECOVER";
30 INPUT N
40 FOR J=1 TO N
50 ENTER (6,*)A$
60 P=POS(A$,"REM ")+4
70 IF P=4 THEN 90
80 GOTO 120
90 P=POS(A$,"REM")+3
100 FORMAT 80B
110 IF P=3 THEN 150
120 A$=A$[P]
130 TRANSFER A$ TO A[(J,1]
140 PRINT A$
150 NEXT J
160 REM RESTORES CP-30 AS A PRINTER
170 WRITE(6,100)256;
180 END
```

procedure for generating text as REM program lines is excellent
never a whole line has to be added or deleted but is less useful,
eed tedious, when you want to add or delete just a few words near the
rt of a paragraph and need to readjust the length of each succeeding
e to the end of the paragraph. This problem can be corrected in sev-
l ways. Only one will be described here and that is to adjust the
e lengths under program control as they are read out of the CP-30
fer at the same time that the line numbers and REMs are removed.

do this the symbols ^ and ^^ are put in the text to delineate
spectively a point at which a line adjustment is desired and the end
a paragraph. To illustrate, suppose you have a REM text which is
anned to have a maximum of 65 characters per line as follows:

```
0 REM The adjustment of text lines whenever just a few words are added
0 REM to, or deleted from, a given line in the early part of a para-
0 REm graph is not easily done until the lines have been listed and
0 REM are being read out of the CP-30 buffer as strings.^^
```

w suppose that you want to clarify the text by adding "the lengths
the REM " following "of" in Line 100. The first step would be
duplicate Line 100 as Line 105 and, with the EDIT keys, add the
sired phrase, remove the extra words, and add appropriate ^ symbols
that the text becomes:

```
00 REM The adjustment of the lengths of the REM ^
05 REM lines whenever just a few words are added ^
10 REM to, or deleted from, a given line in the early part of a para-
20 REM graph is not easily done until the lines have been listed and
30 REM are being read out of the CP-30 buffer as strings.^^
```

ote that the lines to be adjusted are made flush left. If you have any
ndentations at the left margins or large spaces between words, these

will be retained. Also note that the words at the ends of the segments in Lines 100 and 105 have spaces before the terminating ^. If no spaces are put in these locations, the program assumes that you do not want them. Also, a hyphen at the end of the line (as in Line 110) is assumed to be an indicator of word division and is treated accordingly. A more extensive, illustratively edited REM text follows:

```
100 REM The adjustment of the lengths of the REM ^
105 REM text lines whenever just a few words are added ^
110 REM to, or deleted from, a given line in the early part of a para-
120 REM graph is not easily done until the lines have been listed and
130 REM are being read out of the CP-30 buffer as strings.^^
140 REM To generate printed text from REM program ^
145 REM lines you need only to execute WRITE(6,*)WBYTE27,WBYTE38; which
150 REM disengages the print function of the CP-30, followed by LIST #6,
160 REM and then load the program steps given below.^^
170 REM It is a good idea to store your ^
175 REM text material as a program be- ^
180 REM fore print^
185 REM ing it out so that you can edit again if errors turn up.^^
```

These REM lines when you load and run the program which starts on the next page come out as follows:

> The adjustment of the lengths of the REM text lines whenever just a few words are added to, or deleted from, a given line in the early part of a paragraph is not easily done until the lines have been listed and are being read out of the CP-30 buffer as strings.
>
> To generate printed text from REM program lines you need only to execute WRITE(6,*)WBYTE27,WBYTE38; which disengages the print function of the CP-30, followed by LIST #6, and then load the program steps given below.
>
> It is a good idea to store your text material as a program before printing it out so that you can edit again if errors turn up.

```
0 DIM A$[84],B$[240],AI[60,42]
0 DISP "DESIRED CHARACTERS/LINE";
0 INPUT L
0 DISP "NUMBER OF LINES TO RECOVER";
0 INPUT N
0 J1=N
0 J=1
0 GOSUB 730
0 IF L1 THEN 130
00 FORMAT 80B
10 A$=" "
20 GOTO 510
30 IF L2 OR  NOT L3 THEN 510
40 L4=L3
50 B$=A$
60 GOSUB 730
70 L5=L4+L1
80 IF L5=L4 THEN 160
90 B$[L4]=A$
00 IF L5>L THEN 270
10 IF  NOT POS(B$,"^^") THEN 380
20 A$=B$
30 GOTO 510
40 IF L2 THEN 500
50 L4=L5
60 GOTO 160
70 FOR K=L TO 1 STEP -1
80 IF B$[K,K]=" " THEN 300
90 NEXT K
00 A$=B$[1,K]
10 TRANSFER A$ TO A[J,1]
20 J=J+1
30 B$=B$[K+1]
40 IF L2 THEN 400
50 IF J1>0 THEN 380
60 TRANSFER B$ TO A[J,1]
70 GOTO 540
80 L4=LEN(B$)
90 IF B$[L4,L4]#"^^" THEN 420
00 TRANSFER B$ TO A[J,1]
10 GOTO 520
20 IF B$[L4,L4]="-" THEN 160
```

```
430 IF B$[L4,L4]="^" THEN 160
440 IF B$[L4,L4]#" " THEN 470
450 TRANSFER B$ TO A[J,1]
460 GOTO 160
470 B$[L4+1]=" "
480 L4=L4+2
490 GOTO 160
500 A$=B$
510 TRANSFER A$ TO A[J,1]
520 J=J+1
530 IF J1>0 THEN 80
540 REM PROGRAM ASSUMES HERE THAT THE CP-30 IS POWERED AND PRINT
550 REM PRINT PARAMETERS ARE SET.
560 DISP "PRINT";J;"LINES, OR #";
570 INPUT N
580 WRITE (6,100)256;
590 FOR J=1 TO N
600 TRANSFER A[J,1] TO A$
610 P=POS(A$,"^^")-1
620 IF P<1 THEN 660
630 WRITE (6,100)A$[1,P]
640 WRITE (6,*)
650 GOTO 700
660 P=POS(A$,"^")
670 IF  NOT P THEN 690
680 A$=A$[1,P-1]
690 WRITE (6,100)A$
700 NEXT J
710 STOP
720 END
730 ENTER (6,*)A$
740 P=POS(A$,"REM ")+4
750 IF P=4 THEN 730
760 GOTO 790
770 P=POS(A$,"REM")+3
780 IF P=3 THEN 730
790 A$=A$[P]
800 L1=LEN(A$)
810 L2=POS(A$,"^^")
820 L3=POS(A$,"^")
830 J1=J1-1
840 RETURN
850 END
```

```
)0 REM RECOVER VALUES, STORE IN ARRAY A.
)0 FOR J=1 TO 100
)0 FOR K=1 TO 100
20 ENTER(6,110)A(J,K)
30 NEXT K
40 NEXT J
50 REM RESTORE OPERATION OF CP-30 AS A PRINTER.
50 WRITE(6,100)256;
70 END
```

ne readout from the CP-30 is not as fast as from a file stored in the NFOTEK FD-30 floppy disk, but is more convenient and faster than from tape cassette. The readout from the buffer in the above program of 000 F10.6 numbers requires just 30 seconds.

nother but slower way to read numbers out of the buffer and one which bviates the need for inclusion of the line-feed character, provided the umbers were written in a specified format, is RBYTE. This command in he form of A=RBYTE6 (6 is the Select Code) removes one byte at a time rom the buffer and returns it as A in the form of the decimal equi- alent of the ASCII code of the corresponding character. The values of so obtained can then be assembled in a string and VAL of the string s the desired number. Illustrative program steps for storing an array (100,10) in the buffer using a format of F10.6 and recovering it again ollow:

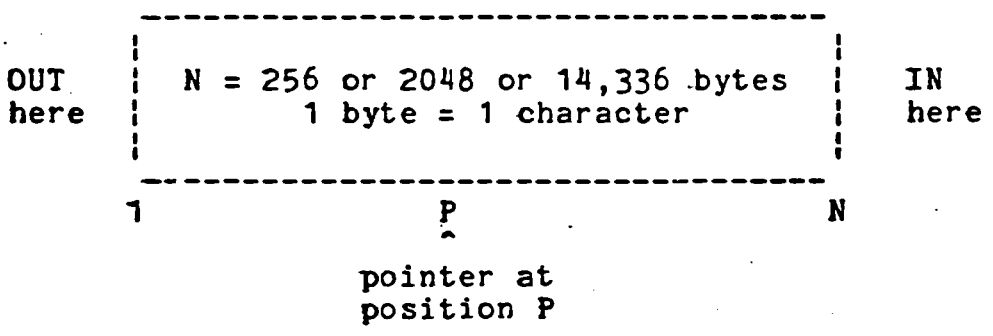
```
)0 DIM N[100,10],B[100,10],A$[10]
100 FORMAT 80B
110 FORMAT 2F10.6
200 WRITE (6,100)27,38;
210 FOR J=1 TO 100
220 FOR K=1 TO 10
230 REM SEMICOLON REQUIRED AFTER WRITE(6,110)N(J,K)
240 WRITE (6,110)N[J,K];
250 NEXT K
260 NEXT J
270 REM RECOVER NUMBERS AND TRANSFER TO ARRAY B
280 FOR J=1 TO 100
290 FOR K=1 TO 10
300 REM UPPER LIMIT OF L IS SAME AS LENGTH OF FORMAT (HERE f10.6)
310 FOR L=1 TO 10
```

```
320 A=RBYTE6
330 REM TRANSFER ASCII CODE FOR CHARACTER TO STRING
340 OUTPUT (A$[L,L],100)A;
350 REM OUTPUT(A$(L,L),*)RBYTE6; IS EQUIVALENT TO STEPS 320 AND 340
360 NEXT L
370 B[J,K]=VAL(A$)
380 NEXT K
390 NEXT J
400 REM RESTORE CP-30 AS A PRINTER
410 WRITE (6,*)WBYTE256;
420 END
```

It requires about 5.5 minutes (with L=10) to generate the array B from the numbers of the array N stored in the buffer.

Once the storage capacity of the CP-30 is reached, no further material can be written in and any further write commands will cause the 9830 to "hang up". Control can be regained by STOP but the buffer will not take in more until WRITE(6,100)256; is received (which also restores the printer function) or some of the buffer contents are removed by ENTER(6,*). For this reason, it is important not to overrun the buffer capacity.

A final important capability of the CP-30 buffer is the way that it can act as a "Pointer Buffer". To understand how this works you will need to know how the buffer is organized and how the pointer works. The organization of the buffer can be represented as follows:



key lines in the foregoing program are 130-490 and if you look at se you will see that when ^ is encountered (i.e. L3#L2#0) in a ing entered from the CP-30 buffer, the line lengths are adjusted to as close to the desired number of characters per line as the spaces ween the words will permit. This process continues through lines not minated by ^ until either some line comes out even or a string minated by ^^ is encountered which denotes the end of a para- ph.

is important to know that program material sent to the CP-30 buffer be recovered as program rather than in the form of strings by the mand PTAPE6. This possibility permits the regeneration of REM text om text in the form of strings. The first step is to devise a FOR,NEXT op which will add to each string a line number and REM. The revised rings are then written into the buffer and finally read out as REM xt by execution of PTAPE6.

should not be thought that the CP-30's buffer is only useful for orage of text material. It can also be used for storage of numerical rays. Suppose for example that you wish to store an array N(100,10) the buffer in a F10.6 format. The numbers can be written into the ffer after execution of WRITE(6,100)27,38; by FOR,NEXT loops for the dices J and K and WRITE(6,110)N(J,K) where you have already entered 0 FORMAT F10.6. What is actually stored in the buffer is twelve bytes byte = 8 bits) and of these ten are the ASCII codes for the digits, cimal point, and sign of N(J,K) and two are the codes for line feed SCII Code 10) and carriage return (ASCII Code 13). These extra haracters are of course necessary for printing N(J,K) as a single umber on a line by itself. Clearly this method of storing numbers is omewhat inefficient because it takes ten ASCII coded characters to tore a F10.6 number which as an element of a full-precision array is

stored in the computer with the use of only eight bytes of memory space.

The storage of numbers in the CP-30 buffer can be made somewhat more efficient by writing in N(J,K) as WRITE(6,110)N(J,K); (note semicolon) where the appropriate format statement is 110 FORMAT 2F10.6. The 2F10. is important here because with F10.6, execution of WRITE(6,110)N(J,K); will result in both line-feed and carriage-return characters being fed into the buffer. With the 2F10.6 format statement, only the ten digits of N(J,K) will go to the buffer, but this must then be followed by WRITE(6,100)10; which sends out the line-feed character (ASCII Code 10 alone. The reason this is important is that when you read the material back from the buffer with appropriate J,K loops into array N(100,10) b ENTER(6,*)N(J,K), or by ENTER(6,110) N(J,K), the line feed-character i used by the 9830 to determine when the transmission of each item is complete. If no line-feed character is encountered the 9830 will "hang up".

The following brief program can be used to demonstrate the steps by which a numerical array can be stored in the buffer and then recovered

```
10 DIM N[100,10],A[100,10]
20 REM GENERATION OF ARRAY ELEMENTS
30 FOR J=1 TO 100
40 FOR K=1 TO 10
50 N[J,K]=J/K*SIN(K)
60 NEXT K
70 NEXT J
80 REM STORAGE OF ARRAY ELEMENTS IN CP-30 BUFFER
90 REM DISENGAGEMENT OF PRINT FUNCTION
100 FORMAT 80B
110 FORMAT 2F10.6
120 WRITE (6,100)27,38;
130 FOR J=1 TO 100
140 FOR K=1 TO 10
150 WRITE (6,110)N[J,K];
160 WRITE (6,100)10;
170 NEXT K
180 NEXT J
```

hen you execute the Pointer Buffer Control command, WRITE(6,100)27,25, OD(INT(P/256),256),MOD(P-INT(P/256)*256),256); the printing function of he CP-30 is disengaged and the pointer is set at location P of a articular byte in the buffer. Now if you read material into the buffer ither as individual ASCII codes, by a LIST, or by writing in strings, he pointer advances to successive locations from P as each byte is eceived. The number of locations that the pointer moves is not the same s the length of the string that you write in unless these are intended o be concatenated and if so you execute WRITE(6,100)A$;. The semicolon ere means that a subsequent input of WRITE(6,100)B$; produces a com- ined string and the pointer moves over by LEN(A$)+LEN(B$). On the other and, WRITE(6,100)A$ without a trailing semicolon produces in the buffer he characters of A$ in ASCII code terminated by the characters for line eed (ASCII Code 10) and carriage return (ASCII Code 13). At a later tage, when the buffer is read by ENTER(6,*)A$, the line feed and car- iage-return characters are used to signal the end of each A$ as it is ead out. As a result, WRITE(6,100)A$ moves the pointer by LEN(A$)+2. It s important to recognize that after you write into the buffer, the ointer is located where the next byte of information will be ccepted.

ou can locate the pointer by "Buffer Pointer Value" (Code 27,37). he sequence to do this is WRITE(6,100)27,37; followed by ENTER(6,*)P. he value of P is the location of the pointer. There is however an mportant precondition for determining P and that is to cancel the ointer Buffer Control by execution of WRITE(6,100)256;. The usual lternative to WRITE(6,100)256; which is WRITE (6,*)WBYTE256; must ot be used prior to WRITE(6,100)27,37; and ENTER(6,*)A because its

format results in a line feed and carriage return and when these are transmitted to the buffer, they can cause loss of items stored at or near pointer location = 1.

To read out material from the buffer when it has been written in using Pointer Buffer Control you have to set the pointer to the value of P which corresponds to where you want the readout to begin. If you initially set the pointer with P=1 and you want to read out all of the material you wrote in, then you have to first execute WRITE(6,100) 256; followed by WRITE(6,100)27,25,MOD(INT(P/256),256),MOD(P-INT(P/256) *256,256);. Then you can remove material from the buffer with ENTER (6,*)A or the like as described earlier. One advantage of Pointer Buffer Control, which is not shared by the FIFO mode, is that you can reread the buffer as many times as you would like by resetting the buffer pointer to the desired initial read point. Furthermore, if, at the end of a series of WRITE operations into the buffer, you WRITE(6,100)256; and then determine the position of the pointer, you can later reset the pointer to that location and write in additional material.

PART IV

SUMMARY OF CP-30 STATEMENTS

ble 1 (pages 3-4) contains a summary of CP-30 operations in response
different codes arranged in numerical order. The following summary is
ranged by function.

PRINT AREA AND PAGE SETUP

p of Form (27,84)

ecution of this statement or the result of powerup or reset is setting
e Top of Form at the current vertical position of the platen. Defines
e upper bound of the print area.

eft Margin (27,77)

ecution of this statement sets the Left Margin at the current position
the print head. The only way to move to the left of this margin with-
ut a reset, is by Backspace (described under PRINT OPERATIONS). The
efault Left Margin is the zero position of the print head.

ext Width (27,87,MOD(INT(X*120/64),256),MOD(INT(X*120),256))

he value of X in inches is the width of the print area from the left
argin. If, in the course of printing a line of text, the print head
eaches the right limit of the print area then a line feed and a
arriage return result and the printing of text will continue from the
eft margin. The default Text Width is 13.2 inches.

Text Length (27,76,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256))

The value of Y in inches is the length of the print area from the Top o Form. If a line feed would cause the print head to cross the lower boun a Form Feed (see the PRINT OPERATIONS SECTION) will occur. Other ways of moving the print head to the lower bound of the print area result in a beep from the CP-30 and a halt at that boundary. The default Text Length is 11 inches.

Form Length (27,25,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256))

The value of Y in inches determines the length of the form and whenever a line feed exceeds the Text Length (see above) the platen turns to a new position which is Y inches from the previous Top of Form and is itself a new Top of Form position. The default Form Length is 11 inches.

SETTING AND CLEARING OF TABS

Clear All Horizontal Tabs (27,51)

Clears all horizontal tabs without movement of the print head (with use of a trailing semicolon).

Clear All Vertical Tabs (27,55)

Clears all vertical tabs without movement of the print head or platen (with use of a trailing semicolon).

ear Horizontal Tab (27,50)

ears a horizontal tab if one has been previously set at the current sition of the print head. No motion of the print head follows execu- on of the command (with use of a trailing semicolon).

ear Vertical Tab (27,54)

ears a vertical tab if one has been previously set at the current sition of the platen. No movement of the print head or platen occurs ter the execution of the command (with use of a trailing semicolon).

rizontal Tab (27,49)

ts a horizontal tab at the current position of the print head. No vement of the print head or platen occurs (with use of a trailing micolon).

rtical Tab (27,53)

ets a vertical tab at the current position of the platen. No movement f the print head or platen occurs (with a trailing semicolon).

PRINT OPERATIONS

orizontal Spacing (27,72,INT(120/X/64),INT(120/X))

he value of X is the number of characters per inch that the TP-30 rints. Default is 10 per inch.

Vertical Spacing (27,86,INT(2*48/Y/64),INT(2*48/Y))

The value of Y is the vertical spacing in lines per inch. Default is 6 lines per inch.

Variable Spacing (27,83)

Execution of this command sets a mode for the CP-30 whereby each character has to be followed by a parameter that determines the space between the center of the current character and the next one to be printed. The distance parameter should be an integer and it represents the number of 1/120's of an inch between the centers of the characters. If the distance is small, characters may overlap seriously. Variable Spacing is only canceled by a reset and bidirectional printing is disabled when the CP-30 is in the Variable Spacing mode. The use of this mode is described in detail on pages 27-30.

Form Feed (12)

Execution of this command causes the platen to roll the paper up to the position which corresponds to the previously entered Form Length.

Line Feed (10)

This command causes the platen to turn up one line as specifed by Line Spacing without a carriage return.

Reverse Line Feed (27,10)

Moves the platen back down one line as specified by Line Spacing without a carriage return.

Carriage Return (13)

The Carriage Return command causes the print head to move to the left margin without a line feed.

top (256)

tops the printer immediately and makes the buffer appear to be cleared but, in fact, sets the buffer pointer back to the location of its first yte). Does not change the print parameters nor reset Top of Form.

ackspace (8)

auses the print head to backspace one character and can be used to go eyond the left margin. However, if Backspace takes the print head to he limit of its mechanical travel, the CP-30 will perform an automatic ESET and assume the default print parameters.

orizontal Tab Right (9)

xecution of this command takes the print head to the next horizontal ab to the right and, if there is no tab, to a stop at the right margin. f a WRITE command is then encountered, the CP-30's beeper will sound nd there will be a carriage return and a line feed.

orizontal Tab Left (27,52)

his command results in the print head moving to the next horizontal tab o the left and, if there is none, to the left margin.

ertical Tab Down (11)

auses the platen to turn the paper to the next vertical tab or, if one, to the bottom of the print area. Any line feed operation at the ottom of the operation will result in Form Feed.

Vertical Tab up (27,56)

Rolls the paper down to the next vertical tab or, if none, to the Top o Form.

Beep (Bell) (7)

Sounds the beeper in the CP-30.

Character Replacement (27,67,Q,N,Replacement List)
Execution of this command causes the replacement of any single charact Q, which can be entered either as its ASCII code or in quotes, by another character, by a string, by ASCII commands, or combinations of these. The length of the replacement list for Q is entered as N and the replacement list itself either in quotes or as ASCII codes separated by commas (see pages 30-33 for a more detailed discussion of Character Replacement).

Peek (27,68,MOD(INT(N/64),256),MOD(INT(N),256))

The value of N specifies the number of milliseconds after a print operation performed by the CP-30 that the print head will move to the right to permit you to view what it has just printed. To disable the Peek feature enter a negative value of N. RESET also disables Peek.

Reset (27,69)

Resets the CP-30 parameters to the default values which are listed on page 34.

gain (27,32)

xecution of this command causes the contents of the CP-30 buffer to be rinted without need for further processing from the 9830. Again starts rom the buffer-pointer position number 1 and there may be already be onsiderable material in the buffer you do not really want printed gain. If you plan to use the Again feature, this problem can be avoided y WRITE(6,100)256; before you start your first printout.

elf Test (27,32)

ests the CP-30's electronics and prints out all of the characters on he print wheel.

PLOT OPERATIONS

lot Origin (27,79,MOD(INT(X*120/64),256),MOD(INT(X*120),256),
OD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256))

he default position of the Plot Origin is at the lower left-hand corner f the print area. To move it to a new position within the print area, xecute the above command, wherein X and Y are in inches (not in user nits). See pages 36 and 46-47 for discussion of placement of the origin nd the conversion of user units to plotter units.

lot Absolute (27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256),
OD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256))

alues of X and Y in inches move the print head to the specifed location within the print area. If the X,Y point entered is outside the print area the print head will stop at the appropriate margin(s). Note that

positive X moves the print head to the right and positive Y moves the paper down. Negative X,Y values do the opposite. To print a specified character such as "x", "+", or the like at the X,Y position simply follow Plot Absolute with the character in quotes or in the form of its ASCII code.

Plot Relative (27,82,MOD(INT(X*120/64),256),MOD(INT(X*120),256),
MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256))

This command operates just as Plot Absolute except that X and Y specify the position of the new point in inches relative to the current point.

Character Fill (27,46,P1,INT(120/P2/64),INT(120/P2),INT(P3*2*48)

Execution of this command sets up the CP-30 so that in moving from one X,Y position to the next, it prints a specified character at specified intervals. The parameter P1 represents the fill character and it can be enetered either in quotes or as the ASCII equivalent. The default parameter of P1 is the decimal point (ASCII Code 46). P2 is the number of fill characters per inch provided the line between the new and current X,Y has a slope of less than 45 degrees with respect to the X axis or is the number of characters per 1.25 inch if the slope is greater than 45 degrees. P3 is a vertical offset parameter which determines whether or not the fill characters are to be raised (positive values) or lowered (negative values) with respect to the line connecting the new and current X,Y values. Character Fill is discussed in more detail on pages 38-46.

Plot Absolute with Character Fill (27,97,MOD(INT(X*120/64),256),
MOD(INT(X*120),256),MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256))

Execution of this command gives the same result as Plot Absolute except that there will be Character Fill along the line between the current and new X,Y coordinates. It is important to remember that X and Y are in inches, not user units.

Plot Relative with Character Fill (27,114,MOD(INT(X*120/64),256),
MOD(INT(X*120),256),MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256))

This statement is the same as Plot Absolute with Character Fill, except that X and Y are located with respect to the current position of the print head instead of at the origin.

USE OF THE CP-30 BUFFER FOR DATA STORAGE

First-In First-Out (27,38)

The First-In First-Out (FIFO) mode of using the buffer for data storage is activated by Code 27,38 and, while this command causes the printer function to be disengaged, the buffer will still receive WRITE commands and the material so received is stored for your future use. Character Replacements, if any have been activated, are performed by the buffer, but the only other commands that are recognized are WRITE(6,100)256; and ENTER(6,*), RBYTE6, and so on that withdraw data. The command WRITE(6, 100)256; restores the printer function of the CP-30 and "clears" the buffer. The result is that material written in earlier can only be recovered by trying to reset the buffer pointer (see below). However, this procedure may not be successful and, in general, you should stay in the FIFO mode until you have recovered what you want to recover from the buffer. More on FIFO operations is given on pages 48-59.

Pointer Buffer Control (27,25,MOD(INT(P/256),256),MOD(P-INT(P/256)*256,256))

Execution of this command, like the FIFO command above, disengages the print function of the CP-30 but still leaves the buffer able to accept WRITE commands. The value of P which you read in determines the first byte location in the buffer that will be utilized for data storage as the result of a WRITE command. WRITE(6,100)256; (do not use WRITE(6,*) WBYTE256;) restores the printer function. However, if print operations are then carried on, the stored material may be partly or completely lost. It is important to recognize that when the buffer is set up as a pointer buffer, and you write in information, then to try to recover it without resetting the pointer, the ENTER or RBYTE commands will try to remove information from the latest position of the pointer, which will be one byte beyond the end of what you just entered. Thus, you must first reset the pointer to the byte location that you want to remove data from. More on pointer-buffer operation is given on pages 59-61.

Buffer Pointer Value (27,37)

Execution of this command followed by ENTER(6,*) returns the current position of the buffer pointer as the value of A. The command will not execute properly when the CP-30 is under Pointer Buffer Control and if so the WRITE(6,100)256; command should be sent before attempting 27,37 and ENTER(6,*)A. See pages 59-61 for more details.

APPENDIX I

The following program steps are for the spiral plot with Character Fill shown in Figure 3 and made in accord with the discussion on pages 40-41 and 45-46.

```
0 REM RESET
0 WRITE (6,100)27,69;
0 REM SET LEFT MARGIN AT I INCH (TEN SPACES) AND TOP OF FORM
0 WRITE (6,100)"          ",27,77,27,84;
0 REM SET TEXT WIDTH OF 7 INCHES
0 X=7
0 WRITE (6,100)27,87,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
0 REM SET TEXT LENGTH OF 9 INCHES
0 Y=9
00 FORMAT 80B
10 WRITE (6,100)27,76,MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
20 REM SET ORIGIN AT CENTER OF PRINT AREA
30 X=3.5
40 Y=4.5
50 WRITE (6,100)27,79,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
60 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
70 DEG
80 A=0
90 B=0
00 REM SUPPRESSES SPACES AFTER PRINTING CHARACTER
10 WRITE (6,100)27,72,0,0;
20 FOR K=0 TO 1080 STEP 10
30 X=K/12+K*COS(K)
40 Y=K/12+K*SIN(K)
50 REM CONVERTS X USER UNITS TO INCHES
60 X=X/400
70 REM CONVERTS Y USER UNITS TO INCHES
80 Y=Y/400
90 REM PRINTS "o" (CODE 111) AT POINT OF ABSOLUTE PLOT OF X,Y
00 REM FIRST POINT TO BE PRINTED WITHOUT CHARACTER FILL
10 IF K=0 THEN 360
20 GOSUB 1200
30 WRITE (6,100)27,97,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
40 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),111;
50 GOTO 380
60 WRITE (6,100)27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
70 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),111;
80 NEXT K
90 REM RESTORES SPACING AFTER PRINTING OF CHARACTER
00 WRITE (6,100)27,72,INT(120/10/64),INT(120/10);
```

```
410 REM SET CHARACTER FILL, P1=45, P2=20, P3=0
420 P2=20
430 P3=0
440 WRITE (6,100)27,46,45,INT(120/P2/64),INT(120/P2),INT(P3*2*48);
450 REM MOVE TO LEFT END OF X AXIS, PRINT + (CODE 43)
460 REM SUPPRESSES SPACES AFTER PRINTING CHARACTER
470 WRITE (6,100)27,72,0,0;
480 X=-3
490 Y=0
500 WRITE (6,100)27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
510 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),43;
520 REM PRINT + AS TIC EVERY 1/2 INCH WITH HYPHENS AS CHARACTER FILL
530 X=0.5
540 FOR L=1 TO 12
550 REM PLOT RELATIVE WITH CHARACTER FILL (CODE 27,114)
560 WRITE (6,100)27,114,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
570 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),43;
580 NEXT L
590 REM RESTORES SPACING AFTER PRINTING OF CHARACTER
600 WRITE (6,100)27,72,INT(120/10/64),INT(120/10);
610 REM CARrIAGE RETURN, PLOT RELATIVE TO TURN PAPER UP 0.14 INCH
620 X=0
630 Y=-0.14
640 WRITE (6,100)13,27,82,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
650 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
660 REM PRINT X-AXIS LABELS
670 WRITE (6,100)"  -1200          -600              0          ";
680 WRITE (6,100)"  600           1200"
690 REM SET CHARACTER FILL, P1=124, P2=20, P3=0
700 P2=20
710 P3=0
720 WRITE (6,100)27,46,124,INT(120/P2/64),INT(120/P2),INT(P3*2*48)
730 REM MOVE TO TOP END OF Y AXIS, PRINT + (CODE 43)
740 REM SUPPRESSES SPACES AFTER PRINTING CHARACTER
750 WRITE (6,100)27,72,0,0;
760 X=0
770 Y=3
780 WRITE (6,100)27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
790 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),43;
800 REM PRINT + EVERY INCH WITH CODE 124 AS CHARACTER FILL
810 X=0
820 Y=-0.5
830 FOR L=1 TO 12
840 REM PLOT RELATIVE WITH CHARACTER FILL
850 WRITE (6,100)27,114,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
860 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256),43;
870 NEXT L
880 REM RESTORES SPACING AFTER PRINTING OF CHARACTER
890 WRITE (6,100)27,72,INT(120/10/64),INT(120/10);
900 REM CLEAR ALL HORIZONTAL AND VERTICAL TABS
```

```
10 WRITE (6,100)27,51,27,55;
20 REM SET HORIZONTAL TAB FIVE SPACES TO RIGHT OF Y AXIS
30 WRITE (6,100)8,8,8,8,8,27,49;
40 REM GO TO LOWER END OF Y AXIS
50 X=0.5
60 Y=-3.5
70 WRITE (6,100)27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
80 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
90 REM SET VERTICAL TABS 1.5 INCH APART
000 FOR Y=-3 TO 3 STEP 1.5
010 REM SET VERTICAL TAB (CODE 27,53) AND PLOT ABSOLUTE (CODE 27,65)
020 WRITE (6,100)27,53,27,65,MOD(INT(X*120/64),256),MOD(INT(X*120),256);
030 WRITE (6,100)MOD(INT(Y*2*48/64),256),MOD(INT(Y*2*48),256);
040 NEXT Y
050 FOR L=1200 TO -1200 STEP -600
060 IF L=0 THEN 1110
070 REM HORIZONTAL TAB LEFT
080 WRITE (6,100)27,52;
090 WRITE (6,1100)L;
100 FORMAT F5.0
110 REM VERTICAL TAB DOWN
120 WRITE (6,100)11;
130 NEXT L
140 REM FORM FEED
150 WRITE (6,100)12;
160 END
200 IF ABS((Y-B)/(X-A))>1 THEN 1230
210 P2=20
220 GOTO 1240
230 P2=25
240 P3=0.032
250 REM CHARACTER FILL SETUP P1=46, P2=20 <45 DEG OR P2=25 >45 DEG, P3=0.03
260 WRITE (6,100)27,46,46,INT(120/P2/64),INT(120/P2),INT(P3*2*48);
270 A=X
280 B=Y
290 RETURN
300 END
```

INDEX

APPENDIX II

CP-30 OPTION 25

The following program is used to determine the actual buffer size (lines 15 through 37) by writing binary 64 into the buffer until the buffer status goes to 290 (buffer full). Lines 38 through 76 exercise the buffer by first writing to the buffer (lines 45 through 51) and then reading back the previously written data (lines 53-74) which is checked against the expected data. A brief demonstration of the printer is contained in the remainder of the program. Lines 81 through 97 shows how print spacing can be changed. Horizontal and vertical tabs are shown in lines 97 through 158. Lines 160 through 208 demonstrate the plotting features of the CP-30. This section of the program follows part II of this manual (as well as appendix I) in plotting equation set X=K/12+K*COS(K), Y=K/12+K*SIN(K). All characters available are printed in lines 216 through 230.

ADDENDUM TO CP-30 MANUAL

Page 15, LINE 400:

```
400  WRITE(6,100) 27, 67...
```

CHANGE TO:

```
400  WRITE(6,100) 27,76...
```

Page 68, change SELF TEST(27,32) to :

SELF TEST(27,122)

Page 14, change in first line of second paragraph:

INFOTEK OPT. 140

TO

INFOTEK OPT. 214

Page 33, add to last paragraph

Self test causes the printing of each character, double spaced, across the width of the paper. After the characters are printed, the CP-30 internal ROMs are tested, then the CP-30 buffer RAM is tested. Failure of an internal ROM is indicated by a single beep following the character print line. Failure of the CP-30 buffer RAM is indicated by multiple beeps. The memory check portion of SELF TEST is initiated automatically upon power up or RESET.

Page 4, change "27,256 stop, reset pointer "to"; 256 stop, reset pointer" "where SC is the CP-30 select code"

Add Appendix II, Option 25

```
0: "CP-30 buffer and printer exerciser tests":
1: wtb S,7
2: dim H$[10]
3: ent "Buffer test (Y or N) ?",H$[1,1]
4: ent "Character print test (Y or N)",H$[2,2]
5: ent "Plot test (Y or N) ?",H$[3,3]
6: ent "Select code test (Y or N)",H$[4,4]
7: if H$[4,4]="N";gto 12
8: for S=2 to 14
9: dsp "Set select switch to",S;stp
10: wtb S,7;wait 200;wtb S,7
11: next S
12: ent "CP-30 select code ?",S
13: if H$[1,1]="N";gto 76
14: "determine the actual capacity of the buffer":
15: wtc S,32
16: fmt 1,2b,z
17: fmt f10.0,z
18: dsp "Take CP-30 off line"
19: wait 500
20: dsp "Reset CP-30"
21: wait 500
22: ent "Number of passes ?",P
23: wtb S,27,38
24: 0}I
25: I+1}I
26: dsp "buffer size ",I
27: wait 1
28: wtb S,64
29: rds(S)}A
30: A-290}B
31: if B=1;gto 25
32: wtc S,32
33: dsp "Buffer size",I;wait 1000
34: if I<1000;500}Z;gto 37
35: if I<3000;50}Z;gto 37
36: 10}Z
37: dsp "Put CP-30 on line";stp
38: fmt 1,b,z
39: fmt f10.0,z
40: wtc S,32
41: wtb S,7
42: wait 1000
43: wtb S,27,38
44: wait 2000
```

```
45: for K=1 to I/10-5
46: dsp "store # ",K,"Pass ",N
47: wait 1
48: wrt S,K
49: wtb S,10
50: next K
51: dsp "Buffer input complete"
52: wait 500
53: for K=1 to I/10-5
54: red S,A
55: red S,B
56: dsp "Read",A,"Pass ",N
57: if K=A;gto 64
58: dsp A,K
59: wait 1000
60: Y+1}Y
61: if Y<100;gto 64
62: dsp "terminated"
63: stp
64: next K
65: if N=P;gto 70
66: if P<10;gto 70
67: G+1}G
68: if G<Z;gto 72
69: 0}G
70: prt "Pass ",N,"Buffer ",I
71: wtb S,7
72: N+1}N
73: if N<P;gto 38
74: wtc S,32;wait 1000
75: wtb S,27,69;wait 5000
76: if H$[3,3]="N";gto 209
77: "start CP-30 printer demonstration":
78: " ":
79: "Reset printer":wtb S,27,69;wait 1000
80: wrt S,"The printing spacing of the CP-30 can be easily changed from"
81: wrt S,"the normal 6 characters per inch and 10 lines per inch"
82: for N=1 to 5
83: wrt S,"AAAAAAAAAAAAAAAAAAAAAAAAA"
84: next N
85: "line feed":wtb S,10
86: "line feed":wtb S,10
87: wrt S,"to 8 lines per inch and 12 characters per inch"
88: "set horizontal spacing":wtb S,27,72,int(120/12/64),120/12
89: "set vertical spacing":wtb S,27,86,int(2*48/8/64),2*48/8
90: for N=1 to 5
91: wrt S,"AAAAAAAAAAAAAAAAAAAAAAAAA"
92: next N
93: wrt S
94: wrt S
95: "reset character spacing":wtb S,27,72,int(120/10/64),120/10
96: "reset vertical spacing":wtb S,27,86,int(96/6/64),96/6
97: wrt S,"The CP-30 also has horizontal and vertical tabs which"
98: wrt S,"directly in inches [from the top of form position and left margi
99: wrt S,"respectively"
```

```
: "set top of form at current paper position":wtb S,27,84
: "set horizontal tabs at 1,2,4, and 8 inches from the left margin":
: "move carriage 1 inch":1}X;0}Y;gsb "mod"
: wtb S,27,82,A,B,C,D
: "set tab":wtb S,27,49
: "move carriage another inch":1}X;0}Y;gsb "mod"
: wtb S,27,82,A,B,C,D
: "set tab here":wtb S,27,49
: "move carriage another 2 inches":2}X;0}Y;gsb "mod"
: wtb S,27,82,A,B,C,D
: "set tab here":wtb S,27,49
: "move carriage another 4 inches":4}X;0}Y;gsb "mod"
: wtb S,27,82,A,B,C,D
: "set tab at current carriage location":wtb S,27,49
: wrt S
: fmt c8,z
: "tab 1":wtb S,9
: wrt S,"1 inch"
: "tab 2":wtb S,9
: wrt S,"2 inches"
: "tab 3":wtb S,9
: wrt S,"4 inches"
: "tab 4":wtb S,9
: wrt S,"8 inches"
: "clear horizontal tabs":wtb S,27,51
: "carriage return":wtb S,13,10
: fmt 80c

: wrt S,"In addition, the vertical tabs of the CP-30 can also be set"
: wrt S,"under program control"
: wtb S,10
: "set vertical tabs at 1,2,4, and 8 inches from the present location":
: "define top of form":wtb S,27,84
: "roll platen down 1 inch using plot relative":0}X;-1}Y;gsb "mod"
: wtb S,27,82,A,B,C,D
: "set vertical tab 1":wtb S,27,53
: "roll platen down another 2 inches":0}X;-2}Y;gsb "mod"
: wtb S,27,82,A,B,C,D
: "set vertical tab 2":wtb S,27,53
: "roll platen down another 5 inches":0}X;-5}Y;gsb "mod"
: wtb S,27,82,A,B,C,D
: "set vertical tab 3":wtb S,27,53
: "use plot absolute to get to top of form":0}X;11}Y;gsb "mod"
: wtb S,27,65,A,B,C,D
: "tab down 1":wtb S,11;wrt S,"1 inch"
: "tab down to 3 inches":wtb S,11;wrt S,"3 inches"
: "tab down another 5 inches":wtb S,11;wrt S,"8 inches"
: "get back to the top again":
: wtb S,27,65,A,B,C,D
: "top of form":
: "start graph":
: "set left margin and top of form":wtb S,32,32,32,32,32,32,32,32,32,32
: wtb S,27,77,27,84
: "set text width to 7 inches":7}X;gsb "mod"
: wtb S,27,76,A,B
: "set text width to 9 inches":9}Y;gsb "mod"
: wtb S,27,76,C,D
: "set plot origin at center of area":3.5}X;4.5}Y;gsb "mod"
: wtb S,27,79,A,B,C,D
: "suppress spaces after printing ":wtb S,27,72,0,0
```

```
159: 0}E;0}F
160: for K=0 to 1080 by 10
161: K/12+K*cos(K)}X;K/12+K*sin(K)}Y;X/400}X;Y/400}Y
162: "plot absolute with character fill first point -no fill":gsb "mod"
163: if K=0;gto 166
164: gsb "abs"
165: wtb S,27,97,A,B,C,D,111;gto 167
166: wtb S,27,65,A,B,C,D
167: next K
168: "restore spacing after printing":wtb S,27,72,int(120/10/64),int(120/1
169: "set up character fill":45}P;20}Q;0}R
170: wtb S,27,46,P,int(120/Q/64),int(120/Q),int(R*96)
171: "move to left end of X-axis, print +, and suppress space after print"
172: wtb S,27,72,0,0
173: -3}X;0}Y;gsb "mod"
174: wtb S,27,65,A,B,C,D,43
175: .5}X;gsb "mod"
176: for L=1 to 12
177: "plot relative with character fill":wtb S,27,114,A,B,C,D,43
178: next L
179: "restore spacing":wtb S,27,72,int(120/10/64),int(120/10)
180: "cariage return and move platen up .14 inch":0}X;-.14}Y;gsb "mod"
181: wtb S,13,27,82,A,B,C,D
182: "label X-axis":
183: fmt 2x,"-1200",10x,"-600",14x,"0",12x,"600",10x,"1200"
184: wrt S
185: "set up character fill":124}P;20}Q;0}R
186: wtb S,27,46,P,int(120/Q/64),int(120/Q),int(R*96)
187: "suppress spacing after printing":wtb S,27,72,0,0
188: 0}X;3}Y;gsb "mod"
189: wtb S,27,65,A,B,C,D,43
190: "print Y-axis with plot relative and character fill":0}X;-.5}Y;gsb "m
191: for L=1 to 12

192: wtb S,27,114,A,B,C,D,43
193: next L
194: "restore spacing":wtb S,27,72,int(120/10/64),int(120/10)
195: "clear all tabs":wtb S,27,51,27,55
196: "set horizontal tab five spaces from X-axis":wtb S,8,8,8,8,8,27,49
197: "go to lower end of Y-axis":.5}X;-3.5}Y;gsb "mod"
198: wtb S,27,65,A,B,C,D
199: "set vertical tabs 1.5 inches apart with plot absolute":
200: for Y=-3 to 3 by 1.5;gsb "mod"
201: wtb S,27,53,27,65,A,B,C,D
202: next Y
203: fmt f5.0,z
204: for L=1200 to -1200 by -600
205: if L=0;gto 207
206: "tab left":wtb S,27,52;wrt S,L
207: "tab down":wtb S,11
208: next L
209: if H$[2,2]="N";gto 225
210: "character print test":
211: for M=1 to 5
212: wtb S,10
213: next M
```

```
0: "CP-30 buffer and printer exerciser tests":
1: wtb S,7
2: dim H$[10]
3: ent "Buffer test (Y or N) ?",H$[1,1]
4: ent "Character print test (Y or N)",H$[2,2]
5: ent "Plot test (Y or N) ?",H$[3,3]
6: ent "Select code test (Y or N)",H$[4,4]
7: if H$[4,4]="N";gto 12
8: for S=2 to 14
9: dsp "Set select switch to",S;stp
10: wtb S,7;wait 200;wtb S,7
11: next S
12: ent "CP-30 select code ?",S
13: if H$[1,1]="N";gto 76
14: "determine the actual capacity of the buffer":
15: wtc S,32
16: fmt 1,2b,z
17: fmt f10.0,z
18: dsp "Take CP-30 off line"
19: wait 500
20: dsp "Reset CP-30"
21: wait 500
22: ent "Number of passes ?",P
23: wtb S,27,38
24: 0}I
25: I+1}I
26: dsp "buffer size ",I
27: wait 1
28: wtb S,64
29: rds(S)}A
30: A-290}B
31: if B=1;gto 25
32: wtc S,32
33: dsp "Buffer size",I;wait 1000
34: if I<1000;500}Z;gto 37
35: if I<3000;50}Z;gto 37
36: 10}Z
37: dsp "Put CP-30 on line";stp
38: fmt 1,b,z
39: fmt f10.0,z
40: wtc S,32
41: wtb S,7
42: wait 1000
43: wtb S,27,38
44: wait 2000
```

```
214: "print all characters across the page":fmt 60b,z,60b
215: for N=32 to 126 by 2
216: for M=1 to 65
217: wtb S,N
218: next M
219: for M=1 to 65
220: wtb S,N+1
221: next M
222: wtb S,13,10
223: next N
224: wtb S,10,10,13
225: wrt S,"THE END";wtb S,10,10,13;wtb S,27,69
226: dsp "That's all folks"
227: stp
228: ".....":
229: "subroutine converts MOD expression of Infotek FB-III ROM":
230: "mod":
231: int(X*120/64-int(X*120/64/256)*256)}A
232: int(X*120-int(X*120/256)*256)}B
233: int(Y*2*48/64-int(Y*2*48/64/256)*256)}C
234: int(Y*2*48-int(Y*2*48/256)*256)}D
235: ".....":
236: "character fill adjust subroutine":
237: "abs":
238: if X-F=0;F+1}F
239: if abs((Y-E)/(X-F))>1;gto 241
240: 20}Q;gto 242
241: 25}Q;.032}R
242: "character fill set up":
243: wtb S,27,46,46,int(120/Q/64),int(120/Q),int(R*96)
244: X}F;Y}E
245: ret
*27193
```

```
0: "set top of form at current paper position":wtb S,27,84
1: "set horizontal tabs at 1,2,4, and 8 inches from the left margin":
2: "move carriage 1 inch":1}X;0}Y;gsb "mod"
3: wtb S,27,82,A,B,C,D
4: "set tab":wtb S,27,49
5: "move carriage another inch":1}X;0}Y;gsb "mod"
6: wtb S,27,82,A,B,C,D
7: "set tab here":wtb S,27,49
8: "move carriage another 2 inches":2}X;0}Y;gsb "mod"
9: wtb S,27,82,A,B,C,D
0: "set tab here":wtb S,27,49
1: "move carriage another 4 inches":4}X;0}Y;gsb "mod"
2: wtb S,27,82,A,B,C,D
3: "set tab at current carriage location":wtb S,27,49
4: wrt S
5: fmt c8,z
6: "tab 1":wtb S,9
7: wrt S,"1 inch"
8: "tab 2":wtb S,9
9: wrt S,"2 inches"
0: "tab 3":wtb S,9
1: wrt S,"4 inches"
2: "tab 4":wtb S,9
3: wrt S,"8 inches"
4: "clear horizontal tabs":wtb S,27,51
5: "carriage return":wtb S,13,10
6: fmt 80c

7: wrt S,"In addition, the vertical tabs of the CP-30 can also be set"
8: wrt S,"under program control"
9: wtb S,10
0: "set vertical tabs at 1,2,4, and 8 inches from the present location":
1: "define top of form":wtb S,27,84
2: "roll platen down 1 inch using plot relative":0}X;-1}Y;gsb "mod"
3: wtb S,27,82,A,B,C,D
4: "set vertical tab 1":wtb S,27,53
5: "roll platen down another 2 inches":0}X;-2}Y;gsb "mod"
6: wtb S,27,82,A,B,C,D
7: "set vertical tab 2":wtb S,27,53
8: "roll platen down another 5 inches":0}X;-5}Y;gsb "mod"
9: wtb S,27,82,A,B,C,D
0: "set vertical tab 3":wtb S,27,53
1: "use plot absolute to get to top of form":0}X;11}Y;gsb "mod"
2: wtb S,27,65,A,B,C,D
3: "tab down 1":wtb S,11;wrt S,"1 inch"
4: "tab down to 3 inches":wtb S,11;wrt S,"3 inches"
5: "tab down another 5 inches":wtb S,11;wrt S,"8 inches"
6: "get back to the top again":
7: wtb S,27,65,A,B,C,D
8: "top of form":
9: "start graph":
0: "set left margin and top of form":wtb S,32,32,32,32,32,32,32,32,32,32
1: wtb S,27,77,27,84
2: "set text width to 7 inches":7}X;gsb "mod"
3: wtb S,27,76,A,B
4: "set text width to 9 inches":9}Y;gsb "mod"
5: wtb S,27,76,C,D
6: "set plot origin at center of area":3.5}X;4.5}Y;gsb "mod"
7: wtb S,27,79,A,B,C,D
8: "suppress spaces after printing ":wtb S,27,72,0,0
```

```
45: for K=1 to I/10-5
46: dsp "store # ",K,"Pass ",N
47: wait 1
48: wrt S,K
49: wtb S,10
50: next K
51: dsp "Buffer input complete"
52: wait 500
53: for K=1 to I/10-5
54: red S,A
55: red S,B
56: dsp "Read",A,"Pass ",N
57: if K=A;gto 64
58: dsp A,K
59: wait 1000
60: Y+1}Y
61: if Y<100;gto 64
62: dsp "terminated"
63: stp
64: next K
65: if N=P;gto 70
66: if P<10;gto 70
67: G+1}G
68: if G<Z;gto 72
69: 0}G
70: prt "Pass ",N,"Buffer ",I
71: wtb S,7
72: N+1}N
73: if N<P;gto 38
74: wtc S,32;wait 1000
75: wtb S,27,69;wait 5000
76: if H$[3,3]="N";gto 209
77: "start CP-30 printer demonstration":
78: " ":
79: "Reset printer":wtb S,27,69;wait 1000
80: wrt S,"The printing spacing of the CP-30 can be easily changed from"
81: wrt S,"the normal 6 characters per inch and 10 lines per inch"
82: for N=1 to 5
83: wrt S,"AAAAAAAAAAAAAAAAAAAAAAAAA"
84: next N
85: "line feed":wtb S,10
86: "line feed":wtb S,10
87: wrt S,"to 8 lines per inch and 12 characters per inch"
88: "set horizontal spacing":wtb S,27,72,int(120/12/64),120/12
89: "set vertical spacing":wtb S,27,86,int(2*48/8/64),2*48/8
90: for N=1 to 5
91: wrt S,"AAAAAAAAAAAAAAAAAAAAAAAAA"
92: next N
93: wrt S
94: wrt S
95: "reset character spacing":wtb S,27,72,int(120/10/64),120/10
96: "reset vertical spacing":wtb S,27,86,int(96/6/64),96/6
97: wrt S,"The CP-30 also has horizontal and vertical tabs which"
98: wrt S,"directly in inches [from the top of form position and left mar
99: wrt S,"respectively"
```

```
9: 0}E;0}F
0: for K=0 to 1080 by 10
1: K/12+K*cos(K)}X;K/12+K*sin(K)}Y;X/400}X;Y/400}Y
2: "plot absolute with character fill first point -no fill":gsb "mod"
3: if K=0;gto 166
4: gsb "abs"
5: wtb S,27,97,A,B,C,D,111;gto 167
6: wtb S,27,65,A,B,C,D
7: next K
8: "restore spacing after printing":wtb S,27,72,int(120/10/64),int(120/10)
9: "set up character fill":45}P;20}Q;0}R
0: wtb S,27,46,P,int(120/Q/64),int(120/Q),int(R*96)
1: "move to left end of X-axis, print +, and suppress space after print":
2: wtb S,27,72,0,0
3: -3}X;0}Y;gsb "mod"
4: wtb S,27,65,A,B,C,D,43
5: .5}X;gsb "mod"
6: for L=1 to 12
7: "plot relative with character fill":wtb S,27,114,A,B,C,D,43
8: next L
9: "restore spacing":wtb S,27,72,int(120/10/64),int(120/10)
0: "cariage return and move platen up .14 inch":0}X;-.14}Y;gsb "mod"
1: wtb S,13,27,82,A,B,C,D
2: "label X-axis":
3: fmt 2x,"-1200",10x,"-600",14x,"0",12x,"600",10x,"1200"
4: wrt S
5: "set up character fill":124}P;20}Q;0}R
6: wtb S,27,46,P,int(120/Q/64),int(120/Q),int(R*96)
7: "suppress spacing after printing":wtb S,27,72,0,0
8: 0}X;3}Y;gsb "mod"
9: wtb S,27,65,A,B,C,D,43
0: "print Y-axis with plot relative and character fill":0}X;-.5}Y;gsb "mod"
1: for L=1 to 12

2: wtb S,27,114,A,B,C,D,43
3: next L
4: "restore spacing":wtb S,27,72,int(120/10/64),int(120/10)
5: "clear all tabs":wtb S,27,51,27,55
6: "set horizontal tab five spaces from X-axis":wtb S,8,8,8,8,8,27,49
7: "go to lower end of Y-axis":.5}X;-3.5}Y;gsb "mod"
8: wtb S,27,65,A,B,C,D
9: "set vertical tabs 1.5 inches apart with plot absolute":
0: for Y=-3 to 3 by 1.5;gsb "mod"
1: wtb S,27,53,27,65,A,B,C,D
2: next Y
3: fmt f5.0,z
4: for L=1200 to -1200 by -600
5: if L=0;gto 207
6: "tab left":wtb S,27,52;wrt S,L
7: "tab down":wtb S,11
8: next L
9: if H$[2,2]="N";gto 225
0: "character print test":
1: for M=1 to 5
2: wtb S,10
3: next M
```

```
214: "print all characters across the page":fmt 60b,z,60b
215: for N=32 to 126 by 2
216: for M=1 to 65
217: wtb S,N
218: next M
219: for M=1 to 65
220: wtb S,N+1
221: next M
222: wtb S,13,10
223: next N
224: wtb S,10,10,13
225: wrt S,"THE END";wtb S,10,10,13;wtb S,27,69
226: dsp "That's all folks"
227: stp
228: ".....":
229: "subroutine converts MOD expression of Infotek FB-III ROM":
230: "mod":
231: int(X*120/64-int(X*120/64/256)*256)}A
232: int(X*120-int(X*120/256)*256)}B
233: int(Y*2*48/64-int(Y*2*48/64/256)*256)}C
234: int(Y*2*48-int(Y*2*48/256)*256)}D
235: ".....":
236: "character fill adjust subroutine":
237: "abs":
238: if X-F=0;F+1}F
239: if abs((Y-E)/(X-F))>1;gto 241
240: 20}Q;gto 242
241: 25}Q;.032}R
242: "character fill set up":
243: wtb S,27,46,46,int(120/Q/64),int(120/Q),int(R*96)
244: X}F;Y}E
245: ret
*27193
```