PDP-9 REMOTE GRAPHICS TERMINAL

Introduction

James F. Blinn

PDP-9 Remote Graphics Terminal (Introduction))e

1. INTRODUCTION

This manual provides a basic introduction to the facilities of the PDP-9 Remote Graphics Terminal located in room 2512 East Engineering. Users of the terminal fall into two categories: those who write their own PDP-9 assembly language programs and those who program entirely in FORTRAN on MTS and use the SEL Routine subroutine package. The following pages give all information common to both types of users: loading the system, the system command language, and display operations. Details peculiar to assembly language programmers or to SEL Routine users appear in the appropriate programmers manual.

On those occasions where we make explicit reference to subroutine names the following convention will apply: local system routines always begin with a # (e.g. #XI) while the corresponding SEL Routine begins with "SEL" (e.g. SELXI).

2. BASIC SYSTEM FACILITIES

2.1 Hardware

The PDP-9/339 Remote Graphics Terminal is built around a DEC PDP-9 central processing unit. This is a reasonably powerful 18-bit minicomputer with 16K words of core and the usual peripherals of teletype, high speed paper tape reader and high speed paper tape punch. Included in the processor is an Extended Arithmetic Unit which provides hardware multiply and divide operations and greatly enhances the power of the PDP-9 to do such operations as rotations and scaling locally.

Attached to the PDP-9 is a DEC 339 display control. This is essentially a separate, special-purpose processor which cycles through the common memory in parallel with the PDP-9 executing its own instruction set to display lines and points. Included with the display control is a "light pen", a hand held photo sensitive device which detects the light from displayed pictures, and a push-button box for programmed function keys.

Communication with MTS is possible using a 201A dataphone dialed into the Data Concentrator. Records can be sent or recieved at transmission rates up to 2000 bits per second, with automatic error recovery provided for loss of data over the line.

Finally, another mode of communication with the graphics terminal consists of a three channel analog-to-digital convertor and a 16 channel digital-to-analog convertor. These devices can be attached to whatever analog devices the user provides.

2.2 Software

2.2.1 The SEL Executive System

The SEL Executive System serves as the interface between user written PDP-9 programs and the hardware. It provides subroutines for buffered I/O to and from the teletype, paper tape equipment, and the 201 dataset (MTS). System subroutines also manage the free core pool, do task switching, input data from the A/D convertor, push button box, and real time clock, and output data to the D/A convertor and pushbutton lights. The system command language provides various debugging operations and allows communication with MTS as a straight teletype terminal.

The system display support provides routines to draw, erase and move user generated picture elements called "leaves". It also

takes care of all synchronization between the PDP-9 and the 339 necessary due to the multiprocessing nature of the system.

Input comes from the display via the light pen. Upon sensing a light pen hit, the system notifies the user program and provides information about what was hit and where on the screen it was. Continuous X,Y coordinate input can also be obtained by using the system's tracking routines.

2.2.2 User Programs

The SEL System provides only the basic I/O support for the terminal. Actually building data structures and creating display leaves is the job of user written PDP-9 programs. Such user programs can then read lines from MTS (the system treats the dataphone connection to MTS as just another I/O device) and process them to provide whatever special purpose graphical operations are desired. For example, since the dataphone can transmit only about 250 characters per second, it is a good idea for the MTS program to send only the data represented by a picture and for the PDP-9 program to read this in and reformat it into the more lengthy 339 instruction codes for a display leaf. The precise division of labor between the 360 and the PDP-9 can thus be taylored to the users particular application. Generally, the 360 is used for mass storage and number crunching while the PDP-9 does highly interactive I/O operations and fast graphical operations. Applications which require small amounts of computations and very fast response times could perhaps be best done totally in the PDP-9 using the 360 only for assemblies, etc.

2.2.3 The SEL Routines

Those users who do not wish to expend the effort to learn how to write PDP-9 programs can still make use of the Remote Graphics Terminal by using a subroutine package called the SEL Routines. These routines run on the 360 under MTS and are FORTRAN callable. They allow FORTRAN users to do all graphical operations in the 360 and remotely call PDP-9 system subroutines. a call to a SEL routine simply transmits a command to the terminal containing the subroutine name and its parameter values. a small PDP-9 program called the SEL Routine Interpreter reads these commands and calls the appropriate system subroutine with the given parameters. This represents the opposite end of the scale described at the end of the previous section and, of course, cannot provide the fastest response times possible, but it is simpler to use.

3. SYSTEM OPERATION

3.1 Loading the System

The system is loaded by the following procedure:

1) Place the tape labeled SEL EXECUTIVE SYSTEM in the paper tape reader.
2) Set all the white console switches to 0 (down).
3) Depress the READ-IN key.

The tape will be read in and the system will automatically start when the end of the tape is reached. Occasionally the reader will stop after only a few inches of tape have been read. This is because the power-up sequence on the PDP-9 does not initialize some internal registers properly. When this happens, depress the I/O RESET key and repeat the above procedure.

3.2 System Initialization and error comments

Various conditions described below can cause a system reinitialization. All reinitializations except those for TASK QUEUE EMPTY do the following:

Stop all I/O devices and flush their buffers.
Clear the display.
Clear the task queue.
Unlock all system subroutines.

All reinitializations including TASK QUEUE EMPTY do the following:

Turn off pushbuttons.
Stop tracking.
Clear the #BD buffer.
Disable light pen hits.
Routes all subsequent 201 records to the display.
Reinitialize the free storage pool.

A reinitialization is signalled to the user by ringing the teletype bell and typeing "!". A diagnostic message is then placed on the screen followed by a dump of all active registers: Link, Accumulator, Multiplier Quotient, Step Counter, and Program Counter. For the reinitialization diagnostics described below the

contents of the meaningful registers are described. All register displays not mentioned contain garbage.

SYSTEM RELOADED

Initial program load of the system.

PANEL RECOVERY

Panel restart of the system. This is accomplished by the following procedure:

1) Set the left hand bank of panel switches to 00022.
2) Depress the I/O RESET key.
3) Depress the START key.

INVALID INTERRUPT

The I/O interrupt processor was entered but no device flags were set. This can be caused by a hardware error or by the user program making a wild jump into the system core bank.

AC,MQ,LINK,SC: Contents at time of "interrupt".

PC: Contents of location 0 when the error condition was recoginzed

MANUAL INTERRUPT

The user hit the INTERRUPT button on the push button box. This serves as an "attention interrupt" to the local system.

AC,MQ,LINK,SC,PC: Contents at the time the interrupt button was hit.

TRAP

The PDP-9 executed an illegal instruction (op-code 0). This is usually caused by attempting to execute data or by jumping to an undefined symbol.

PC: Address of illegal instruction

AC,MQ,LINK,SC: Contents at execution of illegal instruction

TASK QUEUE EMPTY

A call to #TF was made and there were no further tasks on the task queue. This is the normal way for a user program to terminate.

PC: Address of last call to #TF

TASK QUEUE OVERFLOW

A call to #TS was made and the task queue was already full.

PC: Address of last call to #TS

CRASH

A drastic error was found in the display program. This can arise due to a push down list overflow (a maximum of 64 levels is allowed), or the display control encountering a STOP instruction. This error is almost always caused by the user program inserting a bad leaf into the structure.

AC: Address of offending display instruction.

MQ: Offending display instruction.

BAD PARAMETER

A parameter given to one of the system subroutines did not look at all like it should have. Conditions for each routine to produce this comment appear in the Assembly Language Programmers Manual.

AC: Parameter value the routine thought was passed to it.

PC: Address of offending parameter word.

4. COMMAND LANGUAGE INTERPRETER

After a system reinitialization, the Command Language Interpreter is activated to execute commands entered at the teletype. The CLI indicates readiness for a command by typing "(" as a prompting character. In all command descriptions to follow, only the underlined characters need to be typed; the system will automatically echo the rest. If a character is typed which does not follow the specified sequence for any of the commands, it is totally ignored. At any time during the typing of a command, hitting RUBOUT will echo a question mark and delete the command.

4.1 CLEAR Command

This command requires confirmation before being executed. After typing "CD" or "CU" the user must type "O" (for OK) to confirm the command. typing anything else echoes "NO" and the command is canceled.

CLEAR USER CORE? OK

This command stores a trap instruction (000000 octal) in the entire user core bank from location 20000 through 37777. If execution ever reaches one of these instructions, the system immediately terminates execution with the error comment "TRAP". This command is generally issued before loading an un-debugged program to hopefully catch the program if it goes wild.

CLEAR DISPLAY? OK

This command removes all text from the display screen, thus making room for more text.

4.2 FROM Command

```
          PAPER TAPE     PAPER TAPE
          CORE           CORE
     FROM TELETYPE   TO  TELETYPE
          201            201
                         DISPLAY
```

This command copies data from the "FROM" (source) device to the "TO" (sink) device. The copying function is alphanumeric or

binary depending on the combination of I/O devices specified and on the data read from the source device.

4.2.1 Source Devices

FROM PAPER TAPE TO ...

Paper tape is read for copying until a paper tape end-of-file chatacter is reached. The SEL system uses blank tape (octal 000) as the end-of-file character. The tape can be alphanumeric or binary. Alphanumeric format consists of ASCII character codes with the parity bit forced on. Lines are terminated by carriage return (octal 215) followed by line feed (octal 212). Binary tape consists of origin frames followed by a series of data frames of the form:

```
binary  | *xxx.xxx|
origin  | *xxx.xxx|
        | *xxx.xxx|
binary  |* xxx.xxx|
data    |* xxx.xxx|
        |* xxx.xxx|
        |* xxx.xxx|
        |* xxx.xxx|
        |* xxx.xxx|
        |* xxx.xxx|
```

The tape is determined to be alphanumeric or binary by examining the first character read. If the high order bit is set (punched) it is an ASCII character with the parity bit set. If the high order bit is clear, it is the first character of a binary origin.

FROM CORE TO ...

This command solicits the user for the high and low addresses of the core block to be copied. The user then fills in the blanks in the line:

□BLOCK(_____,_____)

A transfer from core is always binary.

FROM TELETYPE TO ...

Copy operations from the teletype are different for each sink device.

... TO PAPER TAPE
... TO DISPLAY
... TO TELETYPE

The user is prompted for lines by the prifix character "□". Text typed in is subject to the line editing characters:

	Echo	Function
ctl-H	backarrow	Backspace
RUBOUT	#	Delete line
RETURN	cr-lf	End of line
ctl-C	backslash	Terminate copy function

... TO CORE

This command allows the display and modification of core locations. Typing a five digit octal address causes a display of the contents of that address. The user can then type:

6-digit octal number	Store this new value in the current location and display the next sequential core location.
RETURN	Do not change the current location but display the next sequential location.
RUBOUT	Terminate copy to core.

This command is assumed whenever the CLI expects a command (prefix !) and the user types in an octal address.

... TO 201

This command allows communication with MTS. The user is prompted for lines by the prefix character sent by MTS (see description of copying TO 201). Text typed in is subject to the following special character interpretations:

	Echo	Function
ctl-H	backarrow	Backspace
RUBOUT	#	Delete line and retype prefix
RETURN	cr-lf	Send line to MTS
ctl-C	backslash	Send end-of-file to MTS
ctl-E	!	Send attention interrupt to MTS
ctl-shft-K		Invert "surpress echo" switch.
alt-mode		Return to SEL System CLI.

This command can also be given by striking the alt-mode key when the command language interpreter is expecting a command. thus repeatedly striking the alt-mode key alternates between talking to the local system and talking to MTS. the user can always see which system he is talking to by looking at the prompting character:

"\" or "□"	SEL System
"#" or ">" or " " etc	MTS

FROM 201 TO ...

Since records arrive from the dataphone at arbitrary times, this command only sets switches to route future records to the specified sink device. the command will prompt the user with "□" for further information of form:

□BINARY	All binary records received from the 201 will be sent to the specified sink device.
□PREFIX=X	All ASCII records with the specified prefix character are routed to the specified sink device.
□DEFAULT	All other ASCII records will be sent to the specified sink device.

A system reinitialization sets up the routing switches as though the following commands were given.

```
\FROM 201 TO DISPLAY
□DEFAULT
\FROM 201 TO DISPLAY
□BINARY
```

All records recieved from the 360 will then be routed to the display. as another example, the following sequence of commands causes all future binary records recieved from MTS to be punched, all ASCII records prefixed by ">" to be typed and all other ASCII records to be written on the display:

```
)FROM 201 TO PAPER TAPE
]BINARY
)FROM 201 TO TELETYPE
]PREFIX=>
)FROM 201 TO DISPLAY
]DEFAULT
```

The following processing is performed on records read from the 201.

For copying to TELETYPE or DISPLAY:

ASCII records are copied intact (including the MTS prefix character).

For copying to TELETYPE, DISPLAY or CORE:

Binary records are interpreted in PDP-9 paper tape binary format. If non standard binary records are recieved from the 201, an attention is immediately sent to the 360 and an appropriate comment is typed.

For copying to PAPER TAPE:

ASCII records have the first character (MTS prefix character) removed to facilitate copying the tape back into a file later. Binary records are copied intact so that any arbitrary binary format may be punched.

Note that the FROM 201 command does not actually try to read from the 201. See the description of the 201 as a sink device to see how this is done.

4.2.2 Sink Devices

FROM ... TO PAPER TAPE

Characters are punched in the same binary and ASCII formats described in section 3.2.1. Whenever an alphanumeric file is punched, a short length of blank tape is fed out at the beginning and at the end of the punched tape file to ensure separation of individual ASCII files. When a binary file is punched, no such

blank tape is put out. Thus, copying two consecutive binary files to the punch concatenates them into one file.

FROM ... TO CORE

Copying to core is essentially a load operation from the source device and thus is always binary. For example the command: FROM PAPER TAPE TO CORE loads a binary paper tape into core, etc.

FROM ... TO TELETYPE

Copying ASCII files to the teletype is essentially a list operation. Copying binary files produces a core image dump in the format:

```
20000 000000 000000 000000 000000 000000 000000 000000 000000
20010 000000 000000 000000 000000 000000 000000 ...
```

The first number on each line is the memory address. The subsequent numbers are the contents of memory from that address on up to a maximum of eight locations per line.

FROM ... TO DISPLAY

This is similar to copying to the teletype except that the information appears on the face of the display. When the storage allocated for display text is exhausted, the following comment appears:

DISPLAY FULL - CLEAR?

If the user then enters OK, the screen is cleared and the copy operation continues. If the user types anything else, "NO" is echoed and the copy operation is aborted.

FROM ... TO 201

Issueing a copy to the 201 is what actually starts activity at the 201 interface. he copy operation begins by reading records and routing them according to the switches set by the FROM 201 command. It continues to read until it recieves a record terminated by an ETX (octal 203) character. This will be an MTS prefix record which indicates that MTS is waiting for a read. When copying FROM TELETYPE TO 201, the prefix record is typed out to prompt the user for entry of a line. When copying from any other device, the prefix record is swallowed up. A line is then

read from the source device and sent to MTS. When MTS sends another prefix record the next line is read and transmitted.

For example, the following sequence of commands sets things up so that the teletype acts like a normal terminal:

```
(FROM 201 TO TELETYPE
 DEFAULT
(FROM TELETYPE TO 201 (or alt-mode )
```

The following sequence of commands punches the contents of the file OBJ in binary. This is how to get an object tape from an assembly on the 360.

```
(FROM 201 TO PAPER TAPE
 BINARY
(FROM TELETYPE TO 201 (or alt-mode )
#$COPY OBJ TO *SINK*@BIN
```

The following sequence of commands copies an ASCII tape into a file in the 360.

```
#$COPY *SOURCE* TO FILE
>alt-mode
(FROM PAPER TAPE TO 201
```

Note that in all such operations, the user must set up the sink device first. when copying to MTS, give the MTS command first. when copying to the PDP-9, give the SEL command first.

4.3 RUN Command

```
RUN aaaaa
```

If the dataphone is connected, the system first reads from the 201 until ?????? started at the address typed is as aaaaa. Thus, when a user program starts, it is guaranteed that MTS is waiting to recieve a record. Note that if the user has previously stopped MTS from sending prefix characters (by, for example, issueing $COP *SOURCE*@BIN TO -F) the user program cannot start. Therefore, if the user wishes to send such a command he must do so within his program.

5. DISPLAY CONTROL PROGRAMS

In addition to the usual jump, subroutine-call, and subroutine-return instructions, the instruction set of the 339 display control has special instructions to move the electron beam across the CRT to draw lines and points. Execution of a line drawing instruction intensifies points along the line for a brief period of time. To provide a steady image, the entire 339 program must be executed over and over so that the repeated intensifications merge into a solid picture.

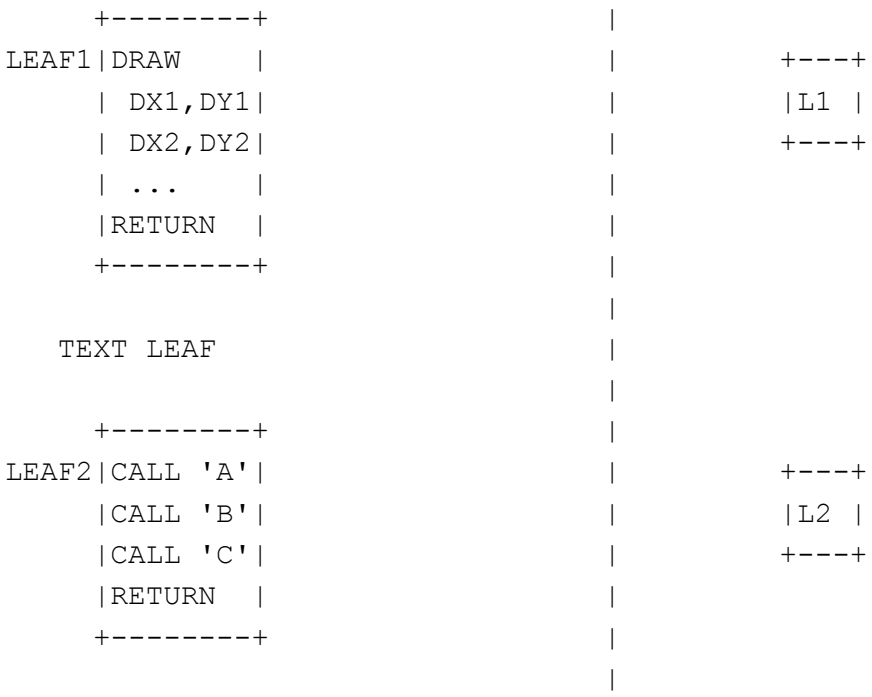
The SEL system maintains the 339 display control program in the form of a tree-like data structure. Thus the operations of drawing, erasing, and translating picture elements become those of inserting, deleting, and modifying elements of the structure. This arrangement not only eases various graphical operations but also allows the user to store some relational data about the picture in the form of the tree structure that he builds.

5.1 Root Node

The root node is the beginning of the display program. the system re-starts the display control here at the end of each frame. Initially it consists of a 339 instruction which jumps to a stop instruction. The node in this state is said to be empty.

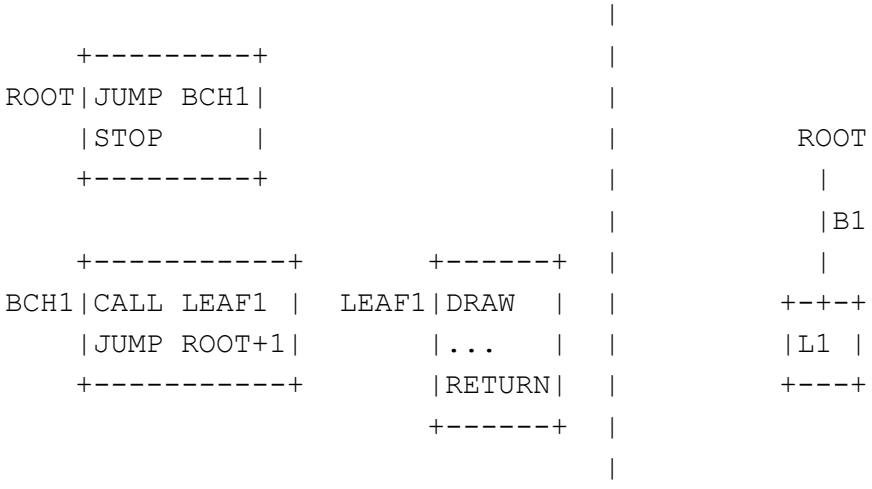
5.2 Leaves

A user written 339 subroutine which draws a part of the picture on the screen is called a leaf. A leaf can, in general consist of any sequence of 339 display control instructions terminated by a return-from-subroutine instruction. Generally a leaf consists of either a list of X,Y coordinates for lines or a sequence of calls on character drawing 339 subroutines. Various system routines are provided to construct line leaves (SELINE) and text leaves (#BD, SELTXT).

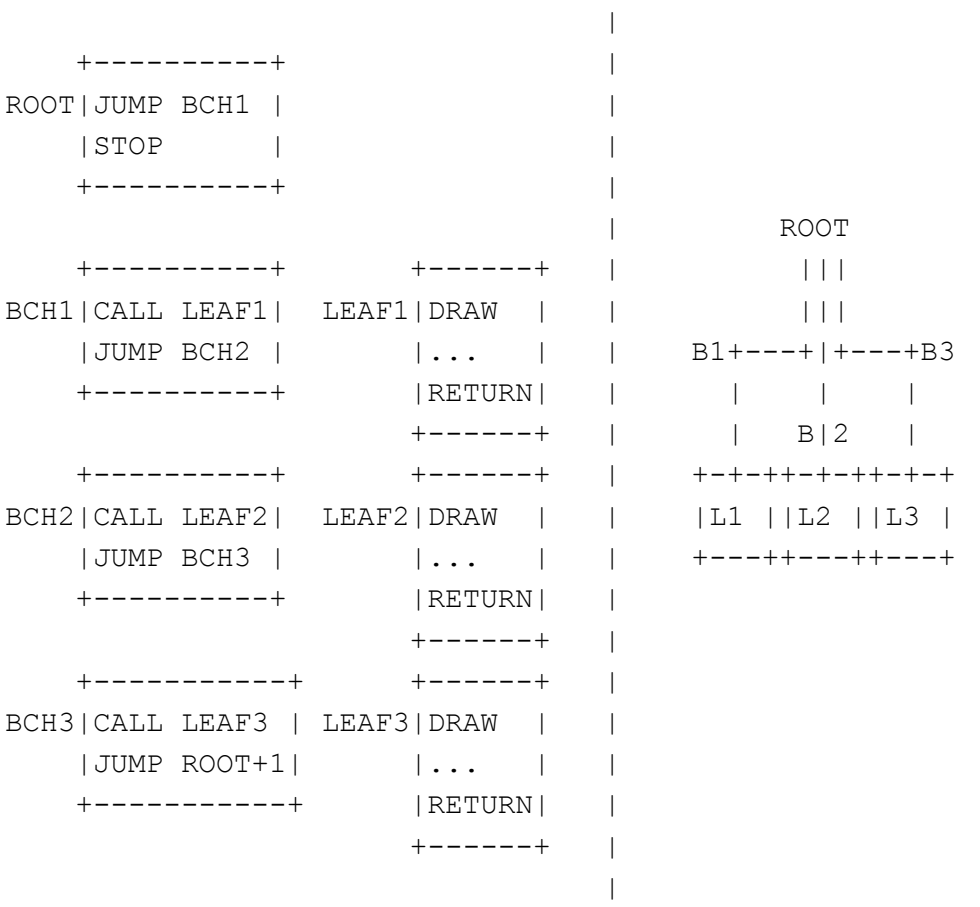


5.3 Branches

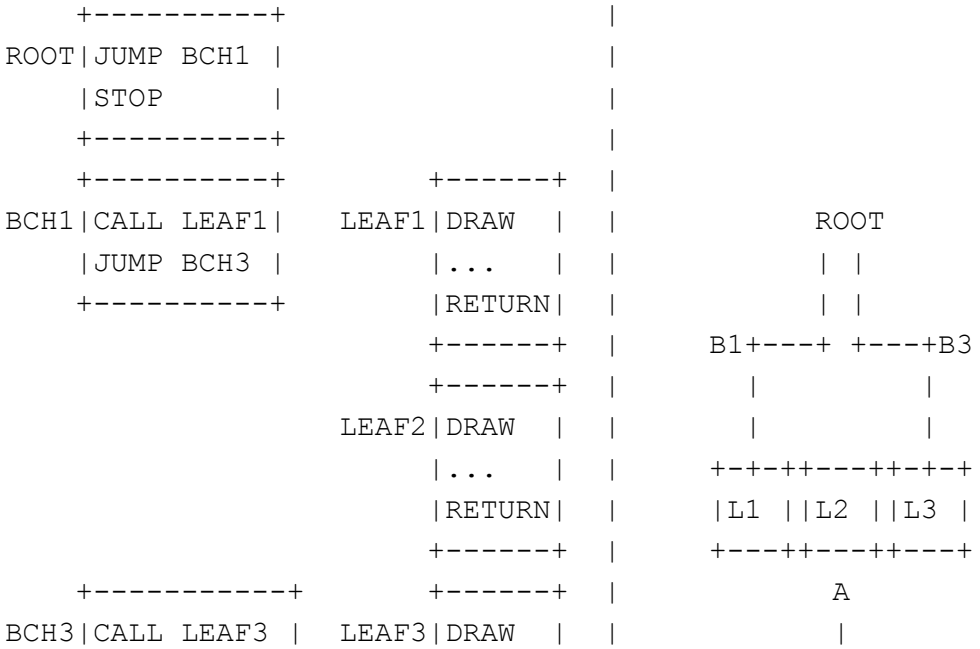
Simply creating the leaf-subroutine will not get it displayed, it must first be linked into the 339's execution loop. Such a link is called a branch. To display a leaf on the screen, the user calls a routine called #SBC (SELBC) to create a branch from the root node to the leaf. This routine modifies the structure to look like the following.

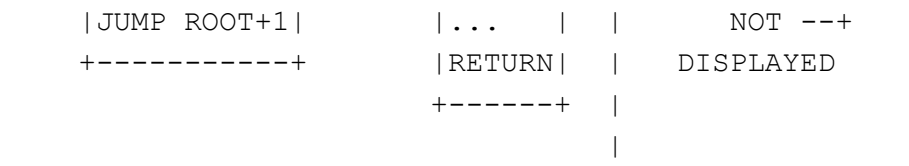


Several different leaves may be similarly linked into the structure to get several pictures on the screen.

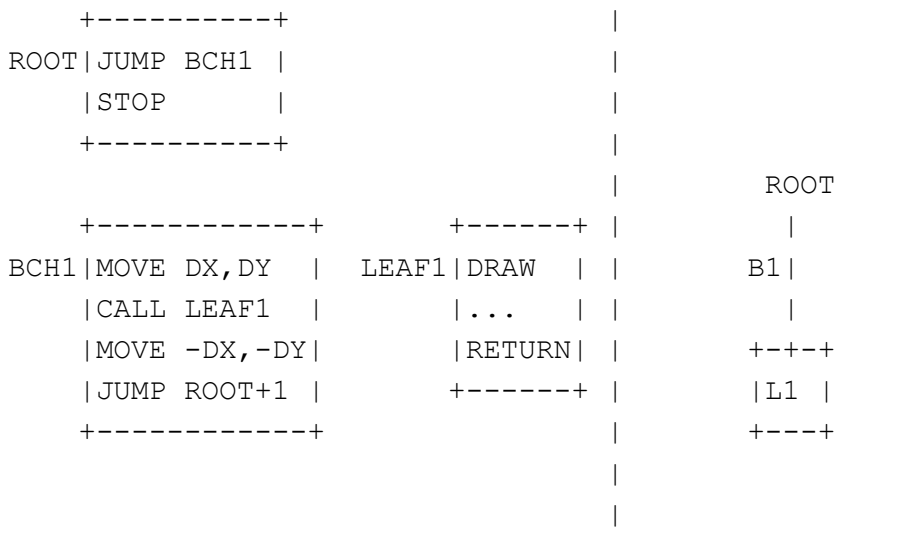


To remove a leaf from the screen, call the routine #SBD (SELBD) to destroy the appropriate branch.

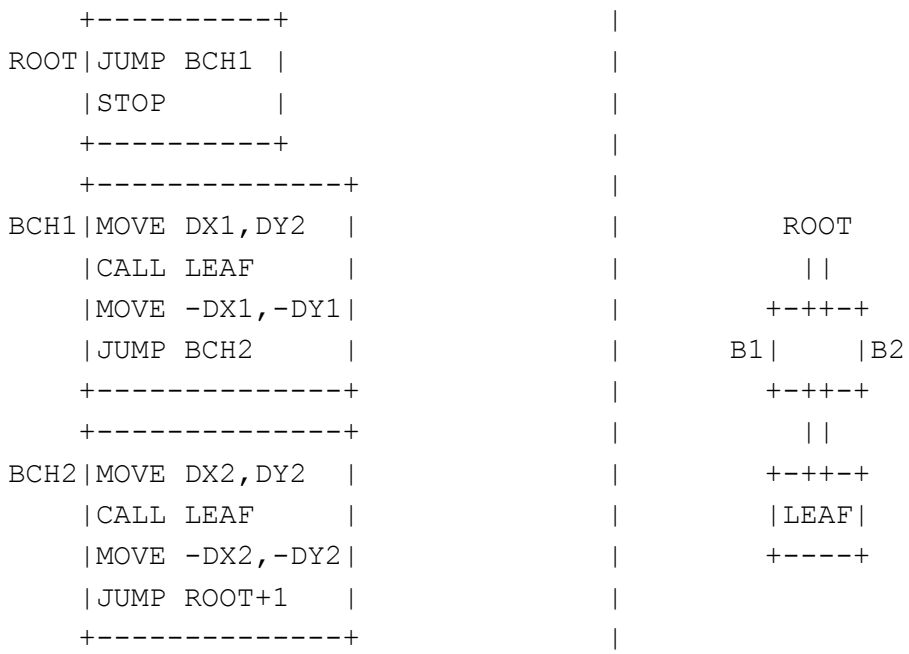




Once a leaf has been placed on the screen, it can be repositioned by calling #SBAC (SELBAC) to apply a translataion to the branch. A translated branch looks like:

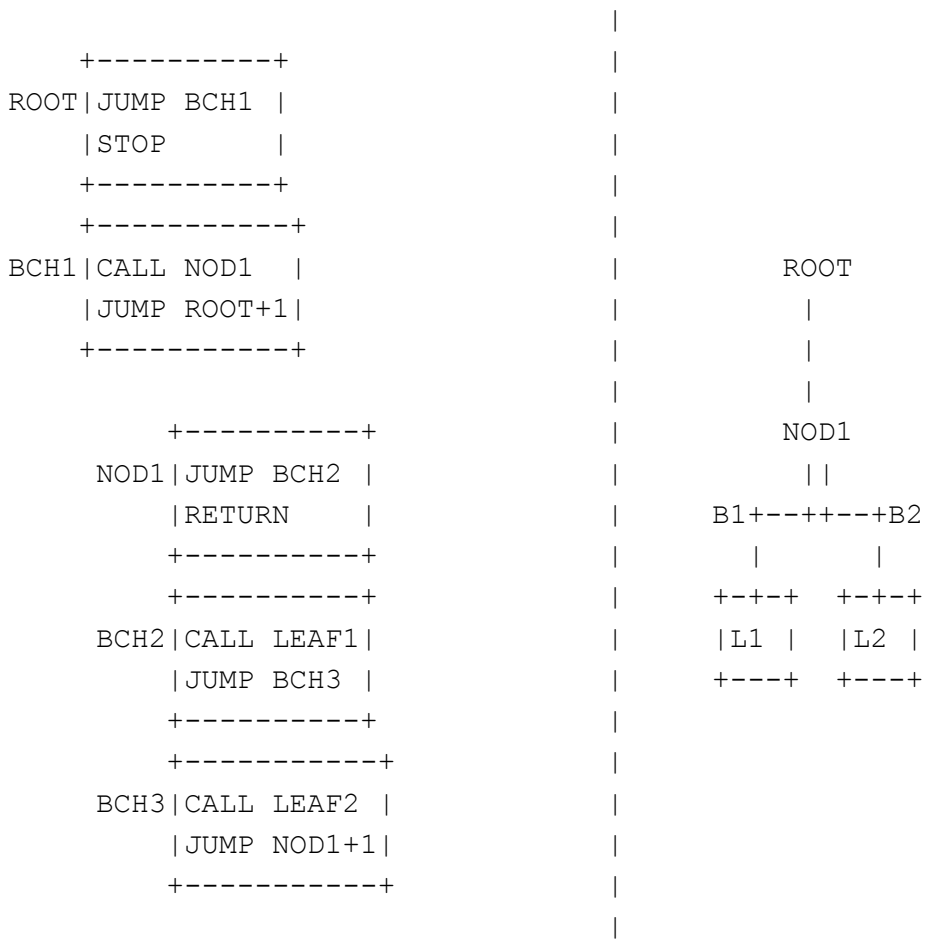


Using branches with translations, you can have one leaf appearing at two places on the screen.



5.4 Nodes

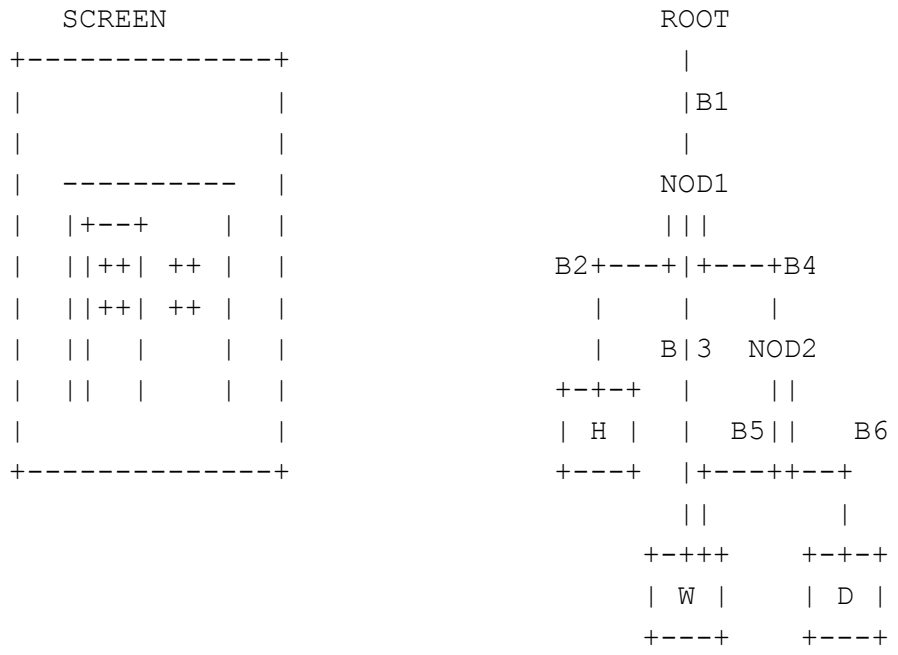
With several leaves attached to the root node it might be disirable to move this entire picture. For this purpose, the user sets up a more complex structure by calling #SNC (SELNC) to create his own nodes. He may then attach leaves to them and then attach the node to the root node.



Then altering the coordinates of B1 moves the whole picture while altering the coordinates of B2 or B3 alter only a part of it.

5.5 Example Program

To illustrate the operation of the display control when executing a typical program and to describe the function of the 339's push down stack mechanism we will consider the following picture and display structure.



Leaf H draws the outer walls of the house, leaf W draws a square window and leaf D draws a door frame. NOD2 represents the entire door consisting of frame and window, while NOD1 represents the entire house consisting of walls, window and door.

The display control starts execution at ROOT. It jumps to branch B1 which calls NOD1 as a subroutine. The 339 subroutine call instruction is called a push-jump, that is, it saves the return address in a core buffer called a push-down-stack. Thus while executing the NOD1 subroutine, the address of B1 is on the push down stack. NOD1 jumps to branch B2 which calls leaf H to draw the walls. While the 339 executes this leaf the push down stack looks like:

B1
B2

When leaf H returns the 339 removes the address of B2 from the stack and jumps to branch B3. This calls leaf W to draw the window in the wall of the house and the push down stack looks like:

B1
B3

Upon return from W the display jumps to branch B4 which calls NOD2. This jumps to branch B5 which calls W again. While the 339 draws the window in the door the push down stack looks like:

B1
B4
B5

W returns and the 339 jumps to B6 and calls leaf D. while the door frame is being drawn the PDS looks like:

B1
B4
B6

The frame finally terminates by NOD2 returning to NOD1, NOD1 returning to ROOT and the display stopping.

In summary, note that while the 339 executes a leaf the push down stack contains the addresses of the branches gone through to get from ROOT to that leaf. This information is quite useful when processing light pen hits on complex display structures.

6. THE LIGHT PEN

The light pen serves as the input device from the display. With it the user can input X,Y coordinate data (via "tracking" or he can indicate parts of the displayed picture (via "light pen hits").

6.1 LIGHT PEN hits

A light pen hit occurs whenever the user points at an intensified portion of the screen while depressing the shutter button of the light pen. Upon detecting a light pen hit, the system saves the entire status of the 339 hardware (339 program counter, X coordinate, Y coordinate, push down stack), schedules a user-specified service task and restarts the display. The service task can then interrogate the saved display status at leisure without the screen going blank.

A light pen hit on a displayed leaf saves, in the push down stack, the path through the branches of the structure taken to get from the root node to the leaf to display it. This information allows the users' program to easily determine what was pointed to as well as where on the screen it is. This information is very hard to obtain from input devices providing only X,Y coordinate input. For example, in one program an electronic circuit appears on the screen with the connectivity of the elements in the circuit built into the display structure. A light pen hit on, say, a resistor will tell not only that it was a resistor (via the leaf name), but also which resistor (via the path from the root node to the leaf), and what its connections are to the other elements on the screen (via the connections between nodes).

6.2 Tracking

A user program can follow motions of the light pen by a process called tracking. This procedure is required due to the fact that the hardware can only detect the position of the light pen if there is light under it. Tracking is usually initiated as a result of a light pen hit by calling the routine #XI which places the a small "tracking cross" at the coordinates of the hit. The system automatically adjusts the cross's position to keep it centered on the light pen. A local user program can continuously monitor the position of the cross by calling #XX and #XY which return the current coordinates of the cross. Tracking continues until explicitly stopped by call to #XT (SELXT) or until the user closes the shutter of the light pen, thus preventing it from seeing the tracking cross. The program can detect this condition by a call to #XS (or SELINP) and use it as a signal to perform some action.