9111T Graphics Tablet User's Manual

T. Ryan
J. Scheetz

Part No. 09111-90050
Microfiche No. 09111-99050

Dec. 23, 1980

Hewlett-Packard Colorado Springs Division
1900 Garden of the Gods Rd., Colorado Springs, Colorado 80907

Copyright by Hewlett-Packard Company 1980

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Computer Museum

Table of Contents

Table of Contents (Cont'd)

Chapter 1
Introduction

General

This manual includes information relating to the use of a 9111T with a 1350A/S. It is not intended to be all-inclusive, and should be used with the 9111A and 1350A manuals. This manual, the Operating and Service manuals for the 1350A and 9111A, appropriate X-Y display and plotter (if used) along with the Operating and Programming manuals for the 1350A and 9111A provide a complete system documentation package.

Throughout this manual it is assumed that the 9111T/1350A is being used with at least one X-Y display. A means of generating hardcopy may be included by adding a plotter. To acheive a thorough understanding of the capabilities of the 9111T/1350A system, this manual is divided into several interrelated areas: system usage, programming, and program examples.

The manual is divided into five chapters as follows:

- **Chapter 1, Introduction**

  Manual organization, concepts of interactive graphics, applications, and advantages of the 9111T/1350A system.

- **Chapter 2, System Overview**

  9111T/1350A interaction.

- **Chapter 3, General Programming**

  9111T instructions, syntax, programming considerations and use of the menu keys.

- **Chapter 4, Hardware**

  System connections and considerations.

- **Chapter 5, Programming Examples**

  Program examples and segments for functions in HPL, BASIC.

The material in this manual is meant to aid in implementing the 9111T/1350A on any computer or controller system. The system must support HP-IB (the HP implementation of IEEE 488) with hardware interface and software device handler.

What is a 9111T?

The 9111T is a 9111A with an enhanced command set to allow it to be used with a 1350A. As such, it becomes the interface in a human/computer interactive system.

Interactive Graphics

An interactive graphics system is ideal for a number of applications. One is in the area of design. Design graphics involves the generation, manipulation, and graphic portrayal of an object or objects. This graphic object is usually associated with non-graphic information, such as calculations or software, which is stored in the computer. Both generation and manipulation are via the interactive device, in this case the 9111T. The level of sophistication for these two operations depends on the level of sophistication of the driving software.

At some point in the design cycle, the need will arise to manipulate existing data. An interactive system simplifies this task greatly. The existing design can be quickly recalled and changed. This might involve deletion or addition of whole sections, which would require large amounts of time to change by hand or by altering a program.

An interactive system involving the 9111T and 1350A gives the ability to create or change by selecting symbols or functions which the programmer has defined for the application. A symbol is firmware defined because it is frequently used, and there is little or no need to rotate or rescale the part. Rather than having to draw an often used part each time, the representation can be defined, stored, and then recalled by the user. These symbols can then be combined to make larger and more complex blocks which can also be moved about or combined with others to create "macros". A macro can be thought of in terms of a combination of symbols or a specially designed symbol which is either not used often or one that requires rescaling. All of this combining and building can be done without having to deal with the individual symbols which make up the whole.

The 9111T utilizes some of the benefits of distributed intelligence by talking directly to the 1350A on some functions. This helps to free the host controller for other tasks, since the controller is not needed to "babysit" the system. Not only does this speed up the updating process on the local level, it allows several interactive systems to be used on the same controller. This architecture eliminates the need for an expensive controller to handle each interactive system.

System Capabilities

Application defined symbols can greatly simplify design tasks. These symbols are configurable by the users software, making them appropriate to different applications. A few of these are:

— electrical/electronic schematics

— hydraulics

— architecture

— radar

— instrumentation

HP-IB

Since the 9111T is an HP-IB device, it can be used with any controller that conforms to IEEE 488. The two letter mnemonics that comprise the command set (HPGL or Hewlett-Packard Graphics Language) are related to the command they represent. Since the 9111T is an HP-IB device, there is no provision for use with a 16-bit interface or RS-232C.

Tablet Advantages

The 9111T has certain advantages over other types of interactive devices.

— user programmable menu keys

— stylus is less fatiguing to use than a light pen

— selectable cursor (alphanumeric character or full screen)

— menu keys and functions can be labled by use of overlays

— limited digitizing capabilities

System Advantages

When combined with a controller, X-Y CRT display, and plotter, the 9111T/1350A system yields some additional advantages:

— easily programmable with HPGL (Hewlett-Packard Graphics Language) via HP-IB

— gives the ability to change and edit information quickly

— hard (paper) or softcopy (CRT)

Chapter 2
System Overview

Concepts

The 9111T/1350A system gives both the operator and programmer a great deal of control, since the 9111T is designed to carry on several functions independent of the controller. For instance, a cursor can be generated on screen by outputting the cursor's (tablet stylus) starting location and then updating the 1350A as the cursor is moved. The 9111T can update the 1350A 60 times a second, so cursor movement appears smooth and continuous. This function's speed is aided by the lack of controller interaction. The 9111T sends the X-Y coordinates to the 1350A in units it can understand. This is done by assigning the 9111T as HP-IB talker and the 1350A as the listener until there is need for the controller to be involved. Data transfer is speeded because the controller doesn't need to "massage" data from the 9111T into a form compatible to the 1350A; The 9111T talks directly to the 1350A. This frees the controller for other functions until needed by the 9111T. When needed, the controller can reconfigure the bus, set itself as talker, and respond to a service request from the tablet. When finished, the controller can reestablish the tablet as talker, the 1350A as listener again, and go about other functions.

Computer Museum

Chapter 3
General Programming

Introduction

This chapter contains a discussion for programming of the 9111T Graphics Tablet in relation to the 1350A. The first section lists formal programming rules. Subsequent sections contain information on programming considerations and use of the soft (menu) keys.

Commands

Table 3-1 contains a list of 9111T special commands for use with the 1350A.

Note

These commands are in addition to the 9111A commands. Appendix A gives a complete list of 9111A/T commands.

EE0

Command EE0 causes the 9111T to output X,Y coordinates of the current stylus position on the tablet to the 1350A. When the command

Table 3-1. 9111T Commands and Functions

ASCII COMMAND	SYNTAX	RANGE	FUNCTION
EE0		13-1010(X) 13-1010(Y)	Cursor. Can move alphanumeric characters around the screen that have been placed in locations 2 and greater in the 1350A memory. The 9111T updates location 1 in the 1350A memory.
EE1		0-1020(X) 0-1020(Y)	Single or double rubber band. For single, the fixed X,Y point must be stored in location 0 in the 1350A memory. For double, the two fixed points must be placed in locations 0 and 2 of 1350A memory. The 9111T updates memory location 1.
EE2	X',Y'	0-1020(X) 0-1020(Y)	Forced horizontal single rubber band. The Y value given to the 1350A by the 9111T is always the same as that specified in the EE command. The 9111T updates location 1 in the 1350A.
EE3	X',Y'	0-1020(X) 0-1020(Y)	Forced vertical single rubber band. The X value given to the 1350A by the 9111T is always the same as that specified in the EE command. The 9111T updates location 1 in the 1350A.
EE4	X',Y'	0-1020(X) 0-1020(Y)	Rubber-banded rectangle. Start point is specified by X',Y' and must be placed in locations 0 and 4 in the 1350A. X1-X3 and Y1-Y3 are the moving corners and are supplied by the 9111T to locations 1, 2, and 3 in 1350A memory.
EE5	S (0-1020)	0-1020(X) 0-1020(Y)	Drawn cursor. Each arm of the cursor is S units wide as specified by the programmer. S value of 1 gives a full-screen cursor. No information need be placed in the 1350A memory by the programmer. 9111T updates locations 1-4 in the 1350A.

is given, the 9111T must be configured as the bus "talker" and the 1350A as the "listener". The tablet then enters coordinates directly into the 1350A memory (location 1) at 60 times per second without controller interaction. The 9111T sends the following command string to the 1350A:

```
FL1,:PA(X),(Y);lf
```

Along with the X,Y information, the 1350A should move something around on the CRT as a cursor character. This must be specified by the programmer and entered into location(s) 2 and greater in the 1350A memory before the EE0 command is sent to the tablet. A double-bright cursor character can be generated by duplicating the cursor characters in consecutive memory locations in the 1350A (see figure 3-1). Cursor characters should be of the "centered" type, ones that do not automatically cause the 1350A to advance the beam to the next character starting point. See 9111A Appendix for a list of available ASCII

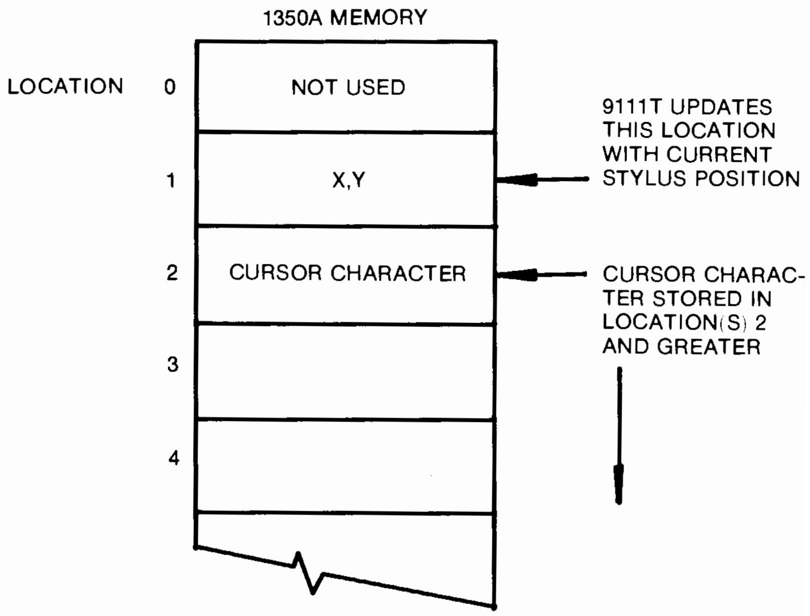

1350A MEMORY
LOCATION
0
NOT USED
1
X,Y
9111T UPDATES THIS LOCATION WITH CURRENT STYLUS POSITION
2
CURSOR CHARACTER
CURSOR CHARACTER STORED IN LOCATION(S) 2 AND GREATER
3
4

Figure 3-1. 1350A Memory Configuration When Using EE0 Command.

characters. Alphanumeric characters can be used to generate a bright cursor if a backspace command (decimal 8) is stored in memory locations between each of the characters. The working bounds of the 1350A are automatically changed to 13-1010 in both the X and Y directions when using EE0. This 12 unit buffer keeps cursor characters from being distorted on the CRT when using characters up to size 2. Consequently, a point on the CRT which is in the X-Y range of 1-12, 1010-1020 cannot be reached with the EE0 cursor.

After putting the 1350A pen down and drawing a cursor character, remember to lift the pen again or a line will be drawn from the old cursor position to the new. This happens because the 1350A scrolls through its memory sequentially (refer to figure 3-1). With the EE0 command, notice the updated cursor position is stored in location 1. The 1350A receives a PA (Plot Absolute) command from the 9111T along with the X, Y information and moves the beam to that point on the CRT. The 1350A then goes to location 2, where the cursor character is stored, and draws that character. If there are more cursor characters in locations 3 and greater, they are drawn at the same point. When the 1350A has drawn all of the cursor characters, the pen must be lifted (PE0) before the 1350A starts at location 1 again, or a line will be drawn from the current point on screen to the next updated point, which is being placed in location 1 by the 9111T at 60 times a second.

EE1

When an EE1 command is given to the 9111T, the tablet outputs the following string to the 1350A:

```
FL1,:PA(X),(Y);lf
```

This stores the moving point of the rubber-banded line in location 1 of the 1350A memory. If generating a single rubber band line, that is one end fixed, the other end moves with the 9111T stylus, location 0 in the 1350A memory is left empty. When generating a double rubber band line, i.e. both ends fixed, middle of the line moves with the 9111T stylus, one fixed point must be stored in location 0 while the other is kept in location 2 (see figure 3-2).

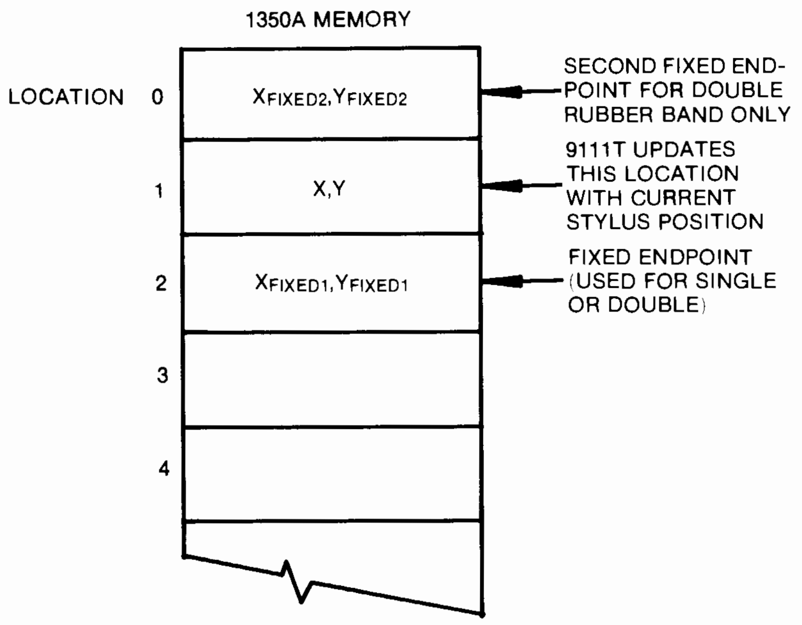
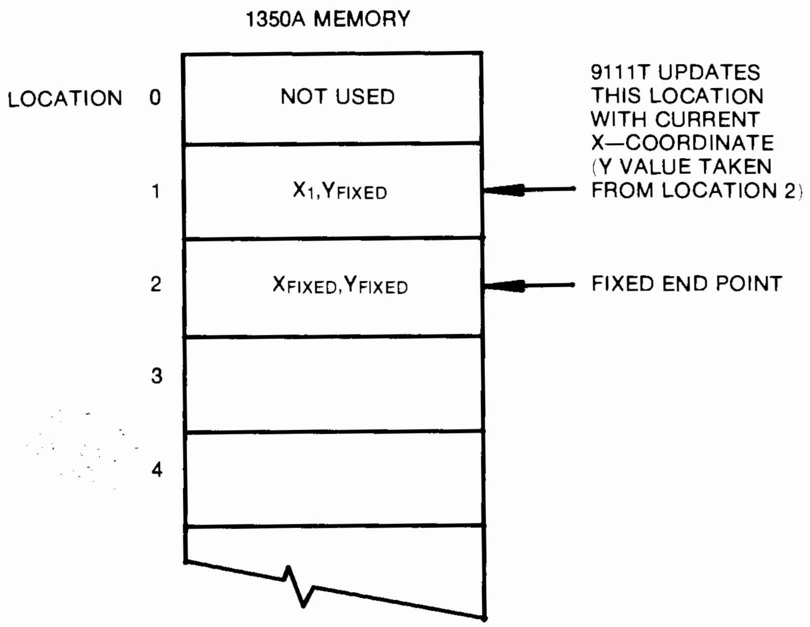

1350A MEMORY
LOCATION 0
1
2
3
4
XFIXED2,YFIXED2
X,Y
XFIXED1,YFIXED1
SECOND FIXED END-POINT FOR DOUBLE RUBBER BAND ONLY
9111T UPDATES THIS LOCATION WITH CURRENT STYLUS POSITION
FIXED ENDPOINT (USED FOR SINGLE OR DOUBLE)

Figure 3-2. 1350A Memory Configuration When Using EE1 Command.

It is possible to add an alpha cursor to the end of a single rubber band line by putting cursor characters in locations 2 and 3 of the 1350A. An alpha character can be placed in the middle of a double rubber band line by placing characters in locations 2 and 3 and writng the second fixed endpoint into location 4.

EE2

The EE2 command allows a single rubber band line (fixed point one end, other end follows tablet stylus) to be forced horizontal; that is, the line drawn on screen will always be horizontal, with no vertical movement. This means that the only value that changes in the 1350A memory is the X value sent by the 9111T to memory location 1. The Y coordinate of the stylus position on the tablet is taken to be the same as that provided



1350A MEMORY
LOCATION 0
NOT USED
1
X1,YFIXED
2
XFIXED,YFIXED
3
4
9111T UPDATES THIS LOCATION WITH CURRENT X—COORDINATE (Y VALUE TAKEN FROM LOCATION 2)
FIXED END POINT

Figure 3-3. 1350A Configuration When Using EE2 Command.

by the programmer in location 2 of the 1350A memory (refer to figure 3-3). When given an EE2 command, the 9111T sends the following string to the 1350A:

```
FL1,:PA(X),(Y);lf
```

EE3

The EE3 command works the same as the EE2 command except that the line if force vertical instead of horizontal. The 9111T takes the X coordinate provided in location 2 and uses it when outputting the current stylus position. The command string sent to the 1350A is the same as for EE2.

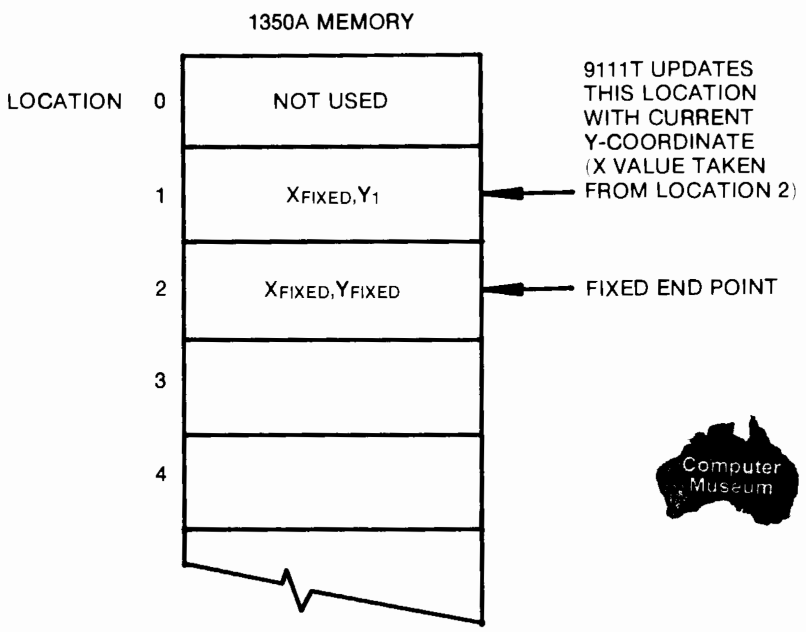

1350A MEMORY
LOCATION
0
NOT USED
1
XFIXED,Y1
2
XFIXED,YFIXED
3
4
9111T UPDATES THIS LOCATION WITH CURRENT Y-COORDINATE (X VALUE TAKEN FROM LOCATION 2)
FIXED END POINT

Figure 3-4. 1350A Memory Configuration When Using EE3 Command.

EE4

The EE4 command produces a rubber-banded rectangle. A rectangle is defined by 5 points, where point number 1 and point number 5 are the same. The start/end point (fixed corner) of the rectangle is specified by the programmer and its coordinates are placed in locations 0 and 4. The 9111T then generates the three moving corners (points) and updates them into locations 1 through 3 of the 1350A memory (figure 3-5). The string sent to the 1350A is:

```
FL1,:PA(X1),(Y1);(X2),(Y2);(X3),(Y3);lf
```

where (X1,Y1), (X2,Y2), (X3,Y3) are the three moving corners of the rectangle.

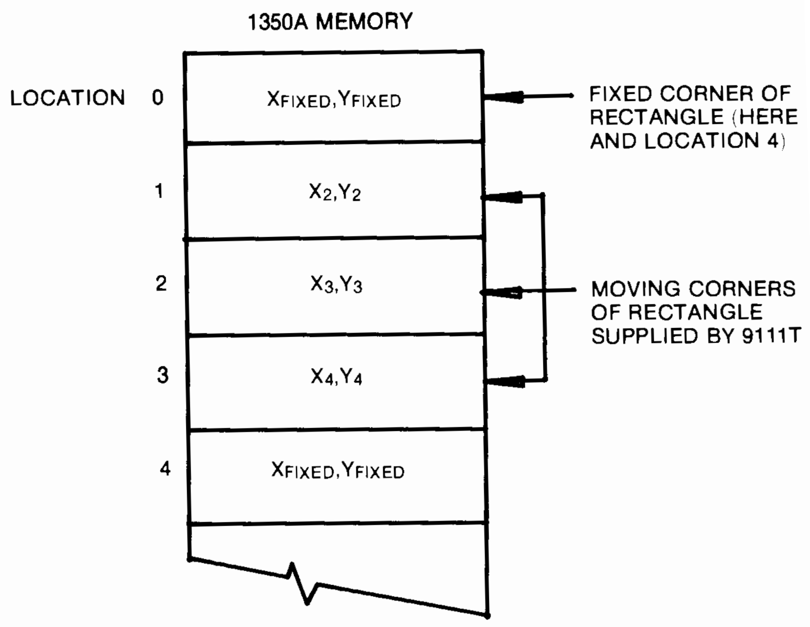

1350A MEMORY
LOCATION 0
1
2
3
4
XFIXED,YFIXED
X2,Y2
X3,Y3
X4,Y4
XFIXED,YFIXED
FIXED CORNER OF RECTANGLE (HERE AND LOCATION 4)
MOVING CORNERS OF RECTANGLE SUPPLIED BY 9111T

Figure 3-5. 1350A Memory Configuration When Using EE4 Command.

EE5

The EE5 command generates a drawn cursor of width and height 2S where S gives the length of one arm. Thus a cursor any size from a single dot (S = 1) to full screen (S = 1020) can be specified. The 9111T updates locations 1-4 (figure 3-6). Generally, the aesthetics of the EE0 cursor are better, but the EE5 cursor allows the operator to access any portion of the CRT within bounds of the 1350A. (The EE0 cursor is kept 12 units from the border to prevent character distortion). If S is not specified, a full screen cursor is drawn. Command string sent to the 1350A is:

```
FL1,:PE0,:PA(X1),(Y1);:PE1,:PA(X2),(Y2);:
  PE0,:PA(X3),(Y3);:PE1,:PA(X4),(Y4);lf
```

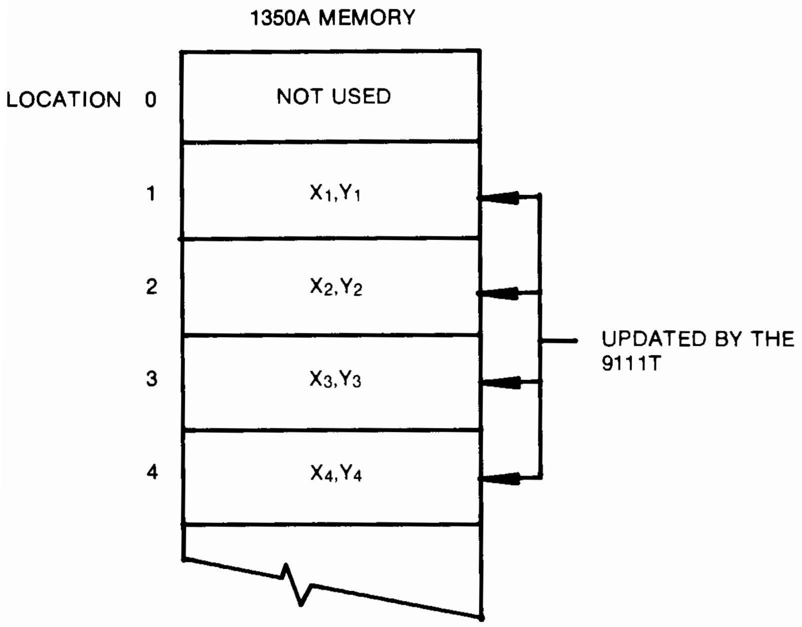


Figure 3-6. 1350A Memory Configuration When Using EE5 Command.

Additional Considerations

The order of execution for all of the "EE" commands is:

1. Place the necessary user supplied data in 1350A memory (locations 0-4).
2. Set the pen in the 1350A (beam on or off).
3. Send the appropriate "EE" command to the 9111T.
4. Configure the bus (9111T = talker, 1350A = listener).
5. Enable the Service Request on HP-IB to cause the user program to handle the interrupt (pen press or digitized point from the 9111T).

When defining areas of the tablet for certain functions, be sure to tell your software what to do with unused areas. For instance, suppose that you have defined the upper half of the tablet surface to invoke various routines or functions. If the pen is pressed in this area, you can read the digitized point from the tablet and depending upon what the value is, the software can be directed to a subroutine which does the appropriate things. If the pen, however, should be pressed in the lower half of the tablet surface, the value read from the tablet will not correspond to any of the numbers in the conditional portion of the software, and the program will "hang". A little "defensive programming" will eliminate this problem by telling the controller to do with points not within the specified area.

When using the EE1, EE2, EE3, and EE4 commands, the command is sent to the 9111T with an X, Y, coordinate which starts the figure at a particular point. The 9111T updates information in memory locations 1-3 of the 1350A. After the figure is drawn on the CRT display and before a new figure is started, the current figure must be stored elsewhere in the 1350A memory if it is to remain on the display. If the current figure is not stored elsewhere, the 9111T will replace the information in memory locations 1-3 with coordinates for the new figure and the current figure will be erased. Consequently, the information should be stored in a file or consecutive memroy locations in the 1350A where there is no chance that it will be erased or written over.

It is a good idea to erase the 1350A memory file into which the 9111T has been writing before invoking a new function. This means that if you have been drawing a rectangle with the 9111T into the 1350A memory and go to the rubber band mode, which only uses locations 0-2, there will be some information left over in locations 3 and 4 and these may be be displayed with the rubber-banded line. Consequently, after finishing the rectangle mode and before using the rubber band mode, erase the file in the 1350A.

Assigning Listener/Talker

Since the 9111T is made to talk directly to the 1350A via the HP-IB bus, it is necessary to configure the 9111T as the bus talker and the 1350A as the listener before transfer of data can take place.

1. Tell all instruments on the bus to UNLISTEN (ASCII "?").
2. Set 9111T as talker (factory set address 06).
3. Set 1350A as listener (factory set address 18).

After the bus is reconfigured as shown, the controller is free to carry on other types of operations until needed by the 9111T, at which time a Service Request is generated. When this happens, the controller reassumes responsiblity as bus talker (figure 3-7).

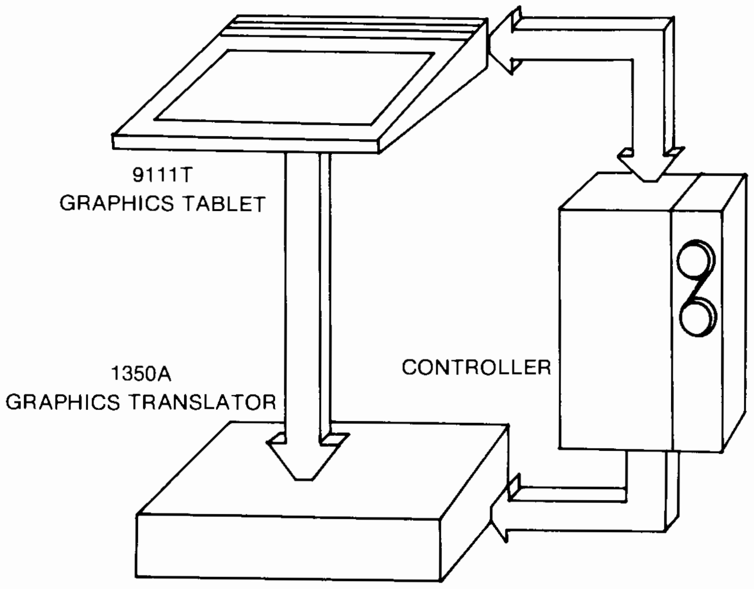


Figure 3-7. Controller Interaction With Assigned Listener/Talker

When the tablet is assigned as the talker and is sending information directly to the 1350A, the controller is free to carry on other tasks, or to talk to other instruments on select codes different than that of the tablet and graphics translator. The controller cannot update the 1350A at the same time that the tablet is doing so. If the system depends on the controller to update dynamic items on the CRT, the controller cannot do this while the tablet is interacting with the 1350A. Consequently, when control returns to the controller, the picture may have changed radically.

Generating Interrupts

Once the 9111T is transferring data directly to the 1350A, the controller is not needed for interaction until a Service Request is generated on the HP-IB bus. When the Service Request is generated, the controller takes control of the bus and routes all data from the 9111T through itself to the 1350A or hardcopy device.

Suppose that you are moving the cursor to a point on screen and want that point to define the start of a rectangle when you press down on the pen. At that time, several things need to happen:

1. The 9111T must stop outputting data to the 1350A.

2. The controller must reassume command of the HP-IB bus.

3. The controller tells the tablet to output the digitized point when the pen is pressed.

4. The controller reads the digitized point and uses it as the fixed corner of the rectangle (software dependent).

5. The controller goes to a pre-defined software routine and stores the digitized point from the 9111T in memory locations 0 and 4 in the 1350A.

6. The controller sends an EE4 command to the 9111T from the software routine.

7. The bus is reconfigured to allow the tablet to output the moving points of the rectangle to the 1350A memory locations 1 through 3.

8. The interrupt in enabled, allowing the next SRQ to be generated.

An SRQ (Service Request) or interrupt is necessary anytime controller interaction of some sort is needed. Some of these may include doing something with a digitized point from the 9111T, doing a calculation before plotting, or selecting a pre-defined function from a softkey.

Using The Softkeys

The softkeys on the 9111T can be used for a number of functions, as mentioned earlier, including pre-defined functions. The 9111A/T is able to interpret a softkey selection by number when the pen is pressed in one of these areas. The system software can interrogate the 9111T to find out which of the softkeys was selected. Depending on the number returned, the program then goes to a routine which relates the softkey number to the pre-defined function. For instance, suppose you want softkey number 1 to define a rectancle. The flow of events may look something like this:

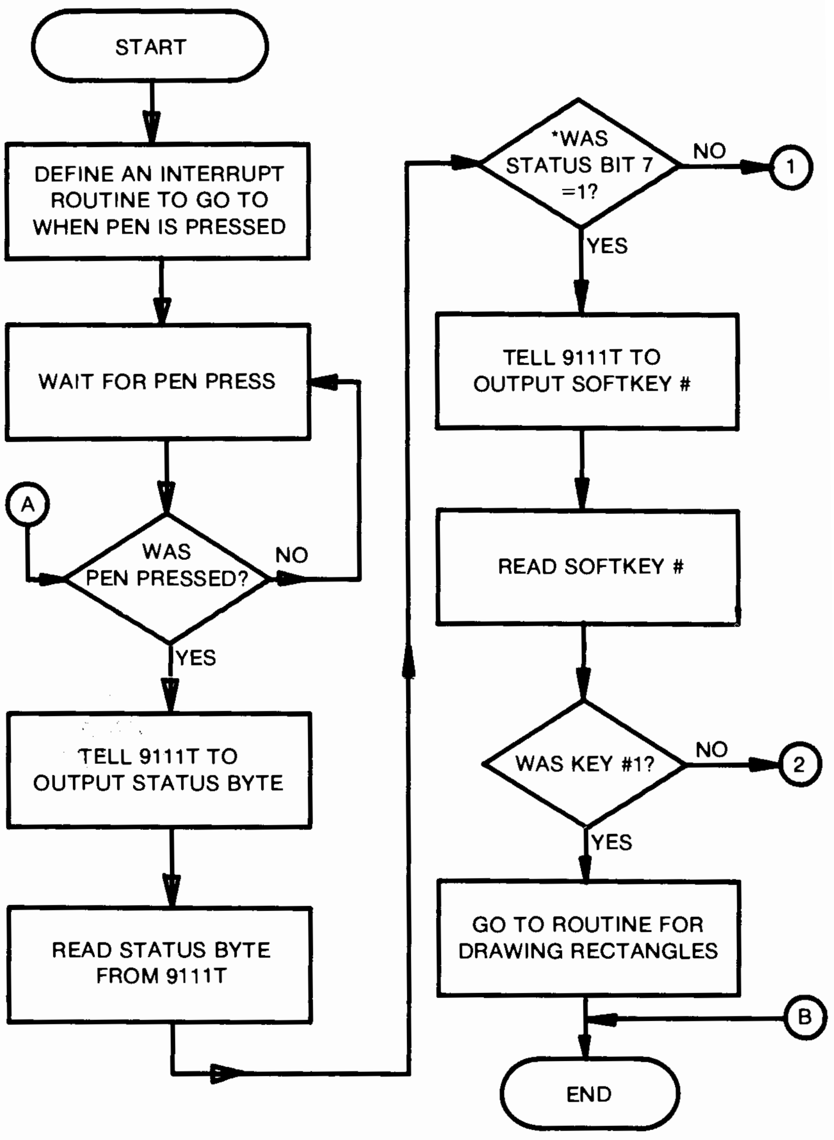
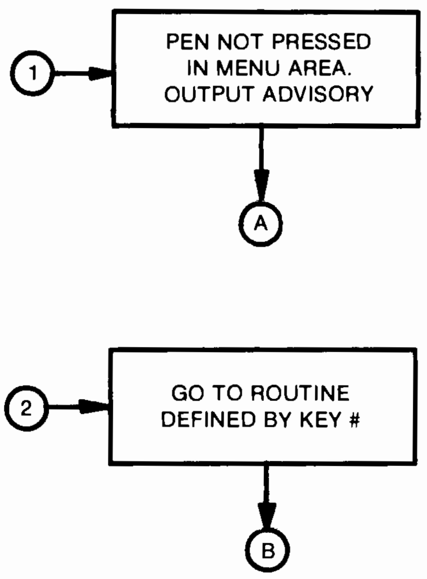

START
DEFINE AN INTERRUPT ROUTINE TO GO TO WHEN PEN IS PRESSED
WAIT FOR PEN PRESS
A
WAS PEN PRESSED?
NO
YES
TELL 9111T TO OUTPUT STATUS BYTE
READ STATUS BYTE FROM 9111T
*WAS STATUS BIT 7 =1?
NO
1
YES
TELL 9111T TO OUTPUT SOFTKEY #
READ SOFTKEY #
WAS KEY #1?
NO
2
YES
GO TO ROUTINE FOR DRAWING RECTANGLES
B
END

Figure 3-8. Interrupt Routine Flowchart


1
PEN NOT PRESSED IN MENU AREA. OUTPUT ADVISORY
A
2
GO TO ROUTINE DEFINED BY KEY #
B

Computer Museum

* Bit 7 of the status byte is 1 when the pen is pressed in a softkey.

Figure 3-8. Interrupt Routine Flowchart (Cont'd.)

Chapter 4
Interfacing

General

This chapter contains information on interfacing the 9111T with the HP-IB bus.

Devices which communicate along the interface bus can be classified into three basic categories:

1. **Talkers** — Devices which are addressed to send information on the bus.

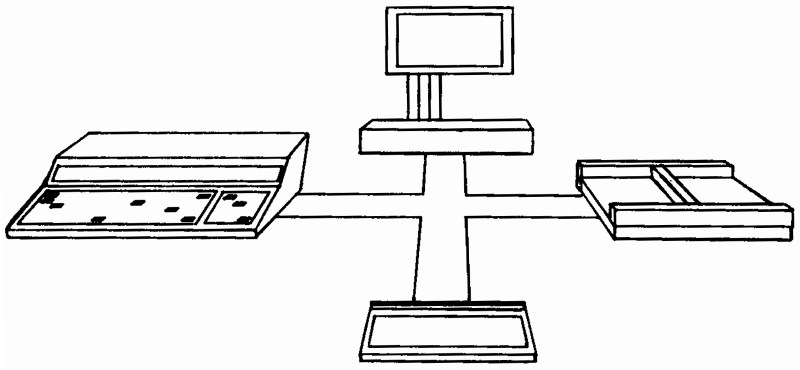
2. **Listeners** — Devices which are addressed to receive information on the bus.

3. **Controllers** — Devices that can specify the talker and listener for an information transfer. Controllers can be categorized as one of two types:

   Active Controller — The current controlling device on the bus.

   System Controller — The controller (which can be active or inactive) that can take priority control of the bus even if it is not the current active controller. Although each bus system can have only one system controller, the system can have any number of devices capable of being the active controller.

A typical HP-IB system with the 9111T and 1350A is shown below:



A plotter, along with the 1350A and X-Y display, gives the means for soft or hardcopy output.

Chapter 5
Programming Examples

Introduction

This chapter contains examples, segments, and complete programs in two languages: (1) HPL, which is used by the HP Model 9825A/B/T Calculators; and (2) BASIC, as implemented by the HP Model 9835A/B.

The programs are broken up into smaller segments to aid their explanation. After each block or segment is explained, the program is shown as a whole.

Not only do these programs explain the operations of the 9111T; when written into and compiled by a computer/controller, they enable the seven functions and six commands that the 9111T/1350A system understands and is able to carry on independent of the controller. The seven functions are:

1. symbol cursor
2. drawn cursor
3. rubber-banded rectangle
4. single rubber-banded line
5. double rubber-banded line
6. forced horizontal line
7. forced vertical line

The programs are heavily commented and intended to give the user/programmer a basic idea how to implement each of these functions in an interactive environment. If you are using a language other than those in the examples, the comments and program segment explanations should show how the general procedure is implemented. If you encounter problems, contact your nearest Hewlett-Packard office.

The HPL examples were done on an HP9825A Calculator Option 002 (23K byte Read/Write memory) equipped with: 98210A String-Advanced Programming ROM and a 98216A General/Extended I/O ROM and 98034A revised HP-IB interface. The BASIC example was done on an HP9835A Calculator with a 98322A I/O ROM.

Dealing With Interrupts

Before starting the actual explanation of operational subroutines, it is convenient to reassign the addresses of the 9111T and 1350A to letter equivalents. This is not only shorter, but makes understanding the program segments just a little easier by eliminating the need to memorize address codes. This is done solely for the sake of convenience and will not change the basic operation of the program.

The 9111T has a factory set address code of 06. The 1350A factory set address is 18. Throughtout this manual we will use select code 7. We can reassign the address in HPL as follows:

```
706→T; 718→G
```

or, in BASIC by:

```
T=706     G=718
```

The interrupt enables the controller to recognize when it should take over the functions of talker when the 9111T has a digitized point to send to the controller. This is done by giving the controller a specific set of instructions or routine to be activated by the interrupt. First, the interrupt routine must be assigned. In HPL the command string might look like this (the 7 in the command is the HP-IB select code):

```
cli 7;oni 7,"pen press"
```

This string first tells the controller to clear the bus (cli 7), then when an interrupt occurs on bus line select code 7 (oni 7), go to the subroutine entitled "pen press".

In BASIC the same instructions would look like:

```
ABORTIO 7       ON INT# 7 GOSUB Pen press
```

Where ABORTIO 7 clears the bus and ON INT# 7 tells the controller to go to the routine called "Pen press".

Throughout this manual, an interrupt originated by the tablet means that the pen has been pressed and the tablet has the coordinates of a digitized point. Because of this, the interrupt routine should cause the tablet to output that point so the controller can read it. Again in HPL, the interrupt routine or "pen press", as we have named the routine, will look like this:

```
0.  "pen press":
1.  sfg 1
2.  wrt T, "OD";red T,X,Y,P
3.  wrt T, "BP 37"
4.  iret
```

Line 1 causes the HPL controller to set an internal flag number 1. This will be explained later when a "wait" loop is defined.

Line 2 instructs the tablet to output the digitized point by means of the "OD" command. When an "OD" command is sent to the 9111T, it outputs three pieces of information: the X and Y coordinates of the digitized point, and a binary 0 or 1 to tell the condition of the pen (either up or down). The second part of line 2 tells the controller to read the X and Y coordinates along with the pen enable bit "P". Even though we tell the controller to read "P", "P" will not be used in our program.

Line 3 is not really necessary, but provides "audible feedback" to let the user know when the pen has been pressed on the tablet surface by causing the 9111T to emit one tone.

Line 4 is the interrupt routine return which causes the program to return to the line immediately following the line that branched to the interrupt routine.

The same routine in BASIC can be done with:

```
10  Pen press:
20    Flag(1)=1
30    OUTPUT T;"OD"
40      ENTER T;X,Y,P
50      OUTPUT T;"BP37"
60      RETURN
```

Enabling the Interrupt

Having told the controller what to do when an interrupt occurs, we now need to enable the interrupt. In HPL, this is done by a simple:

```
eir 7
```

On the 9835A (BASIC) the interrupt is enabled by:

```
CONTROL MASK 7;128
CARD ENABLE 7
```

Invoking the "EE" Commands

Once an "EE" command is sent to the 9111T, it is necessary to assign the 9111T as the bus "talker" and the 1350A as the "listener", enabling the tablet to transfer data directly to the 1350A without controller interaction. This is done immediately after the "EE" command is sent. Let's write a subroutine to assign listener/talker and name this subroutine "talk". In HPL it might look like this:

```
0.  "talk":
1.  cmd 7, "?F2"
2.  cfg 1;eir 7
3.  if not flg1;gto +0
4.  ret
```

Computer Museum

Line 1 stops all bus activity and assigns address 706 (9111T) as the talker and 718 (1350A) as the listener.

Line 2 clears the internal controller flag for the "wait" loop (see explanation for line 3 of this routine). The interrupt is enabled in the second portion of the line.

Line 3 is the actual "wait" loop. After the 9111T is assigned as talker, it transfers data to the 1350A until the pen is pressed. While the tablet is transferring data to the 1350A, we don't want the controller to continue executing the program. This forced wait is accomplished by clearing an internal flag in the controller and telling it to execute the same line over and over until the flag condition changes. The line:

```
if not flg1;gto +0
```

tells the controller to check flag 1 and see if it is set. If not, it stays on the same line (gto +0) until the flag is set. That only happens when the pen is pressed and the interrupt routine "PEN PRESS" is executed. Notice that the first thing subroutine "PEN PRESS" does is set flag 1. So when the pen is pressed on the tablet surface, signalling an interrupt, the interrupt routine is executed, the controller sets flag 1 and exits the "wait" loop.

In BASIC, the routine can be implemented by the following:

```
10  Talk: CONFIGURE 7 TALK = 6 LISTEN = 18
20    Flag(1)=0
30  REM Enable the SRQ on HP-IB
40      CONTROL MASK 7;128
50      CARD ENABLE 7
60  REM Define the "wait" loop
70    IF Flag(1)=0 THEN GOTO 70
80    RETURN
```

Assigning a Data Transfer File In the 1350A Memory

Before the 9111T can actually transfer information to the 1350A, a work file should be set aside in the 1350A memory specifically for this purpose. This file, as explained in chapter 3, is where the EE commands are implemented and where the 9111T places updated X,Y information relating to the current stylus position. This is done by naming the file and assigning the first 5 locations to it. Because the 9111T always addresses location 1 in the 1350A, location 1 should be in this work file. The first 5 locations are set aside by the HPL string:

```
wtb G,"nf1,:tx_____",3,":sn:"
```

The five blanks after the TX command cause the 1350A to write blanks into the 5 locations of file 1. It is important to remember that file 1 must not be used for anything else.

Note

All unassigned memory locations in the 1350A will be in file 0 by default.

For the sake of simplicity, the following format statements are referred to by a number of the HPL subroutines.

```
fmt 1,"pe0,:pa",f4.0,",",f4.0,";:"
fmt 2,"pe1,:pa",f4.0,",",f4.0,";:"
fmt 4,"fl",f4.0,";:"
fmt 6,"uf",f2.0,";:"
```

The formats are accessed by the HPL statement:

```
wrt G+.A
```

where A is the format number. To output an X,Y coordinate using format 1, the syntax is:

```
wrt G+.1,X,Y
```

Interpreting Areas on the Tablet

The program shown in these examples does not use the actual menu keys on the tablet surface. Rather, it interprets sections of the tablet work area. The reason is that when the 9111T is used with a 1350A, the work area is not the same size as that shown on the tablet platten. Therefore, the menu keys displayed on screen and those on the tablet do not correspond. The same goes for the work area displayed on screen and that on the tablet surface.

To interpret an area of the tablet that is not within the menu area, your program must state what to do when the pen is pressed within a defined space. First, we must read the digitized point to see if its X-Y coordinates lie within that space. If it does, tell the program to go to a subroutine that accomplishes the function defined by that space on the tablet platten. If not, we may wish to inform the user that the pen has been pressed in an unallowed or inappropriate area. This is easily done by causing the 9111T to issue a series of beeps or by displaying an advisory message on screen.

An example of an area interpretation subroutine in HPL is given below:

```
 0.  "interpret menu":
 1.  if Y<953 or Y>992;cll 'bad beep';ret
 2.  if X>20 and X<60;cll 'rect';ret
 3.  if X>80 and X<120;cll 'sing';ret
 4.  if X>140 and X<180;cll 'doub';ret
 5.  if X>200 and X<240;cll 'horiz';ret
 6.  if X>280 and X<320;cll 'vert';ret
 7.  if X>340 and X<380;cll 'full';ret
 8.  if X>380 and X<900;cll 'bad beep';ret
 9.  if X>900 and X<940;cll 'erase';ret
10.  if X>960 and X<1005;cll 'abandon';ret
11.  if X<20 or X>1005;cll 'bad beep';ret
12.  ret
```

This subroutine assumes that (1) the pen has been pressed, (2) the "OD" command has been sent to the 9111T, (3) and the controller has read the X-Y coordinates of the point. The program can then call subroutine "interpret menu" which compares the X-Y values and takes the appropriate action.

Line 1 checks the Y value to see if the pen press occured within the menu key area, i.e. 954-991. If not, the program causes the 9111T to emit two tones that indicate an improper action. This is the subroutine called "BAD BEEP".

Line 2 checks for X values in the range of 21 to 59. If X is in this range, the program goes to the rectangle drawing routine entitled "RECT".

Lines 3-7 do the same thing as line 2 for different subroutines.

Line 8 checks for a pen press in one of the undefined and unlabeled menu key areas on screen. If found, the program calls subroutine "BAD BEEP".

Line 9 checks for a pen press in the "ERASE" menu key. If found, the 1350A erases all drawings displayed on screen by erasing file 0.

Line 10 causes the program to abandon the current selected function and redisplay the menu.

Line 11 checks for a pen press outside of the work area.

The same routine is implemented in BASIC as follows:

```
 10  Interpret menu:
 20    IF NOT ((Y<953) OR (Y>992)) THEN 50
 30      GOSUB Bad beep
 40      RETURN
 50    IF NOT ((X>20) AND (X<60)) THEN 80
 60      GOSUB Rect
 70      RETURN
 80    IF NOT ((X>80) AND (X<120)) THEN 110
 90      GOSUB Sing
100      RETURN
110    IF NOT ((X>140) AND (X<180)) THEN 140
120      GOSUB Doub
130      RETURN
140    IF NOT ((X>200) AND (X<240)) THEN 170
150      GOSUB Horiz
160      RETURN
170    IF NOT ((X>280) AND (X<320)) THEN 200
180      GOSUB Vert
190      RETURN
200    IF NOT ((X>340) AND (X<380)) THEN 230
210      GOSUB Full
220      RETURN
230    IF NOT ((X>380) AND (X<900)) THEN 260
240      GOSUB Bad beep
250      RETURN
260    IF NOT ((X>900) AND (X<940)) THEN 290
270      GOSUB Abandon
280      RETURN
290    IF NOT ((X<20) AND (X>1005) THEN 320
300      GOSUB Bad beep
310      RETURN
320    RETURN
```

Generating a Symbol Cursor

When drawing a cursor, the programmer must place a cursor character in location 2 of the 1350A memory. Provided that file 1 is larger than 5 locations, it is possible to make a cursor an entire word, a string of numbers, or a symbol. Each letter, number, or ASCII symbol requires one location in memory. For example, to use the word "banana" as a cursor word would require 8 locations in memory; i.e. 6 locations to accommodate "banana", 1 location for the 9111T to update, and 1 unused location (location 0 — See figure 3-1).

It is also possible to visually overlay characters on top of one another on the CRT screen. In order to do so, you must insert a backspace character (decimal 8) between each of the drawn characters, or use characters which do not cause the character generator in the 1350A to advance. The latter is the technique used in the programs in this manual. To draw a "+" cursor, use the vertical tick mark (decimal 11) with the horizontal tick mark (decimal 12) overlayed. These characters do not advance the 1350A character generator.

To draw a cursor, the HPL subroutine looks like this:

```
0.  "cursor":
1.  wtb G,"ef1,:fl1,:pe0,:pa0,0;:cs2,:pe1,:tx",11,12,3,10,",:pe0,:"
2.  wrt T,"EE0";cll 'talk'
3.  ret
```

Line 1 tells the 1350A to (consecutively) erase file 1, go to location 1, pick the pen up, go to point 0,0 on the CRT, set the character size, enable the pen, draw the cursor character, pick the pen up again. It is a good idea to erase file 1 before starting any function, to clear out any extraneous information from the last function. We then tell the 1350A to find location 1, where the 9111T will put the coordinates of the current stylus position. Next, the pen is placed in the "up" position to prevent the 1350A from drawing a line from the cursor starting point (0,0) to current stylus position. Then, after selecting the size of the cursor with the character size command, the pen is put down and the cursor character (or characters) is drawn at the coordinate point in location 1 (provided by the 9111T). The last step in the process is to raise the pen again so a line will not be seen between the current point and the next updated point.

Line 2 tells the tablet to initiate the "EE0" function. The program then goes to subroutine "talk", which assigns listener/talker and starts data transfer from the 9111T to the 1350A.

In BASIC, the string is the same with the exception of an applicable output format:

```
10  Cursor:
20    OUTPUT G USING (applicable format);
        ":ef1,:fl1,:pe0,:pa0,0;:cs2,:pe1,:tx",11,12,3,10,",:pe0,:"
30  OUTPUT T;"EE0"
40  GOSUB Talk
50  RETURN
```

Generating a Drawn Cursor

While the EE0 command generates a symbol cursor, the EE5 command causes the 1350A to actually "draw" a cursor. With this command, only a vertical/horizontal cursor is possible. The size size, however, can be changed from a single dot to full-screen width. As mentioned before, this has two distinct advantages over the symbol cursor (EE0). First, the drawn cursor is not limited to the 4 character sizes inherent in the 1350A as is the symbol cursor. Second, the symbol cursor is prevented internally by the 9111T's firmware from displaying a cursor closer than 12 units to the 1350A drawing boundaries. Consequently, the drawn cursor can access any point in the 1350A's working boundaries. (The EE0 cursor can return an X-coordinate to the 1350A that is less that 12 or greater than 1010, but it will not be displayed properly on the CRT screen).

The subroutines that follow give a full-screen cursor. The cursor is easily made smaller by replacing the parameter immediately following the EE5 command with a smaller number.

In HPL:

```
0.  "full":
1.  wrt T,"EE5,1020";cll 'talk'
2.  gto -1
3.  ret
```

Line 1 invokes the actual drawn cursor command and enables the talker/listener configuration.

Line 2 loops the subroutine back to line 1. This builds a closed loop routine which can only be exited by selecting the ABANDON menu key.

In BASIC:

```
10  Full:
20    OUTPUT T;"EE5,1020"
30      GOSUB Talk
40      GOTO 20
50    RETURN
```

Storing Pictures in the 1350A Memory

Before proceeding with explanations on the usage of the remaining "EE" functions, all of which actually draw, several points need to be considered. When using the EE1, EE2, EE3, and EE4 commands, the command is sent to the 9111T with the X,Y coordinate which starts the figure at that point. The 9111T updates information in memory locations 1-3 of the 1350A. This means that the current figure is wholly contained in file 1. After the figure is drawn on the CRT display and before a new figure is started, the current figure must be stored elsewhere in the 1350A memory (somewhere other than file 1) if it is to remain on screen. If the current figure is not stored elsewhere, the 9111T will replace the information in file 1 with the new figure and the current figure will be written over. Consequently, the information should be stored in a file or consecutive memory locations in the 1350A memory, where there is no chance of their being destroyed.

One way of making certain a figure doesn't get written over is to store the current figure in memory starting a some location "M". M must be deep enough in 1350A memory to guarantee it will not be replaced by other information. After the current figure is drawn beginning at location M, M should be incremented by the number of vector endpoints needed to draw the figure.

For instance, suppose you have just drawn a rectangle on screen, and want it to stay on screen while you draw another rectangle. When the pen is pressed once, the rectangle begins. As soon as the pen is pressed a second time, the rectangle is defined and contained in 5 locations of file 1. Before the second rectangle is started, the first must be moved out of file 1 to a place in memory beginning at location M. The second rectangle can now be drawn in file 1 without writing over the first rectangle. Now suppose you want to draw a third rectangle. The second must now be moved out of file 1. If however, it is drawn beginning at location M, the first rectangle will be written over. This means that the second rectangle should be drawn at location M+5 if the first and second are both to be saved on screen. (Note: it takes 5 points to draw a rectangle, where the first and last points are the same). So each time a figure is drawn, M must be incremented by the number of points it takes to draw that figure.

Another means of storing figures is to move them to another file. This second file needs to be larger than the sum of all the points you anticipate storing in it. Otherwise, figures may be lost if you exceed the number of locations in the storage file.

Generating Rubber Band Lines

Computer Museum

Single
A single rubber band line has a fixed point at one end while the other end moves with the stylus of the tablet. It is like having a nail in a board with a rubber band over it. At the other end, you insert your finger in the rubber band and move it about. As you move your finger in a circular motion around the fixed endpoint (the nail), the rubber band looks somewhat like a line sweeping around a radar screen. As you move closer in or farther away from the nail, the band stretches or contracts accordingly.

When the pen is pressed the first time on the tablet surface, the fixed endpoint of the line is set. Then, as the stylus is moved away from that point, a line is drawn from the current stylus position to the fixed endpoint. This continues until the pen is pressed the second time, fixing the second endpoint at the pen press coordinate. A second single rubber band line can be started at this second point, or a completely different line can be started with a third pen press.

To generate a single rubber band line in HPL, we will use the following routine:

```
0.  "sing":
1.  cll 'cursor'
2.  wrt G+.4,M;wrt G+.1,X,Y
3.  M+1→M
4.  wrt G+.4,0;wrt G+.1,X,Y;wtb G,"pe1,:"
5.  wrt T,"EE1";cll 'talk'
6.  wrt G+.4,M;wrt G+.2,X,Y
7.  gto -5
8.  ret
```

Line 1 calls subroutine "cursor". The cursor enables the user to see where the pen is positioned on the CRT, and where the line will begin when the pen is pressed. "Cursor" in turn calls subroutine "pen press" to get the X-Y coordinate of a pen press.

Line 2 tells the 1350A to find location "M" and write the X-Y coordinate of the first pen press there with the beam off.

Line 3 increments the location pointer M by 1.

Line 4 instructs the 1350A to go to memory location 0 and write the X-Y coordinates there with the beam off. This is the fixed endpoint of the line.

Line 5 initializes the EE1 command and assigns listener/talker by calling subroutine "talk". It also returns the X-Y coordinate after the pen is pressed.

Line 6 signals the 1350A to find location "M" and write the X-Y coordinates there with the beam on. Note that these X-Y coordinates are not the same as those recovered by line 1. Everytime subroutines "talk" or "cursor" (which in turn calls "talk") are called, a new X-Y coordinate is returned to the main program. This is true because "talk" waits in a loop until the pen is pressed, then goes to "pen press" which reads the X-Y coordinate of the point.

Line 7 loops the routine back to line 1, producing a subroutine that starts the new line at the second endpoint of the old.

The single line subroutine in BASIC is:

```
 10  Sing:
 20    GOSUB Cursor
 30    OUTPUT G USING 40;M
 40      IMAGE "fl",4D,",:"
 50    OUTPUT G USING 60;X,Y
 60      IMAGE ":pe0,:pa",4D,",",4D,";:"
 70    M=M+1
 80      OUTPUT G USING 40;0
 90      OUTPUT G USING 60;X,Y
100    OUTPUT G USING (applicable format);"pe1,:"
110      OUTPUT T,"EE1"
120      GOSUB Talk
130    OUTPUT G USING 40;M
140      GOTO 70
150    RETURN
```

Double
A double rubber band line has two fixed ends, both rubber-banded to a single mobile point. Imagine having two nails in a board with rubber band streched between them. Now stretch one side with your finger. As you move farther from the nails, the band stretches more and the lines from the nails become longer. As you move your finger closer to one of the nails, you create a sort of lopsided triangle.

The first pen press defines one fixed end and the second pen press gives the second end. Now the stylus moves the mobile point around these two fixed ends.

One way to produce a double rubber band line in HPL follows:

```
 0.  "doub":
 1.  cll 'cursor'
 2.  X→X[1];Y→Y[1]
 3.  cll 'cursor'
 4.  wrt G+.4,0;wrt G+.1,X[1],Y[1]
 5.  wrt G+.4,1;wrt G+.2,X,Y
 6.  wrt G+.4,2:wrt G+.2,X,Y;X→[2];Y→Y[2]
 7.  wrt T,"EE1";cll 'talk'
 8.  wrt G+.4,M;wrt G+.1,X[1],Y[1]
 9.  wrt G+.2,X,Y;wrt G+.2,X[2],Y[2]
10.  M+3→M;gto -9
11.  ret
```

Line 1 gives the cursor and returns the first set of coordinates to the program after the pen is pressed.

Line 2 stores coordinates for the first pen press in an array to save them for use later in the subroutine. This also permits the variables X and Y to be redefined.

Line 3 calls the cursor for the second fixed point of the line.

Line 4 erases file 1, sets the 1350A memory pointer to location 0 and places the first fixed endpoint there with the beam off.

Line 5 is not necessary, but places information in location 1. This prevents a momentary line from appearing between point (0,0) and the X,Y point in location 2 until the 9111T can update location 1 with a set of stylus position coordinates.

Line 6 places the second endpoint coordinates in location 2 with the beam on. This causes a line to be drawn between the first and second endpoints. These second endpoint coordinates are then stored in an array for later use.

Line 7 enables the rubber band line function and configures the bus for listener/talker.

Line 8 tells the 1350A to write the first endpoint in location "M" with the beam off.

Line 9 causes the 1350A to write the second endpoint in location M+1 with the beam on. This completes the redrawing of the line beginning at location M.

Line 10 increments the location counter so that another figure will not be drawn over this line. It then loops the routine back to the beginning.

The same subroutine is as follows in BASIC:

```
 10  Doub:
 20    GOSUB Cursor
 30      X(1)=X
 40      Y(1)=Y
 50    GOSUB Cursor
 60      OUTPUT G USING (applicable format);"ef1,:"
 70      OUTPUT G USING 80;0
 80      IMAGE "fl",4D,",:"
 90      OUTPUT G USING 100;X(1),Y(1)
100      IMAGE ":pe0,:pa",4D,",",4D,";:"
110    OUTPUT G USING 80;1
120      OUTPUT G USING 130;X,Y
130      IMAGE ":pe1,:pa",4D,",",4D,";:"
140    OUTPUT G USING 80;2
150      OUTPUT G USING 130;X,Y
160      X(2)=X
170      Y(2)=Y
180    OUTPUT T;"EE1"
190      GOSUB Talk
200    OUTPUT G USING 80;M
210      OUTPUT G USING 100;X(1),Y(1)
220      OUTPUT G USING 130;X,Y
230      OUTPUT G USING 130;X(2),Y(2)
240    M=M+3
250      GOTO 20
260    RETURN
```

Generating a Forced Horizontal Line

A forced horizontal line has the same Y value, no matter where on the tablet surface the pen is moved. With the first pen press, one end of the line is fixed. All subsequent movement of the stylus produces a line with the same Y value as the fixed endpoint; The line is fixed horizontally.

When redrawing a forced horizontal line in the 1350A memory, the programmer is responsible for fixing the Y coordinates. That is, you must use the Y value of the first endpoint for the second endpoint as well.

The programmer must also be careful to send the X-Y coordinates of the first pen press to the tablet when initializing the function with the EE2 command. Notice from Table 3-1 that the X-Y coordinates of that point are part of the syntax of the command. The command, as sent to the 9111T, should be in the form:

```
EE2,X,Y
```

where X and Y are the coordinates of the pen press.

```
0.  "horiz":
1.  cll 'cursor'
2.  wtb G,"ef1,:"
3.  wrt G+.4,0;wrt G+.1,X,Y;wtb G,"pe1,:"
4.  X→X[1];Y→Y[1]
5.  wrt T,"EE2,",X,",",Y;cll 'talk'
6.  wrt G;.4,M;wrt G+.1,X[1],Y[1]
7.  wrt G+.2,X,Y
8.  M+2→M;gto -7
9.  ret
```

The same subroutine in BASIC is:

```
10  Horiz:
20      GOSUB Cursor
30    OUTPUT G USING (applicable format);"ef1,:"
```

```
 40   OUTPUT G USING 50;0
        IMAGE "fl",4D,",:"
 50
 60   OUTPUT G USING 70;X,Y
 70     IMAGE "pe0,:pa",4D,",",4D,";:"
 80     OUTPUT G USING (applicable format);"pe1,:"
 90     X(1)=X
100     Y(1)=Y
110   OUTPUT T;"EE2,",X,",",Y
120     GOSUB Talk
130   OUTPUT G USING 50;M
140     OUTPUT G USING 70;X(1),Y(1)
150     OUTPUT G USING 160;X,Y(1)
160     IMAGE "pe1,:pa",4D,",",4D,";:"
170   M=M+2
180     GOTO 20
190   RETURN
```

Computer Museum

Generating a Forced Vertical Line

A forced vertical line has the same X value, no matter where on the tablet surface the pen is moved. With the first pen press, one end of the line is fixed. All subsequent movement of the stylus produces a line with the same X value as the fixed endpoint; The line is fixed vertically.

When redrawing a forced vertical line in the 1350A memory, the programmer is responsible for fixing the X coordinates. That is, you must use the X value of the first endpoint for the second endpoint as well.

The programmer must also be careful to send the X-Y coordinates of the first pen press to the table when initializing the function with the EE3 command. Notice from Table 3-1 that the X-Y coordinates of that point are part of the syntax of the command. The command, as sent to the 9111T, should be in the form:

```
EE3,X,Y
```

where X and Y are the coordinates of the pen press.

The subroutine for drawing a forced vertical line is the same as that for drawing forced horizontal lines except:

(in HPL)

```
0.  "vert":
5.  wrt T,"EE3,",X,",",Y;cll 'talk'
7.  wrt G+.2,X[1],Y
```

(in BASIC)

```
 10  Vert:
110     OUTPUT T;"EE3,",X,",",Y
150        OUTPUT G USING 160;X(1),Y
```

Generating a Rubber Band Rectangle

A rectangle is defined by 5 points, the first and last having the same coordinates. The start/end points (fixed corner) of the rectangle are fixed by the pen press and placed in memory locations 0 and 4 of the 1350A. The 9111T calculates the three moving corners based on movement of the stylus. The corner opposite the fixed corner is the location of the stylus.

As with the forced horizontal and vertical lines, the programmer must supply the coordinates of the rectangles' fixed corner with the EE4 command. Syntax of the command is:

```
EE4,X,Y
```

To draw a rubber-banded rectangle, we will use the following routine in HPL:

```
0.  "rect":
1.  cll 'cursor'
2.  wrt G+.4,0;wrt G;.1,X,Y;wrt G+.4,4;wrt G+.2,X,Y
3.  X→X[1];Y→Y[1]
4.  wrt T,"EE4,",X,",",Y;cll 'talk'
5.  wrt G+.4,M;wrt G+.1,X[1],Y[1];wtb G,"pe1,;"
6.  wrt G+.2,X,Y[1],X,Y,X[1],Y,X[1],Y[1]
7.  M+5→M
8.  gto -8
9.  ret
```

Line 1 invokes the cursor for positioning the first (fixed) corner and enabling the "pen press" subroutine.

Line 2 writes the X,Y coordinates of the pen press into locations 0 and 4 of the 1350A memory. These are the start and stop points of the rectangle.

Line 3 stores X and Y in an array so they are not lost when X and Y are revalued at the next pen press.

Line 4 tells the tablet to start the rectangle function (EE4) and gives it the coordinates of the first pen press.

Lines 5 and 6 write the finished rectangle in memory beginning at location "M". Note: when line 4 called subroutine "talk", the second corner of the rectangle was fixed when the second pen press occurred. The other two corners of the rectangle can then be determined in terms of the first and third corners (first and second pen presses).

Line 7 increments the location counter.

Line 8 loops the routine back to line 1 to begin another rectangle.

Additional Subroutines

Area Interpretation

Most of the subroutines discussed thus far are looped, with no way to exit. The following subroutine checks X and Y values and assigns a value to "Z" dependent on whether X and Y are within certain ranges. This Z is returned to the main program and can be used by other subroutines. In the program examples at the end of the chapter, this subroutine is used to check for ABANDON, ERASE, or an inappropriate pen press. The HPL subroutine is:

```
0.  "checkXY":
1.  if Y>953 and Y<992;gto +3
2.  if Y>992;4→Z
3.  gto +4
4.  if X>900 and X<960;1→Z
5.  if X>960 and X<1005;2→Z
6.  if X<900;3→Z
7.  ret
```

Line 1 checks the Y value from the pen press and if in the range of 954 to 991, the subroutine jumps to line 4. The Y values here represent the range for the menu keys displayed on screen. Thus, if the pen press is above the menu keys, Z will be equal to 4. Below the menu keys is the work area and we will take no action in this subroutine if Y>953.

If the pen press is within the menu key area, we want to react to only two of the keys (ERASE and ABANDON). If the X value is less than 900, Z will be equal to 3.

If the pen press is in the range of 901 to 959, Z will be set to 1. If X is 961 to 1004, Z will be set to 2.

These values need to be used in other subroutines as follows: the subroutine can call "checkXY" and if the Z returned is equal to 3 or 4, call an advisory routine (beep, bad beep, etc.). If Z is equal to 1, call another subroutine to erase all pictures that have been drawn on the CRT. If Z is equal to 2, call a subroutine that causes the program to abandon the current function.

For instance, let's take subroutine "horiz" and include the call "checkXY" statement where appropriate. This gives the ability to check to see when the function is being abandoned, so the subroutine can be exited.

```
 0.  "horiz":
 1.  cll 'cursor'
 2.  cll 'checkXY';if Z=2;cll 'abandon';gto +10
 3.  if Z=1;cll 'erase';gto -2
 4.  if Z=3 or Z=4;cll 'bad beep';gto -3
 5.  wtb G,"ef1,:"
 6.  wrt G+.4,0;wrt G+.1,X,Y;wtb G,"pe1,:"
 7.  X→X[1];Y→[1]
 8.  wrt T,"EE2,",X,",",Y;cll 'talk'
 9.  wrt G+.4,M;wrt G+.1,X[1],Y[1]
10.  wrt G+.2,X,Y
11.  M+2→M;gto -10
12.  ret
```

Line 2 checks for a Z value of 2. If true, subroutine "abandon" is called. After "abandon" is executed, subroutine "horiz" jumps to line number 12 (return).

Line 3 checks for a pen press in the ERASE menu key. If true, file 1 is erased and the subroutine jumps back to the first executable line.

Line 4 checks for an illegal pen press, calls "bad beep", then returns to the first executable line of the subroutine.

Subroutine "checkXY" looks like this in BASIC:

```
10  Checkxy:
20    IF (Y>953) AND (Y<992) THEN GOTO 50
30    IF Y>992 THEN Z=4
40    GOTO 80
50    IF X<900 THEN Z=3
60    IF (X>900) AND (X<960) THEN Z=1
70    IF (X>960) AND (X<1005) THEN Z=2
80    RETURN
```

Abandoning A Function

The programs in this manual go from function to function by means of a menu on screen. Each function is selected by pressing the tablet pen in a menu key. Once a function is selected, the menu goes away. In order to select another function, you must abandon the current one first by pressing the pen in the ABANDON menu key. The subroutine to accomplish this is:

```
0.  "abandon":
1.  wrt T,"bp 30,75,4;bp 40,150;bp 45;"
2.  wtb G, "uf",5,6,7,",:"
3.  ret
```

Line 1 tells the tablet to "beep" three times to let the user know the subroutine has been enabled.

Line 2 unblanks the three files in the 1350A which contain the menu information. It does not erase the picture just drawn.

In BASIC:

```
10  Abandon:
20    OUTPUT T;"bp 30,75,4;bp 40,150;bp 45;"
30    OUTPUT G USING 40;5,6,7
40    IMAGE "uf",2D,",:"
50      RETURN
```

Using "Beeps"

The tonal scale in the 9111T is very useful in signalling the user when something of importance has occurred. "Beeps" or tones can mark the press of the tablet pen, the enabling of a function, or be a prompt to action. They can also be used to tell the user when an unallowed action is taken. The example programs make use of two short subroutines for tones. The first is used when a subroutine is enabled, telling the user by a series of tones, that the subroutine has been activated. The second signals the user, by low frequency tones, that an inappropriate action has been taken.

The "beep" routines in HPL are:

```
0.  "beep":
1.  wrt T,"bp 40,75,4;bp41;bp42;bp43;bp44"
2.  ret
```

```
0.  "bad beep":
1.  wrt T,"bp6,75,5;bp6,150"
2.  ret
```

Computer Museum

In BASIC:

```
10  Beep:
20  OUTPUT T;"bp40,75,4;bp41;bp42;bp43;bp44"
30  RETURN
```

```
10  Bad beep:
20  OUTPUT T;"bp6,75,5;bp6,150"
30  RETURN
```

Erasing Pictures

When a memory location in the 1350A is written into but has not been specifically assigned to a file, it is in file 0 by default. That makes it easy to erase just the pictures drawn with the 9111T without erasing other necessary information stored in other files. It also means that we need not make a special file just for storing pictures, then hope we have allotted enough memory locations to it so that we don't exceed its maximum size. Since in this manual we specify a location "M" which is not assigned to a file, it falls in file 0. We needn't worry about exceeding the files maximum size, since file 0 will occupy all unassigned memory locations (provided file 0 is not set to a specific size by a Name File 0 command). In order to erase vectors stored at locations M and greater, we simply need to erase file 0. Any information which should not be erased must be placed in a named file.

Subroutine "erase" clears all information in file 0, and is specified in HPL by:

```
0.  "erase":
1.  wrt T,"bp40;bp35;bp50;bp45"
2.  wtb G,":ef0,:";420→M
3.  ret
```

"Erase" in BASIC is:

```
10  Erase:
20    OUTPUT T;"bp40;bp35;bp50;bp45"
30    OUTPUT G USING (applicable format);"ef0,:"
40    M=420
50    RETURN
```

When erasing file 0, it is important to remember to reset M to its original value (in this case, 420).

Complete Programs in HPL and BASIC

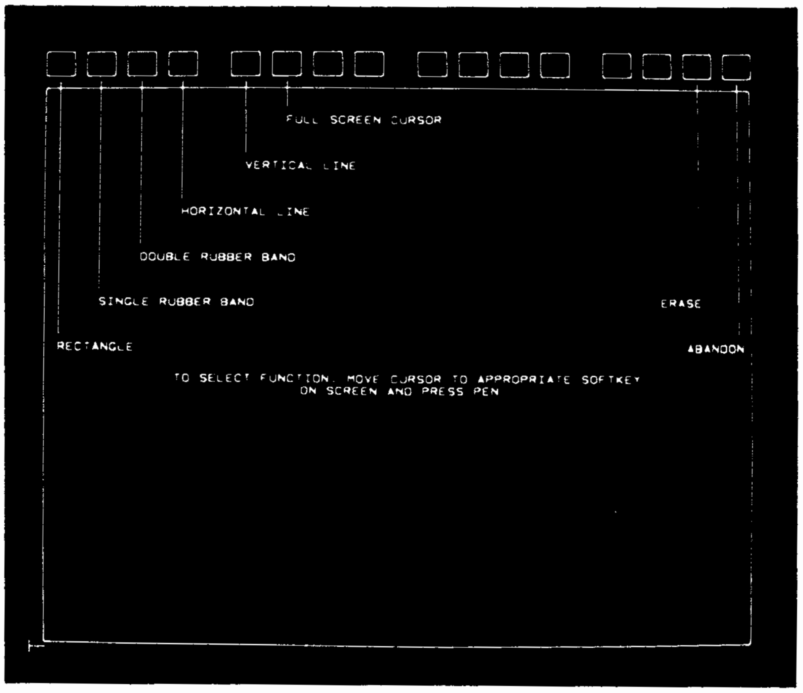


Figure 5-1. Menu Used in Example Programs

9111T Example Program in HPL

```
0: "                                            ":
1: "      9111T EXAMPLE PROGRAM IN HPL          ":
2: "                                            ":
3: 706→T;718→G;cli 7;oni 7,"pen press"
4: dim S[5],T[5],A$[9],B$[3,18],C$[15],D$[13],X[2],Y[2]
5: "RECTANGLE"→A$;"SINGLE RUBBER BAND"→B$[1];"VERTICAL LINE"→D$
6: "HORIZONTAL LINE"→C$;"DOUBLE RUBBER BAND"→B$[2]
7: "FULL SCREEN CURSOR"→B$[3];1→S;415→M
8: 20→S[1]→S[2]→S[5];60→S[3]→S[4]
9: 953→T[1]→T[4]→T[5];993→T[2]→T[3]
10: fmt 1,":pe0,:pa",f4.0,",",f4.0,";:"
11: fmt 2,":pe1,:pa",f4.0,",",f4.0,";:"
12: fmt 3,"pe0,:pa",f4.0,",",f4.0,";:pe1,:cs",f1.0,",:tx",b
13: fmt 4,"fl",f4.0,",:"
14: fmt 5,"bf",f2.0,",:"
15: fmt 6,"uf",f2.0,",:"
16: wrt T,"in;df;im,0,132"
17: wtb G,3,20,"::em::en::ex::sn::sx::um::"
18: wtb G,"nf1,:tx        ",3,":sn:"
19: wtb G,":pe0,:pa300,510;:nf14,:pe1,:cs1,:txTHIS IS THE 9111T",3
20: wrt G+.1,315,450;wtb G,":pe1,:txPress 'Continue'",3,":sn:"
21: wtb G,"nf3,:";wrt G+.1,20,20;wrt G+.2,1003,20,1003,933,20,933,20,20
22: wtb G,":sn:"
23: 1→B
24: "box":1+B→B
25: if B=6;gto "next block"
26: wtb G,"nf4,:";wrt G+.1,S[1],T[1]
27: for I=2 to 5;wrt G+.2,S[I],T[I];next I
28: for D=1 to 3;for P=1 to 5;S[P]+57→S[P];gto "box";next P
29: next D
30: wtb G,":sn:";1→C
```

9111T Example Program in HPL

```
31: "next block":1+C→C;if C=4;stp ;gto "instruct"
32: for P=1 to 5;S[P]+30→S[P];next P;1→B;gto "box"
33: "instruct":wrt G+.5,14
34: wrt G+.1,40,500;wtb G,":nf5,:pe1,:cs0,:tx",A$,3,10,":"
35: wrt G+.1,40,525;wrt G+.2,40,945
36: wrt G+.3,97,575,0,B$[1],3,10,",:"
37: wrt G+.1,97,600;wrt G+.2,97,945
38: wrt G+.3,154,650,0,B$[2],3,10,",:"
39: wrt G+.1,154,675;wrt G+.2,154,945
40: wrt G+.3,211,725,0,C$,3,10,",:"
41: wrt G+.1,211,750;wrt G+.2,211,945
42: wrt G+.3,299,800,0,D$,3,10,",:"
43: wrt G+.1,299,825;wrt G+.2,299,945
44: wrt G+.3,356,875,0,B$[3],3,10,",:"
45: wrt G+.1,356,900;wrt G+.2,356,945;wtb G,":sn:"
46: wtb G,"nf6,:";wrt G+.3,875,575,0,"ERASE",3,10,",:"
47: wrt G+.1,928,600;wrt G+.2,928,945;wtb G,":sn:"
48: wtb G,"nf20,:";wrt G+.3,911,500,0,"ABANDON",3,10,":"
49: wrt G+.1,985,525;wrt G+.2,985,945;wtb G,":sn:"
50: wtb G,"nf7,:"
51: wrt G+.1,200,450;wtb G,":wx14,:pe1,:txTO SELECT FUNCTION, ",3,10,":"
52: wtb G,"pe1,:txMOVE CURSOR TO APPROPRIATE SOFTKEY",3,10
53: wtb G,":pe0,:pa375,430;:pe1,:txON SCREEN AND PRESS PEN",3,10,":sn:sx:"
54: cll 'cursor'
55: cll 'interpret menu'
56: jmp -2
```

9111T Example Program in HPL

```
57: "rect":gsb "beep";wrt G+.5,5,7,8
58: wrt G+.6,8;cll 'cursor'
59: cll 'checkXY';if Z=2;cll 'abandon';gto +8
60: if Z=1;cll 'erase';gto -2
61: if Z=3 or Z=4;cll 'bad beep';gto -3
62: wrt G+.4,0;wrt G+.1,X,Y;wrt G+.4,4;wrt G+.2,X,Y
63: X→X[1];Y→Y[1];wrt T,"EE4,",X,",",Y;cll 'talk'
64: wrt G+.4,M;wrt G+.1,X[1],Y[1];wrt G+.6,0
65: wtb G,":pe1,:";wrt G+.2,X,Y[1],X,Y,X[1],Y,X[1],Y[1]
66: M+5→M;gto -8
67: ret
68: "full":wrt G+.5,5,6,7;gsb "beep"
69: wrt T,"EE5,1022";cll 'talk'
70: cll 'checkXY'
71: if Z=2;cll 'abandon';gto +4
72: if Z=1 or Z=3 or Z=4;cll 'bad beep';gto -3
73: cll 'bad beep';gto -4
74: ret
75: "cursor":
76: wtb G,":ef1,:fl1,:pe0,:pa0,0;:cs2,:pe1,:tx",11,12,3,10,",:pe0,:"
77: wrt T,"EE0";cll 'talk'
78: ret
79: "abandon":wrt T,"bp 30,75,4;bp 40,150,;bp 45"
80: wrt G+.6,5,6,7;ret
81: "bad beep":
82: wrt T,"bp6,75,5;bp6,150";ret
83: "pen press":
84: sfg 1
85: wrt T,"OD";red T,X,Y,P
86: wrt T,"bp37"
87: iret
```

9111T Example Program in HPL

```
88: "interpret menu":
89: if Y<953 or Y>992;cll 'bad beep';ret
90: if X>20 and X<60;cll 'rect';ret
91: if X>80 and X<120;cll 'sing';ret
92: if X>140 and X<180;cll 'doub';ret
93: if X>200 and X<240;cll 'horiz';ret
94: if X>280 and X<320;cll 'vert';ret
95: if X>340 and X<380;cll 'full';ret
96: if X>380 and X<900;cll 'bad beep';ret
97: if X>900 and X<940;cll 'erase'
98: if X>960 and X<1005;cll 'abandon';ret
99: if X<20 or X>1005;cll 'bad beep';ret
100: ret
101: "beep":wrt T,"bp 40,75,4;bp 41;bp 42;bp 43;bp 44"
102: ret
103: "sing":gsb "beep";wrt G+.5,5,7
104: wrt G+.6,9;cll 'cursor'
105: cll 'checkXY';if Z=1;cll 'erase';gto -1
106: if Z=2;cll 'abandon';gto +11
107: if Z=3 or Z=4;cll 'bad beep';gto -3
108: wrt G+.4,M;wrt G+.1,X,Y
109: M+1→M
110: wrt G+.4,0;wrt G+.1,X,Y;wtb G,"pe1,:"
111: wrt T,"EE1";cll 'talk'
112: cll 'checkXY';if Z=1;cll 'erase';gto -3
113: if Z=2;cll 'abandon';gto +4
114: if Z=3 or Z=4;cll 'bad beep';gto -5
115: wrt G+.4,M;wrt G+.2,X,Y
116: gto -7
117: ret
```

Computer Museum

9111T Example Program in HPL

```
118: "doub":gsb "beep";wrt G+.5,5,7
119: wrt G+.6,12
120: cll 'cursor'
121: cll 'checkXY';if Z=2;cll 'abandon';gto +15
122: if Z=1;cll 'erase';gto -2
123: if Z=3 or Z=4;cll 'bad beep';gto -3
124: X→X[1];Y→Y[1]
125: cll 'cursor'
126: cll 'checkXY';if Z=2;cll 'abandon';gto +10
127: if Z=1 or Z=3;cll 'bad beep';gto -6
128: wtb G,"ef1,:";wrt G+.4,0;wrt G+.1,X[1],Y[1]
129: wrt G+.4,1;wrt G+.2,X,Y
130: wrt G+.4,2;wrt G+.2,X,Y;X→X[2];Y→Y[2]
131: wrt T,"EE1";cll 'talk'
132: cll 'checkXY';if Z=2;cll 'abandon';gto +4
133: if Z=1;cll 'erase';gto -13
134: wrt G+.4,M;wrt G+.1,X[1],Y[1];wrt G+.2,X,Y;wrt G+.2,X[2],Y[2]
135: M+3→M;gto -15
136: ret
137: "vert":gsb "beep";wrt G+.5,5,7
138: cll 'cursor'
139: cll 'checkXY';if Z=2;cll 'abandon';gto +8
140: if Z=1;cll 'erase';gto -2
141: if Z=3 or Z=4;cll 'bad beep';gto -3
142: wtb G,"ef1,:"
143: wrt G+.4,0;wrt G+.1,X,Y;wtb G,"pe1,:";X→X[1];Y→Y[1]
144: wrt T,"EE3,",X,",",Y;cll 'talk'
145: wrt G+.4,M;wrt G+.1,X[1],Y[1];wrt G+.2,X[1],Y
146: M+2→M;gto -8
147: ret
```

9111T Example Program in HPL

```
148: "talk":cmd 7,"?F2"
149: cfg 1;eir 7
150: if not flg1;gto +0
151: ret
152: "erase":wrt T,"bp40;bp35;bp50;bp45";wtb G,":ef0,:";420→M;ret
153: "checkXY":0→Z;if Y>953 and Y<992;gto +3
154: if Y>992;4→Z
155: gto +4
156: if X<900;3→Z
157: if X>900 and X<960;1→Z
158: if X>960 and X<1005;2→Z
159: ret
160: "horiz":gsb "beep";wrt G+.5,5,7,8
161: wrt G+.6,13;cll 'cursor'
162: cll 'checkXY';if Z=2;cll 'abandon';gto +8
163: if Z=1;cll 'erase';gto -2
164: if Z=3 or Z=4;cll 'bad beep';gto -3
165: wtb G,"ef1,:"
166: wrt G+.4,0;wrt G+.1,X,Y;wtb G,"pe1,:";X→X[1];Y→Y[1]
167: wrt T,"EE2,",X,",",Y;cll 'talk'
168: wrt G+.4,M;wrt G+.1,X[1],Y[1];wrt G+.2,X,Y[1]
169: M+2→M;gto -8
170: ret
*32756
```

9111T Example Program in Basic

```
10    REM
20    REM   9111T EXAMPLE PROGRAM IN BASIC
30    REM
40    OPTION BASE 1
50    COM Rv(0:0),Flag(0:15)
60    DEG
70      T=706
80          G=718
90          ABORTIO 7
100         ON INT #7 GOSUB Pen_press
110     DIM S(5),T(5),A$[9],B$(3)[18],C$[15],D$[13],X(2),Y(2)
120     A$="RECTANGLE"
130         B$(1)="SINGLE RUBBER BAND"
140         D$="VERTICAL LINE"
150     C$="HORIZONTAL LINE"
160         B$(2)="DOUBLE RUBBER BAND"
170     B$(3)="FULL SCREEN CURSOR"
180         S=1
190         M=415
200     S(5)=S(2)=S(1)=20
210         S(4)=S(3)=60
220     T(5)=T(4)=T(1)=953
230         T(3)=T(2)=993
240     IMAGE ":pe0,:pa",4D,",",4D,";:"
250     IMAGE ":pe1,:pa",4D,",",4D,";:"
260     IMAGE ":pe0,:pa",4D,",",4D,";:pe1,:cs",1D,",:tx",18A
270     IMAGE "fl",4D,",:"
280     IMAGE "bf",2D,",:"
290     IMAGE "uf",2D,",:"
300     OUTPUT T;"in;df;im,0,132"
310     OUTPUT G USING "#,B,B,K";3,20,"::em::en::ex::sn::sx::um::"
```

9111T Example Program in Basic

```
320    OUTPUT G USING "#,K,B,K";"nf1,:tx         ",3,":sn:"
330    OUTPUT G USING "#,K,B";":pe0,:pa300,510;:nf14,:pe1,:cs1,:txTHIS IS
THE 9111T",3
340    OUTPUT G USING 240;315,450
350       OUTPUT G USING "#,K,B,K";":pe1,:txPress 'Continue'",3,":sn:"
360    OUTPUT G USING "#,K";"nf3,:"
370       OUTPUT G USING 240;20,20
380       OUTPUT G USING 250;1003,20,1003,933,20,933,20,20
390    OUTPUT G USING "#,K";":sn:"
400    B=1
410 Box: B=1+B
420    IF B=6 THEN GOTO Next_block
430    OUTPUT G USING "#,K";"nf4,:"
440       OUTPUT G USING 240;S(1),T(1)
450    FOR I=2 TO 5
460       OUTPUT G USING 250;S(I),T(I)
470       NEXT I
480    FOR D=1 TO 3
490       FOR P=1 TO 5
500       S(P)=S(P)+57
510       NEXT P
520       GOTO Box
530    NEXT D
540    OUTPUT G USING "#,K";":sn:"
550       C=1
```

9111T Example Program in Basic

```
560 Next_block: C=1+C
570       IF NOT (C=4) THEN 600
580       PAUSE
590       GOTO Instruct
600    FOR P=1 TO 5
610       S(P)=S(P)+30
620       NEXT P
630       B=1
640       GOTO Box
650 Instruct: OUTPUT G USING 280;14
660    OUTPUT G USING 240;40,500
670       OUTPUT G USING "#,K,K,B,B,K";":nf5,:pe1,:cs0,:tx",A$,3,10,":"
680    OUTPUT G USING 240;40,525
690       OUTPUT G USING 250;40,945
700    OUTPUT G USING 260;97,575,0,B$(1)
710    OUTPUT G USING "#,B";3,10
720    OUTPUT G USING 240;97,600
730       OUTPUT G USING 250;97,945
740    OUTPUT G USING 260;154,650,0,B$(2)
750    OUTPUT G USING "#,B";3,10
760    OUTPUT G USING 240;154,675
770       OUTPUT G USING 250;154,945
780    OUTPUT G USING 260;211,725,0,C$
790    OUTPUT G USING "#,B";3,10
800    OUTPUT G USING 240;211,750
810       OUTPUT G USING 250;211,945
820    OUTPUT G USING 260;299,800,0,D$
830    OUTPUT G USING "#,B";3,10
840    OUTPUT G USING 240;299,825
850       OUTPUT G USING 250;299,945
860    OUTPUT G USING 260;356,875,0,B$(3)
```

9111T Example Program in Basic

```
870     OUTPUT G USING "#,B";3,10
880     OUTPUT G USING 240;356,900
890        OUTPUT G USING 250;356,945
900        OUTPUT G USING "#,K";":sn:"
910     OUTPUT G USING "#,K";"nf6,:"
920        OUTPUT G USING 260;875,575,0,"ERASE"
930        OUTPUT G USING "#,B";3,10
940     OUTPUT G USING 240;928,600
950        OUTPUT G USING 250;928,945
960        OUTPUT G USING "#,K";":sn:"
970     OUTPUT G USING "#,K";"nf20,:"
980        OUTPUT G USING 260;911,500,0,"ABANDON"
990        OUTPUT G USING "#,B";3,10
1000    OUTPUT G USING 240;985,525
1010       OUTPUT G USING 250;985,945
1020       OUTPUT G USING "#,K";":sn:"
1030    OUTPUT G USING "#,K";"nf7,:"
1040    OUTPUT G USING 240;200,450
1050       OUTPUT G USING "#,K,B,B,K";":wx14,:pe1,:txTO SELECT FUNCTION, ",3,
10," :"
1060    OUTPUT G USING "#,K,B,B";"pe1,:txMOVE CURSOR TO APPROPRIATE SOFTKEY",
3,10
1070    OUTPUT G USING "#,K,B,B,K";":pe0,:pa375,430;:pe1,:txON SCREEN AND PRE
SS PEN",3,10,":sn:sx:"
1080    GOSUB Cursor
1090    GOSUB Interpret_menu
1100    GOTO 1080
```

9111T Example Program in Basic

```
1110 Rect:OUTPUT G USING 280;5,7,8
1120        GOSUB Beep
1130    OUTPUT G USING 290;8
1140        GOSUB Cursor
1150    GOSUB Checkxy
1160        IF NOT (Z=2) THEN 1190
1170        GOSUB Abandon
1180        GOTO 1400
1190    IF NOT (Z=1) THEN 1220
1200        GOSUB Erase
1210        GOTO 1130
1220    IF NOT ((Z=3) OR (Z=4)) THEN 1250
1230        GOSUB Bad_beep
1240        GOTO 1130
1250    OUTPUT G USING 270;0
1260        OUTPUT G USING 240;X,Y
1270        OUTPUT G USING 270;4
1280        OUTPUT G USING 250;X,Y
1290    X(1)=X
1300        Y(1)=Y
1310        OUTPUT T;"EE4,";X;",";Y
1320        GOSUB Talk
1330    OUTPUT G USING 270;M
1340        OUTPUT G USING 240;X(1),Y(1)
1350        OUTPUT G USING 290;0
1360    OUTPUT G USING "#,K";":pe1,:"
1370        OUTPUT G USING 250;X,Y(1),X,Y,X(1),Y,X(1),Y(1)
1380    M=M+5
1390        GOTO 1130
1400    RETURN
```

9111T Example Program in Basic

```
1410 Full:OUTPUT G USING 280;5,6,7
1420        GOSUB Beep
1430   OUTPUT T;"EE5,1022"
1440        GOSUB Talk
1450   GOSUB Checkxy
1460   IF NOT (Z=2) THEN 1490
1470        GOSUB Abandon
1480        GOTO 1540
1490   IF NOT ((Z=1) OR (Z=3) OR (Z=4)) THEN 1520
1500        GOSUB Bad_beep
1510        GOTO 1430
1520   GOSUB Bad_beep
1530        GOTO 1430
1540   RETURN
1550 Cursor: !
1560   OUTPUT G USING "#,K,B,B,B,B,K";":ef1,:fl1,:pe0,:pa0,0;:cs2,:pe1,:tx",
11,12,3,10,",:pe0,:"
1570   OUTPUT T;"EE0"
1580        GOSUB Talk
1590   RETURN
1600 Abandon:OUTPUT T;"bp 30,75,4;bp 40,150,4;bp 45;"
1610   OUTPUT G USING 290;5,6,7
1620        RETURN
1630 Bad_beep: !
1640   OUTPUT T;"bp6,75,5;bp6,150,5"
1650        RETURN
1660 Pen_press: !
1670   Flag(1)=1
1680   OUTPUT T;"OD"
1690        ENTER T;X,Y,P
1700   OUTPUT T;"bp37"
1710   RETURN
```

9111T Example Program in Basic

```
1720 Interpret_menu: !
1730   IF NOT ((Y<953) OR (Y>992)) THEN 1760
1740      GOSUB Bad_beep
1750      RETURN
1760   IF NOT ((X>20) AND (X<60)) THEN 1790
1770      GOSUB Rect
1780      RETURN
1790   IF NOT ((X>80) AND (X<120)) THEN 1820
1800      GOSUB Sing
1810      RETURN
1820   IF NOT ((X>140) AND (X<180)) THEN 1850
1830      GOSUB Doub
1840      RETURN
1850   IF NOT ((X>200) AND (X<240)) THEN 1880
1860      GOSUB Horiz
1870      RETURN
1880   IF NOT ((X>280) AND (X<320)) THEN 1910
1890      GOSUB Vert
1900      RETURN
1910   IF NOT ((X>340) AND (X<380)) THEN 1940
1920      GOSUB Full
1930      RETURN
1940   IF NOT ((X>380) AND (X<900)) THEN 1970
1950      GOSUB Bad_beep
1960      RETURN
1970   IF (X>900) AND (X<940) THEN GOSUB Erase
1980   IF NOT ((X>960) AND (X<1005)) THEN 2010
1990      GOSUB Abandon
2000      RETURN
2010   IF NOT ((X<20) OR (X>1005)) THEN 2040
2020      GOSUB Bad_beep
2030      RETURN
2040   RETURN
```

9111T Example Program in Basic

```
2050 Beep:OUTPUT T;"bp40,75,4;bp41;bp42;bp43;bp44"
2060   RETURN
2070 Sing:OUTPUT G USING 280;5,7
2080      GOSUB Beep
2090   OUTPUT G USING 290;9
2100      GOSUB Cursor
2110   GOSUB Checkxy
2120      IF NOT (Z=1) THEN 2150
2130      GOSUB Erase
2140      GOTO 2090
2150   IF NOT (Z=2) THEN 2180
2160      GOSUB Abandon
2170      GOTO 2420
2180   IF NOT ((Z=3) OR (Z=4)) THEN 2210
2190      GOSUB Bad_beep
2200      GOTO 2090
2210   OUTPUT G USING 270;M
2220      OUTPUT G USING 240;X,Y
2230   M=M+1
2240   OUTPUT G USING 270;0
2250      OUTPUT G USING 240;X,Y
2260      OUTPUT G USING "#,K";"pe1,:"
2270   OUTPUT T;"EE1"
2280      GOSUB Talk
2290   GOSUB Checkxy
2300      IF NOT (Z=1) THEN 2330
2310      GOSUB Erase
2320      GOTO 2230
2330   IF NOT (Z=2) THEN 2360
2340      GOSUB Abandon
2350      GOTO 2420
```

9111T Example Program in Basic

```
2360   IF NOT ((Z=3) OR (Z=4)) THEN 2390
2370     GOSUB Bad_beep
2380     GOTO 2230_
2390   OUTPUT G USING 270;M
2400     OUTPUT G USING 250;X,Y
2410   GOTO 2230
2420   RETURN
2430 Doub:OUTPUT G USING 280;5,7
2440     GOSUB Beep
2450   OUTPUT G USING 290;12
2460   GOSUB Cursor
2470   GOSUB Checkxy
2480     IF NOT (Z=2) THEN 2510
2490     GOSUB Abandon
2500     GOTO 2910
2510   IF NOT (Z=1) THEN 2540
2520     GOSUB Erase
2530     GOTO 2460
2540   IF NOT ((Z=3) OR (Z=4)) THEN 2570
2550     GOSUB Bad_beep
2560     GOTO 2460_
2570   X(1)=X
2580     Y(1)=Y
2590   GOSUB Cursor
2600   GOSUB Checkxy
2610     IF NOT (Z=2) THEN 2640
2620     GOSUB Abandon
2630     GOTO 2910
2640   IF NOT ((Z=1) OR (Z=3)) THEN 2670
2650     GOSUB Bad_beep
2660     GOTO 2470_
```

9111T Example Program in Basic

```
2670   OUTPUT G USING "#,K";"ef1,:"
2680      OUTPUT G USING 270;0
2690      OUTPUT G USING 240;X(1),Y(1)
2700   OUTPUT G USING 270;1
2710      OUTPUT G USING 250;X,Y
2720   OUTPUT G USING 270;2
2730      OUTPUT G USING 250;X,Y
2740      X(2)=X
2750      Y(2)=Y
2760   OUTPUT T;"EE1"
2770      GOSUB Talk
2780   GOSUB Checkxy
2790      IF NOT (Z=2) THEN 2820
2800      GOSUB Abandon
2810      GOTO 2910
2820   IF NOT (Z=1) THEN 2850
2830      GOSUB Erase
2840      GOTO 2460
2850   OUTPUT G USING 270;M
2860      OUTPUT G USING 240;X(1),Y(1)
2870      OUTPUT G USING 250;X,Y
2880      OUTPUT G USING 250;X(2),Y(2)
2890   M=M+3
2900      GOTO 2460
2910   RETURN
2920 Vert:OUTPUT G USING 280;5,7
2930      GOSUB Beep
2940   GOSUB Cursor
2950   GOSUB Checkxy
2960      IF NOT (Z=2) THEN 2990
2970      GOSUB Abandon
```

9111T Example Program in Basic

```
2980      GOTO 3180
2990    IF NOT (Z=1) THEN 3020
3000      GOSUB Erase
3010      GOTO 2940
3020    IF NOT ((Z=3) OR (Z=4)) THEN 3050
3030      GOSUB Bad_beep
3040      GOTO 2940
3050    OUTPUT G USING "#,K";"ef1,:"
3060    OUTPUT G USING 270;0
3070      OUTPUT G USING 240;X,Y
3080      OUTPUT G USING "#,K";"pe1,:"
3090      X(1)=X
3100      Y(1)=Y
3110    OUTPUT T;"EE3,";X;",";Y
3120      GOSUB Talk
3130    OUTPUT G USING 270;M
3140      OUTPUT G USING 240;X(1),Y(1)
3150      OUTPUT G USING 250;X(1),Y
3160    M=M+2
3170      GOTO 2940
3180    RETURN
3190 Talk:CONFIGURE 7 TALK = 6 LISTEN = 18
3200    Flag(1)=0
3210      CONTROL MASK 7;128
3220      CARD ENABLE 7
3230    IF NOT Flag(1) THEN GOTO 3230
3240    RETURN
3250 Erase:OUTPUT T;"bp40;bp35;bp50;bp45"
3260      OUTPUT G USING "#,K";":ef0,:"
3270      M=420
3280      RETURN
```

9111T Example Program in Basic

```
3290 Checkxy:Z=0
3300       IF (Y>953) AND (Y<992) THEN GOTO 3330
3310   IF Y>992 THEN Z=4
3320   GOTO 3360
3330   IF X<900 THEN Z=3
3340   IF (X>900) AND (X<960) THEN Z=1
3350   IF (X>960) AND (X<1005) THEN Z=2
3360   RETURN
3370 Horiz:OUTPUT G USING 280;5,7,8
3380      GOSUB Beep
3390   OUTPUT G USING 290;13
3400      GOSUB Cursor
3410   GOSUB Checkxy
3420      IF NOT (Z=2) THEN 3450
3430      GOTO 3640
3440      RETURN
3450   IF NOT (Z=1) THEN 3480
3460      GOSUB Erase
3470      GOTO 3390
3480   IF NOT ((Z=3) OR (Z=4)) THEN 3510
3490      GOSUB Bad_beep
3500      GOTO 3390
3510   OUTPUT G USING "#,K";"ef1,:"
3520   OUTPUT G USING 270;0
3530      OUTPUT G USING 240;X,Y
3540      OUTPUT G USING "#,K";"pe1,:"
3550      X(1)=X
3560      Y(1)=Y
3570   OUTPUT T;"EE2,";X;",";Y
3580      GOSUB Talk
3590   OUTPUT G USING 270;M
```

9111T Example Program in Basic

```
3600      OUTPUT G USING 240;X(1),Y(1)
3610      OUTPUT G USING 250;X,Y(1)
3620   M=M+2
3630      GOTO 3390
3640   RETURN
3650 !
```

Appendix A

Mnemonic	Function
BP	BeeP
CN	CoNtinuous sampling mode
CR	Cursor Rate
DC	Digitizer Clear
DF	DeFault
DP	Digitize Point
EE0	Symbol cursor
EE1	Rubber band lines
EE2	Forced horizontal line
EE3	Forced vertical line
EE4	Rubber band rectangle
EE5	Drawn cursor
IM	Input Masks
IN	INitialize
IP	Input Points
OA	Output Actual stylus position
OC	Output Cursor
OD	Output Digitized point
OE	Output Error
OF	Output Factor
OI	Output Identification
OK	Output Key
OP	Output Points
OR	Output Resolution
OS	Output Status
RC	Read Cursor
RS	Read Softkey
SF	Switch Follow
SG	SinGle sample mode
SK	Set Key
SN	Switch Normal
TD	Test Digitizer
TP	Take Point

Computer Museum

Subject Index