```
10     ! *****************************************************
20     ! *                                                   *
30     ! *  9111A-9845B Graphics Tablet System Tools Tape 2  *
40     ! *  P/N:  09111 -  10005
50     ! *  REV A                                            *
60     ! *  August 1, 1980                                   *
70     ! *                                                   *
80     ! *  Programs on the tape:                            *
90     ! *            GPLOT:  Plots the menu needed for the  *
100    ! *                    EDITOR program and allows you  *
110    ! *                    to link up your menu elements. *
120    ! *            EDITOR: Lets you place and manipulate  *
130    ! *                    elements from the nenu.        *
140    ! *            AND:    Example element.               *
150    ! *            NAND:   Example element.               *
160    ! *            OR:     Example element.               *
170    ! *            NOR:    Example element.               *
180    ! *            CIRCUT: Example menu of elements.      *
190    ! *            ALIGN:  Allows you to align your       *
200    ! *                    document to the platen.        *
210    ! *                                                   *
220    ! *****************************************************
```

```
10     ! *****************************************************
20     ! *                                                   *
30     ! *  9111A - 9845B Graphics Tablet Systen Tools Tape  *
40     ! *  P/N:   09111 - 10004                              *
50     ! *  REV A                                            *
60     ! *  August 1, 1980                                   *
70     ! *                                                   *
80     ! *  Programs on the tape:                            *
90     ! *            PLOT:  Plots the menu needed for the   *
100    ! *                   drawing program.                *
110    ! *            DRAW:  Allows you to create a drawing  *
120    ! *                   or object out of lines, arcs,   *
130    ! *                   circles, rectangles, and labels.*
140    ! *            MENU:  Creates a menu data base for    *
150    ! *                   your own designed menu.         *
160    ! *            DRIV:  Program driver for your own     *
170    ! *                   designed menu.                  *
180    ! *            Binary:  Lets you do quick printing    *
190    ! *                   and erasing in graphics mode.   *
200    ! *            Cirarc:  Lets you draw fast circles    *
210    ! *                   and arcs on the graphics screen.*
220    ! *            DB1:  Example menu data base.          *
230    ! *                                                   *
240    ! *****************************************************
```

9111A-9845B
System Tutorial

Does This Thing Really Work?
Interactive User Self Test

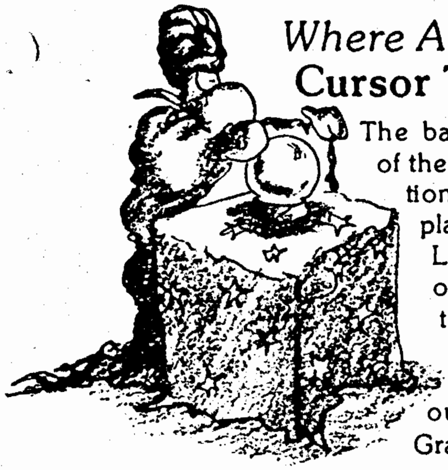
First, place the stylus (the ball-point-pen-shaped-thing on the end of the cable) in the Stylus Groove.

Now push the Switch labeled SELF TEST on the rear of the machine on, and immediately back off. This initiates the self test. The Graphics Tablet runs through the same test that is run when power is applied. When the tone finishes, the Graphics Tablet is waiting for you to digitize the Self Test Dot. To do this , press the tip of the stylus aginst the dot in the lower right corner of the platen of the Graphics Tablet. The Graphics Tablet then plays the same ascending sequence of tones that it does when it passes the first section of the self test. The proper operation of the Stylus and the internal circuitry in the Graphics Tablet has now been tested.

NOTE

If the Graphics Tablet does not perform the Self Test as described above, consult the "Errors and What They Mean" section of the 9111A User's Manual.

Now that the Graphics Tablet is working properly, lets look at what to do with it.



Where Am I Now?
Cursor Tracking

The ball-point pen device on the end of the cable is called a Stylus. The location of the tip of the stylus on the platen is referred to as the Cursor Location. This is Digitizer terminology which has been carried over to the Graphics Tablet so that people with experience in digitizer programming can apply their previous experience more easily to Graphics Tablet programming.

There are several ways to track the cursor using the HPGL langauge implemented on the 9111A. However, the simplest Graphics Tablet programs are written using the graphics statements resident in the 9845B Graphics ROM.

The basic operations involved in tracking the cursor may be broken down into two sections:

1. Set Up
2. Tracking Loop.

Elaborating on these gives us:

1. Set Up
   A) Set up Graphics Tablet
   B) Set up Display
2. Tracking Loop (repeated as long as necessary)
   A) Read Cursor Location
   B) Display Cursor Location

Using the Graphics ROM statements included in the 9845B Extended BASIC greatly simplifies implementing these procedures. If you go into the EDITLINE mode and type in each program line as it is described in the text, you will have a simple cursor tracking program when you finish this section.

First, set up the Graphics Tablet. Simply declare it to be a 9872A Plotter (the 9845B Graphics ROM was written before the 9111A existed, but the 9111A speaks essentially the same language as the 9872). Include the select code of the 9111A to allow specifying whether the 9845 is supposed to deal with the 9111A or the internal graphics display. Then select millimetre scaling with a 10 mm offset on each axis (this is because the P1 scaling point on the 9111 is ten mm from the lower left hand corner of the artwork).

```
10 Setup:    !
20        PLOTTER IS 7,6,"9872A"
30           MSCALE -10,-10
```

Second, we set up the graphics display on the 9845B to correspond to the dimensions of the 9111A artwork in millimetres. This is done using a SHOW statement to set up the scaling on the display. Finally, a frame is drawn on the CRT to represent the active digitizing area on the platen.

```
40        PLOTTER IS 13,"GRAPHICS"
50           SHOW 0,301,0,237
60           CLIP 0,301,0,218
70           FRAME
```

This completes the set up of the Graphics Tablet and the display. Now the program needs to turn on the display.

```
80        GRAPHICS
```

Now let's get on to the tracking loop. It starts with a label.

```
90 Begin_loop:!
```

Then read the cursor location.

```
100       PLOTTER 7,6 IS ON
110          CURSOR X,Y
```

Now we will move the pointer to the position on the display that best represents the location of the stylus on the platen.

```
120      PLOTTER 13 IS ON
130         POINTER X,Y,2
```

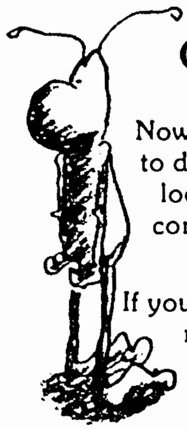
Notice that turning one plotter on turns the other plotter off. Having completed the update procedure, go back and start the loop over again.

```
140 GOTO Begin_loop
150 END
```

This completes the cursor tracking program. Hit RUN, and move the stylus around on the platen. A small blinking cross on the CRT should track your motion. Easy, isn't it?

Here is the whole program.

```
10 Setup:    !
20       PLOTTER IS 7,6,"9872A"
30          MSCALE -10,-10
40       PLOTTER IS 13,"GRAPHICS"
50          SHOW 0,301,0,237
60          CLIP 0,301,0,218
70          FRAME
80       GRAPHICS
90 Begin_loop:!
100      PLOTTER 7,6 IS ON
110         CURSOR X,Y
120      PLOTTER 13 IS ON
130         POINTER X,Y,2
140 GOTO Begin_loop
150 END
```



Where Have I Been?
Cursor Driven Plotting.

Now that you are tracking the cursor, it might be nice to draw some lines on the display. The first method to look at is the simplest. It illustrates some of the concepts involved in the Graphics Tablet System.

If you modify one line in the previous program, you can read in the **pen string**. This is a string that contains a wealth of data, but right now just look at the first character in it. This character is a "1" if the pen in the stylus is pressed down, and a "0" if the pen is not pressed down. The pen parameter for the 9845B PLOT statement is 1 for draw and 0 for move without drawing. Therefore, if you convert the string value to a numeric, you can plug it right into the PLOT statement.

Modify line 110 to read as follows:

```
   110 CURSOR X,Y,P$
```

Now insert the following line before line 140:

```
   131 PLOT X,Y,VAL(P$)
```

(I hear a comment "You said just look at the first character!" No need to worry. The VAL function reads characters for conversion until it encounters a non-numeric character. The second character in P$ is a comma. So only the first character is converted. Neat trick, isn't it. And it's the fastest method of accessing a substring for conversion.)

Now press RUN, and the cursor tracking should be operating normally. In addition, you should be able to 'draw' on the CRT by pressing down on the stylus tip like you would draw with a pen.

And you thought this would be hard!

Here is the whole program, (after renumbering it):

```
10 Setup:    !
20       PLOTTER IS 7,6,"9872A"
30          MSCALE -10,-10
40       PLOTTER IS 13,"GRAPHICS"
50          SHOW 0,301,0,237
60          CLIP 0,301,0,218
70          FRAME
80       GRAPHICS
90 Begin_loop:!
100      PLOTTER 7,6 IS ON
110         CURSOR X,Y,P$
120      PLOTTER 13 IS ON
130         POINTER X,Y,2
140         PLOT X,Y,VAL(P$)
150 GOTO Begin_loop
160 END
```

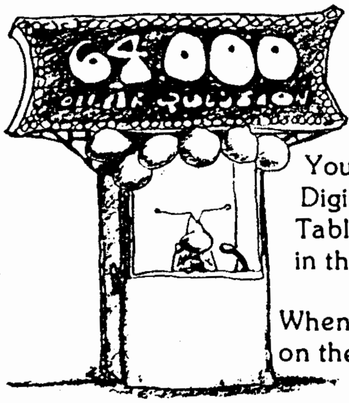
There are two problems with the program we just developed.

1. The image exists only on the display.
2. You're not really digitizing.

Let's go into a little more detail on both those concepts.

First — all you have so far are some lines on the CRT. The data is not in machine readable form. This means you can't do anything but look at it. The real power is gained by converting the data into a machine readable form. This enables you to use the computer to analyze and manipulate the data.

Second — you are looking at the Physical Pen. This means you are trying to do timing operations far removed from the activities you are timing. The Graphics Tablet contains some very sophisticated timing and analysis algorithms to handle exactly these operations. The following section shows you how to use it.



Where Am I Really At?
Digitizing

You are going to look at two concepts now — Digitizing, which takes place in the Graphics Tablet, and a simple Data Base, which exists in the 9845B. First, a look at digitizing.

When you press down on the tip of the stylus on the 9111A, a switch in the stylus is closed. Assuming that the 9111A is in a suitable mode, it digitizes when the switch closes. This means it measures exactly where the stylus is when the pen closes. It then takes the location, some mode information, and various other information, and combines them into a position and pen status bit in the Digitize register. This register can be read using a DIGITIZE command on the 9845B. The question is, when to execute a DIGITIZE statement. That's where the Status Word comes in. (The next four paragraphs describe the Status Word in some detail. You can skip over them if it's more detail than you feel you want to go into.)

The Status Word is the primary mechanism for communicating the inner workings of the Graphics Tablet to the outside world. If we look at the pen parameter passed back from the Graphics Tablet after an OC command is received, it might typically look like this:

pen parameter string

0,00,1024,0

0 = pen not down
1 = pen down

decimal value of status word
$= 2^{10}$ i.e. bit 10
pen press

The third group of digits is the decimal equivalent of the bit pattern in the status register. This is the status word, and is 11 bits long, with 9 of the bits representing significant conditions within the 9111A. A map of the status word looks like this:

Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pen Press	New Cursor Position	Proximity	Menu Pick	SRQ	Error	Ready	Initialized	Digitized Point Available	Clear	Clear

By checking bit 2 of the status word each time we read the cursor position, it is possible to determine if a point has been digitized since the last digitized point was read. This allows the program to sit in the cursor update loop and wait for digitizing to be done by the 9111A. The program then branches to a digitized point routine which updates our data base and display.

It is quite simple to analyze the P$ string you have already read in to isolate the Status Word and then use the VAL function to convert the string to a numeric. The numeric can then be analyzed using either of two methods, depending on whether or not an I/O ROM is available.

Insert the following line before line 120:

? start reading the 6th character
of pen string

```
111        Status_word = VAL(P$[6])
```

Now, if you have an I/O ROM, change line 140 to:

```
140        IF BIT(Status_word,2) THEN GOSUB Digitized_point
```

If you have no I/O ROM, use the following line:

```
140        IF Status_word MOD (4*2) - Status_word Mod 4
              THEN GOSUB Digitized_point
```

The I/O ROM considerably speeds up the process. Whichever method is used, you still must have a routine to respond to the digitized point being detected. This brings us to the data base.

A data base is simply an orderly method for storing data. The data base used in this program consists of three large arrays. The arrays are dimensioned large enough to hold whatever you intend to put in them (and isn't that a circular specification?) To start with, use 100 elements in each array. To do this, insert the following lines at the beginning of the program.

```
1       OPTION BASE 0
2        SHORT X(100),Y(100)
3        INTEGER P(100)
4        X(0)=Y(0)=P(0)=0
```

Now add the following routine to the end of the program:

```
1000 Digitized_point:!
1010            PLOTTER 7,6 IS ON
1020            I=I + 1
1030            DIGITIZE X(I),Y(I),P$
1040            P(I)=VAL(P$)
1050            PLOTTER 13 IS ON
1060            PLOT X(I),Y(I),P(I)
1070            POINTER X(I),Y(I),2
1080   RETURN
```

Before you run this program, you need to set up a digitizing mode on the 9111A. If you have an I/O ROM, add this line before line 80:

```
71      OUTPUT 706;"CN;SF"
```

If you don't have an I/O ROM you can "print" the commands to the 9111A. Use these lines:

```
71      PRINTER IS 7,6
72      PRINT "CN;SF"
73      PRINTER IS 16
```

Now RUN the program. It should operate much as it did before, except that it will eventually overflow the array, halting operation of the program. "That's an Improvement?!" I hear you say. Well, actually it is, but not because you can overflow the array. Rather it is because you have it stored in the array. To see why this matters, change line 150 to read

```
150   IF I<100 THEN GOTO Begin_loop
```

Now add these lines immediately following it.

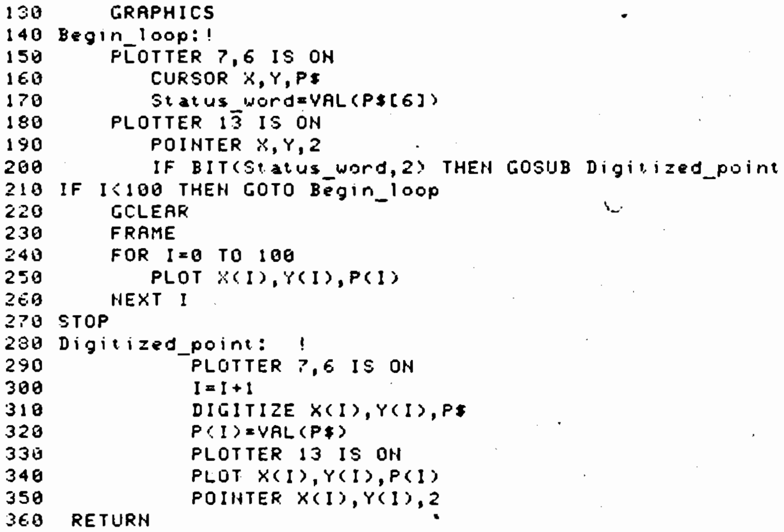
```
151   GCLEAR
152   FRAME
153   FOR I=0 TO 100
154       PLOT X(I),Y(I),P(I)
155   NEXT I
```

Now change line 160 to STOP.

RUN the program again. This time, when the array is filled, the screen will automatically be cleared, and the drawing re-plotted. The data is now in a data base (albeit a primitive one) and can be re-used by the machine.

If you renumber the program, it should look like this:

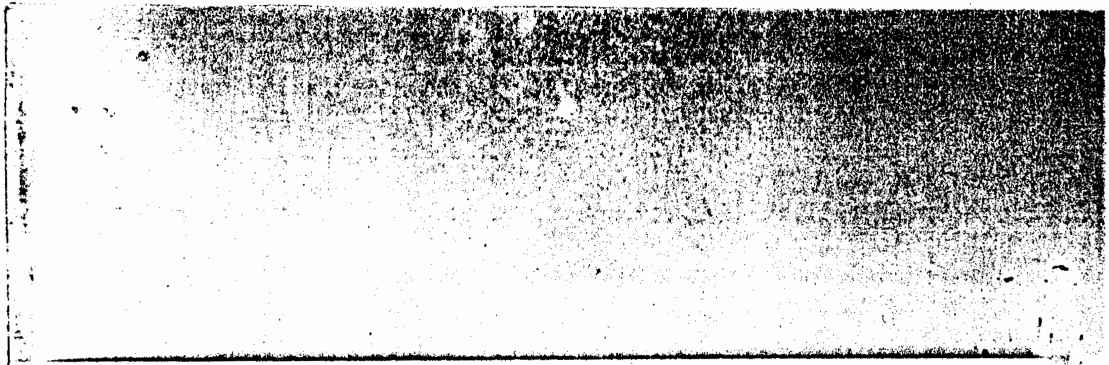
```
10    OPTION BASE 0
20    SHORT X(100),Y(100)
30    INTEGER P(100)
40    X(0)=Y(0)=P(0)=0
50 Setup:   !
60      PLOTTER IS 7,6,"9872A"
70         MSCALE -10,-10
80      PLOTTER IS 13,"GRAPHICS"
90         SHOW 0,301,0,237
100        CLIP 0,301,0,218
110        FRAME
120     OUTPUT 706;"CN;SF"
```



```
130     GRAPHICS
140 Begin_loop:!
150     PLOTTER 7,6 IS ON
160        CURSOR X,Y,P$
170        Status_word=VAL(P$[6])
180     PLOTTER 13 IS ON
190        POINTER X,Y,2
200        IF BIT(Status_word,2) THEN GOSUB Digitized_point
210 IF I<100 THEN GOTO Begin_loop
220     GCLEAR
230     FRAME
240     FOR I=0 TO 100
250        PLOT X(I),Y(I),P(I)
260     NEXT I
270 STOP
280 Digitized_point:  !
290           PLOTTER 7,6 IS ON
300           I=I+1
310           DIGITIZE X(I),Y(I),P$
320           P(I)=VAL(P$)
330           PLOTTER 13 IS ON
340           PLOT X(I),Y(I),P(I)
350           POINTER X(I),Y(I),2
360  RETURN
```

Having the data around in machine readable form enables it to be replotted on another plotter, analyzed by routines, transmitted to other computers, stored, or in other words, generally manipulated by computers. The next step is telling the computer what to do, without using the keyboard on the 9845B.

Softkeys — coming up.



What Do I Do Now?
Softkeys

One big advantage of the 9111A is the built in softkey structure The sixteen numbered boxes along the upper edge of the platen can be interpreted by the Graphics Tablet as Keys, and used to control the execution of a program on the 9845B. This allows the user to go completely into a 'Graphics' mode of operation without ever having to come out of it to use the 9845 keyboard.

The Status Word of the 9111A also has a bit which indicates a softkey has been selected. If a point is selected within one of the Softkey boxes at the top of the platen, bit seven goes true (=1) and you can test for this the same way you tested for the digitized point available bit.

If you have an I/O ROM, add the following line before line 200:

```
191        IF BIT(Status_word,7) THEN GOSUB Which_key
```

no. ?

If you have an I/O ROM, add the following line before line 220:

```
211        IF Status_word MOD (128*2)-Status_word MOD 128
           THEN GOSUB Which_key
```

The I/O ROM considerably speeds up the process. Whichever method is used, you still must have a routine that responds to the Softkey being selected. Two versions of a routine to respond to the softkey selection are provided below, one for use with an I/O ROM, and one for use without it.

They both call a second routine (Key1) which redraws the image on the screen. Add the appropriate Which_key routine, and then add the Key1 subroutine.

```
1000             !
1010 Which_key:! For I/O ROM programs
1010             OUTPUT 706;"RS1"
1030             ENTER 706;Key
1040             ON Key GOSUB Key1,Key2,Key3,Key4,Key5,Key6,
                              Key7,Key8,Key9,Key10, Key11,
                              Key12,Key13,Key14,Key15,Key16
1050 RETURN
```

```
1000            !
1010 Which_key:!  For Non I/O ROM programs
1020            Key=VAL(P$[3])
1030            PRINTER IS 7,6
1040            PRINT "SK0"
1050            PRINTER IS 16
1060            ON Key GOSUB Key1,Key2,Key3,Key4,Key5,Key6,
                             Key7,Key8,Key9,Key10, Key11,
                             Key12,Key13,Key14,Key15,Key16
1070    RETURN
```

(The perceptive reader will have noticed that the I/O ROM routine sends "RS1" to the Graphics Tablet while the non-I/O ROM routine sends "SK0". The RS instruction loads the menu number into the I/O buffer, and then clears the status bit for menu selection available and the menu register itself. The SK command clears the bit and register, but does not load the buffer. The value in the pen string is used instead.)

The subroutine that both Which-key routines call is:

```
1100        !
1110 Key1:! This routine Dumps Graphics to the internal
            printer
1120        DUMP GRAPHICS
1130 RETURN
```

Now that the routines have been added, RUN the program. Draw a picture, and then press the stylus tip down in the square labeled 1. The picture on the screen should be dumped to the internal printer.

Now add the following routine to the program:

```
1200        !
1210 Key2:! Clear screen and re-initialize array pointers
1220        PLOTTER 13 IS ON
1230        GCLEAR
1240        FRAME
1250        I=0
1260 RETURN
```

Now you can clear the screen, too — Softkey program control, and not too difficult. It is obvious that 14 other routines could be invoked by using the other subroutine calls in the ON Key GOSUB construct.

Here's the current program as it looks after it is renumbered.

```
10     OPTION BASE 0
20     SHORT X(100),Y(100)
30     INTEGER P(100)
40     X(0)=Y(0)=P(0)=0
50  Setup:     !
60       PLOTTER IS 7,6,"9872A"
70          MSCALE -10,-10
80       PLOTTER IS 13,"GRAPHICS"
90          SHOW 0,301,0,237
100         CLIP 0,301,0,218
110         FRAME
120      OUTPUT 706;"CN;SF"
130      GRAPHICS
140 Begin_loop:!
150      PLOTTER 7,6 IS ON
160         CURSOR X,Y,P$
170         Status_word=VAL(P$[6])
180      PLOTTER 13 IS ON
190         POINTER X,Y,2
200         IF BIT(Status_word,7) THEN GOSUB Which_key
210         IF BIT(Status_word,2) THEN GOSUB Digitized_point
220 IF I<100 THEN GOTO Begin_loop
230      GCLEAR
240      FRAME
250      FOR I=0 TO 100
260         PLOT X(I),Y(I),P(I)
270      NEXT I
280 STOP
```

```
290 Digitized_point:  !
300          PLOTTER 7,6 IS ON
310          I=I+1
320          DIGITIZE X(I),Y(I),P$
330          P(I)=VAL(P$)
340          PLOTTER 13 IS ON
350          PLOT X(I),Y(I),P(I)
360          POINTER X(I),Y(I),2
370  RETURN
380   !
390 Which_key:  !
400           OUTPUT 706;"RS1"
410           ENTER 706;Key
420           ON Key GOSUB Key1,Key2,Key3,Key4,Key5,Key6,
                           Key7,Key8,Key9,Key10,Key11,
                           Key12,Key13,Key14,Key15,Key16
430  RETURN
440   !
450 Key1:  ! This routine dumps graphics to the internal
             printer.
460        DUMP GRAPHICS
470  RETURN
480   !
490 Key2:  ! Clear the screen and re-initialize the array
             pointers.
500        PLOTTER 13 IS ON
510        GCLEAR
520        FRAME
530        I=0
540  RETURN
```

Softkeys do not operate if array is filled

Excuse Me, But ...
Interrupt Driven Processing

Now that you have covered the basics of operating the Graphics Tablet, it's time to look at speeding up and simplifying the program by using Interrupt processing. Interrupts on the 9845B require an I/O ROM, so the following section (and all the sections that follow it) assumes that you have an I/O ROM.

Interrupt driven operations eliminate the constant checking of the bits in the status word. Instead, we have the 9111A keep track of the status, and generate an SRQ (Service Request) when it requires attention. This Service request is then allowed to interrupt the HP-IB interface which can cause an end of line branch in the operation of the program.

Here's an outline for an interrupt driven program:

1. Set Up Graphics Tablet
2. Set Up Plotter
3. Define End of Line Branch Service Routine
4. Set SRQ generating conditions in Graphics Tablet
5. Enable Interrupts
6. Track Cursor (as long as necessary)

When the interrupt is received, a suitable interrupt response must be generated. This is handled by what is called a Service Routine. The Service Routine must:

1. Determine the cause of the End Of Line Branch

   If the cause was an SRQ

2. Determine the cause of the SRQ in the device that generated the SRQ
3. Respond appropriately to the SRQ

4. Re-enable interrupts

From this outline, modify the program you already have to respond to interrupts.

First, delete lines 170,200, and 210. Now change line 220 to read:

```
220        GOTO Begin_loop
```

Now delete lines 230 to 280. You are back to a cursor tracking loop. Don't delete the subroutines, as you will use them later. Next, add the interrupt set up routines. Insert the following lines before line 130:

```
121      !
122      ON INT #7 GOSUB What_happened
123      OUTPUT 706;"IM,";4+128
124      CONTROL MASK 7;128
125      CARD ENABLE 7
126      !
```

Line 122 sets up the response to the interrupts from card 7. Line 123 tells the 9111A when to generate an SRQ (2^2 = 4 for digitized point $+2^7$ = 128 for menu selected.) Line 124 tells the interface card what to interrupt the controller about, and line 125 enables interrupts from the card.

Now insert the routine to service the interrupt before line 290:

```
221          !
222   What_happened:!
223          OUTPUT 706;"OS"
224          ENTER 706;Status_word
225          IF BIT(Status_word,2) THEN GOSUB Digitized_point
226          IF BIT(Status_word,7) THEN GOSUB Which_key
227          CARD ENABLE 7
228    RETURN
229          !
```

The What_happened routine reads the status from the Graphics Tablet to determine what generated the SRQ, and then branches to an appropriate subroutine to handle the Service Request.

Since the program is using interrupt processing, it is possible to update the pointer position in the digitize subroutine, and then go back to an old location in the cursor tracking loop. To handle this, merely update the X and Y values before leaving the digitize loop. Insert the following lines before line 370.

```
361       X=X(I)
362       Y=Y(I)
```

It will also help to redimension the arrays in lines 22 and 30 from 100 to 10000 elements each to create a larger workspace. (This is for a 9845T — smaller memories will not allow this large a workspace.)

```
20    SHORT X(10000),Y(10000)
30    INTEGER  P(10000)
```

Now, when you run the program, the program should respond as before, but does not require as complicated a series of tests for branching to the routines needed by the system. This can greatly simplify the program implementation. The whole program, after renumbering, is shown below.

```
10    OPTION BASE 0
20    SHORT X(10000),Y(10000)
30    INTEGER P(10000)
40    X(0)=Y(0)=P(0)=0
50 Setup:    !
60      PLOTTER IS 7,6,"9872A"
70         MSCALE -10,-10
80      PLOTTER IS 13,"GRAPHICS"
90         SHOW 0,301,0,237
100        CLIP 0,301,0,218
110        FRAME
120     OUTPUT 706;"CN;SF"
130     !
```

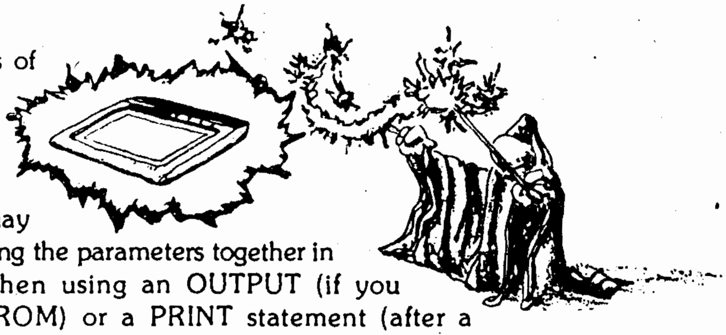
```
140     ON INT #7 GOSUB What_happened
150     OUTPUT 706;"IM,";4+128
160     CONTROL MASK 7;128
170     CARD ENABLE 7
180     !
190     GRAPHICS
200 Begin_loop:!
210     PLOTTER 7,6 IS ON
220        CURSOR X,Y,P$
230     PLOTTER 13 IS ON
240        POINTER X,Y,2
250 GOTO Begin_loop
260      !
270 What_happened:!
280       OUTPUT 706;"OS"
290       ENTER 706;Status_word
300       IF BIT(Status_word,2) THEN GOSUB Digitized_point
310       IF BIT(Status_word,7) THEN GOSUB Which_key
320       CARD ENABLE 7
330 RETURN
340      !
350 Digitized_point:  !
360           PLOTTER 7,6 IS ON
370           I=I+1
380           DIGITIZE X(I),Y(I),P$
390           P(I)=VAL(P$)
400           PLOTTER 13 IS ON
410           PLOT X(I),Y(I),P(I)
420           POINTER X(I),Y(I),2
430           X=X(I)
440           Y=Y(I)
450  RETURN
460    !
470 Which_key:   !
480            OUTPUT 706;"RS1"
490            ENTER 706;Key
500            ON Key GOSUB Key1,Key2,Key3,Key4,Key5,Key6,
                         Key7,Key8,Key9,Key10,Key11,
                         Key12,Key13,Key14,Key15,Key16
510  RETURN
520    !
530 Key1:   ! This routine dumps graphics to the internal
              printer.
540         DUMP GRAPHICS
550  RETURN
560    !
570 Key2:   ! Clear the screen and re-initialize the array
              pointers.
580         PLOTTER 13 IS ON
```

```
590         GCLEAR
600         FRAME
610         I=0
620   RETURN
```

The next step is to add some sound to the program. Beeping The Beeper, coming up.

What's That Sound?
Audible Prompts

One nice feature on the 9111A is an easily controlled Beeper. The Beep instruction enables you to select from a set of musical tones and easily specify their duration and amplitude. This makes it simple to design attention grabbing prompts for feedback to the operator of the Graphics Tablet.



Short sequences of tones are easier for users to deal with than individual beeps. They may be sent by packing the parameters together in a string, and then using an OUTPUT (if you have an I/O ROM) or a PRINT statement (after a PRINTER IS statement) to send the string. By selecting a set of standard prompt sequences for the various responses expected by the program (Ready for point, Point accepted etc.) it is easier for a user to deal with the programs on a long term basis. Certain conventions may also be carried over from an area which has dealt with sequences of tones for millenia, the field of Music. Use ascending frequency sequences to indicate inquiry, and descending sequences to indicate acknowledgement. Using a modified version of the 9111A error tone sequence to indicate errors encountered can provide some continuity between the hardware and software. The error tone sequence shown in Woops$ below is derived from the error tone sequence used by the 9111A.

Now add some audio prompts to the program you have been working on.

First, insert the following lines before line 20 in the program.

```
                                    4
11       DIM  Softkey_in$[25],Woops$[25]                    delete
12       Softkey_in$="BP36,50,3;BP34;BP32;BP30"
13       Woops$="BP18,25,5;BP21"&RPT$(";BP18;BP21",3")
```

Now insert the following line before line 500

```
491               IF Key > 2 THEN GOTO Error
492               OUTPUT 706; Softkey_in$
```

and insert this error routine before line 530

```
521 Error: !
522       OUTPUT 706;Woops$
523  RETURN
524       !
```

Now RUN the program and select a menu item. You should get the soft key recognized prompt if you select a valid softkey (1 or 2) or an error tone if an undefined key is selected.

Next, a look at Digitizing Modes, and what they mean.

What's in a Mode?
Digitizing Modes

The 9111A has two digitizing modes, Single (SG) and Continuous (CN). The following program modifications demonstrate the Single and Continuous modes. It also shows the two switch modes that affect the Continuous mode, Switch Normal (SN) and Switch Follow (SF).

In the Single Point mode, a point is taken when the tip of the stylus is pressed against the platen (this is sometimes called a ''picking a point'' or ''pick''). In the Continuous mode, points are loaded into the I/O buffer at a rate set by the Data Rate Register. If a point is not read before a new one is ready, the point is replaced by the new value.

In the Continuous Mode, it is possible to start and stop the flow of points using the switch in the Stylus. Two different control modes are provided, Switch Normal and Switch Follow. Switch follow operates very intuitively — while you press down on the Stylus, points are read into the I/O buffer, and when the pressure is released, points are no longer taken — rather like drawing with a pen or a pencil. In the Switch Normal mode, the Stylus switch is used to start and stop the process of taking points — you press once to start the flow, and press again to stop it.

These various modes are selected using the SG, CN, SF, and SN commands. Since you already have a program running to handle digitizing and softkey commands, it is a simple matter to add a new set of softkeys to change the digitizing modes.

You are adding three keys to the Softkey Menu, one each for Single, Switch Normal, and Switch Follow (Continuous is set by selecting either Switch Normal or Switch Follow.) Since softkeys one and two are already used, the new routines are added on softkeys three, four, and five. Add the following lines to the end of the program:

```
1000       !
1010 Key3: ! Set Single Point Mode, Begin new line
1020       OUTPUT 706;"SG"
1030       P(I)=0
1040       PLOTTER 13 IS ON
1050       PENUP
1060 RETURN
1070       !
```

```
1080 Key4:! Set Continuous Mode, Switch Follow, Begin new
          line
1090       OUTPUT 706;"CN;SF"
1100       P(I)=0
1110       PLOTTER 13 IS ON
1120       PENUP
1130 RETURN
1140       !
1150 Key5:! Set Continuous Mode, Switch Normal, Begin new
          line
1160       OUTPUT 706;"CN;SN"
1170       P(I)=0
1180       PLOTTER 13 IS ON
1190       PENUP
1200 RETURN
```

In each of the mode selecting subroutines, the mode selecting instructions are sent to the Graphics Tablet, the line contour just completed is terminated in the data base by forcing the pen parameter to 0, and then the pen is lifted on the CRT, so that the current line is terminated on the display.

You also need to allow selection of the added softkeys, so change line 530 to:

```
530             IF Key > 5 THEN GOTO Error
```

Now RUN the program. It should operate as it did before — it will default to Continuous Mode and Switch Follow. Now select Softkey 3. Press the stylus down in the active area of the platen. A small dot should appear on the CRT. Press the stylus at a new location on the platen, and a line will appear between the last point and the new one. This interconnecting will continue until a new mode Softkey is selected, which will start a new contour. Selecting the mode already in effect will continue operation in the mode, but start a new contour. Use the table below to get used to the various modes of the Graphics Tablet, and to get used to Softkey program control.

Softkey 1 — Dump Graphics
Softkey 2 — Erase Picture
Softkey 3 — Single Point (Line) Mode
Softkey 4 — Switch Follow Continuous Mode
Softkey 5 — Switch Normal Continuous Mode

Using a table to keep track of what the Softkeys are doing is a little clumsy. There are three ways to get around this.

1. Write on the Platen Directly
   A Felt Tip or Grease pen can be used to write on the platen. This is quick and easy to do, but is a bit messy.
2. Generate an artwork overlay
   If a 9872 Plotter or illustrating facilities are available, an art work overlay can be generated. This is very good for projects which will see repeated use, but is an overkill for a simple demo.
3. Use a CRT Soft Menu
   It is possible to simply draw the menu on the CRT, and then use the pointer-cursor tracking loop for selection. This is the best technique for fast, temporary operations.

The first method is very simple, as long as the marking pen used can be easily cleaned from the platen. The last method is the next topic to be covered. Labels coming up.

What's Going On?

Labels

Soft Menus (appearing only on the CRT) can be very useful in making easily transportable programs for the 9111A.

The first step is generating the Softkey Menu. The Label_screen routine below draws the Softkeys and labels them. There are two nested loops in the program, because the Softkey boxes are not uniformly spaced. They are grouped in blocks of four squares each. The Block loop provides the inter-block spacing, and the Square loop spaces the individual squares within the block. As each block is drawn, a label is read for it from the data statement at the end of the routine. Once the Softkeys are all drawn and labeled, the active digitizing area is outlined.

```
1000          !
1010 Label_screen:!
1020          GRAPHICS
1030          Guard=4.1
1040          Radius=13.7
1050          Left=0-Guard-Radius/2
1060          CSIZE 2.8
1070          LORG 5
1080          RESTORE Labels
1090          FOR Block=0 to 3
1100             FOR Square=1 TO 4
1110                X=Left+Block*(4*Radius+3*Guard+10.6)+
                    Square*(Radius+Guard)
1120                CLIP X-Radius/2,X+Radius/2,222.6,235.8
1130                FRAME
1140                READ Label$
1150                MOVE X,230
1160                LABEL USING "K";Label$
1170             NEXT Square
1180          NEXT Block
1190          !
1200          CLIP 0,301,0,218
1210          FRAME
1220          !
1230 Labels:  DATA DMP,CLR,SNG,SF,SN," "," "," "," "," ",
              " "," "
1240          DATA " "," "," "," "
1250 RETURN
```

Notice the blanks enclosed in quotes in the data statements. These are for unassigned Softkeys, to be added at a later time.

The routine is not very useful without calls to it. First it must be called to initialize the CRT. Change line 130 to:

```
130       GOSUB Label_screen
```

Delete line 140 and 220. Now change line 150 to:

```
150       GOSUB Key4
```

It must also be called after clearing the screen. Delete line 690 and insert the following line before line 710:

```
701       GOSUB Label_screen
```

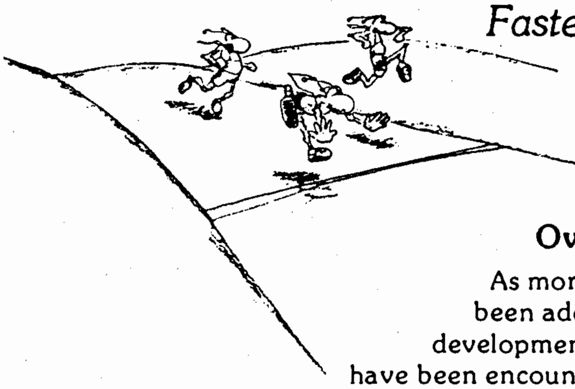
Generally any time you erase the screen, the Label_screen routine must be called.

Now run the program. A map of the 9111A will be drawn on the CRT, along with labels for the Softkey functions. Otherwise, the program should operate as before.

```
10     OPTION BASE 0
20     DIM Softkey_in$[25],Woops$[45]
30     Softkey_in$="BP36,50,3;BP34;BP32;BP30"
40     Woops$="BP18,25,5;BP21"&RPT$(";BP18;BP21",3)
50     SHORT X(10000),Y(10000)
60     INTEGER P(10000)
70     X(0)=Y(0)=P(0)=0
80 Setup:    !
90        PLOTTER IS 7,6,"9872A"
100          MSCALE -10,-10
110       PLOTTER IS 13,"GRAPHICS"
120          SHOW 0,301,0,237
130          GOSUB Label_screen
```

```
650         PLOTTER 13 IS ON
660         GCLEAR
670         I=0
680         GOSUB Label_screen
690   RETURN
700    !
710 Key3: ! Set Single Point Mode, Begin New Line
720        OUTPUT 706;"SG"
730        P(I)=0
740        PLOTTER 13 IS ON
750        PENUP
760   RETURN
770        !
780 Key4: ! Set Continuous Mode, Switch Follow, Begin New
Line
790        OUTPUT 706;"CN;SF"
800        P(I)=0
810        PLOTTER 13 IS ON
820        PENUP
830   RETURN
840        !
850 Key5: ! Set Continuous Mode, Switch Normal, Begin New Lin
e
860        OUTPUT 706;"CN;SN"
870        P(I)=0
880        PLOTTER 13 IS ON
890        PENUP
900   RETURN
910        !
920 Label_screen: !
930        GRAPHICS
940        Guard=4.1
950        Radius=13.7
960        Left=0-Guard-Radius/2
970        CSIZE 2.8
980        LORG 5
990        RESTORE Labels
1000       FOR Block=0 TO 3
1010          FOR Square=1 TO 4
1020             X=Left+Block*(4*Radius+3*Guard+10.6)+Square*
(Radius+Guard)
1030             CLIP X-Radius/2,X+Radius/2,222.6,235.8
1040             FRAME
1050             READ Label$
1060             MOVE X,230
1070             LABEL USING "K";Label$
1080          NEXT Square
1090       NEXT Block
1100       !
1110       CLIP 0,301,0,218
1120       FRAME
1130       !
1140 Labels: DATA DMP,CLP,SNG,SF,SN," "," "," "," "," "
1150          DATA " "," "," "," "," "," "
1160 RETURN
```

You may have noticed the program has slowed down considerably since the first cursor tracking loop. There are some ways to overcome this — and that's the next topic. Speed — coming up.



Faster Horses... Speeding up Graphics Tablet Operations

Overall Speed

As more and more features have been added to the program under development, certain speed penalties have been encountered. Several techniques may be employed to speed up the data transfer operations occurring in various parts of this program. The simplest is to put the 9845B into OVERLAP mode. Insert the following line before line 10 of the program.

```
1       OVERLAP
```

Cursor Speed

The next step is to speed up the cursor tracking loop. This is accomplished by using the Binary Transfer that is the default response of the 9111A. If no other data transfer is being undertaken, the 9111A will provide six bytes of binary data representing the X and Y location of the cursor (two bytes each, twos complement) and the status word of the 9111A (two bytes also.) This information is always ready if no command has been sent to the 9111A which requires other data to be loaded into the I/O register.

To use the Binary Transfer, three integers are needed. Modify line 60 to include declarations for M,N, and Z.

```
60     INTEGER P(10000),M,N,Z
```

Next, replace the cursor tracking loop. Delete lines 220 to 250. Then insert the following lines before line 260:

```
211 ENTER 706 USING "W,W,W";M,N,Z
212 POINTER M/40,N/40,2
```

Line 211 uses a two byte binary integer handshake specifier (W) to enter the binary transfer information. Line 212 divides by 40 (there are 40 digitizer units per millimetre) to scale the data to be displayed on the CRT. Add this line before line 200:

```
199    PLOTTER 13 IS ON
```

If you run the program, the cursor tracking operation should be considerably faster.

Digitizing Speed

Even with the faster cursor tracking, continuous digitizing is slowed down by the overhead of plotting a pointer on the CRT and returning to the cursor tracking loop after each point is received. To minimize this overhead, a second digitize routine can be added. Add this Fast_digitize routine to the end of the program.

```
2000 !
2010 Fast_digitize:!
2020         POINTER 1000,1000
2030 Loop2:!
2040         I=I+1
2050         PLOTTER 7,6 IS ON
2060            DIGITIZE X(I),Y(I),P$
2070            P(I)=VAL(P$)
```

```
2080          PLOTTER 13 IS ON
2090             PLOT X(I),Y(I),P(I)
2100 IF P(I) THEN GOTO Loop2
2110          X=X(I)
2120          Y=Y(I)
2130          POINTER X,Y,2
2140  RETURN
```

The first POINTER statement throws the pointer entirely off the CRT. A Fast digitizing loop is then used until the pen parameter goes to zero (indicating the end of a stream of points in the continuous mode.) Once the fast digitizing loop is finished, the pointer values are updated, and the pointer itself repositioned. Then the program execution is returned to the calling routine.

This routine must be called from somewhere. Add the following line before line 320:

```
311 IF BIT(Status_word,2) AND Continuous THEN GOSUB Fast_
    digitize
```

Then change line 310 to:

```
310 IF BIT(Status_word,2) AND NOT Continuous THEN GOSUB
    Digitize
```

Now add the following lines to the various mode selection routines to set the Continuous flag to an appropriate value:

Key3

```
731        Continuous=0
```

Key4

```
801        Continuous=1
```

Key5

```
871        Continuous=1
```

Now the program selects a fast digitizing routine in the continuous mode, and the normal routine in the single point mode. The completed tutorial program is listed below.

It is possible to speed up the program even further by using only the binary transfer data, and decoding it directly. This reguires you to write your own scaling routines and to interpret the physical pen parameter included in the status bytes transferred at the end of the binary transfer. The communication is done entirely with the I/O ROM, and can speed up the operation, at some expense in ease of use. Such a program is beyond the scope of this tutorial.

This concludes the tutorial section of your 9111A-9845B Systems manual. The next section deals with various AGL commands (AGL is the 9845B Graphics Language) and how they affect the Graphics Tablet.

```
1      OVERLAP
10     OPTION BASE 0
20     DIM Softkey_in$[25],Woops$[45]
30     Softkey_in$="BP36,50,3;BP34;BP32;BP30"
40     Woops$="BP18,25,5;BP21"&RPT$(";BP18;BP21",3)
50     SHORT X(10000),Y(10000)
60     INTEGER P(10000),M,N,Z
70     X(0)=Y(0)=P(0)=0
80 Setup:   !
90       PLOTTER IS 7,6,"9872A"
100         MSCALE -10,-10
110      PLOTTER IS 13,"GRAPHICS"
120         SHOW 0,301,0,237
130         GOSUB Label_screen
140      GOSUB Key4
150      !
160      ON INT #7 GOSUB What_happened
170      OUTPUT 706;"IM,";4+128
180      CONTROL MASK 7;128
190      CARD ENABLE 7
191      PLOTTER 13 IS ON
200      !
210 Begin_loop:!
211      ENTER 706 USING "W,W,W";M,N,Z
212      POINTER M/40,N/40,2
260 GOTO Begin_loop
270      !
```

```
280 What_happened:!
290       OUTPUT 706;"OS"
300       ENTER 706;Status_word
310       IF BIT(Status_word,2) AND NOT Continuous THEN GOSUB
 Digitized_point
311       IF BIT(Status_word,2) AND Continuous THEN GOSUB
 Fast_digitize
320       IF BIT(Status_word,7) THEN GOSUB Which_key
330       CARD ENABLE 7
340 RETURN
350     !
360 Digitized_point:  !
370           PLOTTER 7,6 IS ON
380           I=I+1
390           DIGITIZE X(I),Y(I),P$
400           P(I)=VAL(P$)
410           PLOTTER 13 IS ON
420           PLOT X(I),Y(I),P(I)
430           POINTER X(I),Y(I),2
440           X=X(I)
450           Y=Y(I)
460  RETURN
470   !
480 Which_key:  !
490            OUTPUT 706;"RS1"
500            ENTER 706;Key
510            IF Key>5 THEN GOTO Error
520            OUTPUT 706;Softkey_in$
530            ON Key GOSUB Key1,Key2,Key3,Key4,Key5,Key6,Key
7,Key8,Key9,Key10,Key11,Key12,Key13,Key14,Key15,Key16
540  RETURN
550   !
560 Error:  !
570      OUTPUT 706;Woops$
580 RETURN
590     !
600 Key1:  ! This routine dumps graphics to the internal
printer.
610        DUMP GRAPHICS
620  RETURN
630   !
640 Key2:  ! Clear the screen and re-initialize the array
pointers.
650        PLOTTER 13 IS ON
660        GCLEAR
670        I=0
680        GOSUB Label_screen
690  RETURN
700   !
710 Key3: ! Set Single Point Mode, Begin New Line
720       OUTPUT 706;"SG"
730       P(I)=0
731       Continuous=1
```

```
740         PLOTTER 13 IS ON
750         PENUP
760   RETURN
770         !
780 Key4:   ! Set Continuous Mode, Switch Follow, Begin New
Line
790         OUTPUT 706;"CN;SF"
800         P(I)=0
801         Continuous=1
810         PLOTTER 13 IS ON
820         PENUP
830   RETURN
840         !
850 Key5:   ! Set Continuous Mode, Switch Normal, Begin New
Line
860         OUTPUT 706;"CN;SN"
870         P(I)=0
871         Continuous=0
880         PLOTTER 13 IS ON
890         PENUP
900   RETURN
910         !
920 Label_screen: !
930         GRAPHICS
940         Guard=4.1
950         Radius=13.7
960         Left=0-Guard-Radius/2
970         CSIZE 2.8
980         LORG 5
990         RESTORE Labels
1000        FOR Block=0 TO 3
1010           FOR Square=1 TO 4
1020              X=Left+Block*(4*Radius+3*Guard+10.6)+Square*
(Radius+Guard)
1030              CLIP X-Radius/2,X+Radius/2,222.6,235.8
1040              FRAME
1050              READ Label$
1060              MOVE X,230
1070              LABEL USING "K";Label$
1080           NEXT Square
1090        NEXT Block
1100        !
1110        CLIP 0,301,0,218
1120        FRAME
1130        !
1140 Labels: DATA DMP,CLR,SNG,SF,SN," "," "," "," "," "
1150           DATA " "," "," "," "," "," "
1160 RETURN
2000  !
2010 Fast_digitize: !
2020          POINTER 1000,1000
2030 Loop2: !
2040          I=I+1
```

```
2050        PLOTTER 7,6 IS ON
2060           DIGITIZE X(I),Y(I),P$
2070           P(I)=VAL(P$)
2080        PLOTTER 13 IS ON
2090           PLOT X(I),Y(I),P(I)
2100  IF P(I) THEN GOTO Loop2
2110        X=X(I)
2120        Y=Y(I)
2130        POINTER X,Y,2
2140  RETURN
```

Language Reference

Language Reference

9111A Instruction Set

The instruction set for the 9111A Graphics Tablet consists of 27 Hewlett-Packard Graphics Language (HPGL) instructions. Each instruction is a two-letter mnemonic which can be either upper or lower case. Depending on the instruction, some of the mnemonics allow numeric parameters. If more than one parameter is allowed with an instruction, the parameters must be separated with a comma. Spaces and carriage return (CR) characters within the data string are ignored by the graphics tablet.

Data transfer to and from the graphics tablet is in 8-bit ASCII code. Data placed on the bus by the graphics tablet is terminated with the carriage return/linefeed (CR/LF) characters. Parameters within the data string are separated with a comma. Instructions received by the graphics tablet must be terminated with a linefeed (LF) character, semicolon (;) or the HP-IB END method. Data termination is discussed next.

Data Termination

The graphics tablet responds to three types of data (instruction) termination. The three types are explained next. See your controller manual for the output format of your controller.

1. Whenever the graphics tablet receives a data string followed by a linefeed character (ASCII decimal 10), the data is interpreted as a complete instruction (two letter mnemonic with any allowable parameters). Any additional data characters received by the graphics tablet are inter- - preted ~~rupted~~ as another instruction. HP Desktop Computers generate the CR/LF characters internally for the control of peripheral devices. This is an operating system function and to avoid the output of these characters you must specify certain formats (see the operating manual for your computer for more information on its output format).

2. Whenever the graphics tablet receives a data string followed by a semicolon (ASCII decimal 59) character, the data is interpreted as a complete instruction (two letter mnemonic with any allowable parameters). Any additional data characters received by the graphics tablet are interpreted as another instruction. The semicolon character is available on the keyboard of the HP Desktop Computers and must be typed in along with the graphic tablet instruction.

3. HP-IB END refers to a third method of data termination available with the graphics tablet. This method uses the EOI (end of identify) interface signal line in conjunction with the last character in the data string. If EOI is set true (signal condition) prior to the graphics tablet receiving the last character in a data string, the last character serves its initial function (mnemonic or parameter) as well as acting as the data terminator.

NOTE

HP-IB END is a method of termination involving hardware as well as software functions. See the IEEE Std. 488-1978 for more information on this method.

HPGL Compatibility

The graphics tablet HPGL language differs from the 9874A's (HP Digitizer) language in the following respects:

1. Any instruction with more than one allowable parameter can have any parameter change without re-specifing the other parameters again. See the following example.

   Assuming we have specified Input Points to be the following values.

   IP 600,600,11000,8000

Later we need to re-specify IP. We want to change the 11000 to 8000. This can be done by just specifing the 8000 as shown next.

IP,,8000

With the 9874A Digitizer you had to re-specify each parameter whether it changed or not. The graphics tablet allows you to specify just the parameter you want to change providing you position it through the use of commas.

2. The IP instruction sent to the graphics tablet without parameters sets IP to default. Default for Input Points is 400,400,11632,8340.

   Sending "IP,,," does not change the current existing values for Input Points.

3. The binary transfer is unique to the graphics tablet. This is a (controller read initiated) binary output mode for fast data transfer. See the section titled "Binary Data Transfer" in this syntax section. Section 5.45

Methods Used to Represent Syntax

This syntax section uses two methods of representing the instruction set for the graphics tablet. The conventions of each form are as follows.

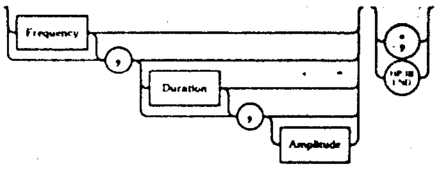
Pictorial Representation (the other is Linear Representation)

All items bolded and enclosed by a rounded envelope must be received by the graphics tablet **exactly as shown** (e.g., Mnemonics, Commas and Semicolon). Items in lighter text and enclosed by a rounded envelope refer to a termination character or termination method (e.g., Linefeed and HP-IB END). Items enclosed by rectangular boxes are names of parameters used in the instruction. A description of each parameter is given in the text following the drawing. Instruction elements are connected by lines. Each line can only be followed in one direction, from left to right. Any combination of instruction elements that can be generated by following the lines in the proper direction is syntactically correct. An instruction element is optional if there is a valid path around it. This form of syntax representation is easy to use, and in some cases, more formally correct than the alternate form described as "Linear Representation" which follows the next example.

The Beep instruction syntax is presented next. It is highlighted in three different ways. The accompanying text describes the highlighting as well as what the graphics tablet needs to receive if this example were actually encountered.

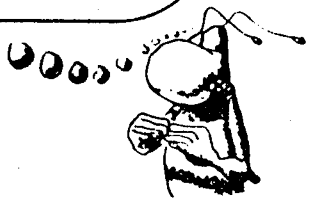
I need to program that certain sound, but I don't know the parameters. I'll start by sending BP with default parameters. Default parameters are 12,150,4.

This is the path on the graphic representation that I'll follow.



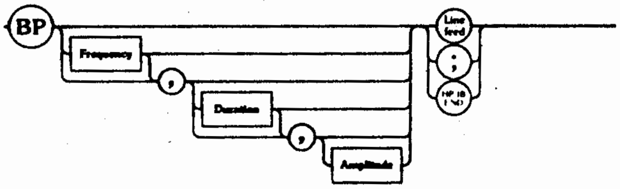
This is what the graphics tablet needs to receive.

BP Linefeed



Oh No! That's the wrong frequency; its not long enough and its too loud. I'll change all the parameters. I'll send note "A" above middle C (value 33), specify a duration of 1 second (value 1000), and soften the tone a little (value 3).

This is the path on the graphic representation that I'll follow.

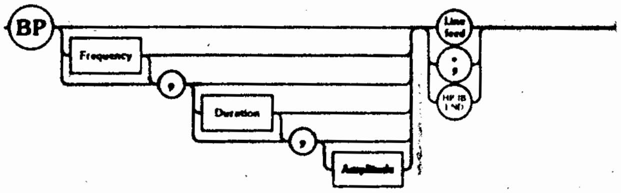


This is what the graphics tablet needs to receive.

BP33,1000,3 Linefeed

That's it; no, its still too loud. I'll soften the tone just a little. Now I don't want to change the other parameters so I need to send a comma specifing the place for both the frequency and duration parameters. This causes the graphics tablet to use the last specified parameters. Then I'll send a 2 for the amplitude.

This is the path on the graphic representation that I'll follow.



This is what the graphics tablet needs to receive.

BP,,2 Linefeed

That's the tone I want to hear.

Linear Representation

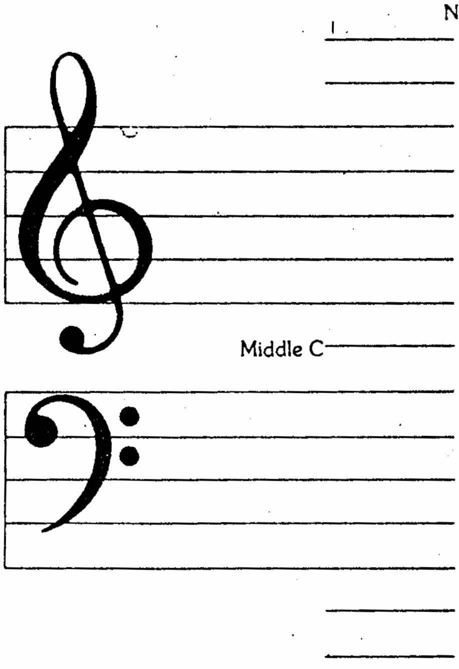
This form of syntax representation is included to be compatible with previous HP manuals. Many user's are accustomed to seeing this form. If both forms are new to you, it is recommended that you concentrate on the Pictorial form.

Bold Type : All items shown in **bold** type must be received by the graphics tablet exactly as shown. The one exception is that the mnemonics can appear in lower case characters.

[ ]: Items within square brackets are optional. If the optional items are used, the comma must preceed the second, third, and fourth items in the optional list.

| : A vertical line between two items reads as "or"; only one of the items may be included.

Parameter values of 49 thru 255 are accepted, but produce the same pitch as 48. The following table shows the parameter values and corresponding notes.



Note	N	Note	N
C	48 to 255		
B	47	A♯,B♭	46
A	45	G♯,A♭	44
G	43	F♯,G♭	42
F	41		
E	40	D♯,E♭	39
D	38	C♯,B♭	37
C	36		
B	35	A♯,B♭	34
A	33	G♯,A♭	32
G	31	F♯,G♭	30
F	29		
E	28	D♯,E♭	27
D	26	C♯,B♭	25
C	24		
B	23	A♯,B♭	22
A	21	G♯,A♭	20
G	19	F♯,G♭	18
F	17		
E	16	D♯,E♭	15
D	14	C♯,B♭	13
C	12		
B	11	A♯,B♭	10
A	9	G♯,A♭	8
G	7	F♯,G♭	6
F	5		
E	4	D♯,E♭	3
D	2	C♯,B♭	1
C	0		

Duration

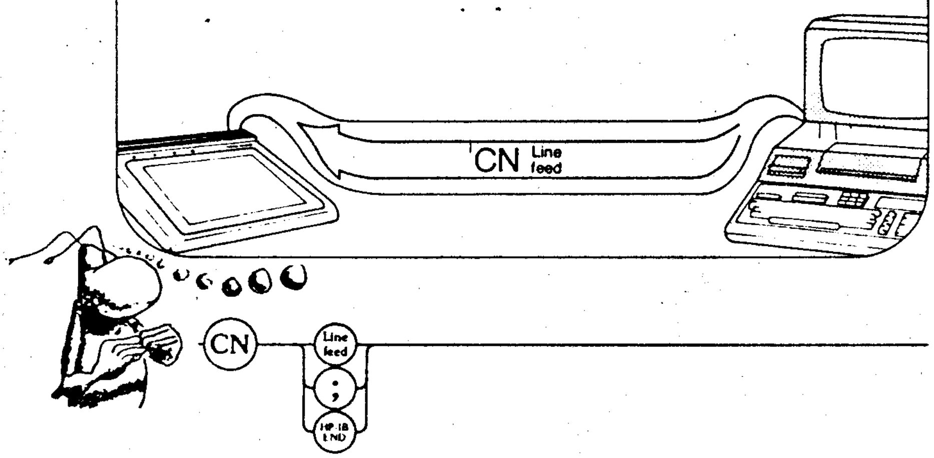
Duration (length the tone is generated) is specified in milliseconds. The values accepted are 1 thru 32 767. 32 767 milliseconds specify almost 33 seconds of the tone generation.

Amplitude

Values 0 thru 5 are accepted. 0 gives no tone whereas 1 gives a soft tone and 5 gives the loudest tone.

BP 12, 150, 4 is in effect at power on and reset. If the **BP** instruction is received without parameters, the tablet beeps using the last specified values for frequency, duration, and amplitude. Spaces and carriage return characters contained within the BP instruction are ignored by the graphics tablet.

Continuous Sampling Mode



CN Linefeed | ; | HP-IB END

The CN instruction sets the graphics tablet's continuous sampling mode. Once the CN mode is selected data sampling is controlled by the digitize switch contained within the stylus. Pressing the pen tip firmly against the graphics tablet's active surface initiates continuous digitizing. To stop the continuous digitizing the pen tip must again be firmly pressed against the tablet's active surface.

The digitizing mode described above is the continuous sampling mode with the stylus digitizing switch set to switch normal (SN). This toggling mode of the digitize switch is the default mode when CN is specified.

The SN and SF instructions allows you to program the response of the digitize switch.

The stylus digitize switch has an alternate mode which is specified using the SF (switch follow) mnemonic. When the SF condition is specified, the continuous digitizing is initiated only when the stylus pen is firmly pressed against the tablet's active surface. When the downward pressure is lessened causing the stylus digitize switch to open, continuous digitizing is stopped.

The mnemonics SF or SN can be specified at any time relative to setting the CN mode. The graphics tablet remembers a digitize switch condition specified prior to receiving a CN instruction.

CN [Δt,ΔD]

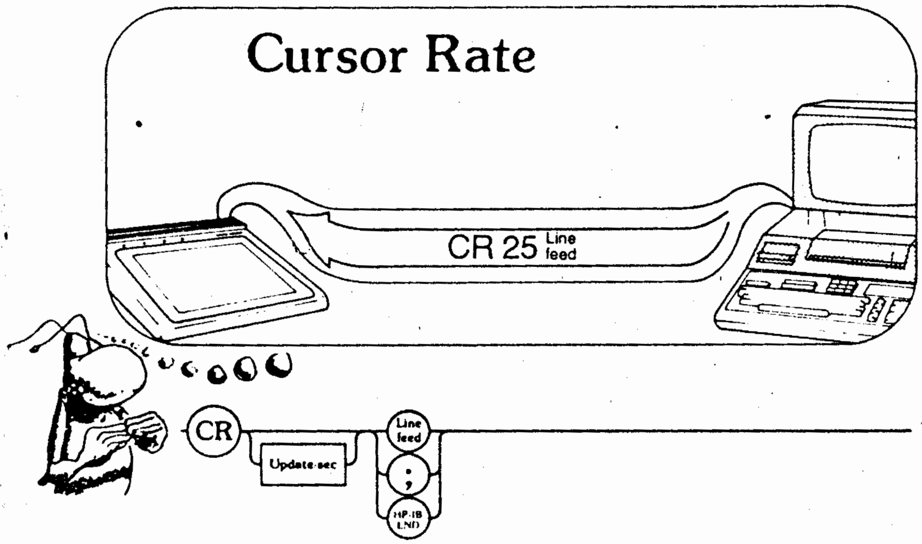
For compatibility reasons the graphics tablet allows two parameters with the CN instruction. The parameters are accepted, but not acted upon.

A recommended sequence for the use of the CN instruction follows:

- Set the CN mode.
- Set the digitize switch mode (SF or SN).
- Check bit 2 of the tablet's status.
- If bit 2 is set, send OD and read X,Y, and pen data.
- If bit 2 is not set, keep checking bit 2.

When a point is digitized, bit 2 is set.

Cursor Rate



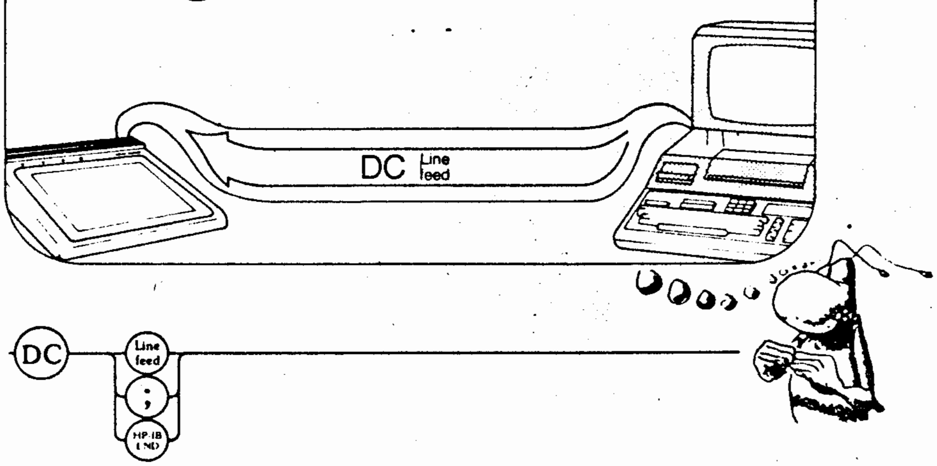
CR [update/sec] Linefeed | ; | HP-IB END

This instruction allows you to specify the data rate of the continuous sampling mode. CR allows you to control the number of data points going into your data base via time control. Another use for the instruction would be to establish or eliminate a stylus cursor lag (time delay between the stylus from the graphics tablet and a CRT cursor).

The values 1 thru 60 are the accepted parameter range and correspond to updates per second.

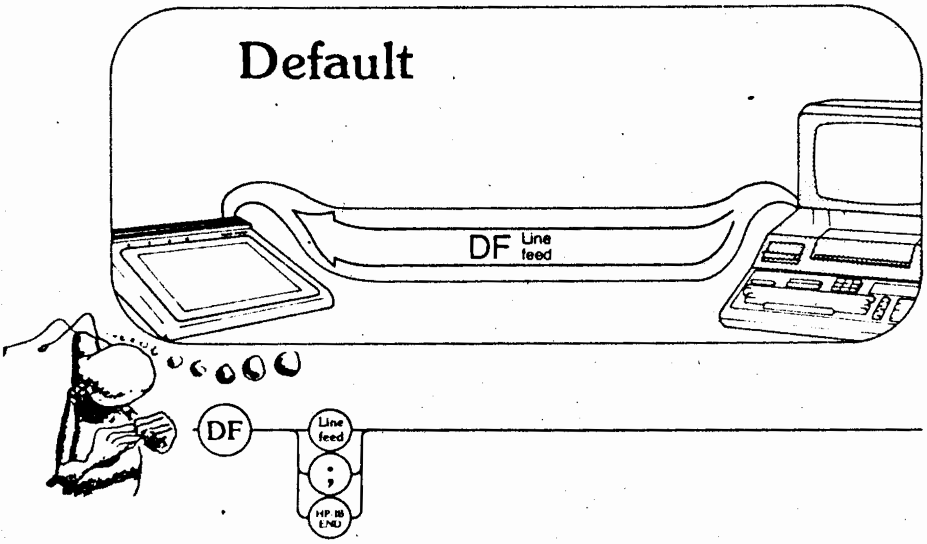
Specifing a CR with no parameters sets the default value which is 60 updates per second.

Digitizer Clear



DC Linefeed | ; | HP-IB END

The digitizer clear instruction clears the modes set by the following mnemonics: DP, SG, CN. In addition to clearing these modes, any digitized point coordinates are cleared as well as bit 2 of the status byte.



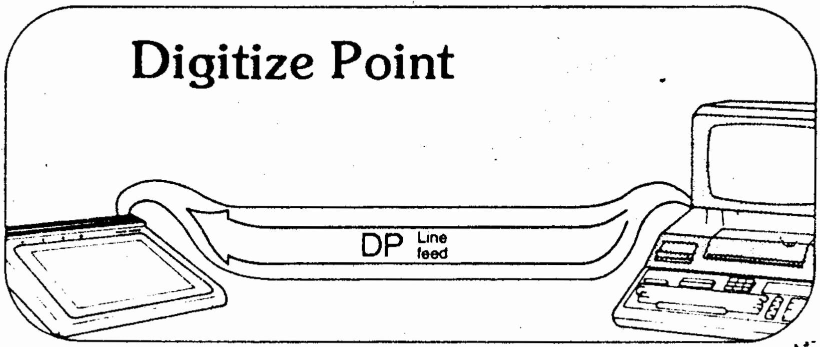
DF Linefeed | ; | HP-IB END

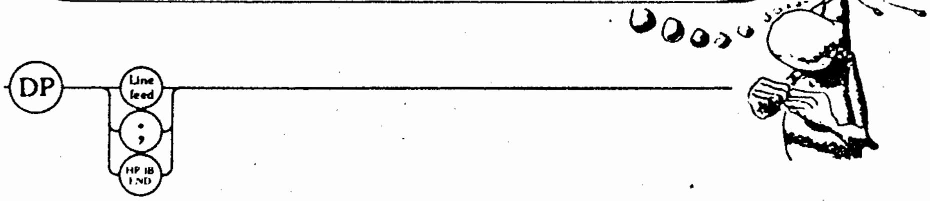
The DF instruction sets the graphics tablet to a predetermined power-on state.

The following conditions are set by the default instruction.

Condition	Default Set Value
Cursor Sample Rate	60/second
E Mask	7
S Mask	0
P Mask	0
Status Byte	16 (Bit 4 is set)
Menu Area	On
Menu Item	0
Digitizing State (CN or SG)	None
Stylus Digitize Switch	Switch Normal Mode

Digitize Point





DP Linefeed | ; | HP-IB END

The digitize point instruction prepares the graphics tablet to recognize the next pen press as a digitized point. The digitize LED is illuminated. This instruction is used without the continuous or single sampling mode. If a CN or SG mode is set the DP instruction is ignored.

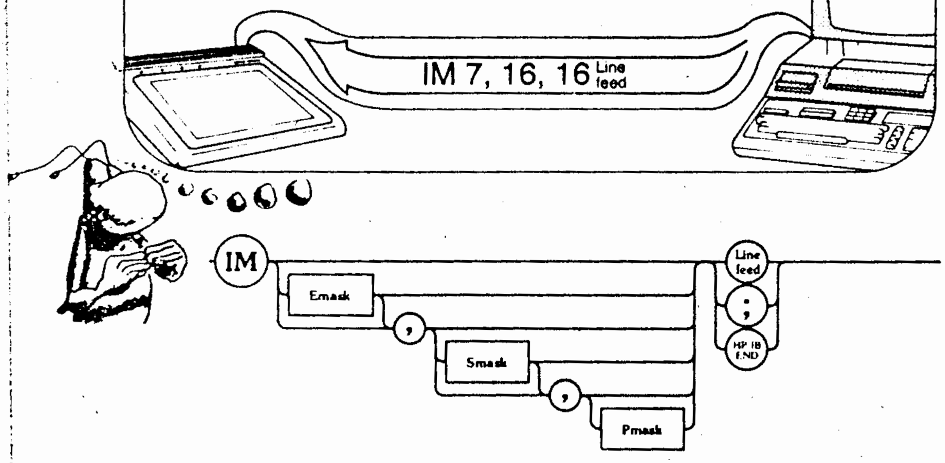
DP is a single point digitizing instruction that is compatible with the digitizing operation provided on some HP plotters. A suggested implementation using DP is shown next.

- Clear the graphics tablet using DF.
- Set the DP mode.
- Check bit 2 of the tablet's status.
- If bit 2 is set, send OD and read X,Y, and Pen data.
- If bit 2 is not set, keep checking bit 2.

Bit 2 is set when a point is digitized.

Input Masks

IM Emask, Smask, Pmask



IM[E-mask][,S-mask][,P-mask]Linefeed | ; | HP-IB END

The input mask instruction is used by your controller to selectively enable the following: the recognized errors, the status conditions that can cause a service request, and to select status conditions that cause a response from a parallel poll.

Error Mask

The summed value of the errors that you want to enable is specified. See the error mask table.

The default error mask is 7.

Error Mask

Value	Bit	Error
0	0	No Error
1	1	Instruction not recognized, instruction exceeded 45 characters, or OD sent with no digitizing mode in effect.
2	2	Wrong number of parameters
4	3	Illegal parameter value
64	7	Inconsistent Stylus Location Data

Status Mask

The S-mask value specifies the status byte conditions that can send the require service message (interface line SRQ). The S-mask value is the decimal equivalent sum of the bit values of the selected status-byte bits. See the following table.

Status Mask

Bit Values	Status Bits	Meaning
1	0	Always Clear
2	1	Always Clear
4	2	Digitize Point Available
8	3	Initialized (Completed Power On Self Test)
16	4	Ready (Completed Power On Self Test, User Self Test, or Beep Instruction)
32	5	Error
64	6	SRQ Sent
128	7	Menu Selection
256	8	Proximity
512	9	New Cursor Information Available
1024	10	Pen Switch is Pressed

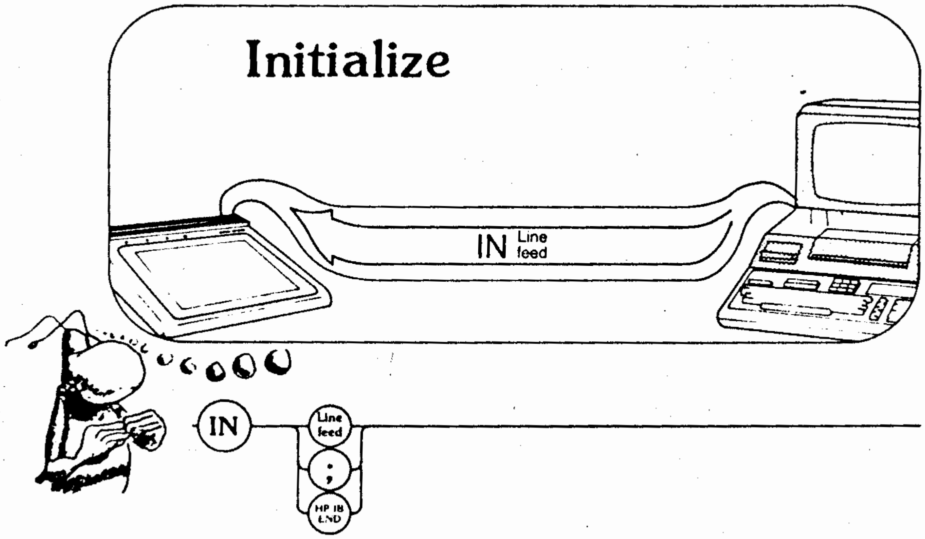
Parallel Poll Mask

The parallel poll response bit is determined through the selection of the addresses switches. See the next table. An affirmative parallel poll response is enabled by the P-mask matching the status word, and is programmed using the same techniques as the S-mask (see table above).

Parallel Poll

Decimal Value Returned	HP-IB Address
128	0
64	1
32	2
16	3
8	4
4	5
2	6
1	7
0	

Initialize



IN Linefeed | ; | HP-IB END

The initialize instruction performs the self test and then sets the graphics tablet to its power-on condition.

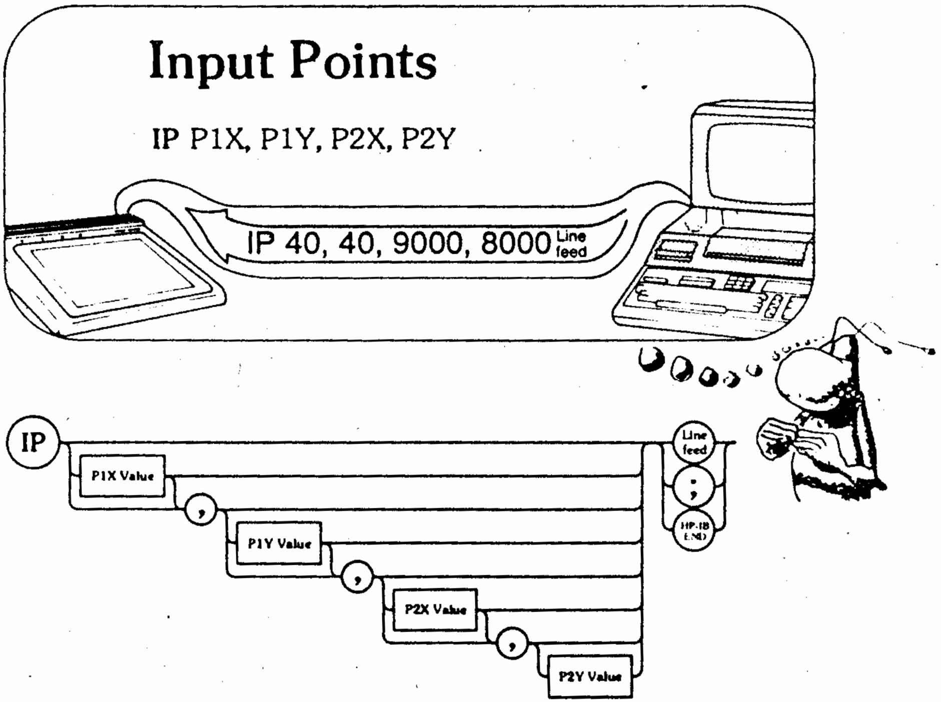
The following conditions exist after the graphics tablet is initialized:

Condition	Set Value
Sample Rate	60/second
E Mask	7
S Mask	0
P Mask	0
Status Byte	16 (Bit 4 is set)
Menu Area	On
Menu Item	0
Digitizing State (CN or SG)	None
Stylus Digitize Switch	Switch Normal Mode
P1 and P2 Values	P1 P2 400,400 11632,8340

It is recommended that you always follow the IN instruction with the DF instruction when you are using the binary data transfer mode.

Input Points

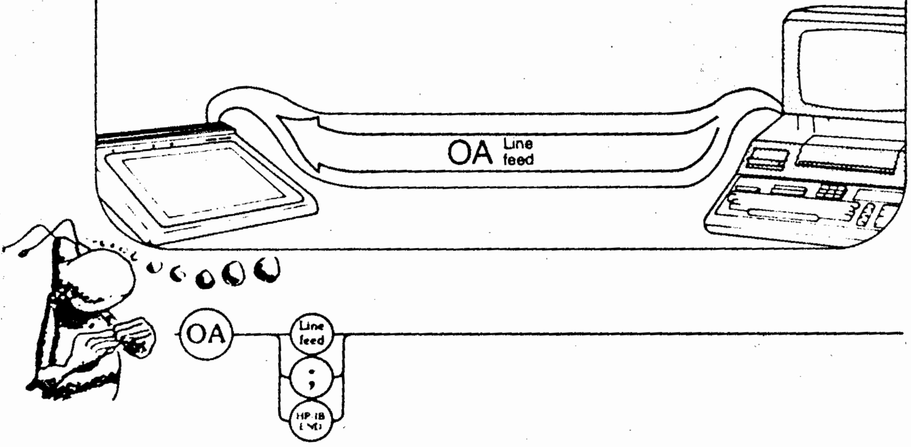
IP P1X, P1Y, P2X, P2Y



IP[P1 X][, P1 Y][, P2 X][, P2 Y] Linefeed | ; | HP-IB END

The input points instruction causes the graphics tablet to store four values specified by your controller program. These values can then be output for scaling purposes.

Output Actual Stylus Position



OA Linefeed | ; | HP-IB END

This instruction is the same as the OC. It causes the graphics tablet to output the last known X, Y, Pen, Menu Selection, Status, and Error information. See the OC instruction.

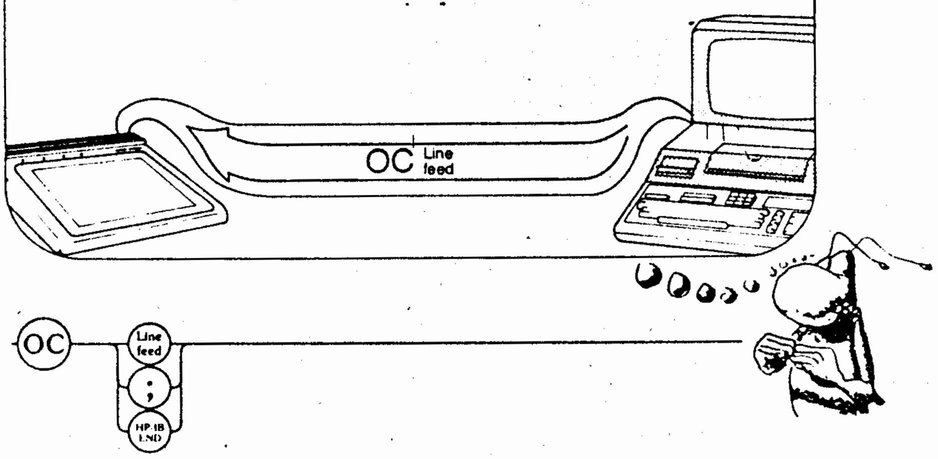
The return parameters and output format is shown next.

XXXXX	,	XXXXX	,	X,	XX,	XXXX,	XXX	CR/LF
X value		Y value		PEN	MENU	STATUS	ERROR	

A recommended sequence using the OA instruction is presented next.

- Set digitizing and digitize switch mode.
- Check bit 2 of the tablet's status.
- If bit 2 is set, send OA and read the X,Y, and Pen data.
- If bit 2 is not set, keep checking until it is set.

Output Cursor



OC Linefeed | ; | HP-IB END

This instruction sets up the graphics tablet to output the following information: X, Y, PEN, MENU, STATUS, and ERROR. The next instruction to the graphics tablet (from the controller) is expected to be a controller input instruction. It is not necessary to read all parameters into your controller. The OC parameters (output and format) is shown next.

X value		Y value	PEN	MENU	STATUS	ERROR	
XXXXX	,	XXXXX	,X,	XX,	XXXX,	XXX	CR/LF

The parameters X, Y, and ERROR are output using a variable length format. The X and Y values can have maximum character field of 5 characters and a minimum field of one character. Smaller numbers (1 or 2 digits) can possibly contain a minus sign if you are digitizing in the lower left hand corner.

5-22 OC

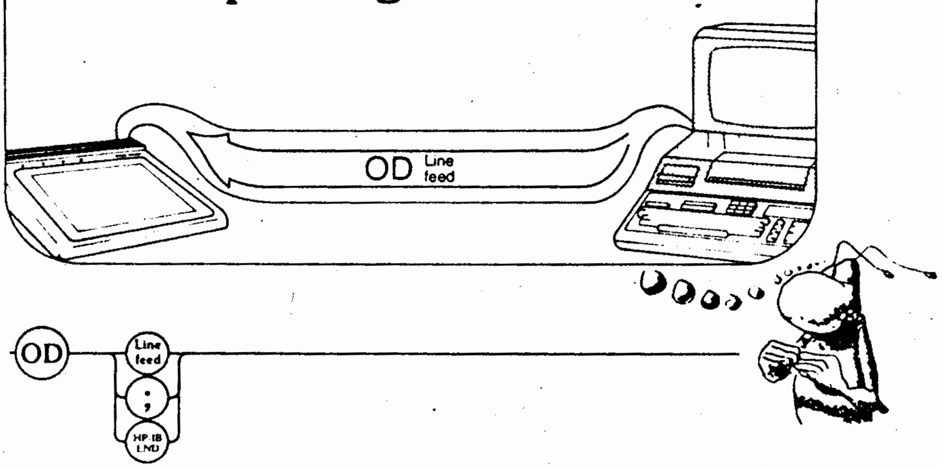
The error value can have a maximum of three characters and a minimum of 1 character.

The pen value is a fixed one character field.

The key and status parameters are fixed in length (key = 2 characters, status = 4 characters) and can contain leading zeros.

The parameters are each separated with a comma and the entire output string is terminated with a carriage return and linefeed.

Output Digitized Point



OD Linefeed | ; | HP-IB END

The output digitized point instruction readies the graphics tablet to output the known stylus position. The next instruction to the graphics tablet is expected to be a controller input instruction. The following parameters are available with the OD instruction. It is not necessary to read all the parameters into your controller.

X value		Y value		PEN	
XXXXX	,	XXXXX	,	X	CR/LF

The X and Y values are output in a variable length field. The field can vary from 5 ASCII characters down to 1 character. Digitizing in the extreme lower or left platen area can cause a minus sign to be sent over with the data.

The pen parameter is a single character field. This character will always be a one or a zero.

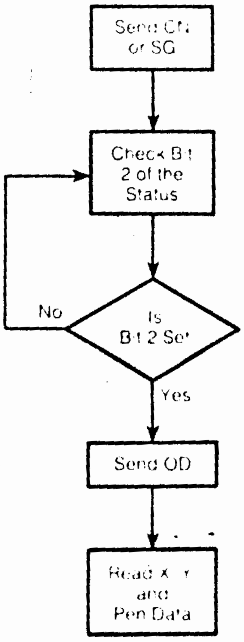
5-24 OD

When digitizing in the CN (continuous) mode, each point will have a pen parameter of one except the final (or last) point. This last point will always have a parameter of 0.

Digitizing in the SG (single) mode, the pen parameter will always be a one.

The output digitized point (OD) instruction is designed to be used with bit 2 of the status byte. The suggested implementation is shown next.

- Set digitizing mode (CN or SG).
- Check bit 2 of the tablet's status.
- If bit 2 is set, send OD and read the X, Y, and PEN Data.
- If bit 2 is clear keep checking until it's set.

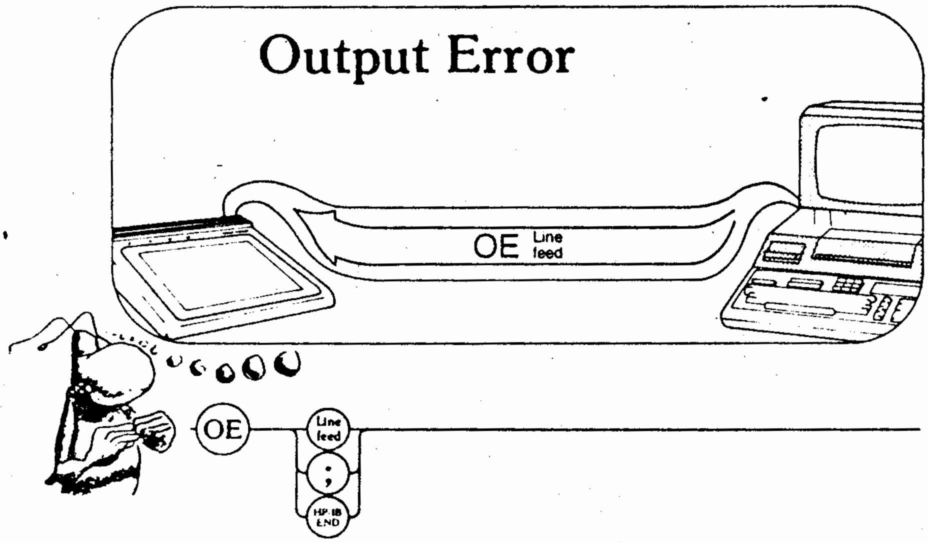


When the graphics tablet receives the CN or SG instruction, the green "Digitize" LED will light. The graphics tablet is now ready to take a point. When a point is digitized, bit 2 of the tablet's status is set to 1. To digitize a point place the stylus tip on the tablet's active surface and press enough to energize the digitize switch. Once bit 2 is set then the OD instruction is sent to the tablet and the controller can read the X, Y, and PEN data.

If OD is received by the graphics tablet and bit 2 of the status byte is not set, the graphics tablet takes control of the HP-IB control lines and stops further data communication until bit 2 is set. System I/O communication is halted until a point is digitized. It is recommended that you not try using this mode of operation if the S mask is set to generate an SRQ (interrupt) on bit 2 of the status byte.

If OD is received by the graphics tablet and a digitizing mode (DP, SG or CN) is not set, the following controller input instruction receives the following data: X = 0, Y = 0, PEN = −1. Error 1 is also generated.

Output Error



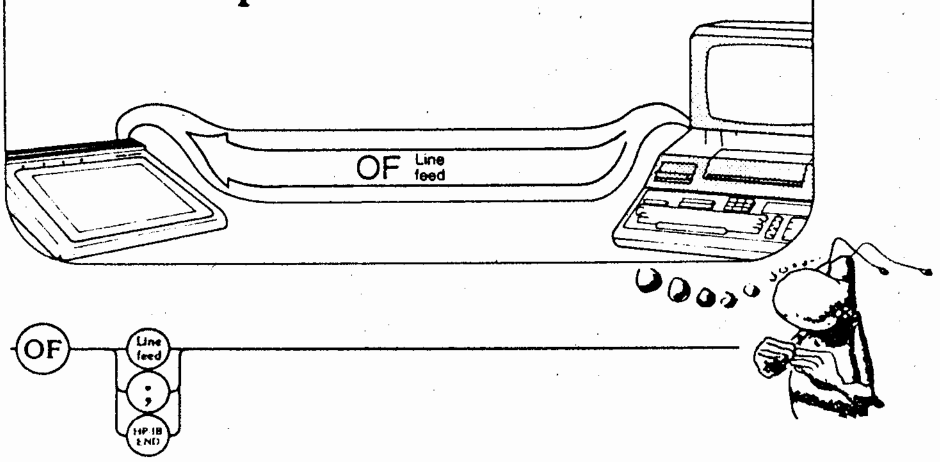
OE Linefeed | ; | HP-IB END

The OE instruction readies the graphics tablet to output its current error condition. With the next controller input instruction (addressed to the graphics tablet), this current error condition is output. The graphics tablet's errors values and meaning are listed next.

Values	Error Meanings
0	No error
1	Instruction not recognized, instruction exceeded 45 characters, "OD" received with no digitizing mode set.
2	Wrong number of parameters
3	Illegal parameter value
7	Inconsistent Stylus Location Data

When an error is generated, bit 5 of the status byte is set. Bit 5 is cleared when the graphics tablet receives the "OE" instruction. Of course, the next instruction to the graphics tablet is expected to be the controller input to receive output error data.

Output Factor



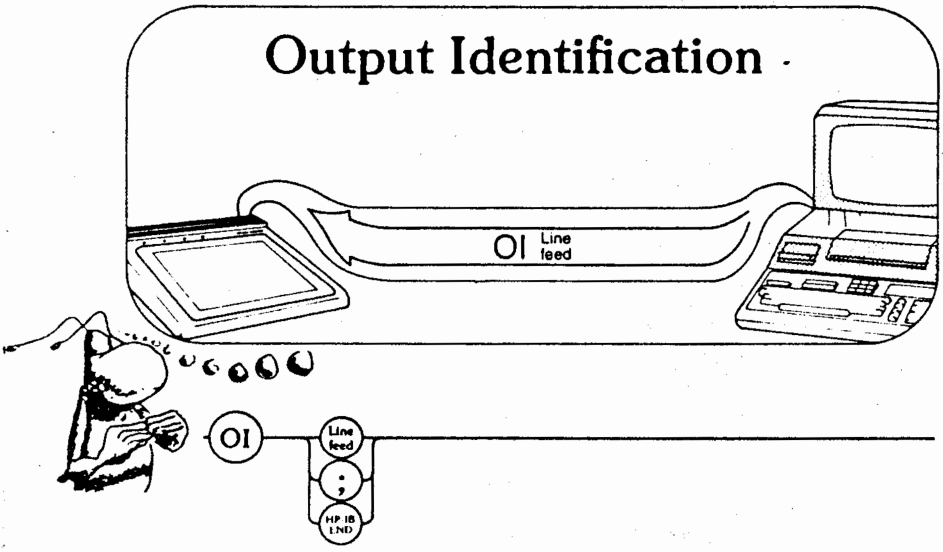
OF Linefeed | ; | HP-IB END

The output factor sets up the digitizer to output two parameters with the next controller input instruction. The two parameters represent the X and Y resolution expressed in lines/millimetres. The values are 40 and 40. The output data string is shown next.

40,40CRLF

This is the apparent resolution of the graphics tablet; however, the data is rounded internally to 10 line/millimetres.

Output Identification

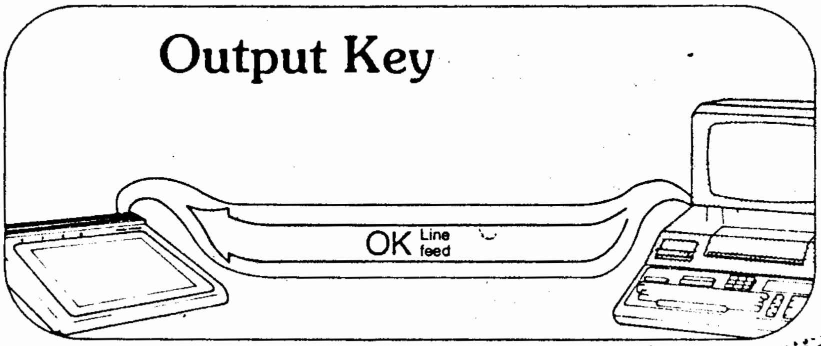


OI Linefeed | ; | HP-IB END

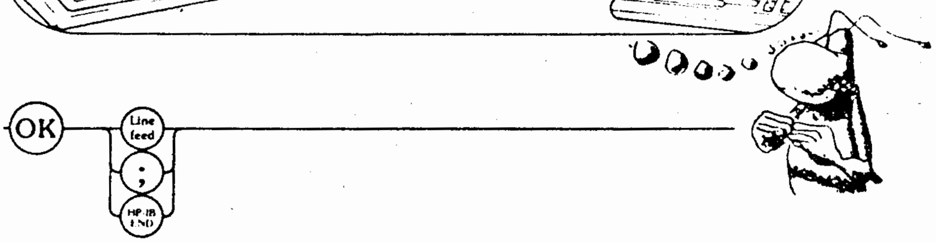
The output indentification instruction gets the graphics tablet ready to output one parameter when the next controller input is excepted. This parameter is 9111A. This instruction can be used to identify this device on a large bus system. The output data string is shown next:

9111ACRLF

Incidentally, this is the only instruction to return a non-numeric data character.



Output Key



OK Linefeed | ; | HP-IB END

This instruction sets the graphics tablet to output the selected menu value upon the receipt of the next controller input instruction. When you select a menu square (energizing the digitize switch, contained within the stylus, within a square area marked on the upper section of the platen), bit seven of the status byte is set. If you are checking bit seven (via program control) you should send an OK instruction once bit seven is set. And this is followed with a controller read instruction.

The OK instruction clears bit seven of the status and readies the graphics tablet to output a value associated with the selected square. The following table shows the value associated with each predefined menu square.

5-30 OK

Menu

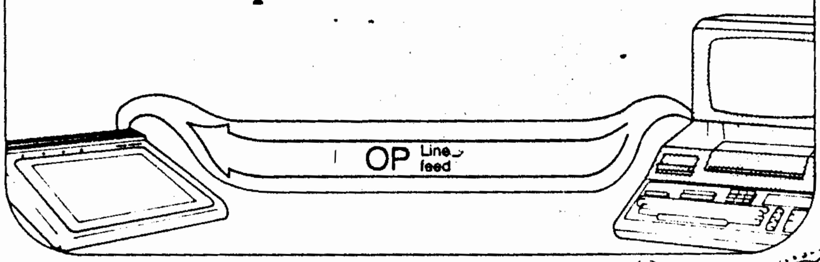
Output Value	Menu Square Selected
1	1
2	2
4	3
8	4
16	5
32	6
64	7
128	8
256	9
512	10
1024	11
2048	12
4096	13
8192	14
16384	15
32768	16

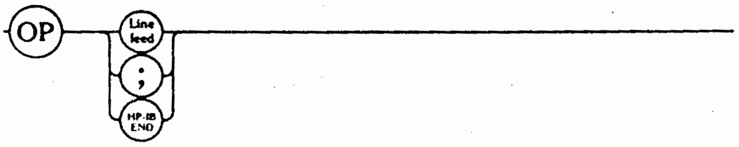
Energizing the digitizing switch in a square menu area will set a value (see the previous table). Energizing the digitize switch in the same square clears the previously set value and bit 7 of the status byte.

For compatibility reasons with the 9874A digitizer this instruction is allowed. When used in this manner this instruction would normally be followed by the SK0 instruction in order to clear the menu value and menu light.

If compatibility is not a concern, it is recommened that you use the RS instruction which is more suited for the graphics tablet.

Output Points





OP Linefeed | ; | HP-IB END

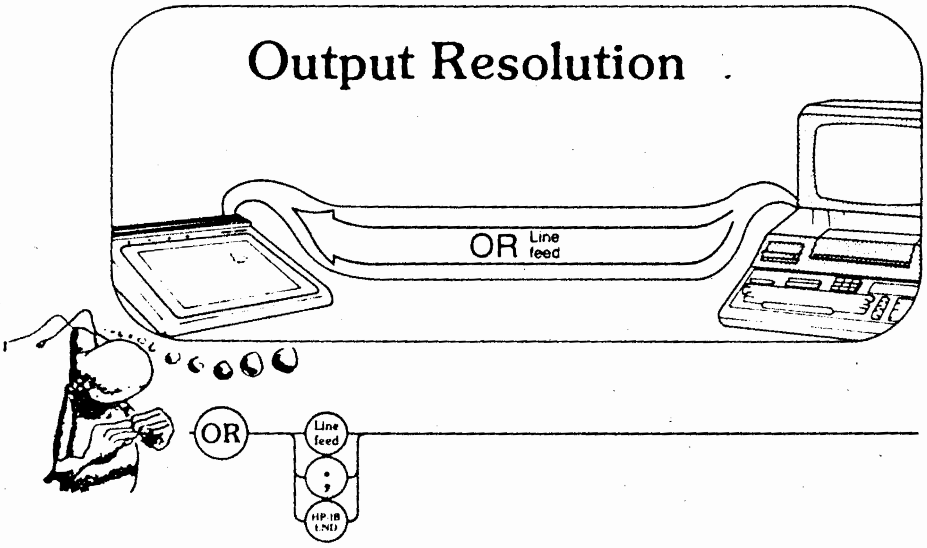
This instruction outputs the scaling points. P1 and P2. These are the same coordinates input with IP (Input Points). The graphics tablet outputs four points with the next controller input instruction. The output data string is shown next.

$$\underbrace{XXXXX}_{P1\,X},\ \underbrace{XXXXX}_{P1\,Y},\ \underbrace{XXXXX}_{P2\,X},\ \underbrace{XXXXX}_{P2Y}\ CR\ LF$$

Each value can vary from 5 characters down to 1 character; the string contain comma delimeters and a CR/LF as the string terminator.

Incidentally, negative values are allowed if they are input with the IP instruction.

Output Resolution



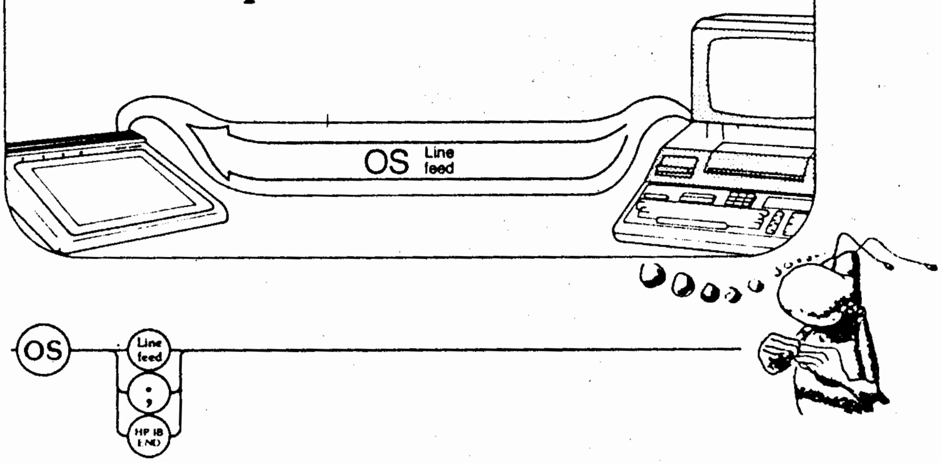
OR Linefeed | ; | HP-IB END

The output resolution instruction sets up the graphics tablet to output two parameters with the next controller input instruction. The two parameters represent the X and Y resolution expressed in lines per millimetres. The values are .025 and .025. The output data string is shown next.

.025,.025 CR LF

The apparent resolution of the graphics tablet is .025 millimetres; however, the data is rounded internally to the nearest .1 millimetre. For scaling purposes, consider all data transferred to and from the tablet as representing .025 millimetre units.

Output Status



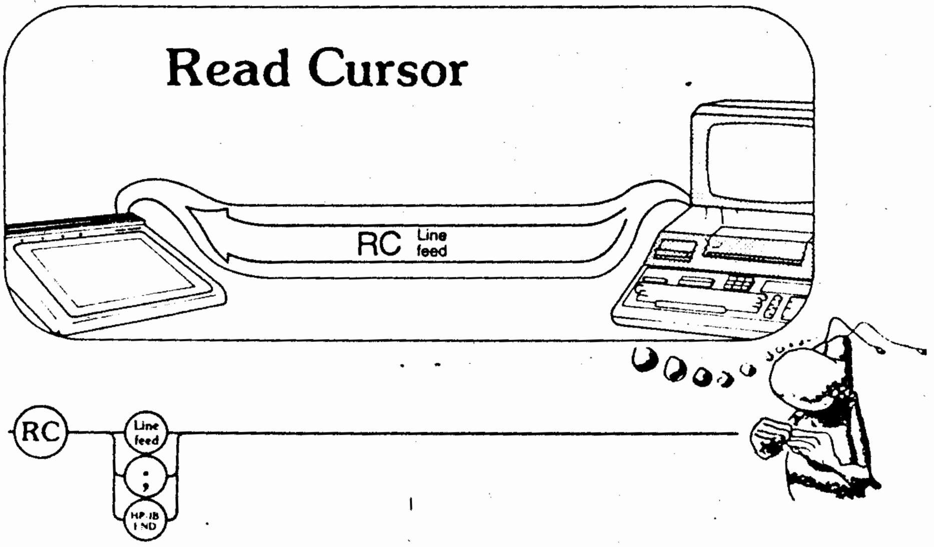
OS Linefeed | ; | HP-IB END

This instruction sets up the graphics tablet to output the decimal sum of the bits which are set in the status word when it receives the next controller input instruction. The status word consists of eleven bits (0 through 10). Different bits are set corresponding to the graphics tablet's internal condition. The summed total weighted value of the set bits is output. See the following status word table.

The Status Word

Weighted Value	Bit	Set Bit Meaning	Instruction to Clear Bit
1	0	Always Clear	
2	1	Always Clear	
4	2	Digitize Point Bit - This bit is set when a point is digitized.	OD, DC, DF, IN
8	3	Initialize Bit - Completed Power on Self Test.	OS and DF
16	4	Ready Bit - Completed Power Self Test, User Interaction Self Test, and Beep.	Initiating the Power on Self Test, User Interaction Self Test, or Beep Instruction.
32	5	Error Bit - Error Detected.	OE, DF, and IN
64	6	Service Request Bit - SRQ Generated.	Clear SRQ
128	7	Softkey Bit - Menu Item Selected.	OK, RS, DF, and IN
256	8	Proximity Bit - Pen Tip within approximately ¼ inch of the active platen area.	Remove the Pen Tip from the active Platen area.
512	9	New Cursor Position Bit - The Buffers containing cursor positional data are updated.	Binary Read, DF, IN, and OC
1024	10	Pen Press Bit - Pen is pressed against the active platen area.	Lift Pen

Read Cursor



RC Linefeed | ; | HP-IB END

This instruction sets up the graphics tablet to output the following information X, Y, PEN, STATUS, and ERROR. Upon the next controller input instruction, the following parameters are read into your controller. The RC parameters (format) are shown next.

X value		Y Value		PEN		Key		STATUS		ERROR	
XXXXX	,	XXXXX	,	X	,	XX	,	XXXX	,	XXX	CR LF

The parameters X, Y, and error are output using a variable length format. The X and Y values can have a maximum character field of 5 characters and a minimum field of one character. Smaller numbers (1 or 2 digit) can possibly contain a minus sign if you are digitizing in the lower left hand corner. The error value can have a maximum of three characters and a minimum of one character.

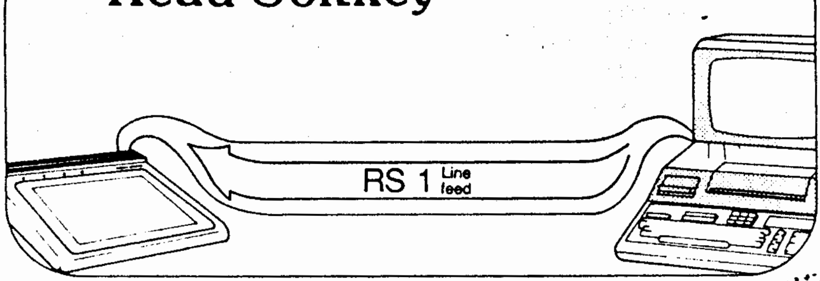
The pen value is a fixed one character field.

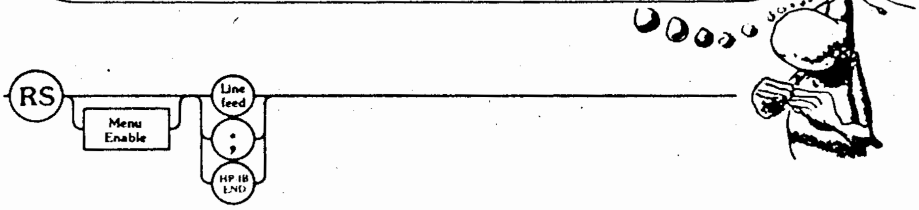
5-36 RC

The key and status parameters are fixed in length (key = 2 characters, status = 4 characters) and can contain leading zeros.

The parameters are separated with comma delimeters and the graphics tablet terminates all outputs with a carriage return and linefeed characters.

Read Softkey





RS [menu enable] Linefeed | ; | HP-IB END

This instruction set up the graphics tablet to output a decimal number corresponding to the selected menu square. This number is output with the next controller input instruction. See the following table for the value output and its corresponding menu square.

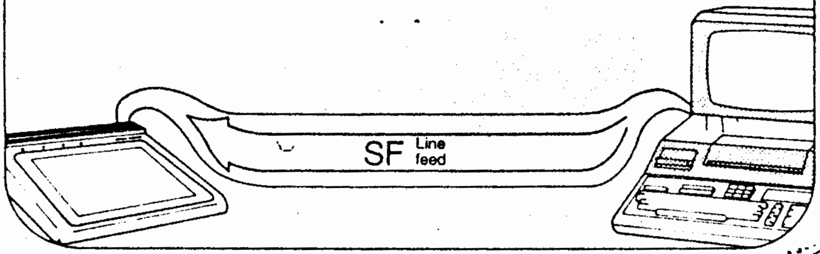
5-38 RS

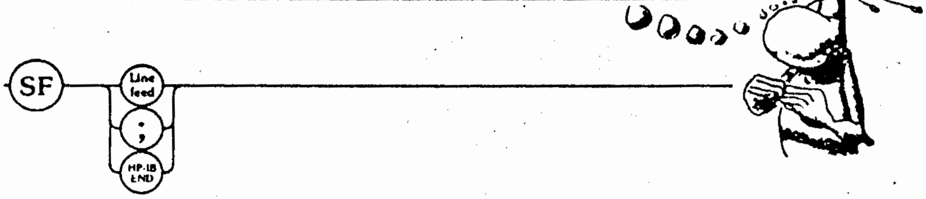
Menu

Output Value	Menu Square Selected
0	No Square Selected
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16

The parameter allowed with the RS instruction can be a one or a zero. A one enables the menu area of the platen, whereas a zero disables the menu area. The output value shown in the previous table is available in any case. RS clears the menu number and status bit 7. Also, the menu value is cleared after it is read.

Switch Follow

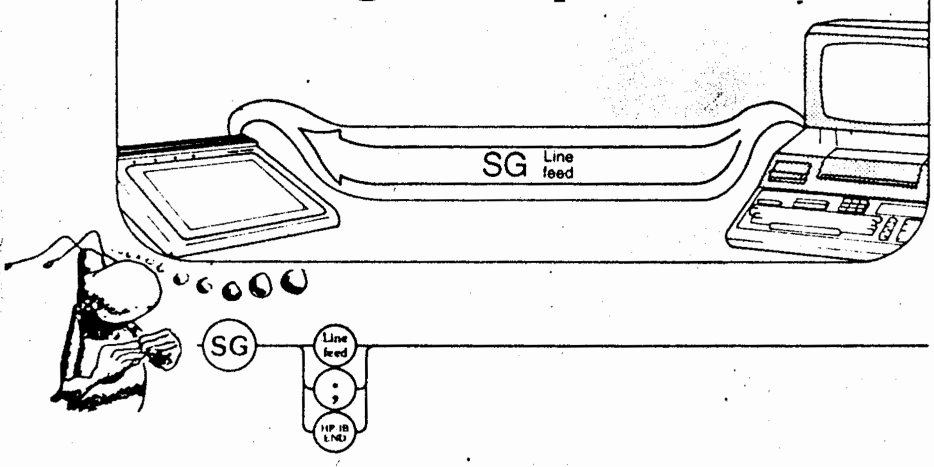




SF Linefeed | ; | HP-IB END

This instruction places the digitize switch (internal to the stylus) into a press to digitize mode. Points are digitized only while the pen is pressed to the platen. This instruction is only used in the continuous sampling mode. The last point sent out using the OD instruction will have a pen parameter value of zero.

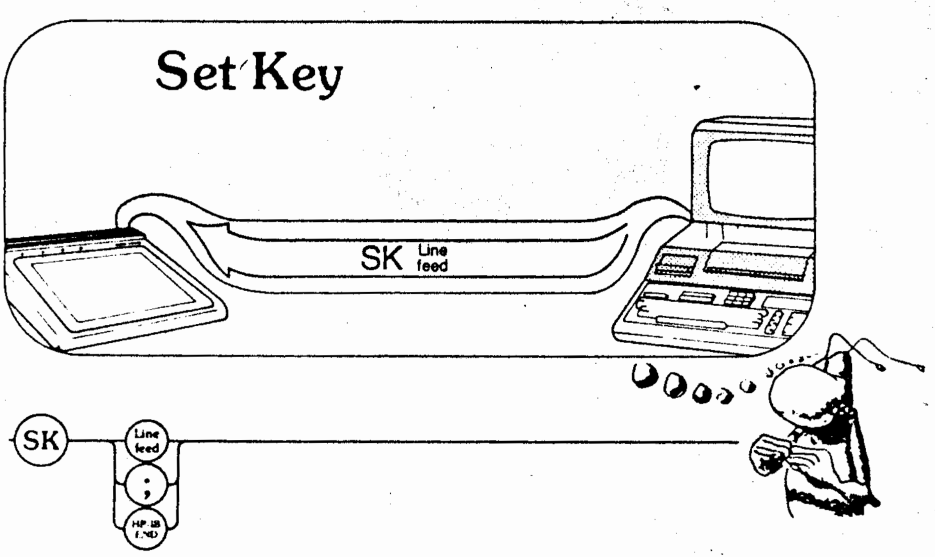
Single Sample Mode



SG Linefeed | ; | HP-IB END

The SG instruction sets the graphics tablet's single sample mode. When this mode is set, the digitize LED above the active area is illuminated and the digitize switch within the stylus is armed. When the digitize switch is energized (pressing the pen tip onto the active area) a coordinate point is stored in the tablet. This process sets bit 2 of the status byte. The data is transferred to your controller via the OD instruction.

Set Key

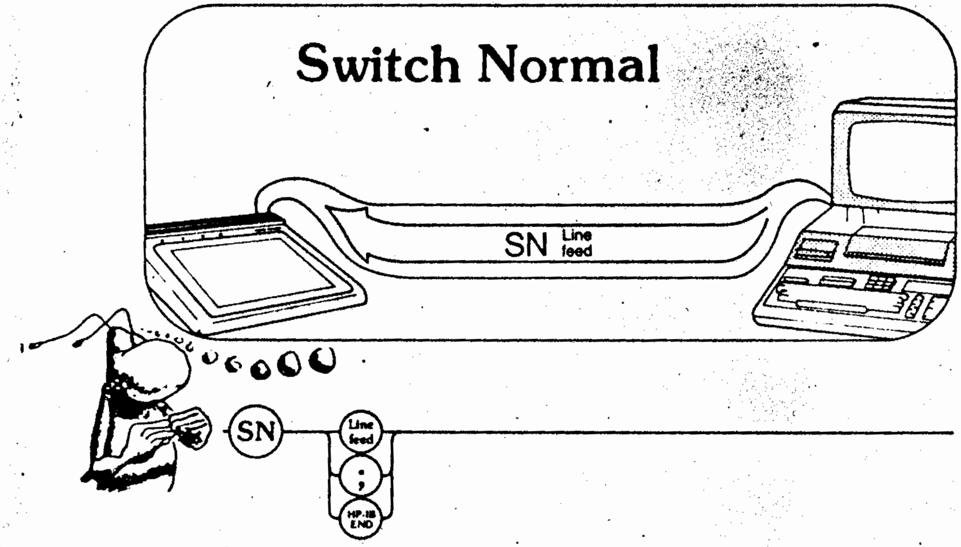


SK Linefeed | ; | HP-IB END

The set key instruction clears any previously picked menu value and also bit seven of the status byte. This instruction is normally used following an "OK" instruction. It would be used to clear the graphics tablet between menu selections.

SK [value] is allowed for compatibility with the HP graphics devices. The [value] is read into the graphics tablet and discarded.

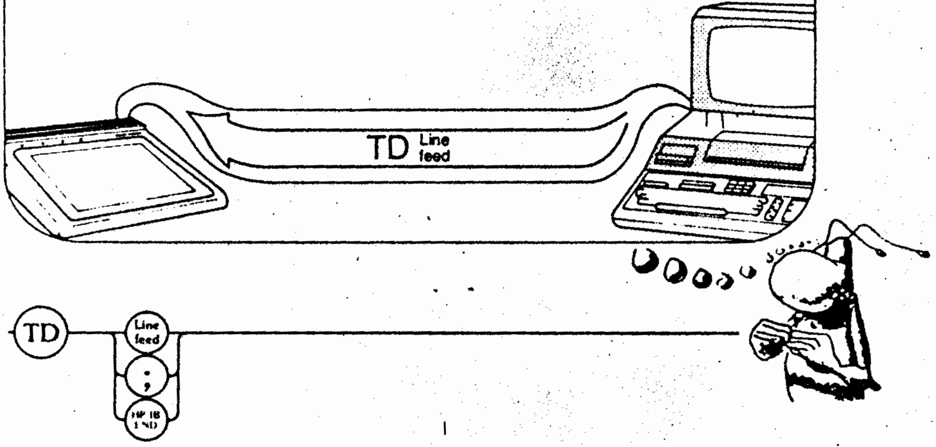
Switch Normal



SN Linefeed | ; | HP-IB END

SN sets the switch normal (default) mode of the digitize switch. The first pen press starts the digitizing process; the next pen press stops the digitizing. This instruction is used in the CN (continuous sampling mode) and the last coordinate value output (with OD) will have a pen parameter of zero.

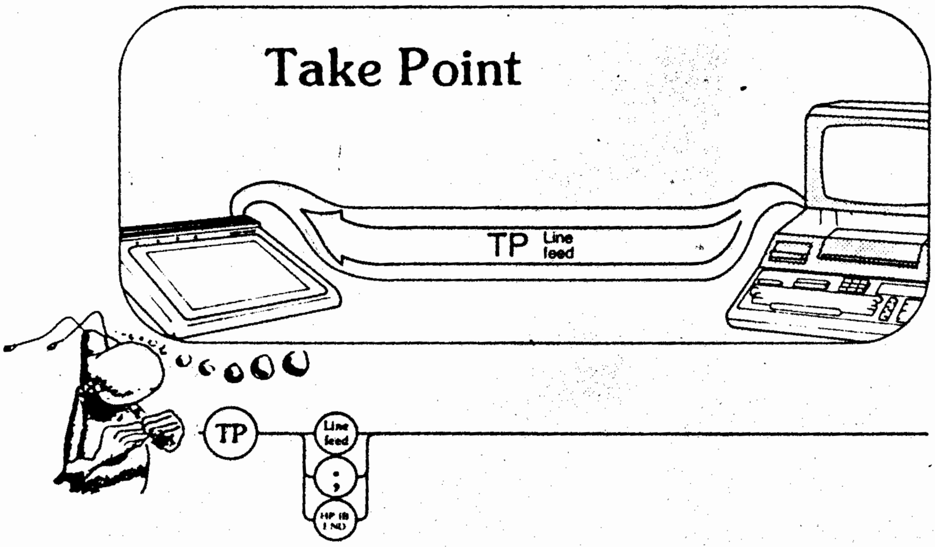
Test Digitizer



TD Linefeed | ; | HP-IB END

This instruction sets up the user interaction self test mode. You must digitize the dot on the active surface, which is checked for accuracy. If any instruction is received by the graphics tablet while it is waiting for the digitized point, the self test is aborted. See the section titled "Errors and What They Mean" for more informtion on the Self Tests.

5-44 TP



TP Linefeed | ;-| HP-IB END

The take point instruction simulates the press of the digitize switch. This is done regardless of the actual pen position (pen down or pen up).

This instruction can be used to force a point to be digitized in the SG mode or to terminate digitizing a string of data in the CN (Switch Normal) mode.

No-operation Instructions

The following instructions are accepted without error, but cause no action within the graphics tablet.

AN
AT
AV
CC
DD
DR
IW Note: OW is not allowed
LB
LT
PA
PC
PD
PG
PU
RV
SL
SP
SR

Binary Data Transfer

A binary data transfer mode is available on the graphics tablet. This binary transfer is the default mode of operation and is available using any of the bus addresses.

Binary transfer is initiated with your controller doing a read operation. This read operation must follow the hardware guidelines of the IEEE 488-1978 Standard. The binary data placed on the bus is 6 bytes of data. The first 2 bytes of data is the binary representation of the stylus X position. This is followed with 2 bytes of Y position and 2 bytes of the tablet's current status. Each 2 bytes is a two's complement binary number sent with the most significant bit first. Another controller read will initiate another output of binary data. For most efficient timing your controller read cycles should approximately match the cursor update cycles. See the CR instruction.